Digital Photography II

The Image Processing Pipeline

EE367/CS448I: Computational Imaging
stanford.edu/class/ee367
Lecture 4

Gordon Wetzstein
Stanford University

Review – “Sensors are Buckets”

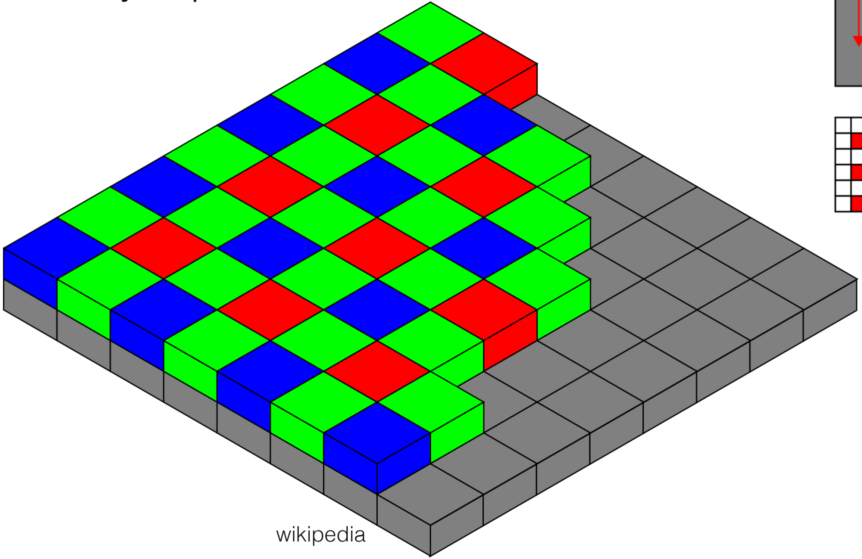
collect photons like a bucket

integrate spectrum

integrate incident directions

Review – Color Filter Arrays

Bayer pattern



wikipedia

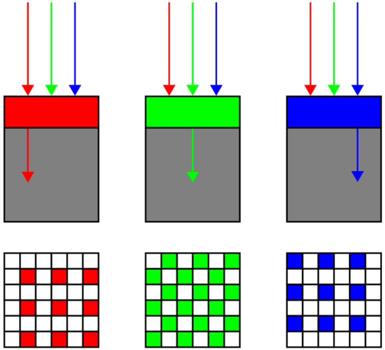


Image Formation

- high-dimensional integration over angle, wavelength, time

plenoptic function

↓

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

plenoptic function:
[Adelson 1991]

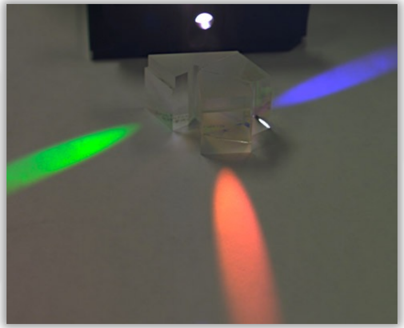
More Ways to Capture Color

field sequential

Prokudin-Gorsky

multiple sensors

green sensor

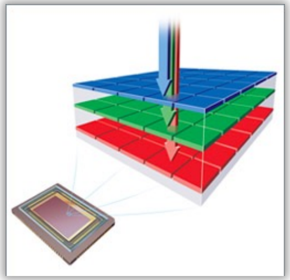


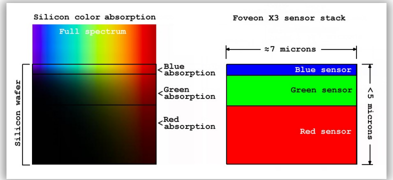
blue sensor

wikipedia

red sensor

vertically stacked

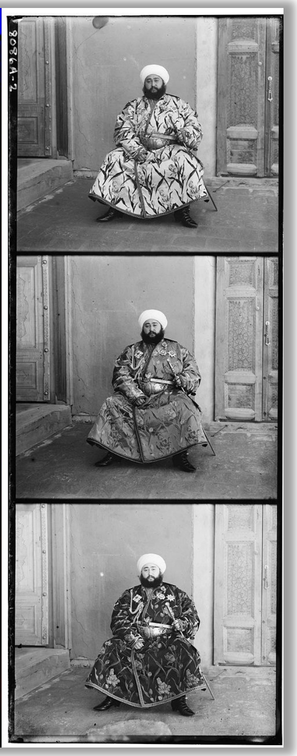




Foveon X3

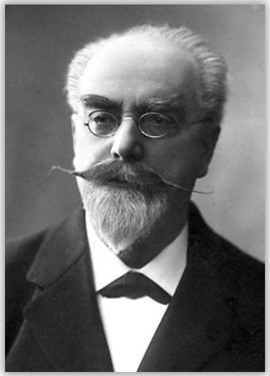
More Ways to Capture Color

Prokudin-Gorsky



Alim Khahn, Emir of Bukhara, 1911

More Ways to Capture Color



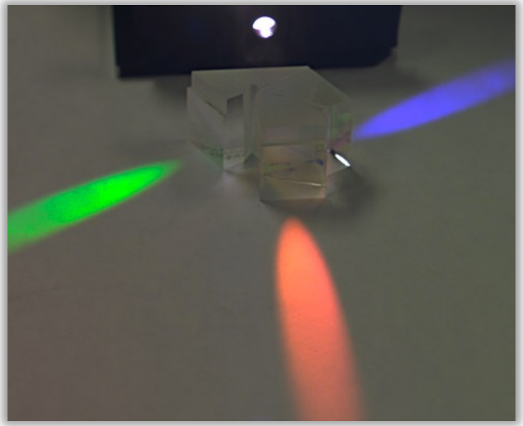
Gabriel Lippmann

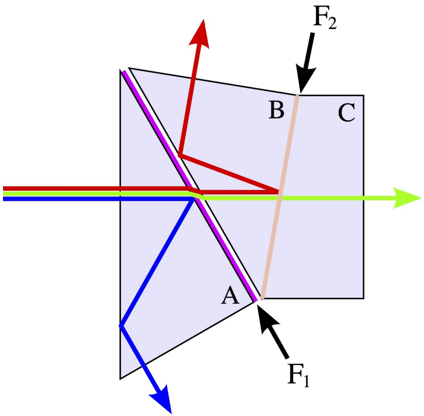
- notable French inventor
- Nobel price for color photography in 1908 = volume emulsion capturing interference
- today, this process is most similar to volume holography!
- also invented integral imaging (will hear more…)

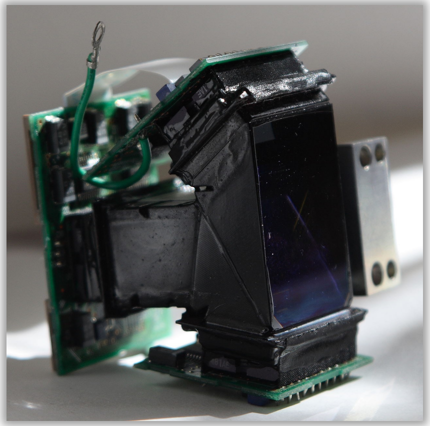
Lippmann's stuffed parrot

Three-CCD Camera

beam splitter prism

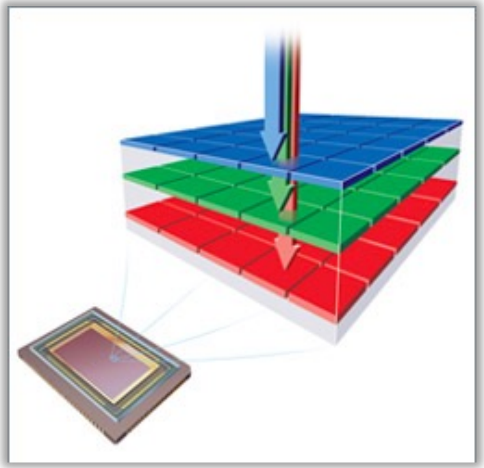




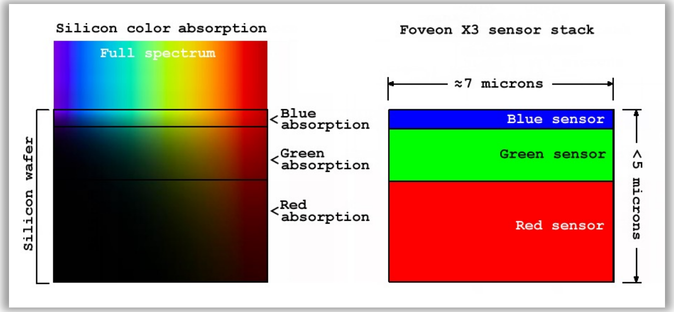


Philips / wikipedia

Stacked Sensor



Foveon X3

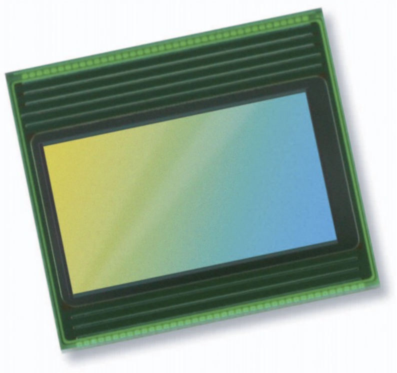


Sigma SD9

Other Wavelengths

- OmniVision:

  RGB + near IR!



Product Specifications

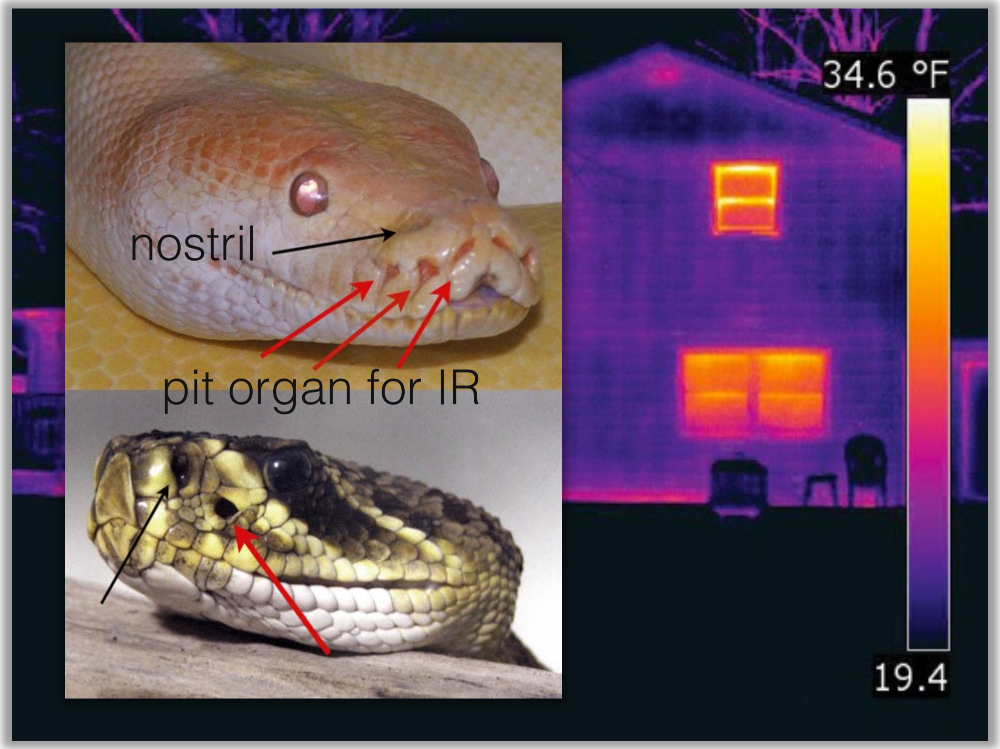
Part Number	**OV4682-G04A**
Resolution	4MP
Chroma	Color
Analog / Digital	Digital
Power Requirement	Active: 163 mA (261 mW) Standby: 1 mA XSHUTDOWN: <10 µA
Temperature Range	Operating: -30°C to +85°C junction temperature Stable image: 0°C to +60°C junction temperature
Output Format	10-bit RAW data
Optical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



Review: Photons to RAW Image

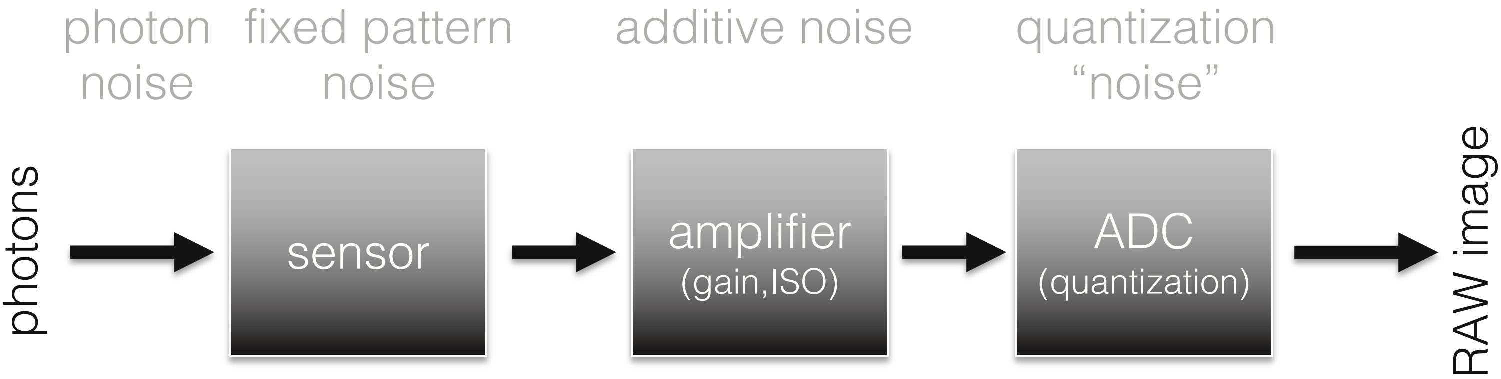


Image Processing Pipeline

RAW image
(dcraw –D)

JPEG image

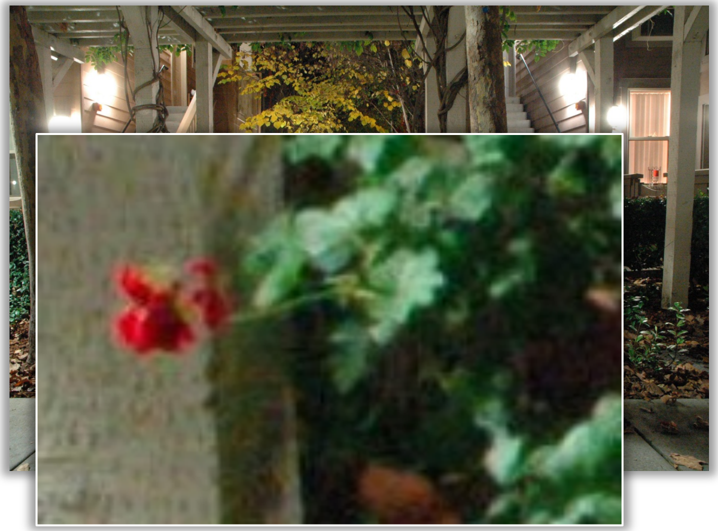
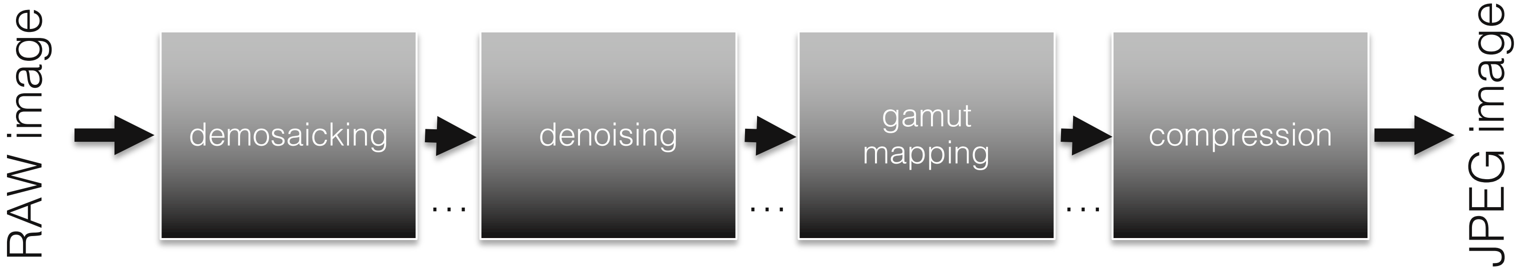


Image Processing Pipeline

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

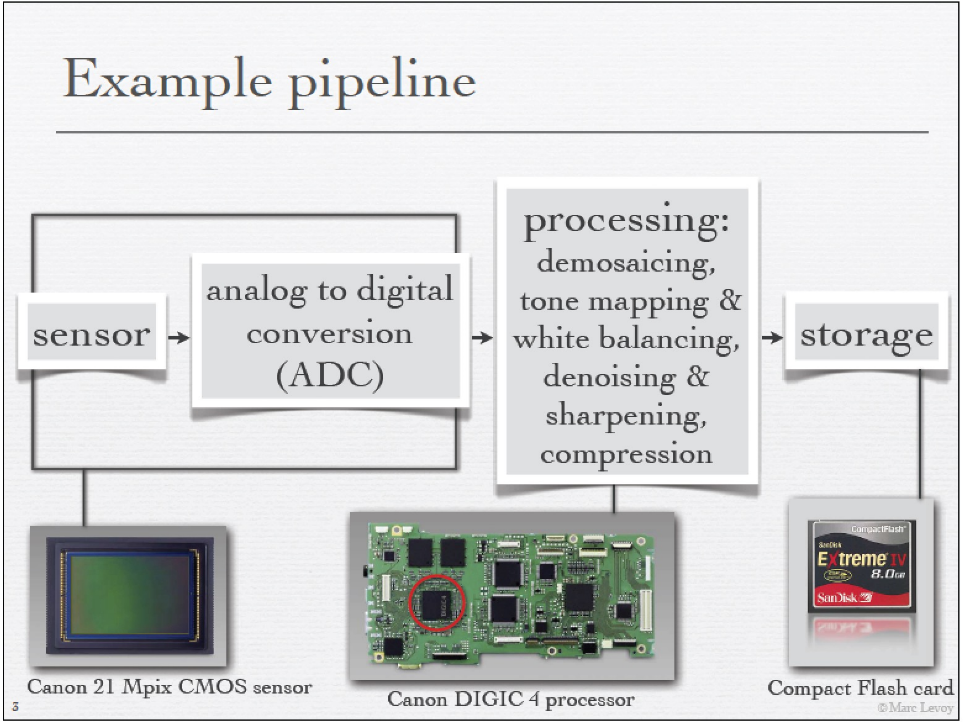
Image Processing Pipeline



also:

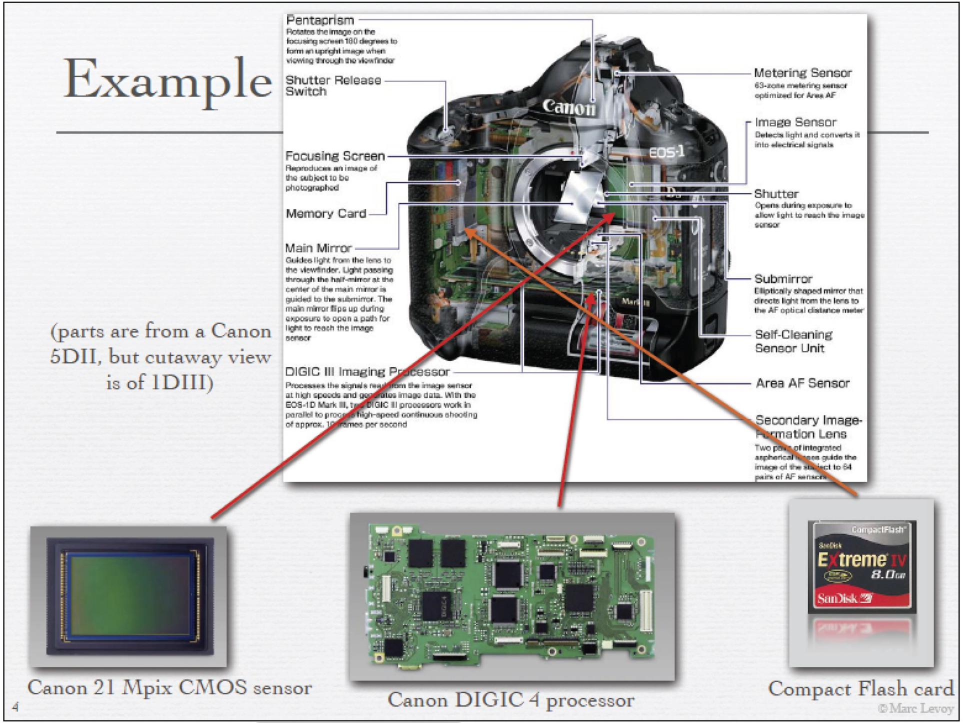
- dead pixel removal
- dark frame subtraction (fixed pattern / thermal noise removal)
- lens blur / vignetting / distortion correction
- sharpening / edge enhancement

Image Processing Pipeline



Marc Levoy, CS 448

Image Processing Pipeline



Marc Levoy, CS 448

Exif Meta Data

exchangeable image file format

Filename - night_nikon.JPG
Make - NIKON CORPORATION
Model - NIKON D70s
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

Filename - night_nikon.JPG
Make - NIKON CORPORATION
Model - NIKON D70s
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

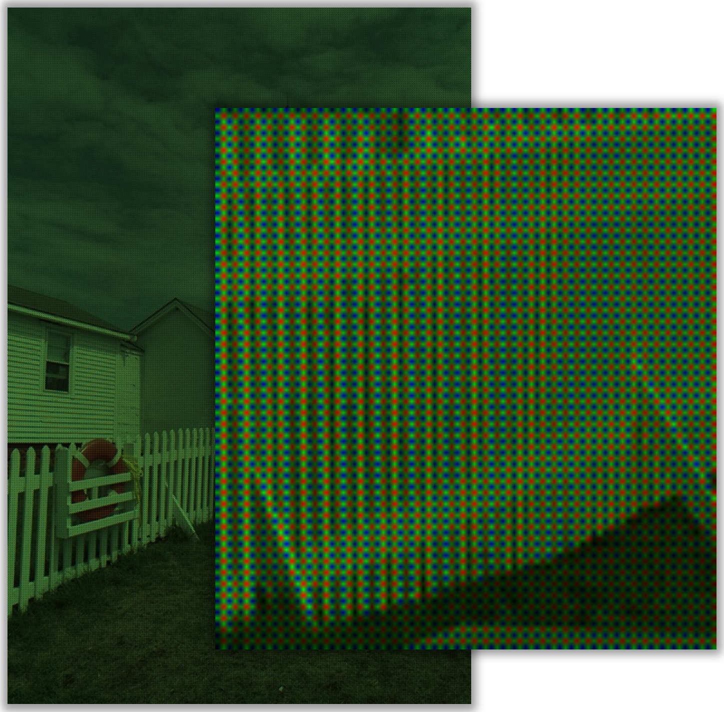
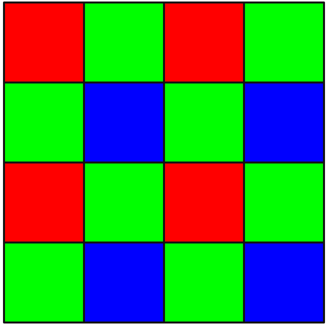


image from Kodac dataset



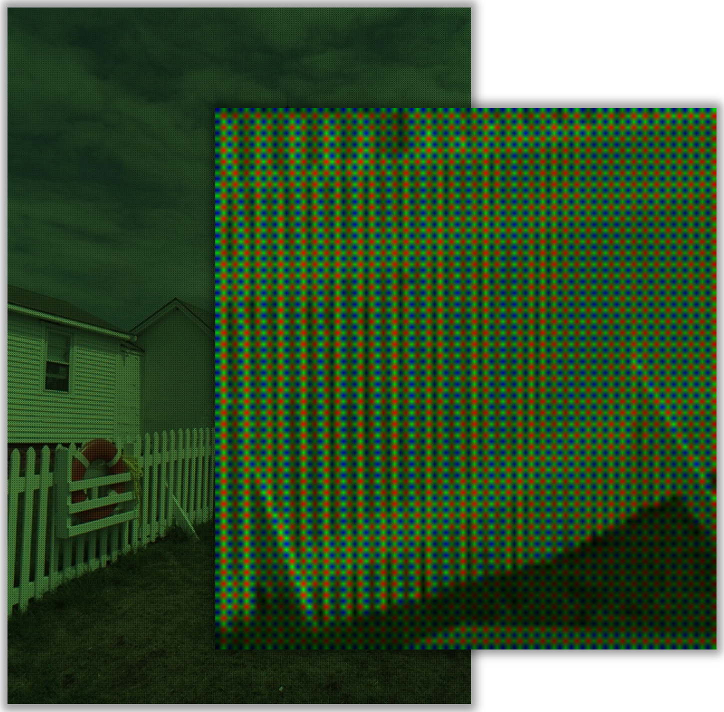
Bayer CFA

Demosaicking (CFA Interpolation)

RAW

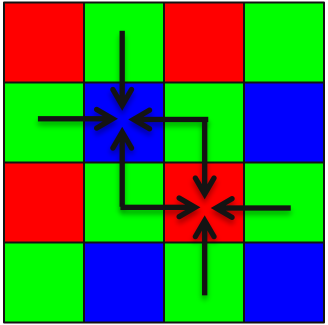
linear interpolation green channel

image from Kodac dataset



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

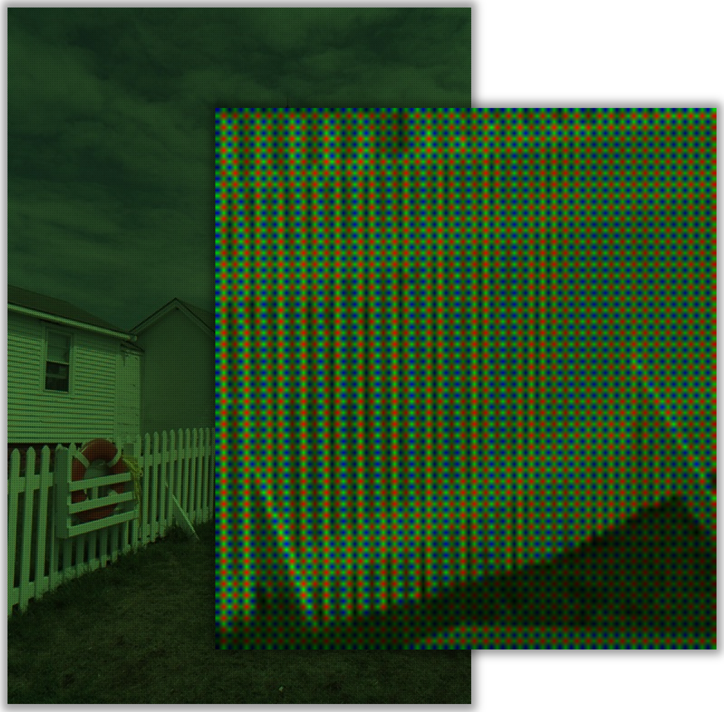
$$(m,n) = \{(0,-1),(0,1),(-1,0),(1,0)\}$$



Bayer CFA

Demosaicking (CFA Interpolation)

RAW



linear interpolation

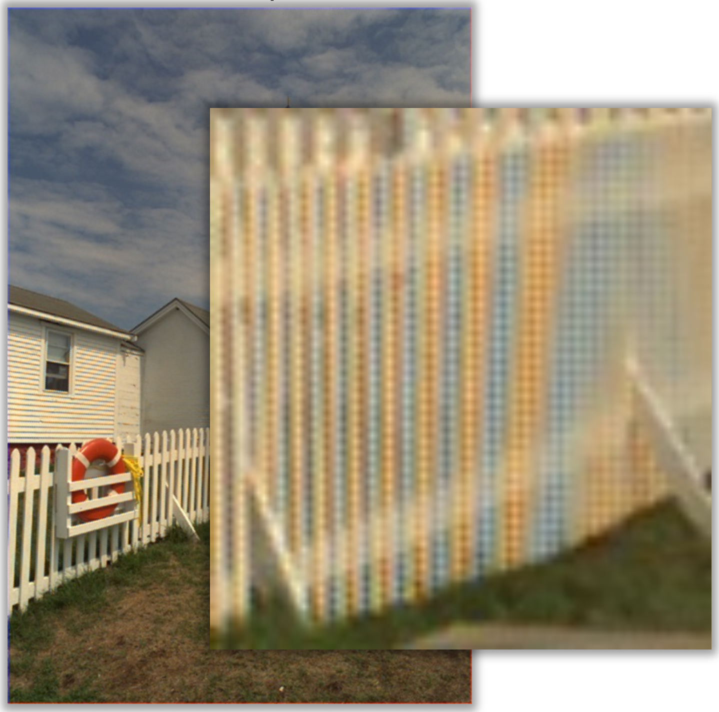


image from Kodac dataset

Demosaicking (CFA Interpolation)

original | RAW | demosaicked

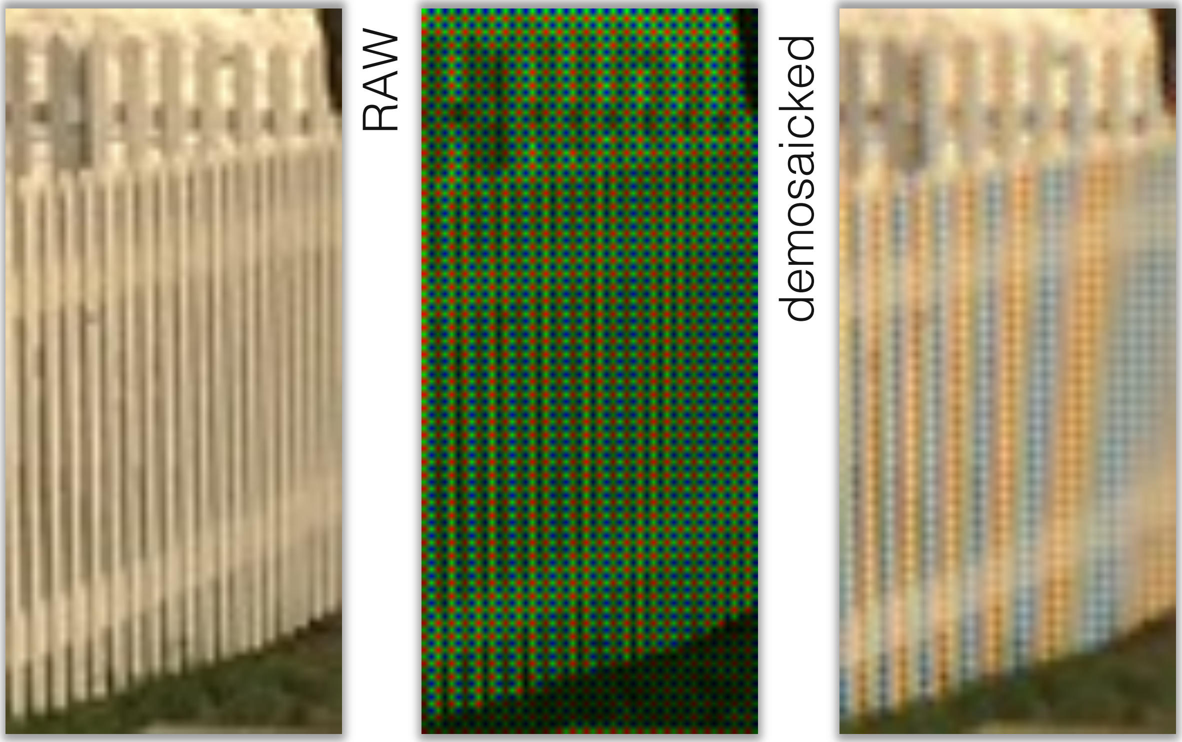
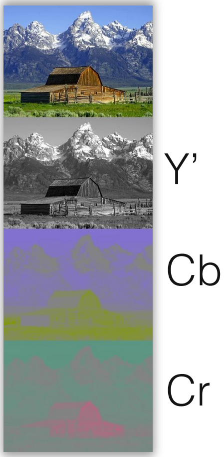


image from Kodac dataset

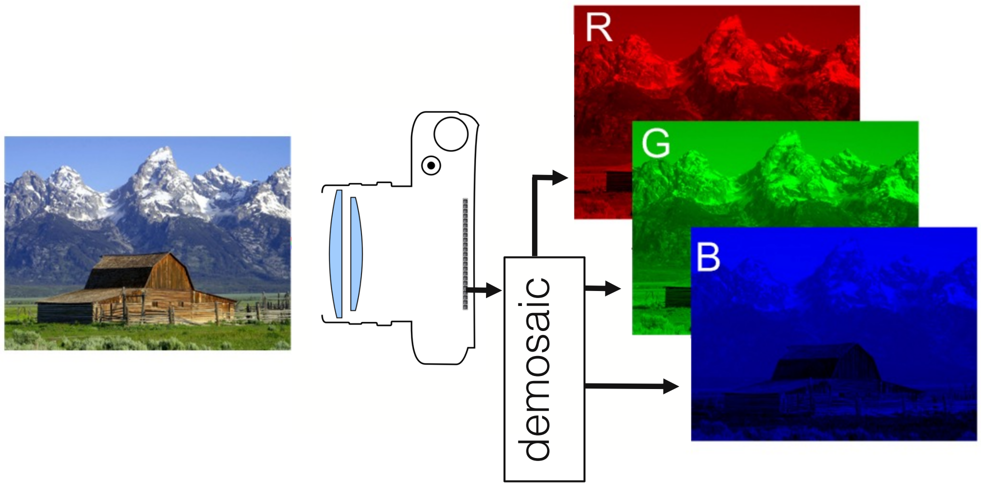
Demosaicing – Low-pass Chroma

- sampling problem (despite optical AA filter): (too) high-frequency red/blue information

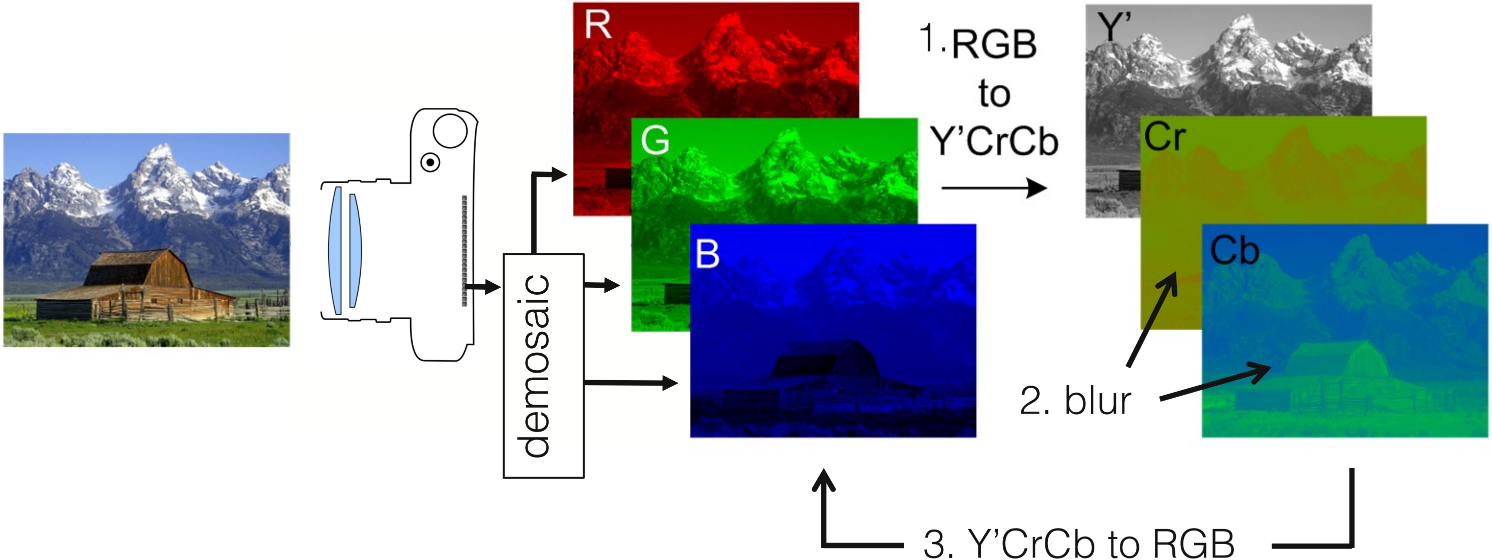
- simple solution: low-pass filter chrominance – humans are most sensitive to “sharpness” in luminance:
  1. apply naïve interpolation
  2. convert to Y’CbCr (related to YUV)
  3. median filter chroma channels: Cb & Cr
  4. convert back to RGB



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:

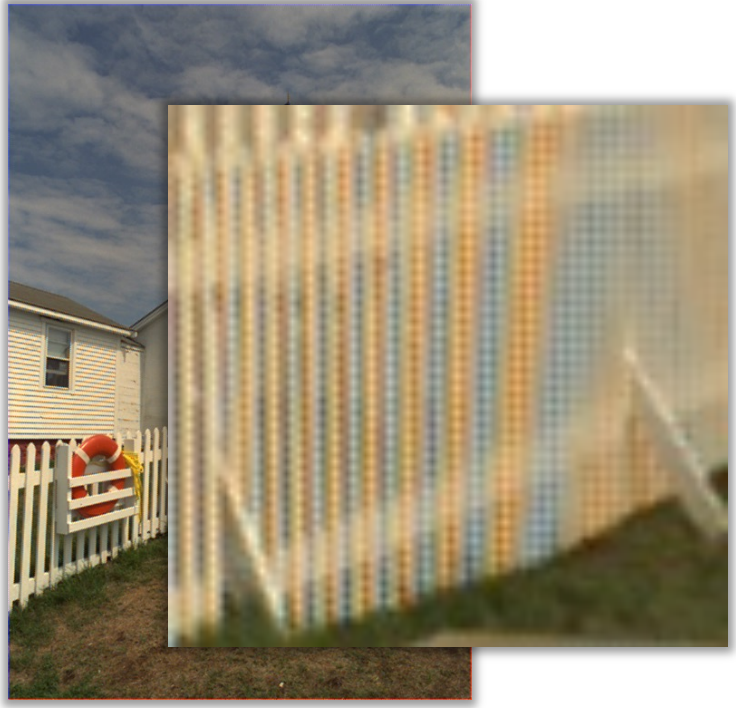
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = M^{-1} \left(\begin{bmatrix} Y' \\ Cb \\ Cr \end{bmatrix} - \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} \right) \cdot \frac{65535}{257}$$

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

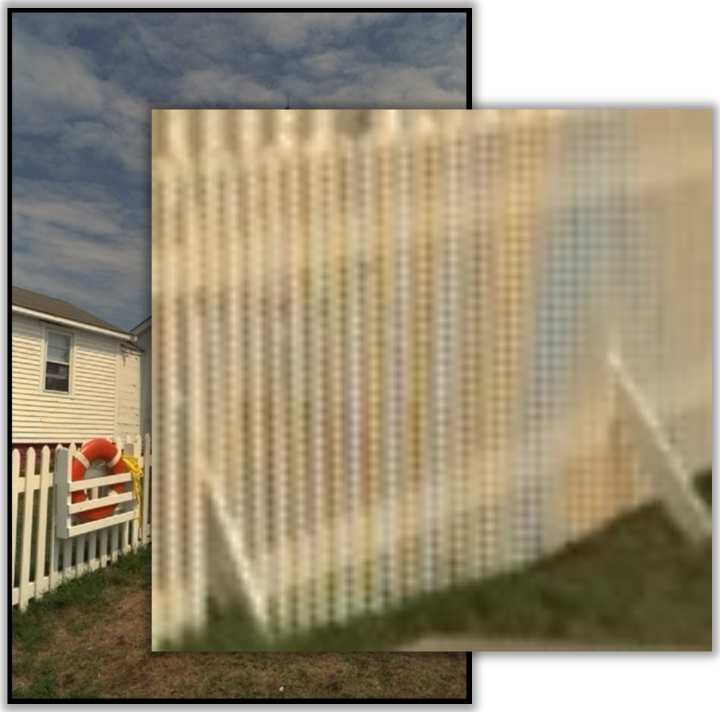
Pixel values for above equations between 0 and 255!

Demosaicing – Low-pass Chroma

linear interpolation

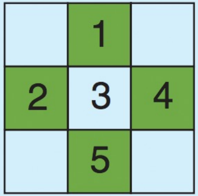


chrominance filtered



Demosaicing – Edge-Directed Interpolation

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

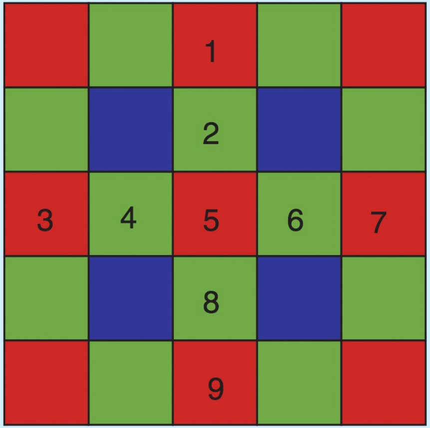


1. Calculate horizontal gradient $\Delta H = |G2 - G4|$
2. Calculate vertical gradient $\Delta V = |G1 - G5|$
3. If $\Delta H > \Delta V$,
   $G3 = (G1 + G5)/2$
   Else if $\Delta H < \Delta V$,
   $G3 = (G2 + G4)/2$
   Else
   $G3 = (G1 + G5 + G2 + G4)/4$

from Gunturk et al. 2005

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

   $G5 = (G2 + G8 + G4 + G6)/4$

from Gunturk et al. 2005

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

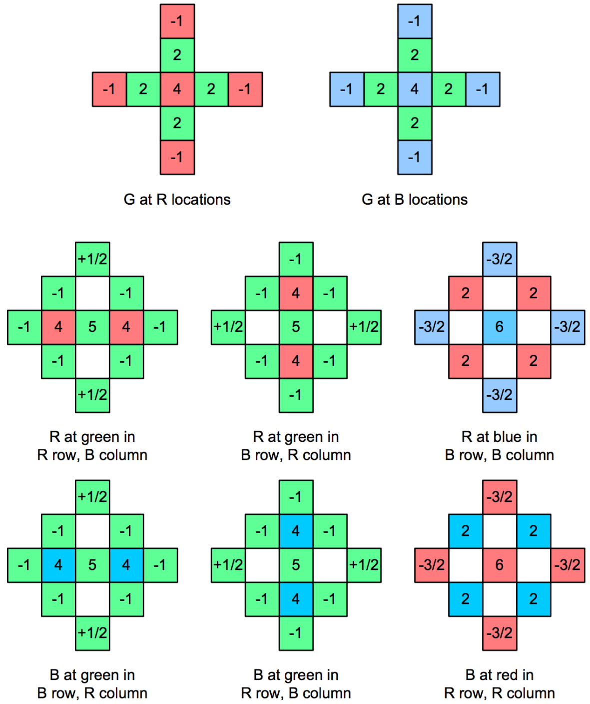
$(m,n) = \{(0,-2),(0,2),(-2,0),(2,0)\}$

- 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

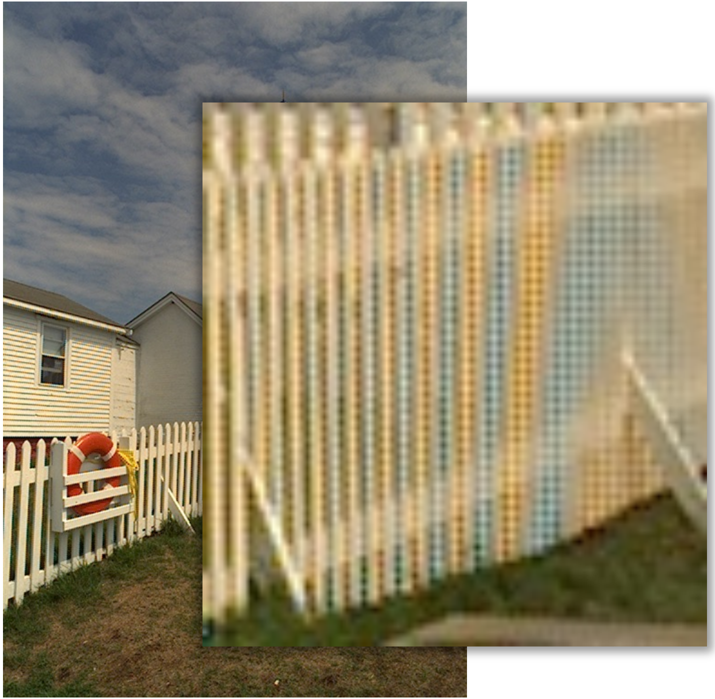
- write out math to get linear filters:
- use normalized filters in practice, i.e. scale numbers by sum of filter



Demosaicing - Malvar et al. 2004

linear interpolation

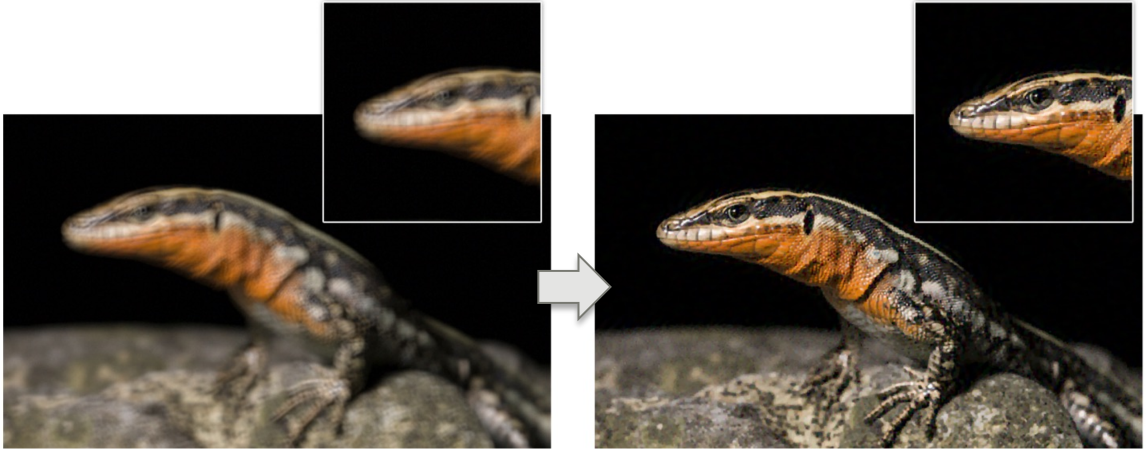
Malvar et al.



Deblurring / Deconvolution

common sources:
out-of-focus blur
geometric distortion
spherical aberration
chromatic aberration
coma

from Heide et al. 2016



Blurred input image

Deblurred / deconvolved image

Denoising

noisy image

(Gaussian iid noise, σ=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

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

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

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

Denoising – 2. Local, Nonlinear Filtering

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

small window of image i_{noisy} centered at x

- almost as naïve: use median filter in <u>local</u> neighborhood

Denoising

noisy image (Gaussian, σ=0.2)

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

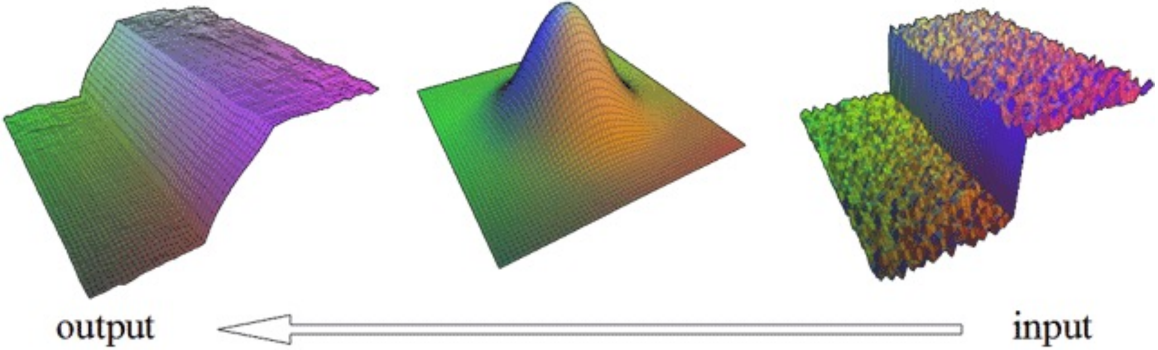
$$w(x,x') = \exp\left(-\frac{\left\|x'-x\right\|^2}{2\sigma^2}\right) \cdot \exp\left(-\frac{\left\|i_{noisy}\left(x'\right) - i_{noisy}\left(x\right)\right\|^2}{2\sigma_i^2}\right)$$

- more clever: average in <u>local</u> neighborhood, but only average similar intensities!

Denoising – Gaussian Filter

J: filtered output (is blurred)
f: Gaussian convolution kernel
I: step function & noise

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

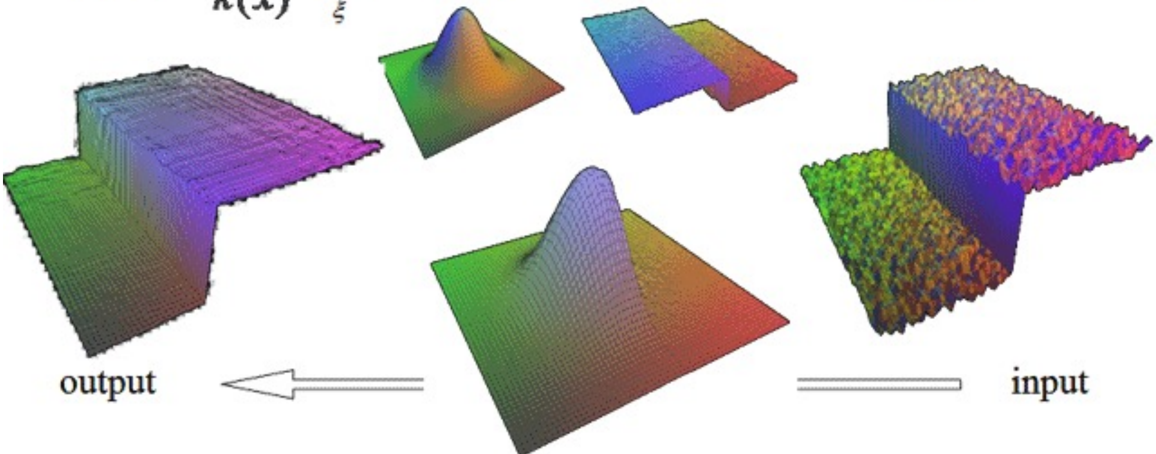


Denoising – Bilateral Filter

J: filtered output (is not blurred)
f: Gaussian convolution kernel
I: noisy image (step function & noise)

difference in intensity as scale!

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



Tomasi & Manduchi, 1998

Denoising – Bilateral Filter

original image

bilateral filter = “edge-aware smoothing”

Denoising – Bilateral Filter

noisy image

bilateral filter = “edge-aware smoothing”

Denoising – 4. Non-local Means

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

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

[Buades 2005]

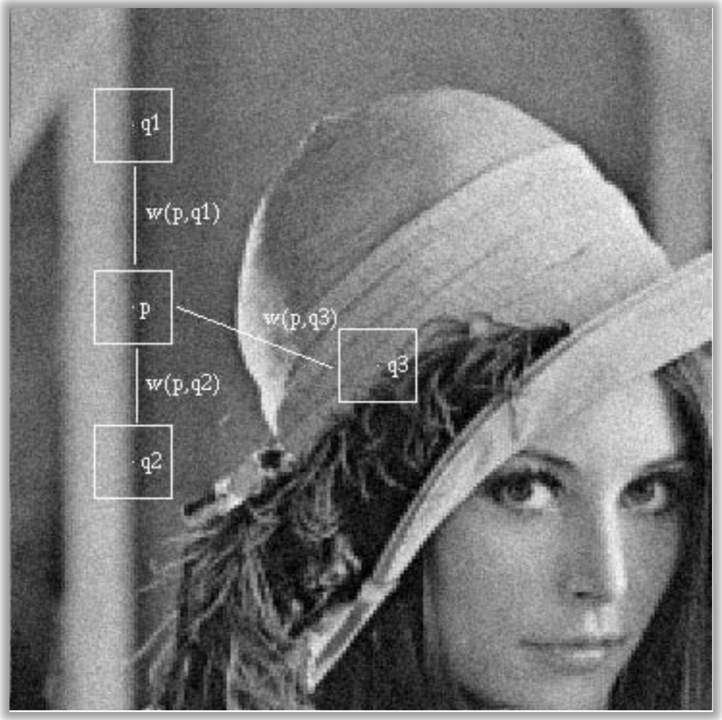
- very powerful approach: exploit self-similarity in image; average pixels with a similar neighborhood, but don't need to be close → non-local

Denoising – 4. Non-local Means

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

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

[Buades 2005]



Denoising – 4. Non-local Means

[Buades 2005]

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

[Dabov 2006]

Denoising

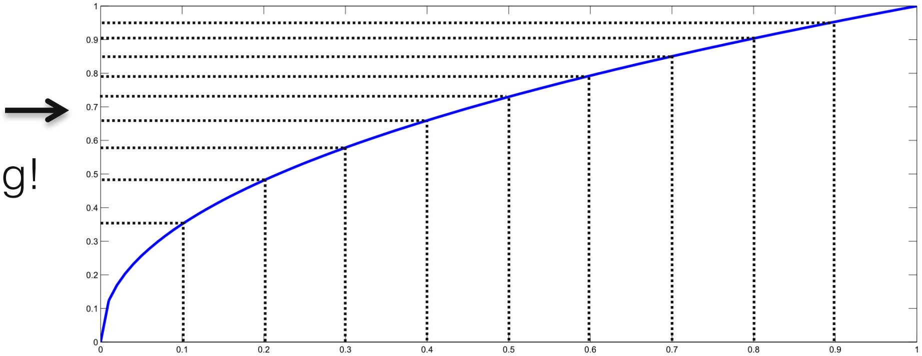
- many methods for denoising (check Buades 2005):
    - filtering wavelet or other coefficients
    - total variation denoising
    - patch-based or convolutional sparse coding …

- state of the art: non-local methods, in particular BM3D

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 γ=2.2

perceptually linear spacing!



Gamma Correction in sRGB

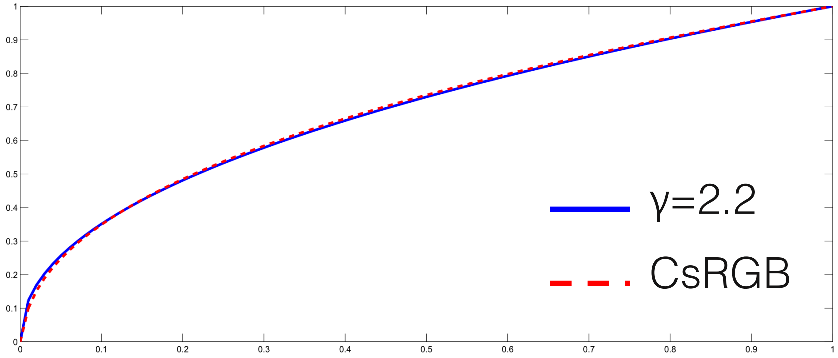
- standard 8 bit color space of most images, e.g. jpeg
- roughly equivalent to γ=2.2

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

linear

gamma

$$a = 0.055$$



Compression – JPEG (joint photographic experts group)

jpeg – ps quality 0

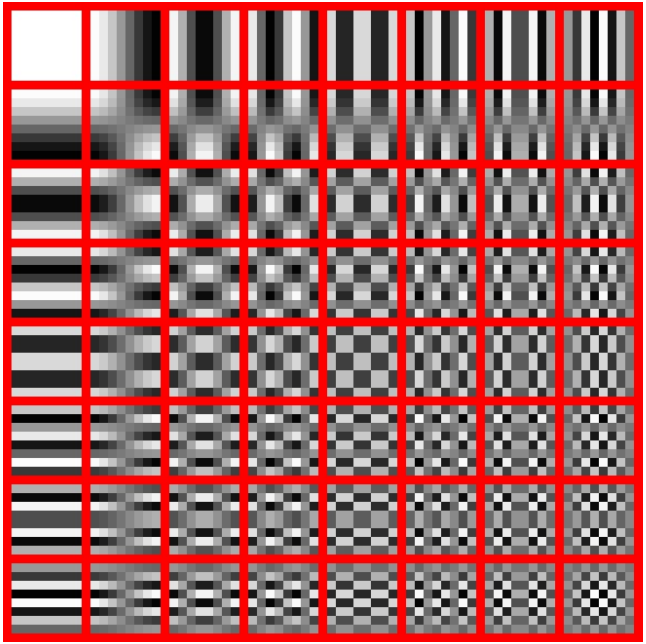
jpeg – ps quality 2

original

Compression – JPEG (joint photographic expert group)

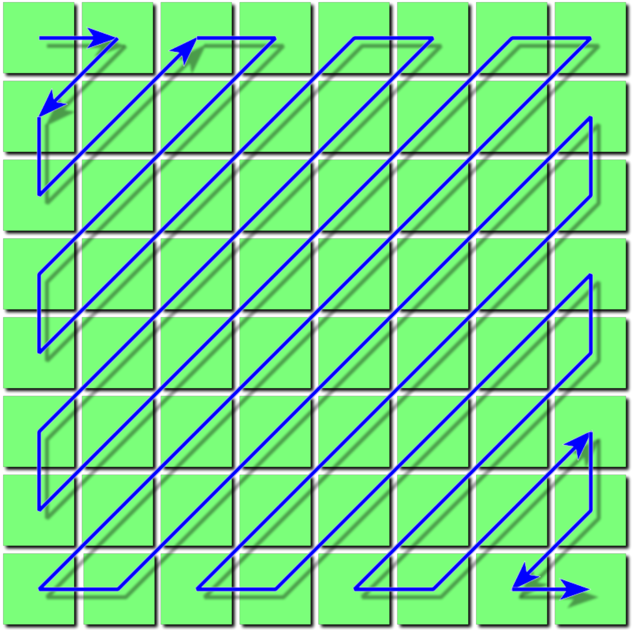
1. transform to YCbCr
2. downsample chroma components Cb & Cr
   - 4:4:4 – no downsampling
   - 4:2:2 – reduction by factor 2 horizontally
   - 4:2:0 – reduction by factor 2 both horizontally and vertically
3. split into blocks of 8x8 pixels
4. discrete cosine transform (DCT) of each block & component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

Compression – JPEG (joint photographic expert group)



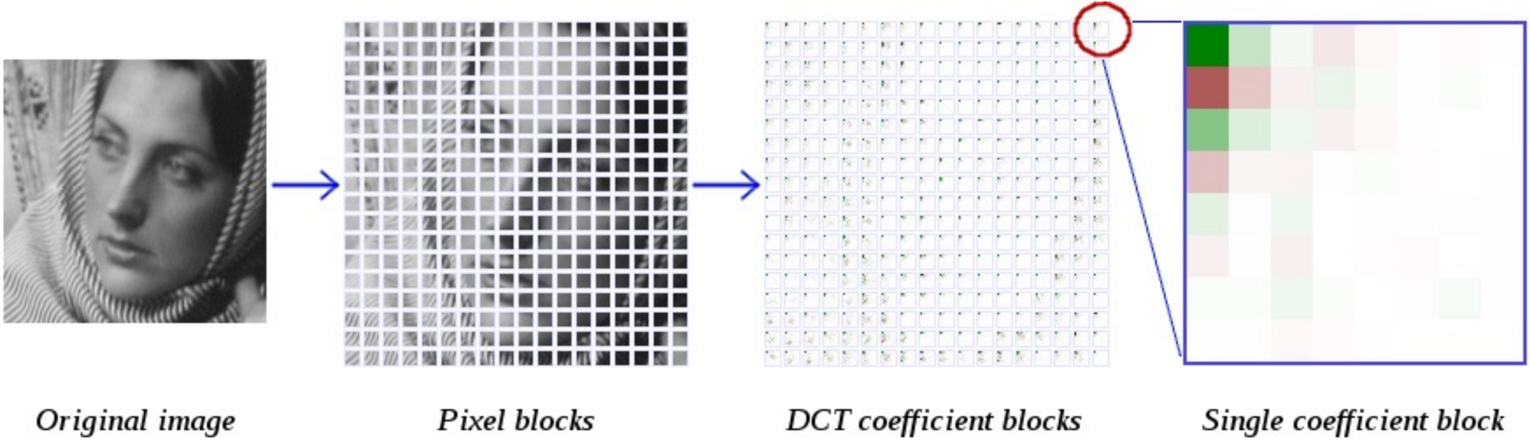
wikipedia

DCT basis functions



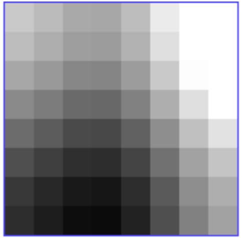
RLE of “same frequency” coefficients

Compression – JPEG (joint photographic expert group)



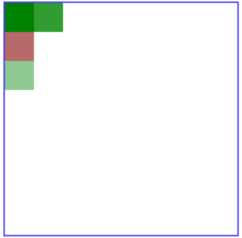
http://xiph.org/~xiphmont/demo/daala/demo1.shtml

Compression – JPEG (joint photographic expert group)



Original pixel data

114	108	100	99	109	129	152	166
109	102	95	94	104	124	146	161
99	93	85	84	94	114	137	151
86	80	72	71	82	102	124	138
73	66	58	57	68	88	110	125
60	53	46	45	55	75	97	112
50	43	36	35	45	65	88	102
45	38	31	30	40	60	82	97

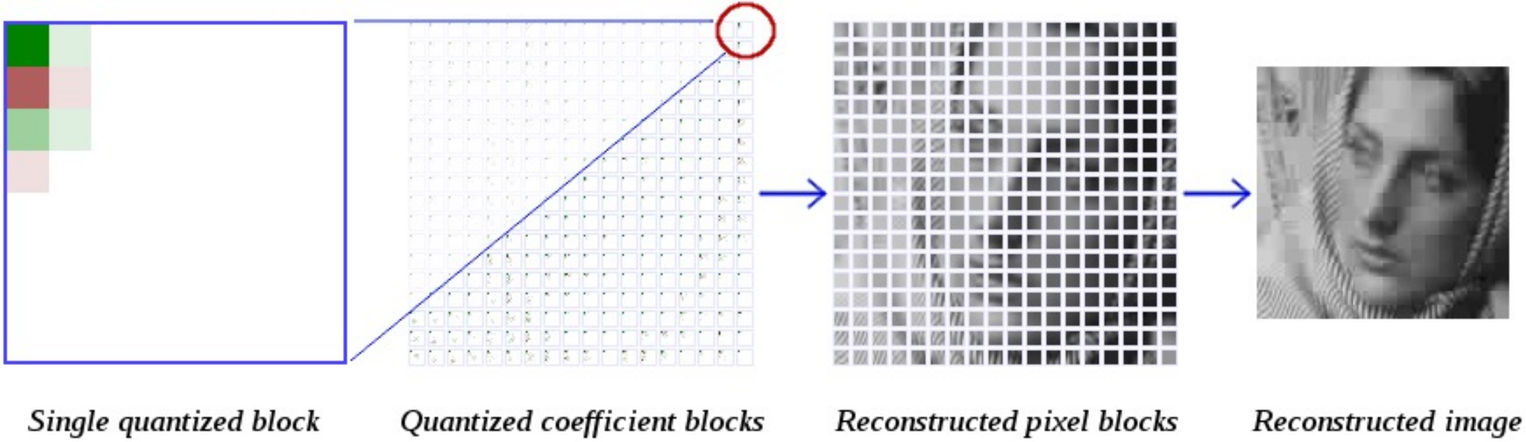


DCT coefficient data

700	200	0	0	0	0	0	0
−150	0	0	0	0	0	0	0
110	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

http://xiph.org/~xiphmont/demo/daala/demo1.shtml

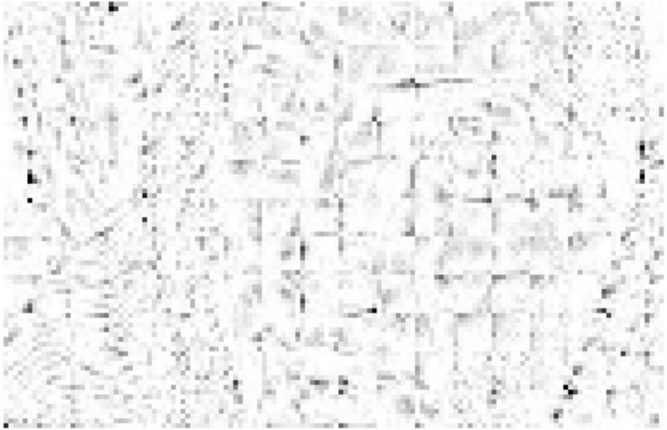
Compression – JPEG (joint photographic expert group)



http://xiph.org/~xiphmont/demo/daala/demo1.shtml

Compression – JPEG (joint photographic expert group)

Closeup of reconstructed image



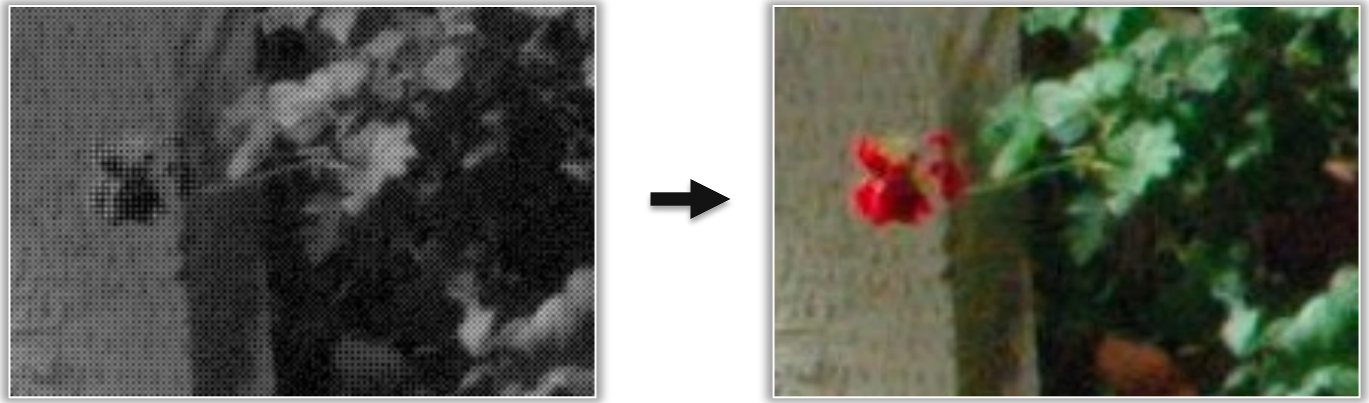
Normalized error distribution within each block

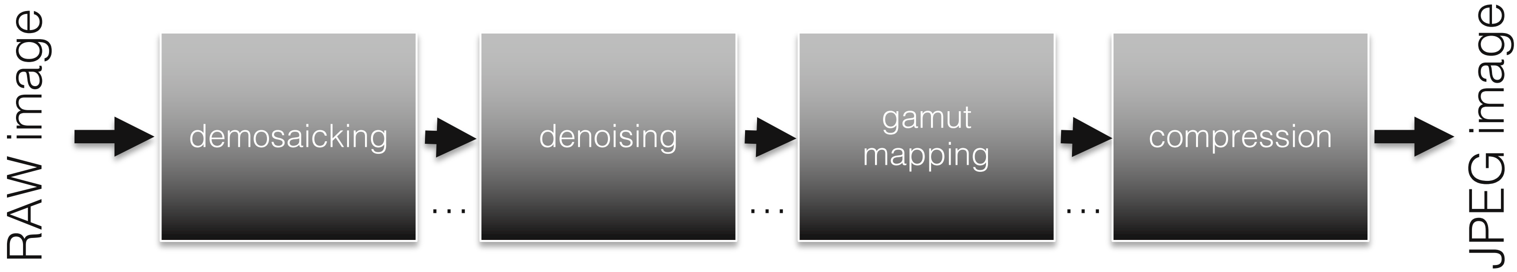
http://xiph.org/~xiphmont/demo/daala/demo1.shtml

Compression – JPEG (joint photographic experts group)

jpeg – ps quality 0 | jpeg – ps quality 2 | original

Image Processing Pipeline





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/