Digital Photography II

color & image processing pipeline

*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

Bayer pattern
Image Formation

- high-dimensional integration over angle, wavelength, time

\[ i(x) \approx \int \int \int l(x, \theta, \lambda, t) d\theta d\lambda dt \]

plenoptic function

plenoptic function: [Adelson 1991]
More Ways to Capture Color

- Field sequential
- Multiple sensors
- Vertically stacked

- Prokudin-Gorsky
- Wikipedia
- Foveon X3
More Ways to Capture Color

• Russian chemist and photographer
• used Maxwell's color photography technique (1855)
• commissioned by Tsar Nicholas II, photo documented diversity of Russian empire from 1909-1915
• ~3500 negatives

Sergei Prokudin-Gorsky

Alim Khahn, Emir of Bukhara, 1911
More Ways to Capture Color

- 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...)

Gabriel Lippmann

Lippmann’s stuffed parrot
Three-CCD Camera

beam splitter prism

Philips / wikipedia
Stacked Sensor

Foveon X3

Sigma SD9
Other Wavelengths

- OmniVision: RGB + near IR!

<table>
<thead>
<tr>
<th>Product Specifications</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Part Number</strong></td>
</tr>
<tr>
<td><strong>Resolution</strong></td>
</tr>
<tr>
<td><strong>Chroma</strong></td>
</tr>
<tr>
<td><strong>Analogue / Digital</strong></td>
</tr>
<tr>
<td><strong>Power Requirement</strong></td>
</tr>
<tr>
<td><strong>Temperature Range</strong></td>
</tr>
<tr>
<td><strong>Output Format</strong></td>
</tr>
<tr>
<td><strong>Optical Format</strong></td>
</tr>
<tr>
<td><strong>Frame Rate</strong></td>
</tr>
<tr>
<td><strong>Pixel Size</strong></td>
</tr>
<tr>
<td><strong>Image Area</strong></td>
</tr>
<tr>
<td><strong>Package</strong></td>
</tr>
<tr>
<td><strong>Package Dimensions</strong></td>
</tr>
<tr>
<td><strong>Product Brief</strong></td>
</tr>
</tbody>
</table>
Other Wavelengths

- 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.

What we call “color” is how we subjectively perceive a very small range of these wavelengths.
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:

$$ R = \int_{\lambda} \Phi(\lambda) f(\lambda) d\lambda $$

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

\[ c(\lambda_i) = (c_s, c_m, c_l) \]

LMS sensitivity functions

“pure beam” (laser)
The retinal color space

c(λᵢ) = (cₛ, cₘ, c_l)

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

“pure beam” (laser)
The retinal color space

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

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

\[ c(\ell_{\lambda_i}) = (c_s, c_m, c_l) \]

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

\[ c(\ell_{\lambda_i}) = (c_s, c_m, c_l) \]

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

= convex combination of pure colors
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
Adjust the strengths of the primaries until they re-produce the test color. Then:

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

equality symbol means “has the same retinal color as” or “is metameric to”
To match some test colors, you need to add some primary beam on the left (same as “subtracting light” from the right)

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

\[
\longrightarrow \quad \mathbf{c}(\ell(\lambda)) = \alpha \mathbf{c}(\ell_{435}) + \beta \mathbf{c}(\ell_{535}) - \gamma \mathbf{c}(\ell_{625})
\]
Repeat this matching experiments for pure test beams at wavelengths $\lambda_i$ and keep track of the coefficients (negative or positive) required to reproduce each pure test beam.

$$\mathbf{c}(\lambda_i) = k_{435}(\lambda)\mathbf{c}(\ell_{435}) + k_{535}(\lambda)\mathbf{c}(\ell_{535}) + k_{625}(\lambda)\mathbf{c}(\ell_{625})$$
Repeat this matching experiments for pure test beams at wavelengths $\lambda_i$ and keep track of the coefficients (negative or positive) required to reproduce each pure test beam.

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

What about “mixed beams”?
Two views of retinal color

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

**Synthetic:** Retinal color is produced by synthesizing color primaries using the color matching functions.
Two views of retinal color

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

Synthetic: Retinal color is produced by synthesizing color primaries using the color matching 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.
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!
Created by the International Commission on Illumination in 1931 based on color matching experiments from 12 people!
CIE XYZ colorspace
CIE $xy$ (chromaticity)

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

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

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

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.
What does the boundary of the chromaticity diagram correspond to?
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

sRGB impossible colors

sRGB realizable colors

sRGB color primaries
Color gamuts

Gamuts of various common industrial RGB spaces

What is this?
The problem with RGBs visualized in chromaticity space

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

- Photons
- Fixed pattern noise
- Additive noise
- Quantization "noise"

1. Sensor
2. Amplifier (gain, ISO)
3. ADC (quantization)

RAW image
Image Processing Pipeline

- demosaicking
- denoising
- gamut mapping
- compression

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
(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

Example pipeline

- **Sensor** → analog to digital conversion (ADC) → 
  - processing: demosaicing, tone mapping & white balancing, denoising & sharpening, compression → 
  - storage

- Canon 21 Mpix CMOS sensor
- Canon DIGIC 4 processor
- Compact Flash card

Marc Levoy, CS 448
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
Exif Meta Data

Filename - night_nikon.JPG
Make - NIKON CORPORATION
Model - NIKON D70S
Orientation - Top left
XResolution - 300
YResolution - 300
ResolutionUnit - Inch
Software - Ver.1.00
DateTimeOriginal - 2005:09:01 12:16:43
YCbCrPositioning - Co-Sited
ExifOffset - 216
ExposureTime - 10 seconds
FNumber - 13.00
ExposureProgram - Manual control
Maker Note (Vendor): -
Data version - 0210 (808595760)
ISO Setting - 1000
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
Thumbail IFD offset - 1430
Flash Compensation - 67072
ISO - 2-1600
Tone Compensation - AUTO

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

LightSource - Auto
Flash - Not fired
FocalLength - 18.00 mm
UserComment - (c) Gordon Wetzstein
SubsecTime - 00
SubsecTimeOriginal - 00
SubsecTimeDigitized - 00
FlashPixVersion - 0100
ColorSpace - sRGB
Demosaicking (CFA Interpolation)

RAW

image from Kodac dataset

Bayer CFA
Demosaicking (CFA Interpolation)

RAW
linear interpolation green channel

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

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

image from Kodac dataset

Bayer CFA
Demosaicking (CFA Interpolation)

RAW

linear interpolation

image from Kodak dataset
Demosaicking (CFA Interpolation)

image from Kodac dataset
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.
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).
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

1. RGB to Y’CrCb
2. blur
3. Y’CrCb to RGB
Demosaicing – Low-pass Chroma

RGB to Y’CrCb:

\[
\begin{bmatrix}
Y' \\
Cb \\
Cr
\end{bmatrix} = \begin{bmatrix}
65.48 & 128.55 & 24.87 \\
-37.80 & -74.20 & 112.00 \\
112.00 & -93.79 & -18.21
\end{bmatrix} \begin{bmatrix}
R \\
G \\
B
\end{bmatrix} \cdot \frac{257}{65535} + \begin{bmatrix}
16 \\
128 \\
128
\end{bmatrix}
\]

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
   
   $G3 = (G1 + G5 + G2 + G4)/4$
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 \texttt{demosaic()} and part of HW2
Demosaicing- Malvar et al. 2004

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

  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: \( \hat{r}(x,y) = \hat{r}_{\text{lin}}(x,y) + \beta \Delta_G(x,y) \)

- Interpolate R at B pixels: \( \hat{r}(x,y) = \hat{r}_{\text{lin}}(x,y) + \gamma \Delta_B(x,y) \)

- Gain parameters optimized from Kodak dataset: \( \alpha = 1/2, \beta = 5/8, \gamma = 3/4 \)
• 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
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

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

noisy image
(Gaussian iid noise, $\sigma=0.2$)
Denoising – Most General Approach

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

- 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

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

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

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

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

- naïve approach: average in local neighborhood, e.g. using a Gaussian low-pass filter
Denoising – 2. Local, Nonlinear Filtering

\[ i_{\text{denoised}}(x) = \text{median}(W(i_{\text{noisy}},x)) \]

- almost as naïve: use median filter in local neighborhood
Denoising

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

Gaussian

Median

$\sigma=0.1$

$\sigma=0.3$

$\sigma=0.5$

$w=1$

$w=3$

$w=5$
Denoising – 3. Bilateral Filtering

original  Gaussian filtering  bilateral filtering
Denoising – 3. Bilateral Filtering

Why is the output so blurry?
Denoising – 3. Bilateral Filtering

Blur kernel averages across edges
Denoising – 3. Bilateral Filtering

Do not blur if there is an edge! How does it do that?
Denoising – 3. Bilateral Filtering

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

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

- 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!

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

\[ \sigma_t = 0.1 \quad \sigma_t = 0.25 \quad \sigma_t = 8 \]
(Gaussian blur)

\[ \sigma_s = 2 \quad \sigma_s = 6 \quad \sigma_s = 18 \]

input
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)
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!

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

[Buades 2005]
Denoising – 4. Non-local Means

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

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

- very powerful approach: exploit self-similarity in image; average pixels with a similar neighborhood, but don’t need to be close \(\rightarrow\) non-local
Denoising – 4. Non-local Means

noisy  Gaussian filtering  anisotropic filtering

TV  bilateral filtering  NL-means

[Buades 2005]
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

[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
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->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 $\gamma = 2.2$

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

- $a = 0.055$

\[\gamma = 2.2\]

\[C_{sRGB}\]
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)

DCT basis functions

RLE of “same frequency” coefficients
Compression – JPEG (joint photographic expert group)

Original image → Pixel blocks → DCT coefficient blocks → Single coefficient block

http://xiph.org/~xiphmont/demo/daala/demo1.shtml
Compression – JPEG (joint photographic expert group)

Original pixel data

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

DCT coefficient data

<table>
<thead>
<tr>
<th>700</th>
<th>200</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>-150</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>110</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

http://xiph.org/~xiphmont/demo/daala/demo1.shtml
Compression – JPEG (joint photographic expert group)

Single quantized block → Quantized coefficient blocks → Reconstructed pixel blocks → Reconstructed image

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)
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

Demosaicking
• Malvar, He, Cutler, “High-quality Linear Interpolation for Demosaicking of Bayer-patterned Color Images”, Proc. ICASSP 2004

Plenoptic function

Other, potentially interesting work
• Kodac dataset (especially good and standard for demosaicking): http://r0k.us/graphics/kodak/