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
Image Formation

- high-dimensional integration over angle, wavelength, time

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

plenoptic function

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

field **sequential**

multiple sensors

vertically stacked

Prokudin-Gorsky

[Images and diagrams representing various color capture methods]
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

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...)
Three-CCD Camera

beam splitter prism
Stacked Sensor

Foveon X3

Sigma SD9
Other Wavelengths

- **OmniVision**: RGB + near IR!

### Product Specifications

<table>
<thead>
<tr>
<th>Specification</th>
<th>OV4682-G04A</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Part Number</strong></td>
<td><strong>OV4682-G04A</strong></td>
</tr>
<tr>
<td><strong>Resolution</strong></td>
<td>4MP</td>
</tr>
<tr>
<td><strong>Chroma</strong></td>
<td>Color</td>
</tr>
<tr>
<td><strong>Analog / Digital</strong></td>
<td>Digital</td>
</tr>
<tr>
<td><strong>Power Requirement</strong></td>
<td>Active: 163 mA (261 mW)</td>
</tr>
<tr>
<td></td>
<td>Standby: 1 mA</td>
</tr>
<tr>
<td></td>
<td>XSHUTDOWN: &lt;10 μA</td>
</tr>
<tr>
<td><strong>Temperature Range</strong></td>
<td>Operating: -30°C to +85°C</td>
</tr>
<tr>
<td></td>
<td>Stable image: 0°C to +60°C</td>
</tr>
<tr>
<td><strong>Output Format</strong></td>
<td>10-bit RAW data</td>
</tr>
<tr>
<td><strong>Optical Format</strong></td>
<td>1/3&quot;</td>
</tr>
<tr>
<td><strong>Frame Rate</strong></td>
<td>Full @ 90 fps</td>
</tr>
<tr>
<td></td>
<td>1080p @ 120 fps</td>
</tr>
<tr>
<td></td>
<td>672x380: 330 fps</td>
</tr>
<tr>
<td></td>
<td>720p @ 180 fps</td>
</tr>
<tr>
<td><strong>Pixel Size</strong></td>
<td>2.0 μm</td>
</tr>
<tr>
<td><strong>Image Area</strong></td>
<td>5440 x 3072 μm</td>
</tr>
<tr>
<td><strong>Package</strong></td>
<td>COB</td>
</tr>
<tr>
<td><strong>Package Dimensions</strong></td>
<td>6600 x 5800 μm</td>
</tr>
<tr>
<td><strong>Product Brief</strong></td>
<td><img src="image" alt="Product Brief" /></td>
</tr>
</tbody>
</table>
Other Wavelengths

- thermal IR
- often use Germanium optics (transparent IR)
- sensors don’t use silicon: indium, mercury, lead, etc.
Review: Photons to RAW Image

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

Sensor

Amplifier (gain, ISO)

ADC (quantization)

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

RAW image → demosaicking → denoising → gamut mapping → compression → JPEG image

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

Exif - night nikon.jpg

Exif Make - NIKON CORPORATION
Model - NIKON D70s
Orientation - Top Left
XResolution - 300
YResolution - 300
ResolutionUnit - Inch
SOFTWARE - Ver 1.00
Date/Time - 2005:09:01 12:16:43
YCbCrPositioning - Co-Sited
ExifOffset - 216

ExposureTime - 10 seconds
FNumber - 13.00
ExposureProgram - Manual control
ExifVersion - 0221
Date/TimeOriginal - 2005:09:01 12:16:43
ComponentsConfiguration - YCbCr
CompressedBitsPerPixel - 1 (bits/pixel)

ExposureBiasValue - 0.50
MaxApertureValue - 3.48
MeteringMode - Center weighted average
LightSource - Auto

Flash - Not fired
FocalLength - 18.00 mm
UserComment - (c) Gordon Wetzstein

Thumbnail: -

Compression - 6 (JPG)
XResolution - 300
YResolution - 300
ResolutionUnit - Inch
JpegIFOffset - 29368
JpegIIFByteCount - 8393
YCbCrPositioning - Co-Sited
Demosaicking (CFA Interpolation)

RAW

image from Kodak dataset

Bayer CFA
Demosaicking (CFA Interpolation)

RAW

linear interpolation green channel

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

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

image from Kodak dataset

Bayer CFA
Demosaicking (CFA Interpolation)

RAW

linear interpolation

image from Kodac dataset
Demosaicking (CFA Interpolation)

original

RAW

demosaicked

image from Kodak 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

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

**RGB to Y'CrCb:**

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

**Y'CrCb to RGB:**

\[
\begin{bmatrix}
    R \\
    G \\
    B
\end{bmatrix} = M^{-1}
\begin{bmatrix}
    Y' \\
    Cb \\
    Cr
\end{bmatrix}
- \begin{bmatrix}
    16 \\
    128
\end{bmatrix}
\times \frac{65535}{257}
\]

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 `demosaic()` and part of HW2
Demosoping - Malvar et al. 2004

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

\[
\Delta_R(x,y) = r(x,y) - \frac{1}{4} \sum_{(m,n)} r(x+m,y+n)
\]
\[
(m,n) = \{(0,-2),(0,2),(-2,0),(2,0)\}
\]

• Red gradient:

• 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: \( \alpha = 1/2, \beta = 5/8, \gamma = 3/4 \)
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

- 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

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

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 \textit{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, $\sigma=0.2$)
Denoising – 3. Bilateral Filtering

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

[Box 1]

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

• more clever: average in \textit{local} neighborhood, but only average similar intensities!
Denoising – Gaussian Filter

\[ J(x) = \sum_{\xi} f(x, \xi) \cdot I(\xi) \]

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

output \quad \longrightarrow \quad input
Denoising – Bilateral Filter

J: filtered output (is not blurred)

f: Gaussian convolution kernel

I: noisy image (step function & noise)

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

difference in intensity as scale!
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

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

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

• 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

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

\[
\hat{i}(x) = \sum_{\text{all pixels } x'} \hat{i}(x') \cdot w(x, x')
\]

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

noisy  Gaussian filtering  anisotropic filtering

TV  bilateral filtering  NL-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 $\gamma=2.2$

perceptually →
linear spacing!
Gamma Correction in sRGB

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

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

linear

$\gamma=2.2$

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)

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

<table>
<thead>
<tr>
<th></th>
<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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></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>
<td></td>
</tr>
</tbody>
</table>

DCT coefficient data

<table>
<thead>
<tr>
<th></th>
<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>
<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>
<td>0</td>
</tr>
<tr>
<td></td>
<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></td>
<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></td>
<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></td>
<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></td>
<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)

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


Plenoptic function


Other, potentially interesting work

- Kodac dataset (especially good and standard for demosaicking): http://r0k.us/graphics/kodak/