CSE 559A: Computer Vision

Fall 2017: T-R: 11:30-1pm @ Lopata 101

Instructor: Ayan Chakrabarti (ayan@wustl.edu).
Staff: Abby Stylianou (abby@wustl.edu), Jarett Gross (jarett@wustl.edu)

http://www.cse.wustl.edu/~ayan/courses/cse559a/

Aug 31, 2017
Everyone needs to fill out survey and join Piazza.
We strongly encourage you to use Piazza over e-mail for questions (mark only to instructors if you wish).
Also read Piazza posts!
Doodle poll to fix office hours was posted Tue. Vote by midnight tonight.
We'll take a snapshot then, and decide office hours and recitation time.
Reminder: slides will be posted after class. (Tuesday slides already online)
Additional Python and Math resources posted on project website. (scroll to end)
LaTeX resources to be posted soon.
Everyone said they have a laptop. If you are concerned about compute power for problem sets, e-mail me.

Everyone on waitlist as of Tue morning was enrolled. Pretty much at limit now.
$I[n_x, n_y] = \int_{t=0}^{T} \left[ \int E(x, y, t) \ p(x - \bar{x}_{n_x}, y - \bar{y}_{n_y}) \ q \ dx \ dy \right] \ dt$

- $n_x, n_y$ are integers indexing pixels in image array.
- $I[n_x, n_y]$ is recorded pixel intensity.
- $(x, y)$ is spatial location
- $(\bar{x}_{n_x}, \bar{y}_{n_y})$ is the "center" spatial location of the pixel / sensor element at $[n_x, n_y]$. 
\[ I[n_x, n_y] = \int_{t=0}^{T} \left[ \int E(x, y, t) \, p(x - \bar{x}_{n_x}, y - \bar{y}_{n_y}) \, q \, dx \, dy \right] \, dt \]

- \( n_x, n_y \) are integers indexing pixels in image array.
- \( I[n_x, n_y] \) is recorded pixel intensity.
- \((x, y)\) is spatial location
- \((\bar{x}_{n_x}, \bar{y}_{n_y})\) is the "center" spatial location of the pixel / sensor element at \([n_x, n_y]\).

\( E(x, y, t) \) is light "power" per unit area incident at location \((x, y)\) on the sensor plane at time \(t\)

\( p(x, y) \) is spatial sensitivity of the sensor (might be lower near boundaries, etc.)

\( q \) is quantum efficiency of the sensor (photons/energy to charge/voltage)

\( T \) is the duration of the exposure interval.
\[ I[n_x, n_y] = \int_{t=0}^{T} \left[ \int E(x, y, t) \ p(x - \bar{x}_{n_x}, y - \bar{y}_{n_y}) \ q \ dx \ dy \right] \ dt \]

\( I[n_x, n_y] \) is the *ideal unquantized* pixel intensity
SENSOR

\[ I^0[n_x, n_y] = \int_{t=0}^{T} \left[ \int E(x, y, t) \ p(x - \bar{x}_{n_x}, y - \bar{y}_{n_y}) \ q \ dx \ dy \right] dt \]

\[ I \leftarrow I^0 \]

SHOT NOISE

- Caused by uncertainty in photon arrival
- Actual number of photons \( K \) is a discrete random variable with Poisson distribution
  \[ P(K = k) = \frac{\lambda^k e^{-\lambda}}{k!} \]
- \( \lambda \) is the "expected" number of photons. In our case, \( \propto I^0 \)
- Property of Poisson distribution: Mean and Variance both equal to \( \lambda \)
- Often, shot noise is modeled with additive Gaussian noise with signal dependent variance:
  \[ I \leftarrow I^0 + \sqrt{I^0} \ \epsilon_1 \]

where \( \epsilon \sim \mathcal{N}(0, 1) \) (Gaussian random noise with mean 0, variance 1).
Signal amplified by gain $g$ before digitization. Based on ISO (higher $g$ for higher ISO).

Some signal-independent Gaussian noise added before and after amplification.

$$ I \leftarrow I^0 + \sqrt{I^0} \epsilon_1 $$

where $\sigma_{2a}$ and $\sigma_{2b}$ are parameters (lower for high quality sensors), and $\epsilon_1, \epsilon_{2a}, \epsilon_{2b}$ are $\mathcal{N}(0, 1)$ noise variables, all independent.

$$ I \leftarrow \underbrace{gI^0}_{\text{Amplified Signal}} + \underbrace{g\sqrt{I^0} \epsilon_1}_{\text{Amplified Shot Noise}} + \underbrace{\sqrt{(g^2\sigma_{2a}^2 + \sigma_{2b}^2)} \epsilon_2}_{\text{Amplified and un-amplified additive noise}} $$
\[ I^0[n_x, n_y] = \int_{t=0}^{T} \left[ \int E(x, y, t) \ p(x - \bar{x}_{n_x}, y - \bar{y}_{n_y}) \ q \ dx \ dy \right] dt \]

\[ I \leftarrow gI^0 + g\sqrt{I^0} \ \epsilon_1 + \sqrt{\left(g^2 \sigma^2_{2a} + \sigma^2_{2b}\right)} \ \epsilon_2 \]

**DIGITIZATION**

- Final step is rounding and clipping (by an analog to digital converter)

\[ I \leftarrow \text{Round}\left(gI^0 + g\sqrt{I^0} \ \epsilon_1 + \sqrt{\left(g^2 \sigma^2_{2a} + \sigma^2_{2b}\right)} \ \epsilon_2\right) \]

\[ I = \min\left(I_{\text{max}}, \ \text{Round}\left(gI^0 + g\sqrt{I^0} \ \epsilon_1 + \sqrt{\left(g^2 \sigma^2_{2a} + \sigma^2_{2b}\right)} \ \epsilon_2\right)\right) \]
WHY STUDY THIS?

- To understand the degradation process of noise (if we want to denoise/recover $I^0$ from $I$).
- To prevent degradation during capture, because we control exposure time $T$ and ISO/gain $g$.
- To understand the different trade-offs for loss of information from noise, rounding, and clipping.
Ignoring noise, what is the optimal $g$ for a given $I^0[n_x, n_y]$?

- Keep $g$ low so that most values of $gI^0[n_x, n_y]$ are below $I_{\text{max}}$.
- But if $g$ is too low, a lot of the variation will get rounded to the same value.
Sensor

\[ I^0[n_x, n_y] = \int_{t=0}^{T} \left[ \int E(x, y, t) p(x - \bar{x}_{n_x}, y - \bar{y}_{n_y}) q \, dx \, dy \right] \, dt \]

\[ I = \min \left( I_{\text{max}}, \ \text{Round}(gI^0 + g\sqrt{I^0} \ e_1 + \sqrt{(g^2 \sigma^2_{2a} + \sigma^2_{2b})} \ e_2) \right) \]

Light vs Amplification

Say we have chosen the optimal target values for the product \( gI^0 \). Is it better:

- To have a higher \( g \) and lower magnitude \( I^0 \)
- To have a lower \( g \) and higher magnitude \( I^0 \)
- Depends, based on \( \sigma_{2a}, \sigma_{2b} \)

Additional Reading (if interested):
So how do we increase $I^0$?

- Better sensors (higher $q$)
- Larger sensor elements: $p(\cdot, \cdot) > 0$ over a larger area.

But we've gone the other way: cameras stuff more 'megapixels' in smaller form factors.
Increase exposure time $T$?

- If scene is static and camera is stationary:
  - $E(x, y, t)$ doesn't change with $t \Rightarrow I_0 \propto T$
- If scene is moving ...
Increase $E(x, y, t)$ itself. How?

- Take pictures outdoors, or under brighter lights.
- Don't use a pinhole camera!
Small Pinhole: Focused beam, very little light.
Large Pinhole: More light, larger 'circle of confusion'
Thin lens model (approximation)

More light & localization of rays to sensor plane but only for objects at the focusing distance D.
Again, point projects to 'circle of confusion'
Can still tradeoff amount of light with amount of blur in out of focus regions (move up or down "f-stops")
Better trade-off than pinhole camera
TRADEOFFS

Photographers think about these tradeoffs every time they take a shot

- Dynamic range and what part of the image should be well exposed (rounding and clipping)

- Choosing between:
  - ISO i.e. Gain & noise
  - Exposure Time & motion blur
  - F-stop i.e. aperture size & defocus blur
We left out at an important term in this equation: wavelength
COLOR

\[ I^0[n_x, n_y] = \int_{t=0}^{T} \left[ \int E(\lambda, x, y, t) \ p(x - \bar{x}_{n_x}, y - \bar{y}_{n_y}) \ q(\lambda) \ d\lambda \ dx \ dy \right] dt \]

- Light carries different amounts of power in different wavelengths
- \( E(\lambda, x, y, t) \) now refers to power per unit area per unit wavelength
  - In wavelength \( \lambda \), incident at \( (x, y) \) at time \( t \)
  - Both spectral and spatial density function
- \( q(\lambda) \): Quantum efficiency also a function of wavelength
  - CMOS/CCD sensors are sensitive (have high \( q \)) across most of the visible spectrum
  - Actually extend to longer than visible wavelengths (near infra red)
  - Why cameras have NIR filter, to prevent NIR radiation from being 'superimposed' on the image

Q: But this measures 'total' power in all wavelengths. How do we measure color?

Ans: By putting a color filter in front of each sensor element.
COLOR

\[ I^0[n_x, n_y, c] = \int_{t=0}^{T} \left[ \int E(\lambda, x, y, t) \ p(x - \bar{x}_{n_x}, y - \bar{y}_{n_y}) \ \Pi_c(\lambda) \ q(\lambda) \ d\lambda \ dx \ dy \right] \ dt \]

for \( c \in \{R, G, B\} \)

- \( \Pi_c \) is the transmittance of a color filter for color channel \( c \)
- E.g., \( \Pi_R \) will transmit power in (be high for) wavelengths in the red part of the visible spectrum and attenuate power in (be low for) other wavelengths.
- Sometimes also called "color matching functions"

![Graph showing color filters for different wavelengths](actual_graph.png)
But we can only put one filter in front of each sensor element / pixel location.

So color cameras "multiplex" color measurements: they measure a different color channel at each location.

Usually in an alternating pattern called the Bayer pattern:

- Note: a disadvantage is that color filters block light, so measured $I^0$ values are lower.
- That's why black and white / grayscale cameras are "faster" than color cameras.
Final steps in camera processing pipelines (except for some DSLR cameras shooting in RAW):

- **Filter Colors to Standard RGB:**
  - Cameras often use their own color filters \( \Pi_c \).
  - Apply a linear transformation to map those measurements to standard RGB.
- **White-balance:** scale color channels to remove color cast from a non-neutral illuminant.
- **Tone-mapping:**
  - The simplest form is "gamma correction" (approximately raising each intensity to the power \((1/2.2)\))
  - Done based on standard developed for what old display devices expected
  - Fits the full set of measurable colors into the gamut that can be displayed / printed
  - Modern cameras often do more advanced processing (to make colors look vibrant)
- **Compression**

And that's how you get your PNG / JPEG images!

**Optional Additional Reading:** Szeliski Sec 2.3
OTHER EFFECTS

Other effects we did not talk about. E.g.,

- Real Lenses not thin lenses and have distortions:
  - Radial Distortion
  - Vignetting
  - Chromatic Aberration

- Rolling Shutter: No explicit shutter but when pixels reset electronically (along scanlines)
Place array of micro-lenses in front of sensor
Different pixels observe the scene "from a different angle"
Let you estimate depth, shift viewpoint post-capture,
refocus post-capture
16 Different Camera Units
Fuse images to get lower noise, higher dynamic range, synthetically control focal distance and aperture size post-capture

No lens: lets the camera be much smaller, sensor can be placed on curved surfaces. Put a mask in the aperture: causes 'patterns' of confusion instead of circles of confusion. Create focused image computationally.
NON-STANDARD CAMERAS

Coded Aperture Photography

NON-STANDARD CAMERAS

Coded Aperture Photography

NON-STANDARD CAMERAS

Coded Aperture Photography

Unbounded High Dynamic Range Photography using a Modulo Camera

Hang Zhao\textsuperscript{1} Boxin Shi\textsuperscript{1,3} Christy Fernandez-Cull\textsuperscript{2} Sai-Kit Yeung\textsuperscript{2} Ramesh Raskar\textsuperscript{1}

\textsuperscript{1} MIT Media Lab \textsuperscript{2}MIT Lincoln Lab \textsuperscript{3}SUTD

ICCP 2015, Houston, TX \[Best Paper runner-up\]
**IMAGES**

- Exist as 2-D (grayscale) or 3-D (color image) arrays

![2-D grayscale image](image1.png) ![3-D color image](image2.png)  
W x H \quad W x H x 3  

- Precision: uint8 (0-255), uint16(0-65535), Floating point
  - We will often treat them as (positive) real numbers.

- Conventions: $I[n_x, n_y, c], I[n_x, n_y] \in \mathbb{R}$ or $\in \mathbb{R}^3$, $I[n]$: where $n \in \mathbb{Z}^2$

- How do you process / manipulate these arrays?
POINT-WISE OPERATIONS

- $Y[n] = h(X[n])$
- $Y[n] = h(X_1[n], X_2[n], \ldots)$
- $Y[n] = h_n(X[n])$ - Might vary based on location.
- $h(\cdot)$ itself might be based on 'global statistics'
POINT-WISE OPERATIONS

$X[n] \in \mathbb{R}^3$

$Y[n] = \begin{bmatrix} 0.7 & 1.05 & 0.7 \end{bmatrix} X[n]$

Linear Color Transforms
POINT-WISE OPERATIONS

\[ X[n] \in \mathbb{R}^3 \]

\[ Y[n] = \begin{bmatrix} 0.7 & 0.7 & \vdots \end{bmatrix} \begin{bmatrix} X[n] \end{bmatrix} \]

Linear Color Transforms
POINT-WISE OPERATIONS

$X[n] \in \mathbb{R}^3$

$Y[n] = \begin{bmatrix} 1.05 & 0.7 \\ 0.7 & 0.7 \end{bmatrix} X[n]$

Linear Color Transforms
POINT-WISE OPERATIONS

\[ X[n] \in \mathbb{R}^3 \]

\[ Y[n] = \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} X[n] \]

Linear Color Transforms
POINT-WISE OPERATIONS

\[ X[n] \in \mathbb{R}^3 \]

\[ Y[n] = 0.9 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} X[n] + 0.1 \]

Linear Color Transforms
POINT-WISE OPERATIONS

$X[n] \in \mathbb{R}^3$

$Y[n] = 0.5 \begin{bmatrix} 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \\ 1/3 & 1/3 & 1/3 \end{bmatrix} X[n] + 0.5 X[n]$

Linear Color Transforms
POINT-WISE OPERATIONS

$X[n] \in \mathbb{R}^3$

\[
Y[n] = -1.0 \begin{bmatrix}
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3 \\
1/3 & 1/3 & 1/3
\end{bmatrix} X[n] + 2.0 X[n]
\]

Linear Color Transforms
POINT-WISE OPERATIONS

\[ X[n] \in \mathbb{R} \]

\[ Y[n] = X[n]^{0.5} \]

Non-linear Transforms
POINT-WISE OPERATIONS

\[ X[n] \in \mathbb{R} \]

\[ Y[n] = X[n]^{2.0} \]

Non-linear Transforms

Tone-maps / Change contrast
POINT-WISE OPERATIONS
POINT-WISE OPERATIONS
Can be arbitrary: but usually monotonic

Uneven distribution of intensities (many too dark or too bright)

Find a monotonic function $h(\cdot)$ such that the intensities of $h(X[n])$ are distributed uniformly in the range $[0, 255]$. 

Histogram Equalization
Consider the Cumulative Distribution (think of intensity as a continuous r.v.)

Use the CDF as the tone map: \[ h(x) = P(X[n] < x) \times 255.0 \]
POINT-WISE OPERATIONS
POINT-WISE OPERATIONS

Image Matting (combine multiple images with alpha matte)

From Szeliski 3.1
Linear operation on spatial neighborhoods

\[ Y[n] = \sum_{n'} X[n - n'] k[n'] \]

Kernel

\[ X[n] \]

\[ Y[n] \]

\[ k[n] \]
CONVOLUTION
CONVOLUTION