Homework 1

CS6550 - Computer Vision

Arc Chang, Hwann-Tzong Chen

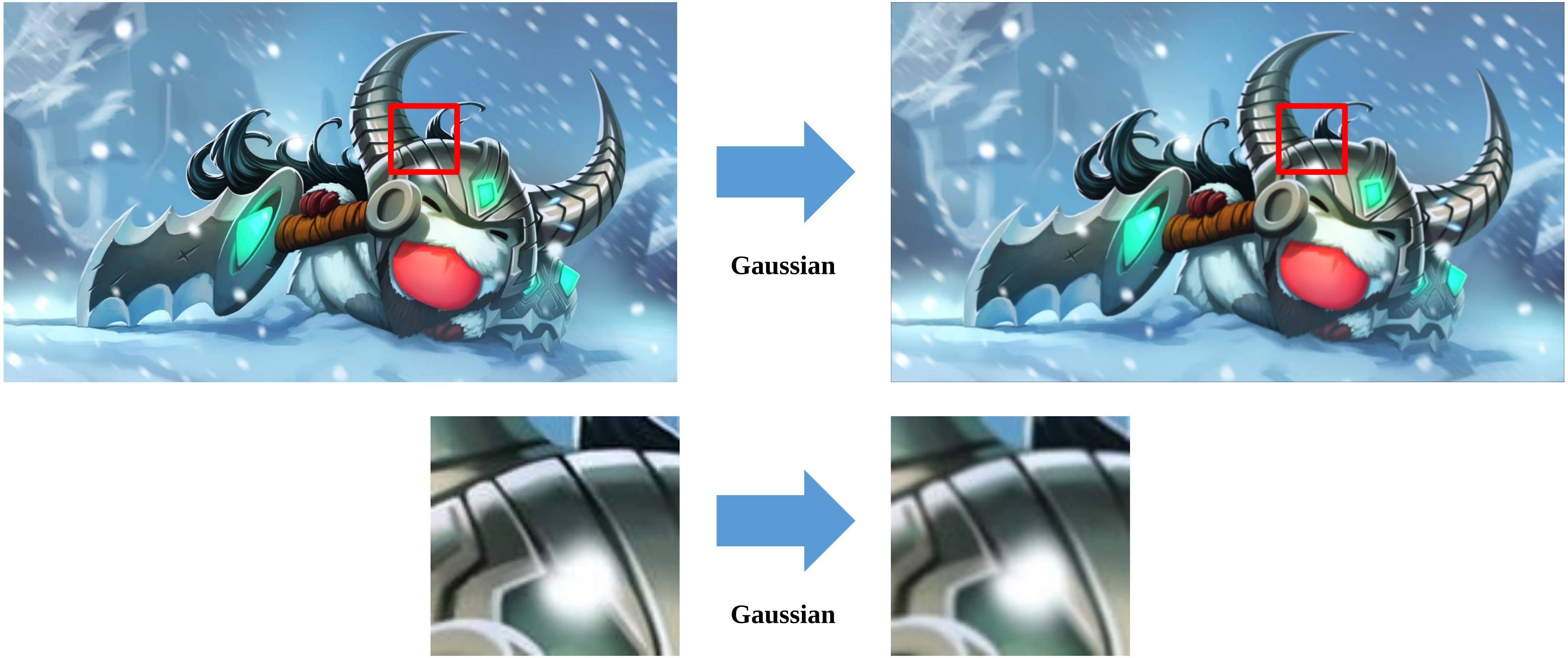
Part 1. Harris Corner Detection

Algorithm

- **Slides p.79**

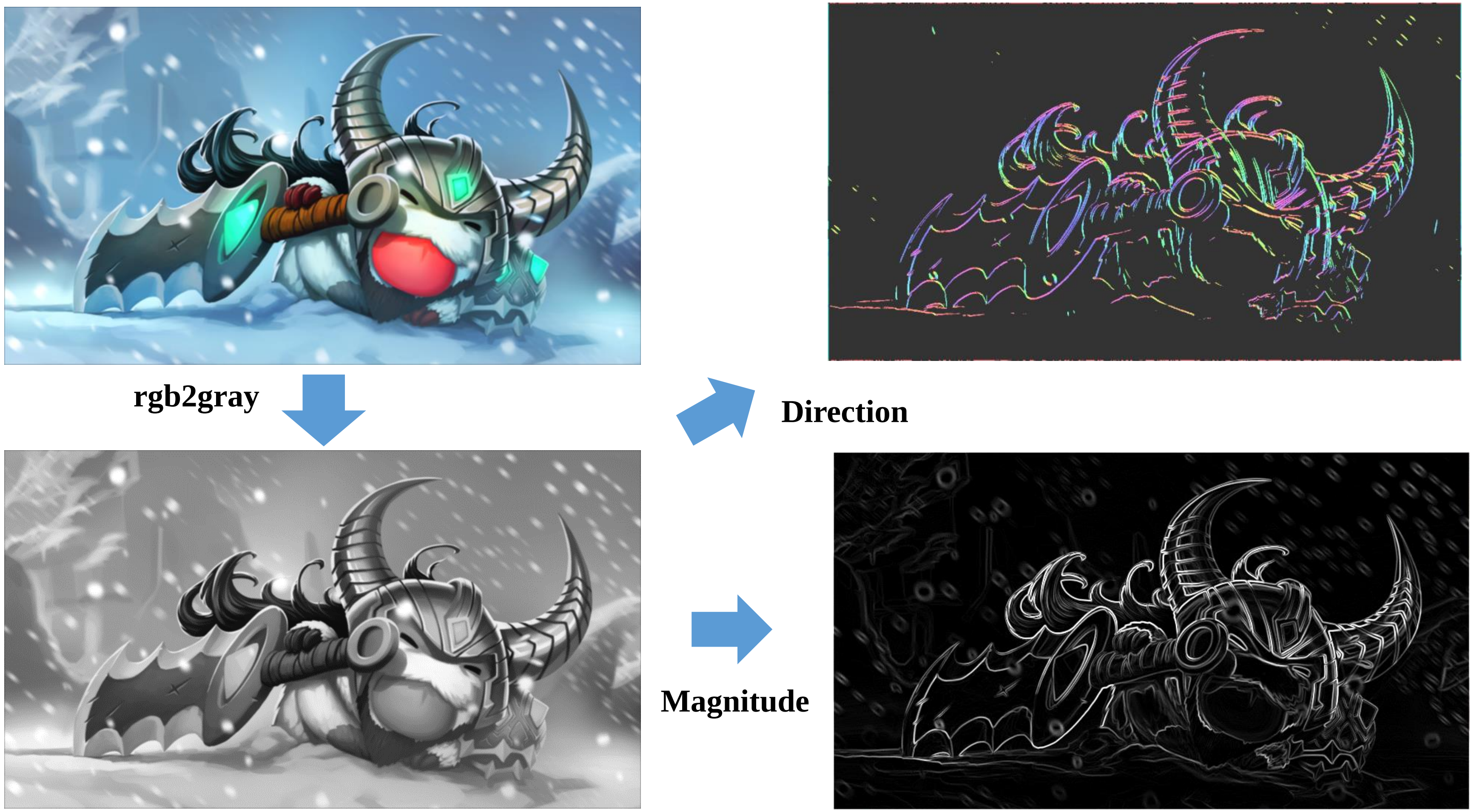
1. Filter the image with a Gaussian.
2. Estimate intensity gradient in two perpendicular directions for each pixel, $\frac{\partial f(x,y)}{\partial x}$, $\frac{\partial f(x,y)}{\partial y}$. This is performed by twice using a 1D convolution with the kernel approximating the derivative.
3. For each pixel and a given neighborhood window:
   - Calculate the local structure matrix A.
   - Evaluate the response function $R(A)$.
4. Choose the best candidates for corners by selecting a threshold on the response function $R(A)$ and perform non-maximal suppression.

Gaussian Smooth

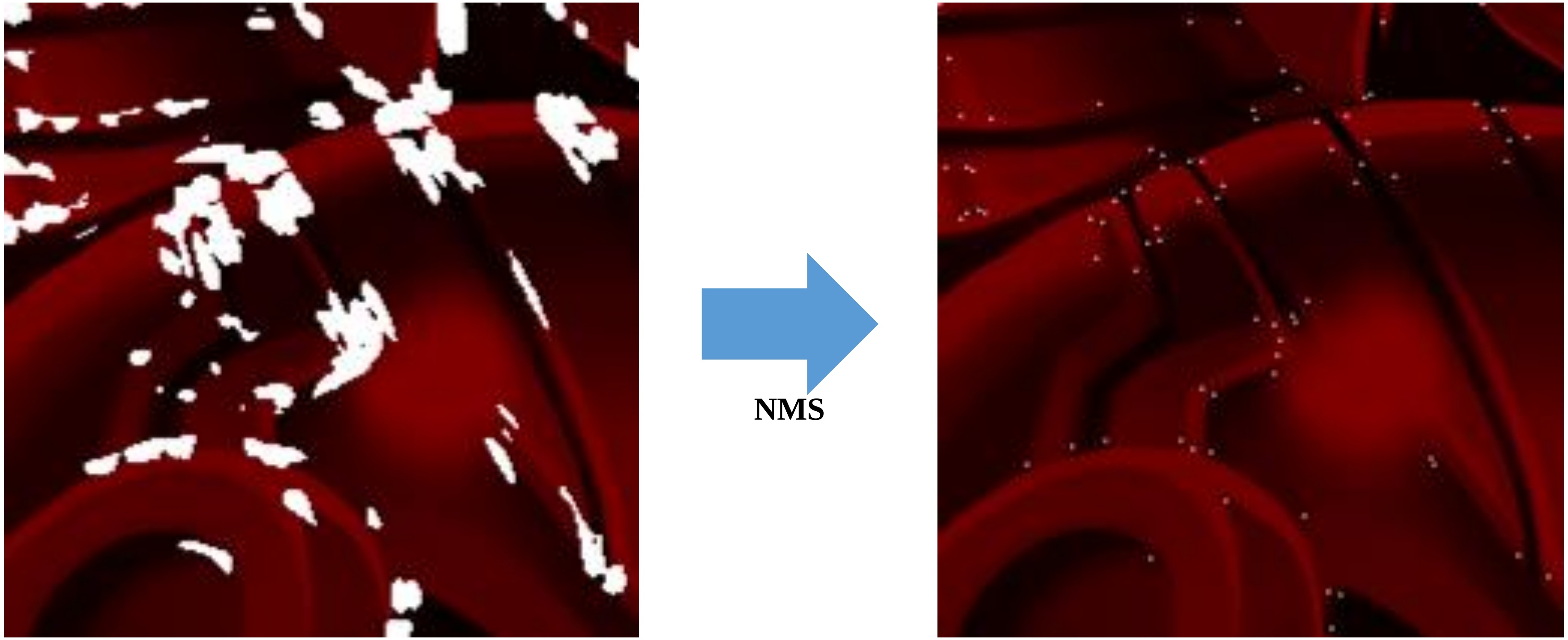


Compute Gradients

https://matplotlib.org/examples/color/colormaps_reference.html



Non-maximal Suppression



Corner Response Function

Results

Origin

Rotate

Scale

Reference

- **Gaussian smooth**

  http://www.librow.com/articles/article-9

- **Sobel edge detection**

  https://medium.com/datadriveninvestor/understanding-edge-detection-sobel-operator-2aada303b900

- **Compute structure tensor**

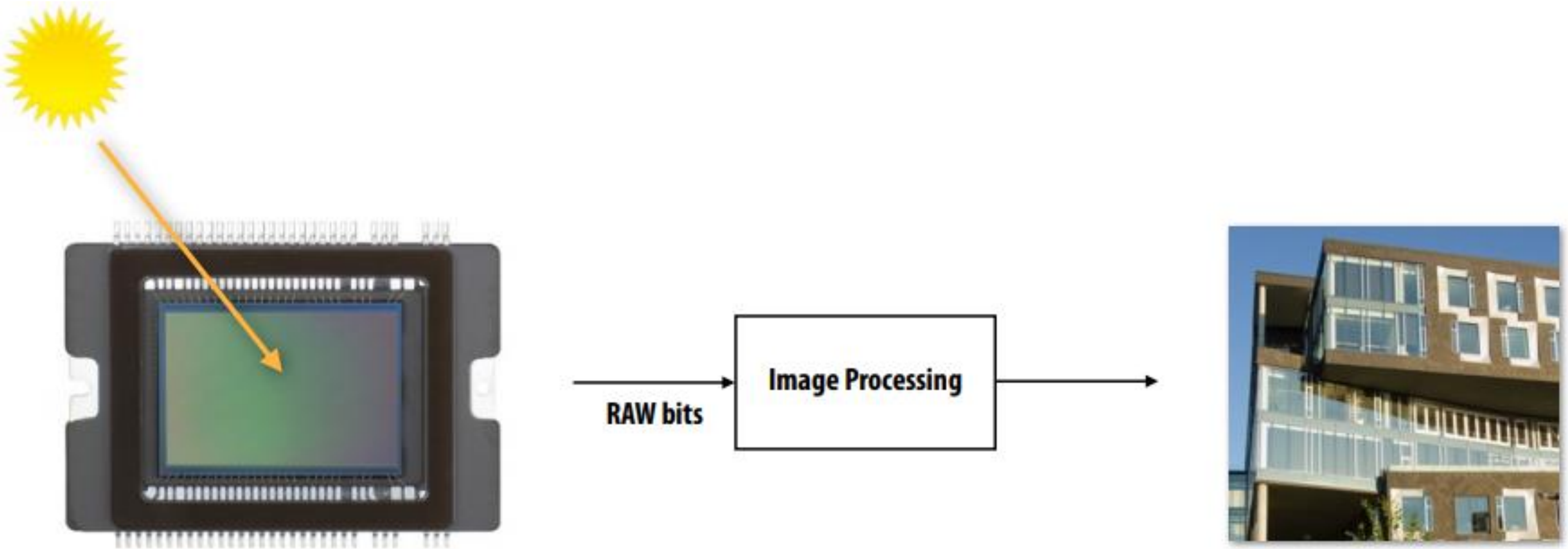
  https://docs.opencv.org/3.4/dc/d0d/tutorial_py_features_harris.html

- **Non-maximal suppression**

  http://www.ipol.im/pub/art/2018/229/article_lr.pdf **(p.313, Step 5.)**

Part 2. Image Sensing Pipeline

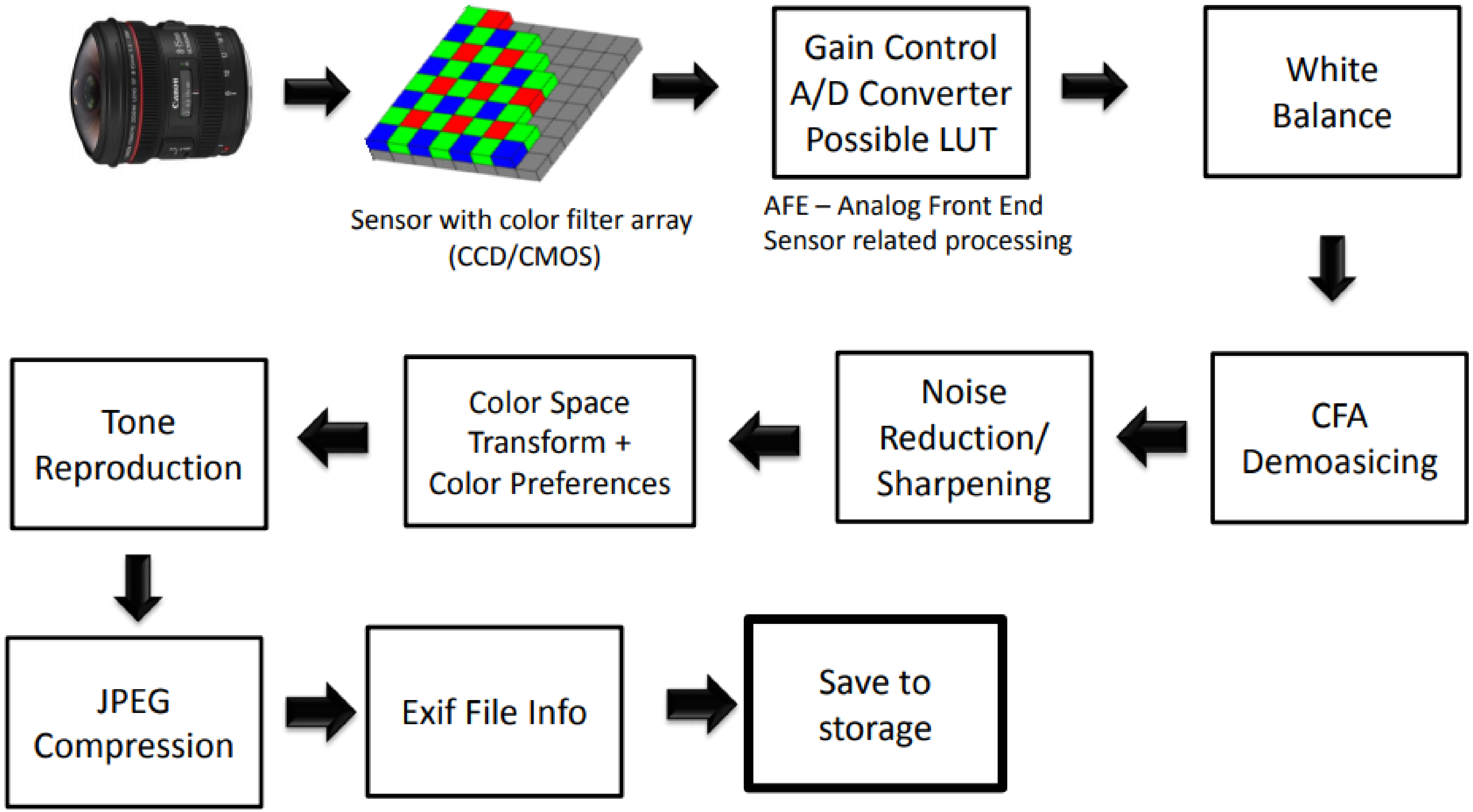
Camera



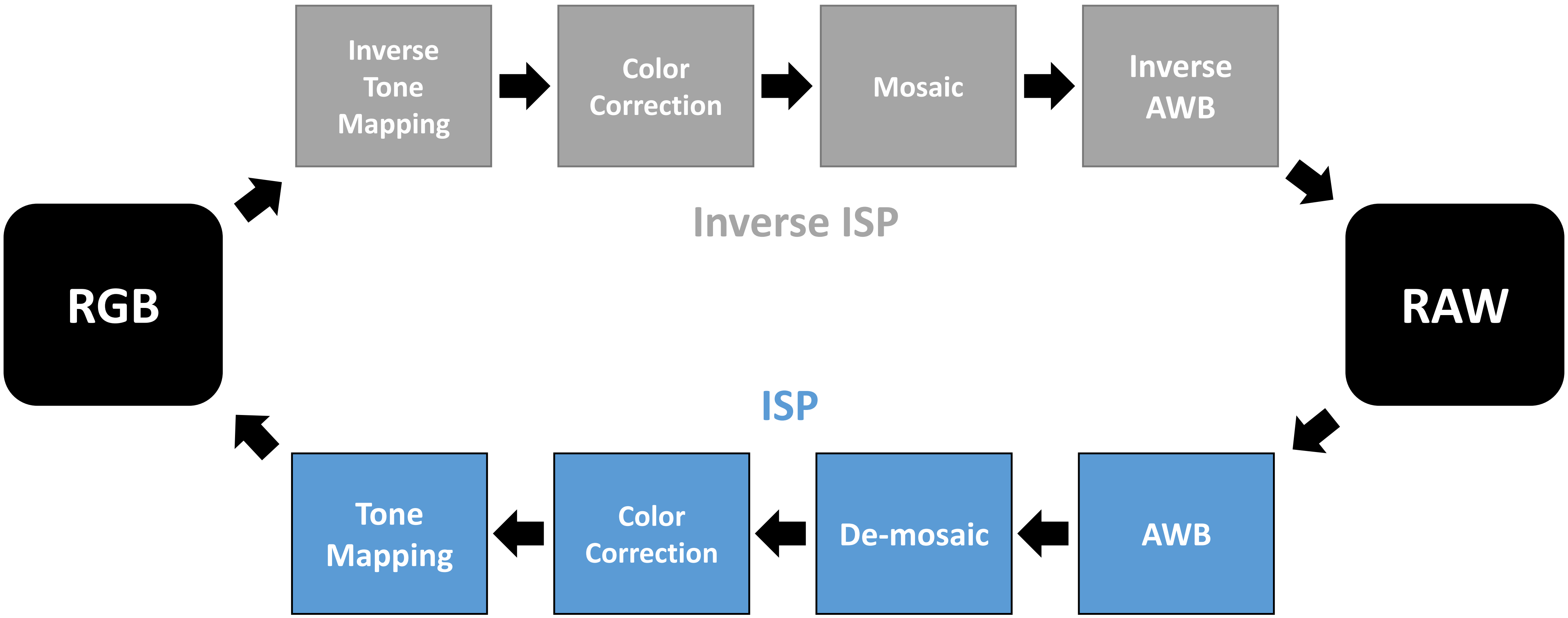
Canon 14 MP CMOS Sensor
(14 bits per pixel)

Final Image Representation
(e.g., JPG file)

Image Sensing Pipeline



In this homework ......



For Example (1/2)

RAW

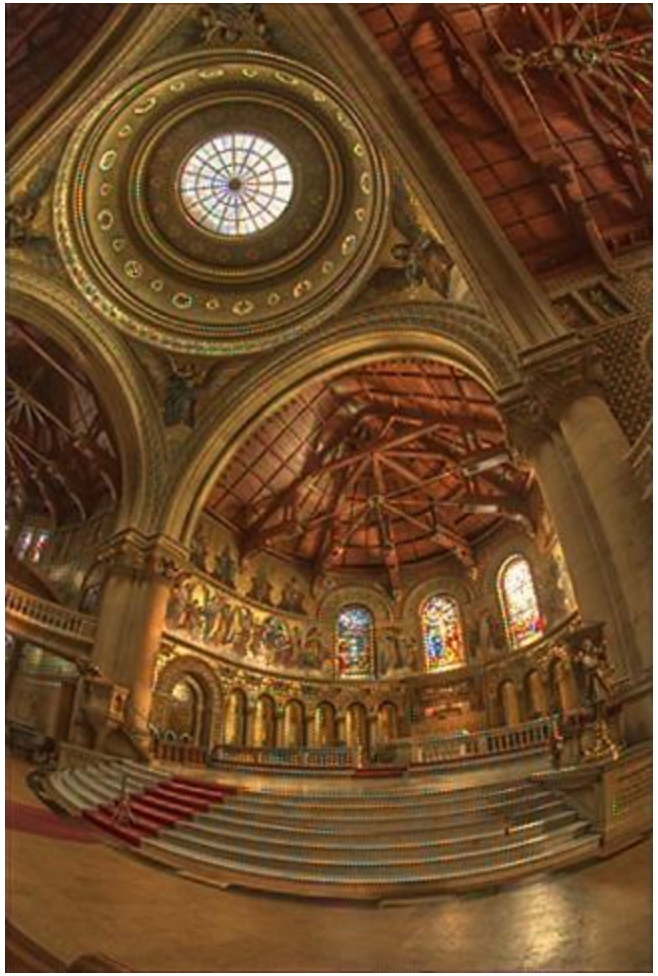
AWB

Demosaic

For Example (2/2)

Color Correction

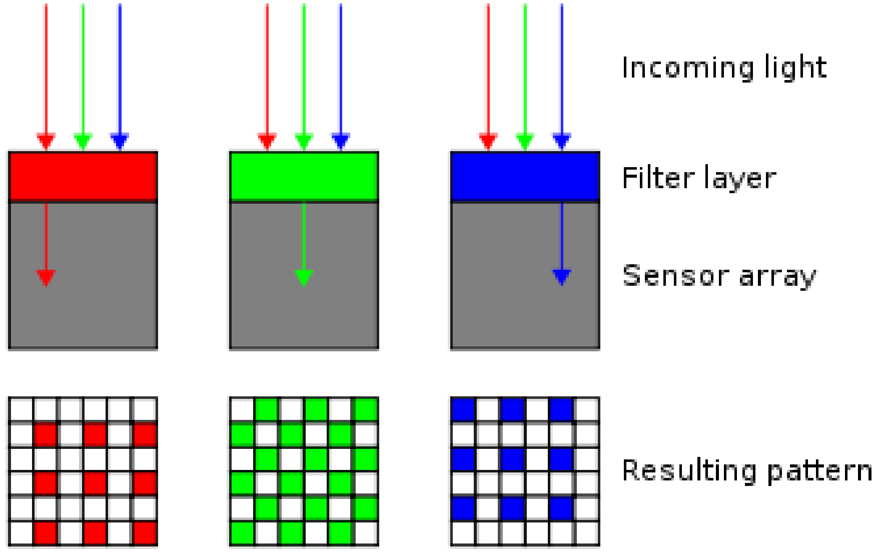
CIE XYZ to RGB

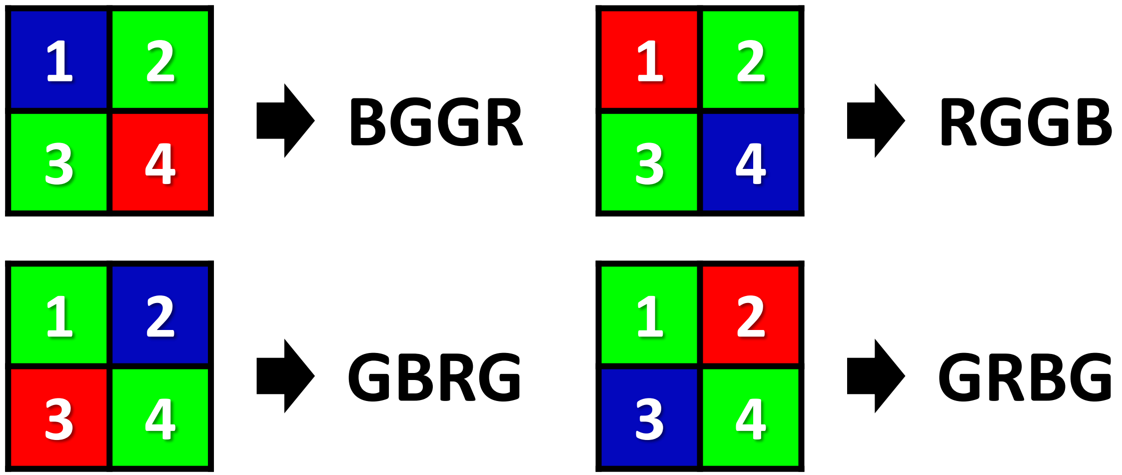


Tone Mapping

Raw Data (rawRGB)

- **Bayer pattern (CFA, Color Filter Arrays)**
  - Green photo-sensors: luminance-sensitive elements
  - Red and blue photo-sensors: chrominance-sensitive elements
  - Twice as many green elements as red or blue to mimic the physiology of the human eye
  - Four patterns: **BGGR, RGGB, GBRG, GRBG**





AWB (Auto White Balance)

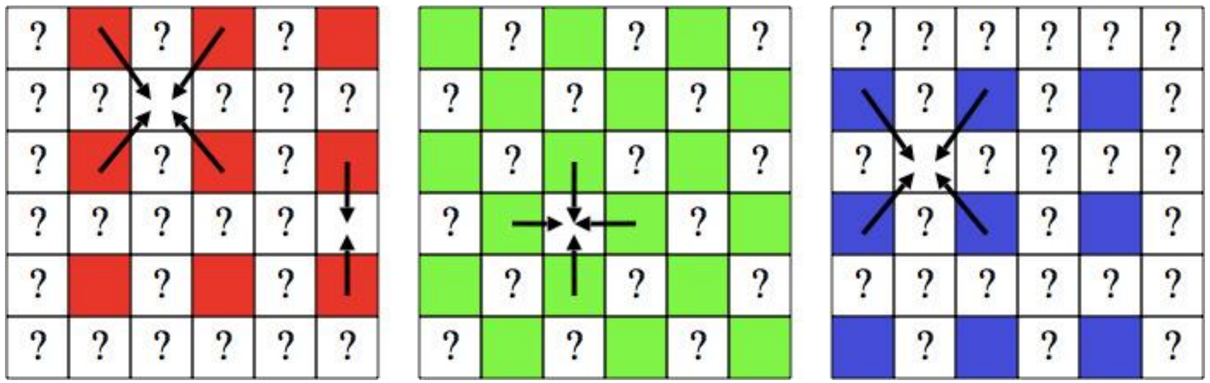
- **WB: mimics chromatic adaptation of the eye**
  - Can be manual settings
  - Stored in metadata
  - Otherwise, we can perform AWB

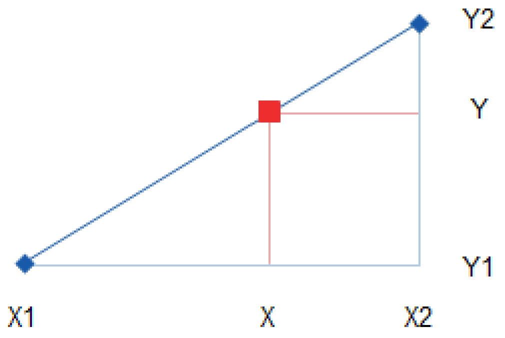
WB SETTINGS	COLOR TEMPERATURE	LIGHT SOURCES
	10000 - 15000 K	Clear Blue Sky
	6500 - 8000 K	Cloudy Sky / Shade
	6000 - 7000 K	Noon Sunlight
	5500 - 6500 K	Average Daylight
	5000 - 5500 K	Electronic Flash
	4000 - 5000 K	Fluorescent Light
	3000 - 4000 K	Early AM / Late PM
	2500 - 3000 K	Domestic Lightning
	1000 - 2000 K	Candle Flame

- **AWB: attempts to make what is assumed to be white map to “pure white”**
  - Two classical methods: gray world algorithm and white patch algorithm
  - In this homework, you only need to implement easiest AWB method
    - See more details in **white_balance.py**

De-mosaic (1/2)

- **Easiest method – Linear Interpolation**





$$\frac{(X - X1)}{(X2 - X1)} = \frac{(Y - Y1)}{(Y2 - Y1)}$$

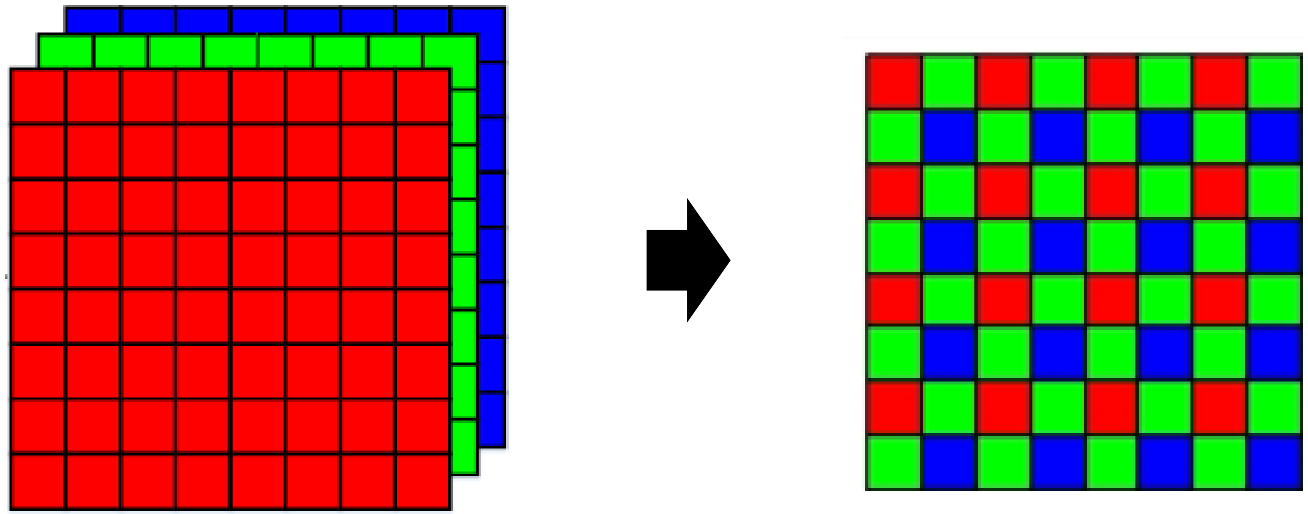
$$Y = Y1 + (X - X1)\frac{(Y2 - Y1)}{(X2 - X1)}$$

De-mosaic (2/2)

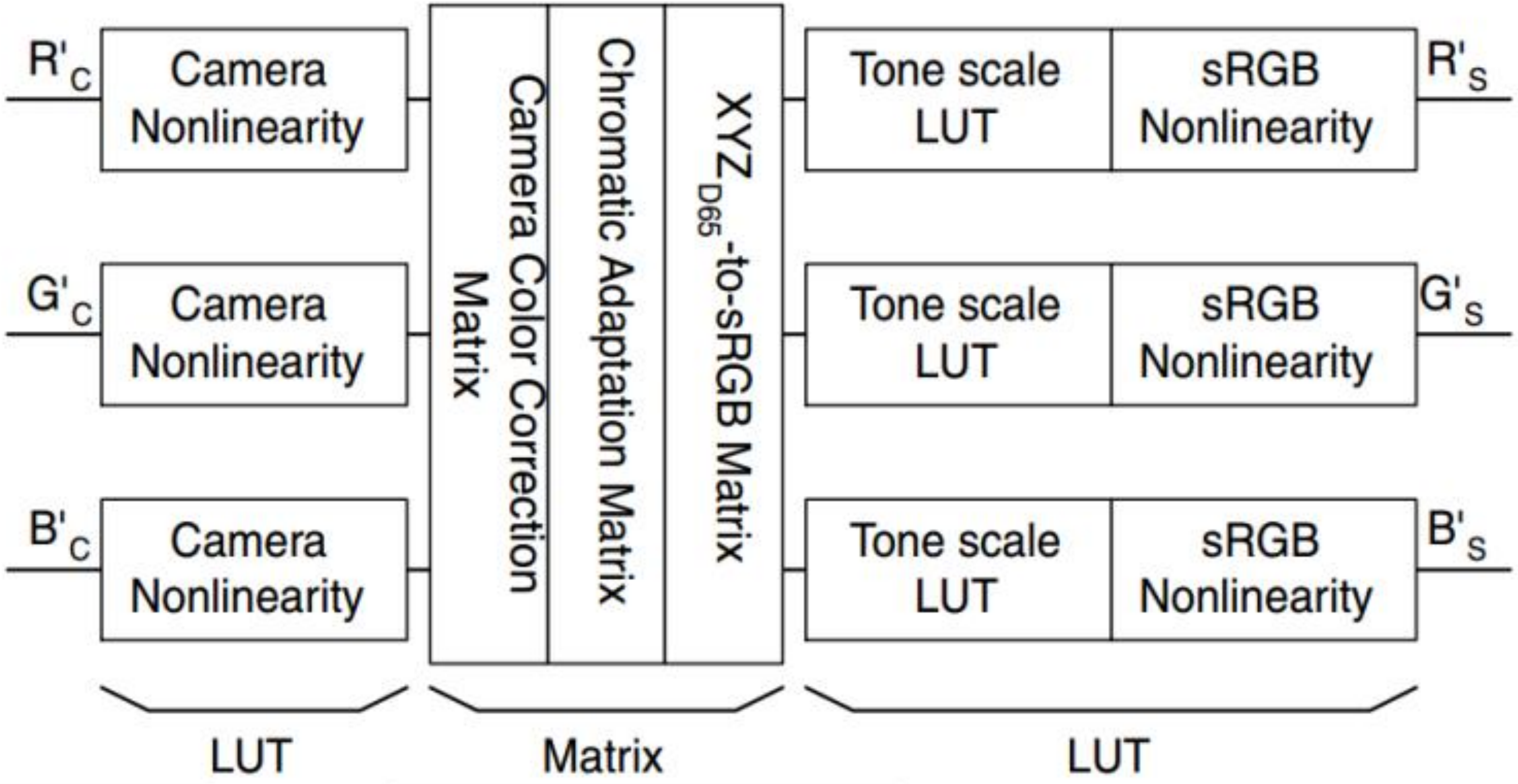
- **Python Library - colour_demosaicing**
  - ***demosaicing_CFA_Bayer_bilinear***
  - ***demosaicing_CFA_Bayer_Malvar2004***
    - *[MHCW04] Henrique S Malvar, Li-Wei He, Ross Cutler, and One Microsoft Way. High-Quality Linear Interpolation for Demosaicing of Bayer-Patterned Color Images. In International Conference of Acoustic, Speech and Signal Processing, 5–8. Institute of Electrical and Electronics Engineers, Inc., May 2004.*
  - ***demosaicing_CFA_Bayer_Menon2007***
    - *[MAC07] Daniele Menon, Stefano Andriani, and Giancarlo Calvagno. Demosaicing With Directional Filtering and a posteriori Decision. IEEE Transactions on Image Processing, 16(1):132–141, January 2007.*

Mosaic

- **Discard the value of other 2 channels**
  - **H*W*3 → H*W**



Color Correction (1/3)



These LUT used to correct sensor RAW

RAW sensor RGB values come in, and sRGB values come out

Tone-scale and sRGB gamma are applied

Color Correction (2/3)

- **CCM (Color Correction Matrix)**
  - Transforms sensor native RGB values into CIE XYZ color space
  - It is important that the white-balance has been performed correctly

Color images are stored in *m* x *n* x 3 arrays (*m* rows (height) x *n* columns (width) x 3 colors). For the sake of simplicity, we transform the color image to a *k* x 3 array, where *k* = *m* x *n*. The Original (uncorrected input) pixel data O can be represented as

$$O = \begin{bmatrix} O_{R1} & O_{G1} & O_{B1} \\ O_{R2} & O_{G2} & O_{B2} \\ & \cdots & \\ O_{Rk} & O_{Gk} & O_{Bk} \end{bmatrix}$$

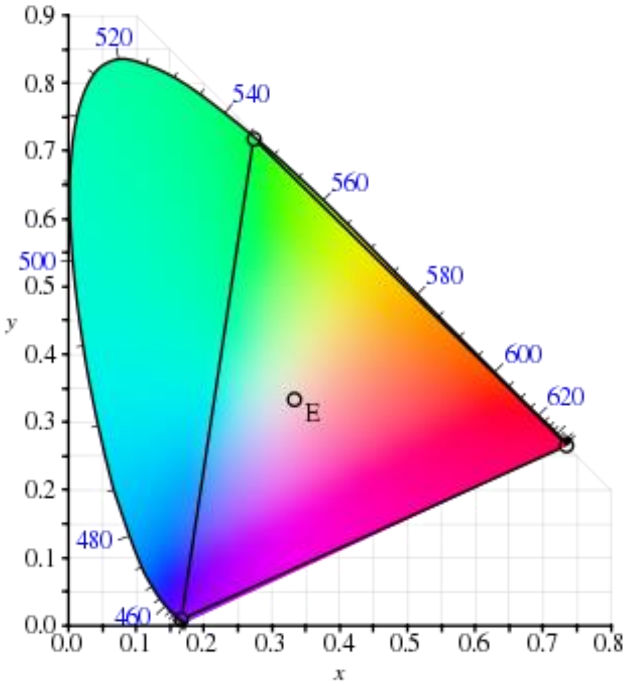
where the entries of row *i*, $[O_{Ri} \quad O_{Gi} \quad O_{Bi}]$, represent the normalized R, G, and B levels of pixel *i*. The transformed (corrected) array is called P, which is achieved by matrix multiplication with A, the **color correction matrix**. Imatest allows you to choose two different forms of A, either 3×3 or 4×3.

Each of the R, G, and B values of each output (corrected) pixel is a linear combination of the three input color channels of that pixel.

$$P = \begin{bmatrix} P_{R1} & P_{G1} & P_{B1} \\ P_{R2} & P_{G2} & P_{B2} \\ & \cdots & \\ P_{Rk} & P_{Gk} & P_{Bk} \end{bmatrix} = \begin{bmatrix} O_{R1} & O_{G1} & O_{B1} \\ O_{R2} & O_{G2} & O_{B2} \\ & \cdots & \\ O_{Rk} & O_{Gk} & O_{Bk} \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{bmatrix}$$

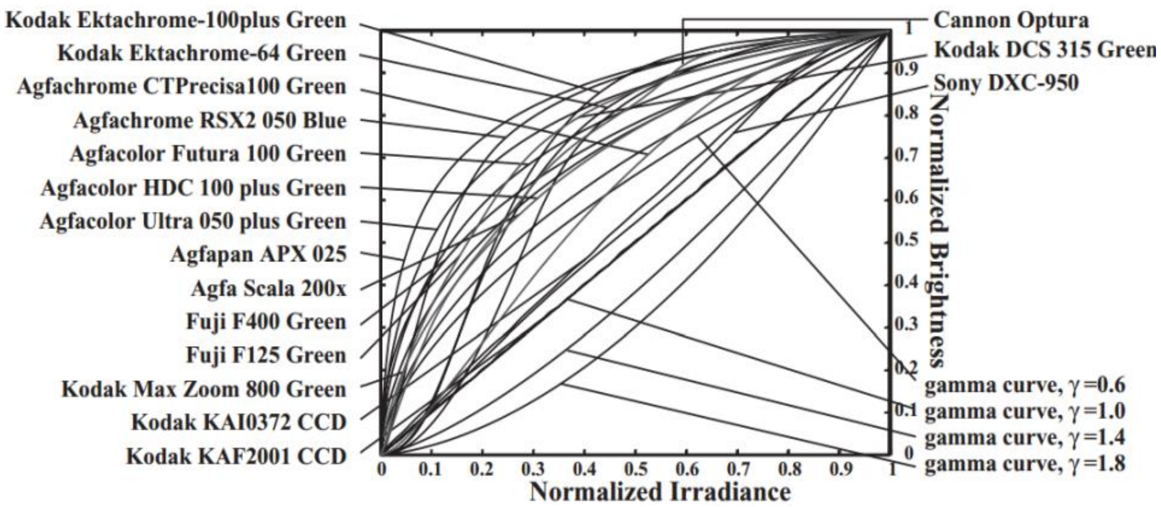
Color Correction (3/3)

- **Transform Color Space**
  - RGB $\longleftrightarrow$ XYZ (CIE 1931)
  - White point defined at X = 1/3, Y = 1/3, Z = 1/3
  - $$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4887180 & 0.3106803 & 0.2006017 \\ 0.1762044 & 0.8129847 & 0.0108109 \\ 0.0000000 & 0.0102048 & 0.9897952 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$



Tone Mapping

- **Display cannot afford the brightness in real world**
  - use more sensor bits to store
- **High Dynamic Range (HDR) → Low Dynamic Range (LDR)**
  - compress sensor bits into 8-bit (RGB24)
  - makes images suitable to be viewed on a digital screen
- **Tone curve: Non-linear mapping of RGB tones**



Rules

- **Due: Wed, 10/16, 23:59**
- **You should use Python to complete this homework**

- **NO CHEAT !!**

For more details ……

https://www.eecs.yorku.ca/~mbrown/CVPR2016_Brown.html

- *M. S. Brown, "Understanding the In-Camera Image Processing Pipeline for Computer Vision", IEEE Computer Vision and Pattern Recognition - Tutorial, June 26, 2016*

http://graphics.stanford.edu/courses/cs178/

- **CS 178 - Digital Photography (Spring 2014)**
- **Marc Levoy: Google Pixel 3**