

# COMP 590/776: Computer Vision

## Lecture 4: Camera Imaging Pipeline

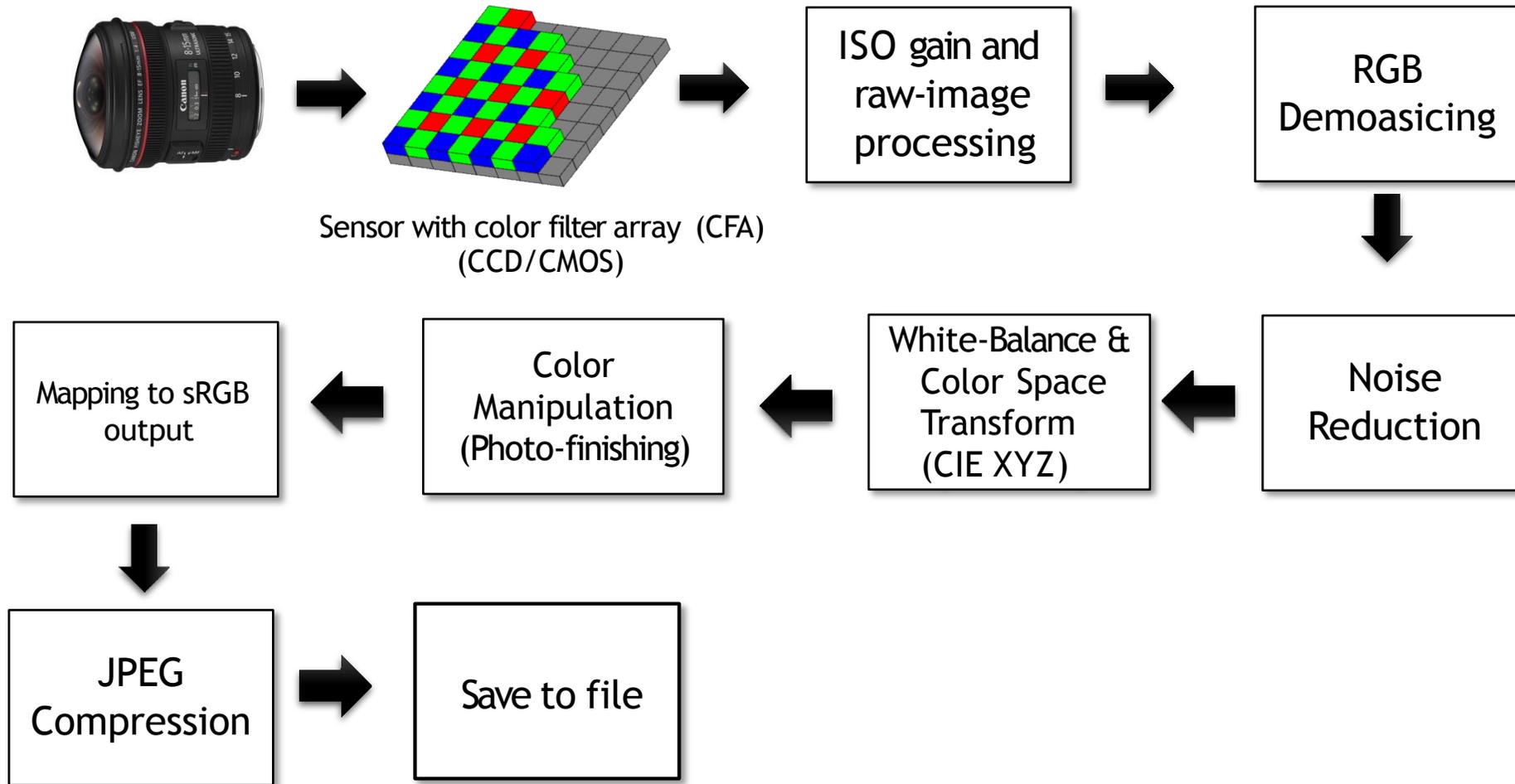
Instructor: Soumyadip (Roni) Sengupta

ULA: Andrea Dunn, William Li, Liujie Zheng



Course Website:  
Scan Me!

# A typical color imaging pipeline

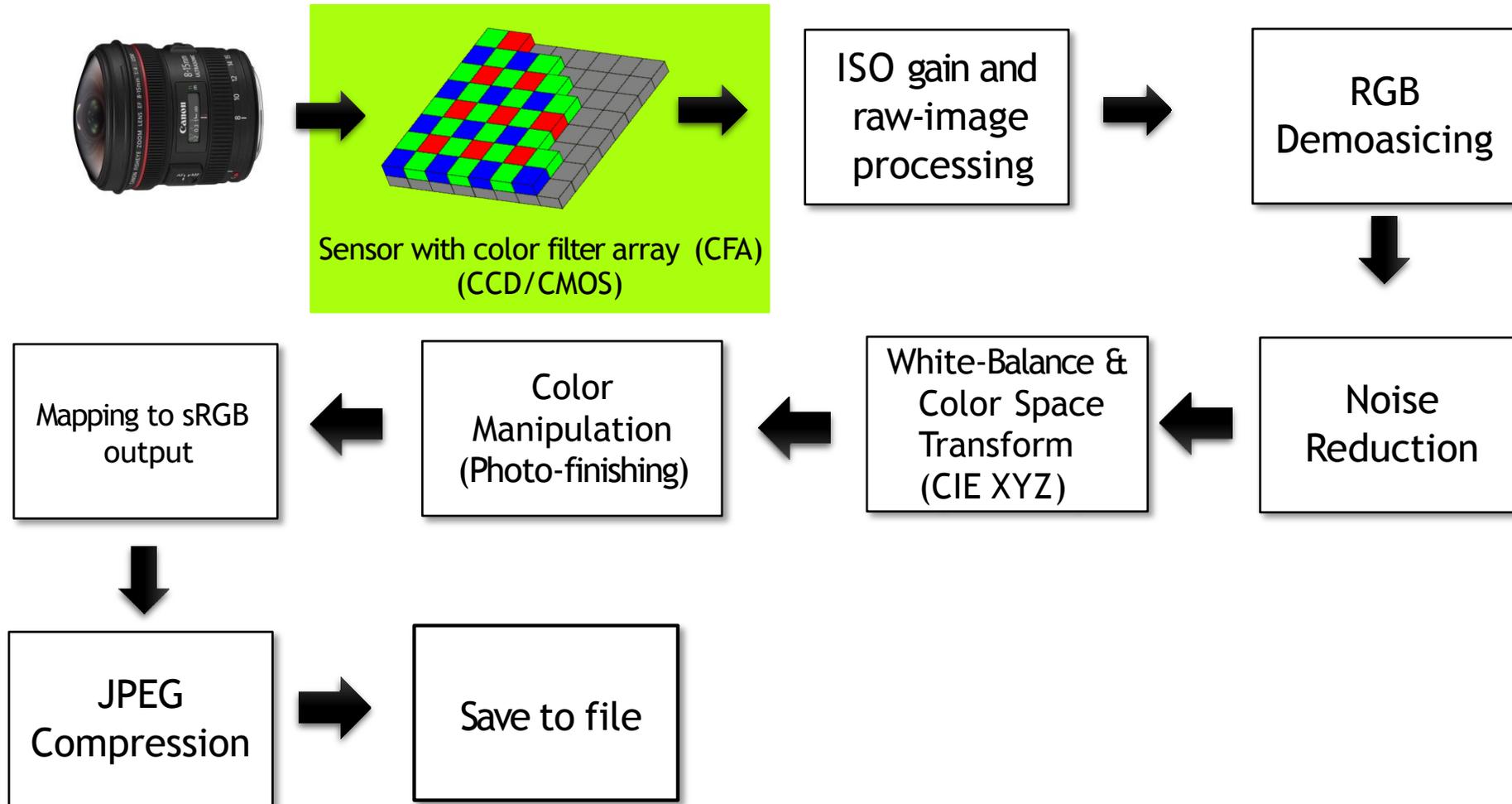


**NOTE:** This diagram represents the steps applied on a typical consumer camera pipeline. ISPs may apply these steps in a different order or combine them in various ways. A modern camera ISP will undoubtedly be more complex, but will almost certainly implement these steps in some manner.

# Integrated signal processor (ISP)

- You will hear the term "ISP" associated with camera pipelines
- An ISP is dedicated hardware used to process the sensor image to produce the final output (JPEG image) that is saved on your device
- The ISP is usually integrated as part of a system on a chip (SoC) that has other modules
- Companies such as Qualcomm, HiSilicon, Intel (and more) sell ISP chips
  - An ISP can be customized by the customer (Samsung, Huawei, LG, Apple, etc)
- Note that it is also possible to perform operations common on an ISP on your device's CPU and GPU

# A typical color imaging pipeline



# Camera sensor



CMOS sensor

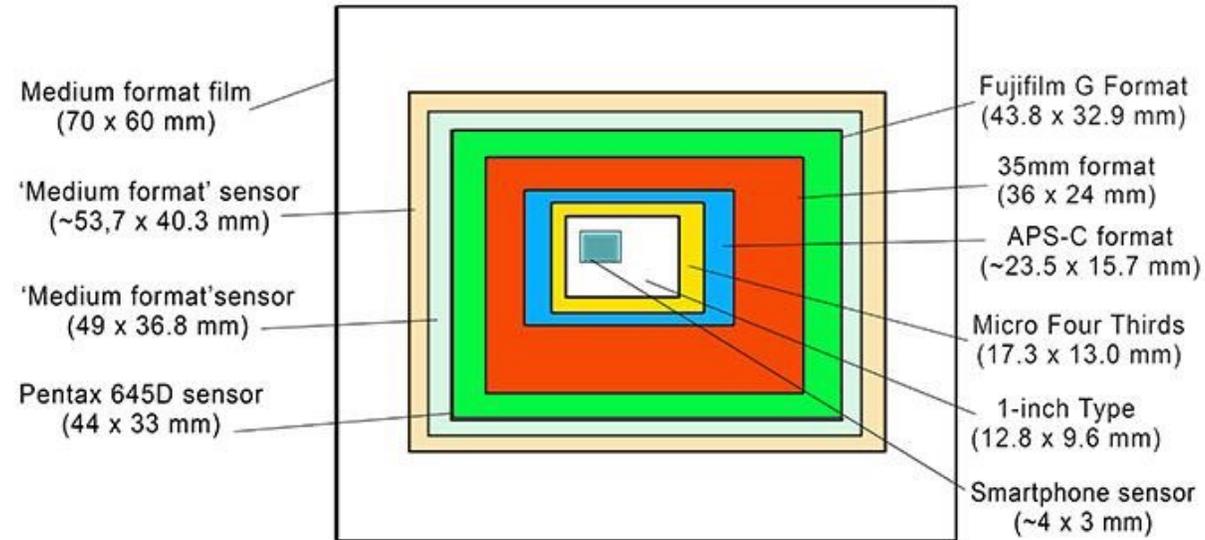
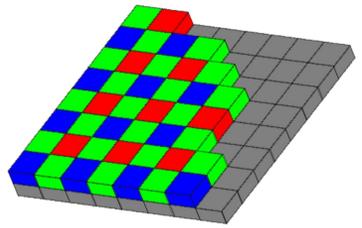


Figure from Photo Review website.

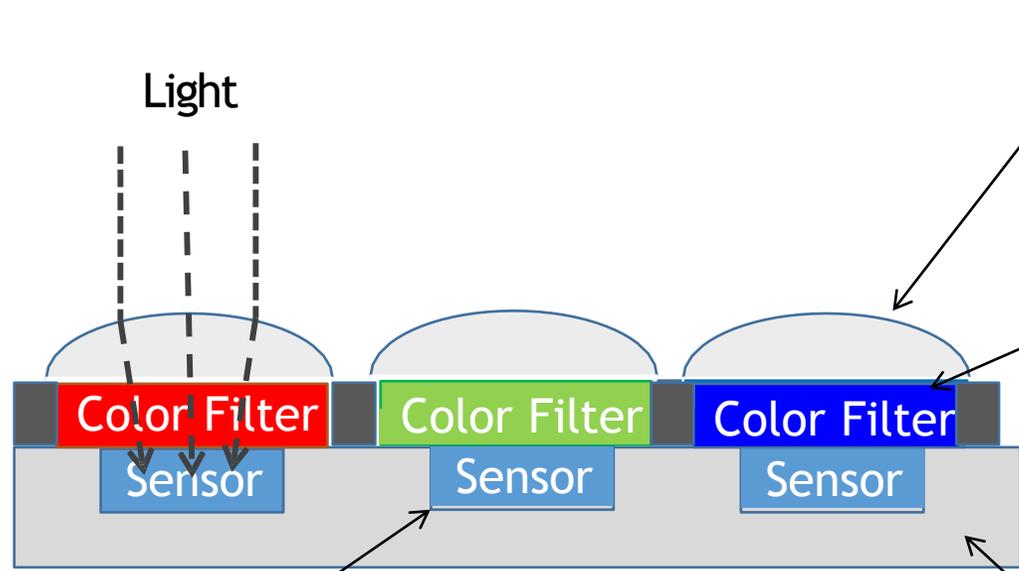
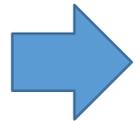
**Almost all consumer camera sensors are based on complementary metal-oxide semiconductor (CMOS) technology.**

- We generally describe sensors in terms of number of pixels and size.
- The larger the sensor, the better the noise performance as more light can fall on each pixel.
- Smart phones have small sensors!

# Camera sensor RGB values



Color filter array or "Bayer" pattern.



Micro-lenses are placed over the diode to help increase light collection on the sensor

Color filters place over the sensor. This forms a Color Filter Array (CFA) also called a "Bayer Pattern" after inventor Bryce Bayer.

**Photodiode**  
Photons hit the diode and force out electrons.  
This design is similar to a solar cell!

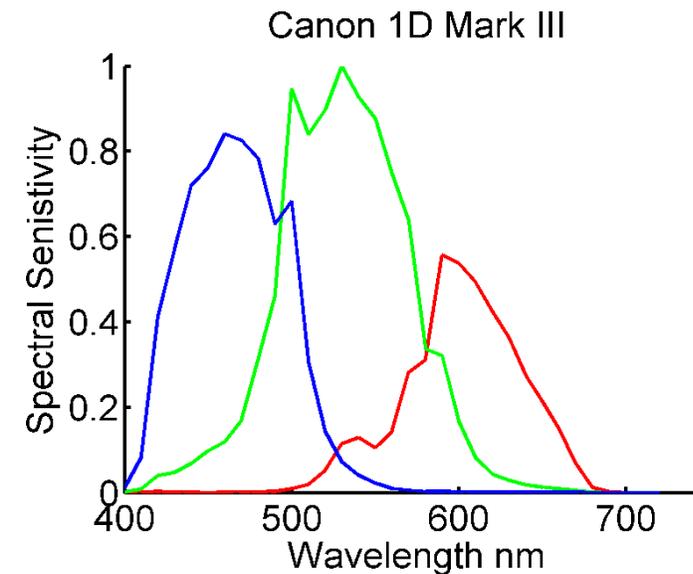
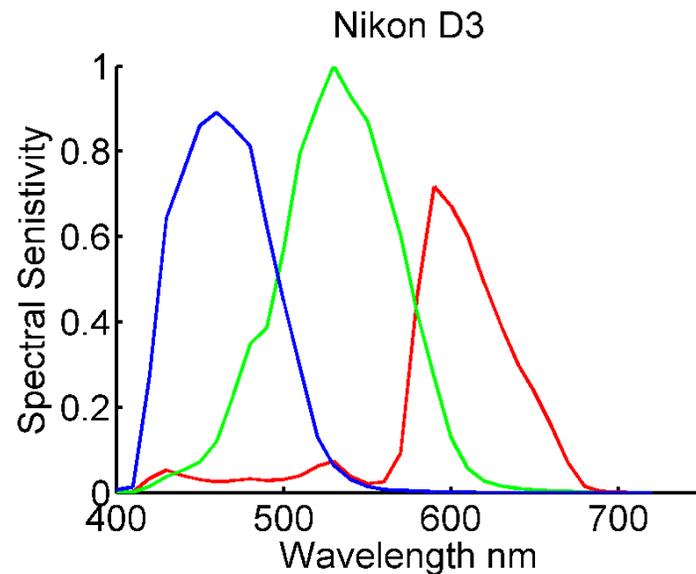
Silicon/Circuitry



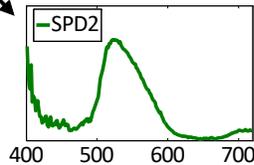
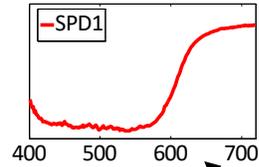
Bryce Bayer

# Camera RGB sensitivity

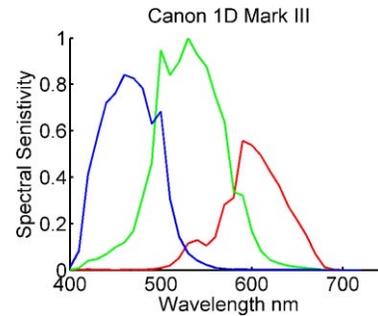
- The color filter array (CFA) on the camera filters the light into three *sensor-specific* RGB primaries



# Sensor raw-RGB image



Remember: physical world is measured by radiometric spectral power distributions.



Color Matching Functions  
CIE XYZ

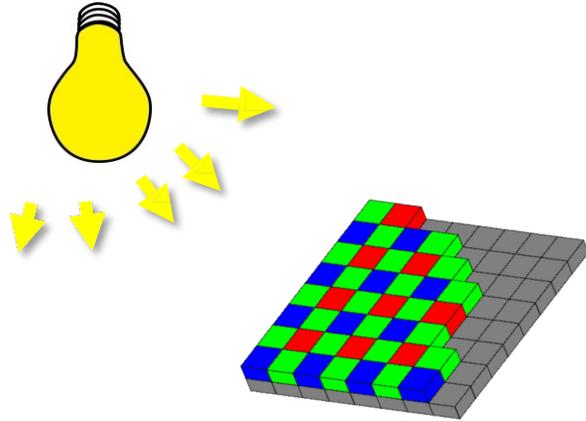
Your camera sensor RGB filter is sensitive to different regions of the incoming SPD.



raw-RGB represents the physical world's SPD "projected" onto the sensor's spectral filters.

# Sensors are linear to irradiance

- Camera sensors are decent light measuring devices
- If you double the amount of light hitting a sensor's pixel, the digital value output of that pixel will double

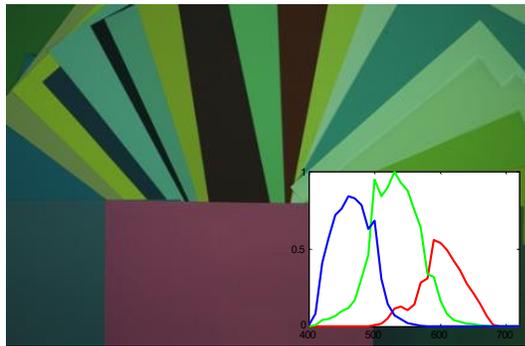


Sensor output is linear with respect to irradiance falling over the sensor over a certain amount of time.

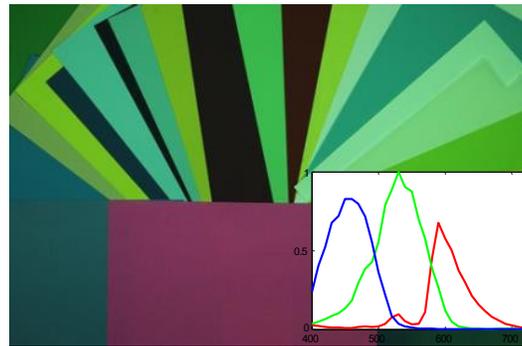
$$I = i * t$$

Digital value  $I$  is a linear function of irradiate  $i$  and exposure  $t$ .

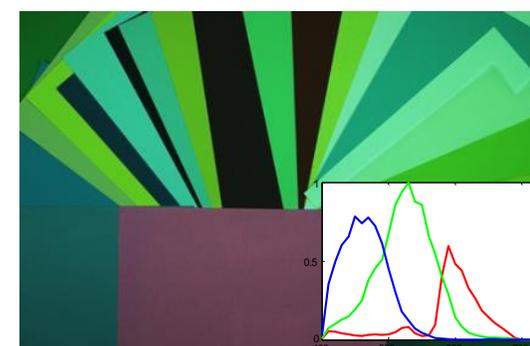
# IMPORTANT: raw-RGB sensor images are not in a standard color space



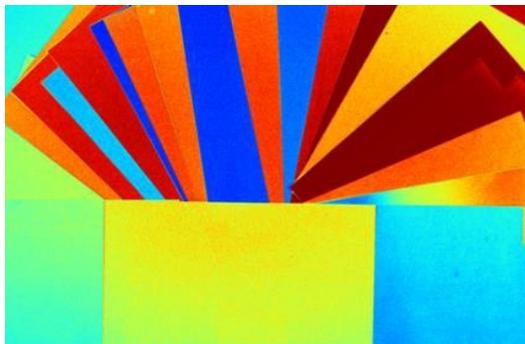
Canon 1D



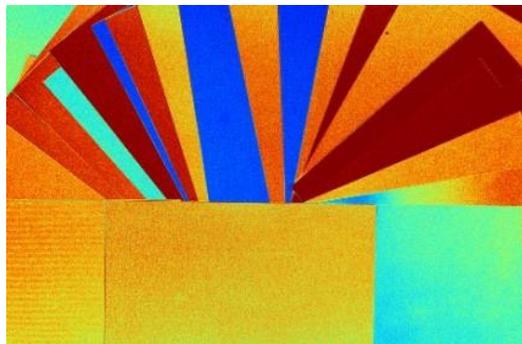
Nikon D40



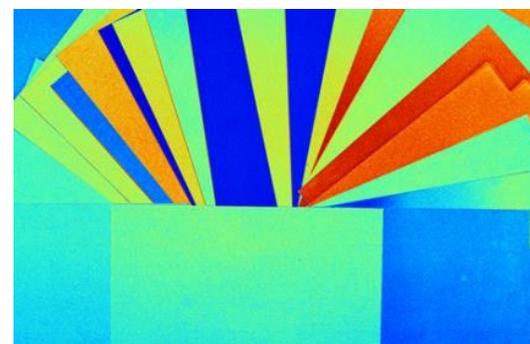
Sony  $\alpha$ 57



$||\text{Canon 1D} - \text{Nikon D40}||_2$



$||\text{Canon 1D} - \text{Sony } \alpha 57||_2$



$||\text{Nikon D40} - \text{Sony } \alpha 57||_2$

Color plots show L2 distance between the raw-RGB values with different cameras.

# Displaying raw-RGB images

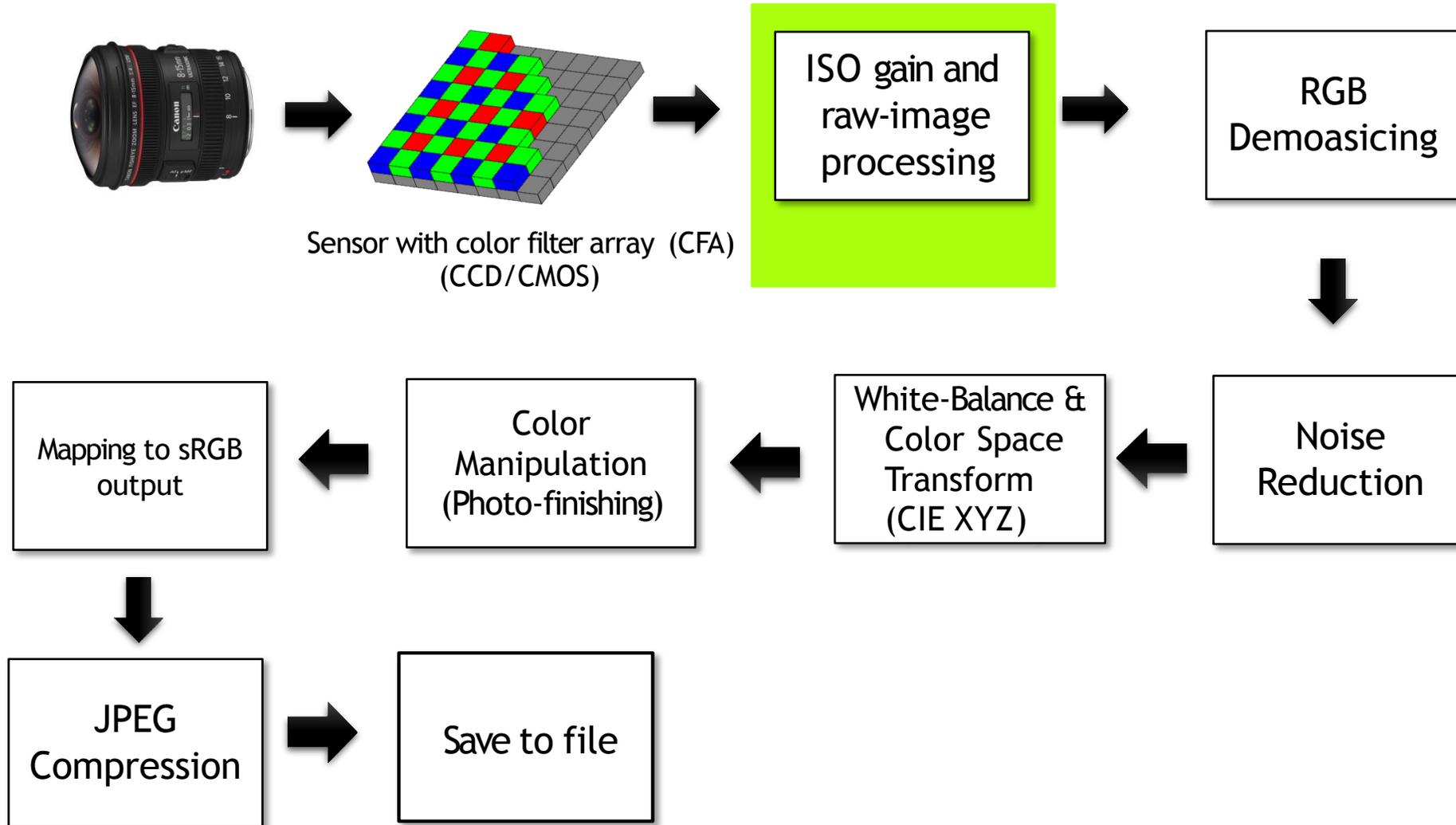
- Inserting a raw-RGB image in your slides, research paper, etc will result in strange colors.
- Why? Our devices (computers, printers, etc) expect the image to be in a standard color space like sRGB.



This is a raw-RGB image. Why does it look bad?  
Because the RGB values are not sRGB values.

**Knowing your color space is important!**

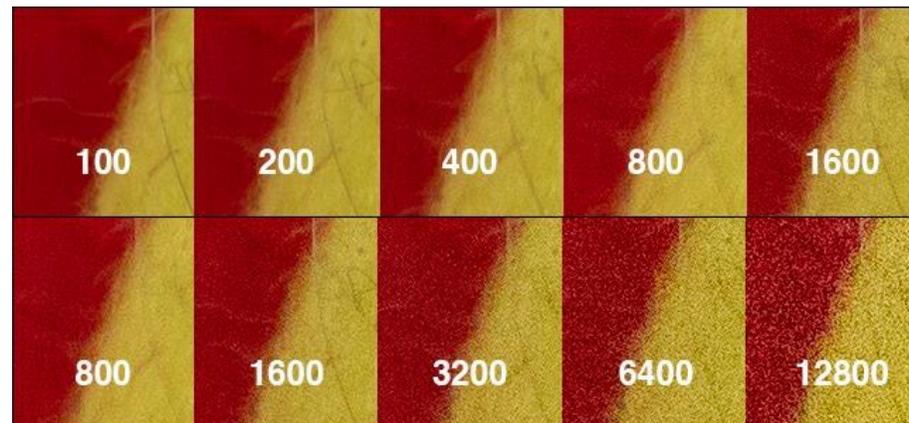
# A typical color imaging pipeline



# ISO signal amplification (gain)

- Imaging sensor signal is amplified and digitized
- Amplification to assist A/D conversion
  - Need to get the voltage to the range required to the desired digital output
- This gain is used to accommodate camera ISO settings
  - Gain to signal applied on sensor
  - Note - gaining the signal also gains image noise

Different ISO settings (note: **the exposure will be shorter for higher ISO**)



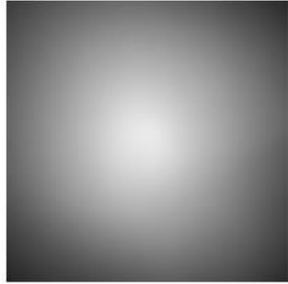
ISO gain and  
raw-image  
processing

Image: Harry Guinness

# Pixel "intensity"

- We often talk about a pixel's intensity, however, a pixel's numerical value has no *unit*
- The digital value of a pixel is based on several factors
  - Exposure (which is a function of both shutter speed and exposure)
  - Gain (ISO setting on the camera)
  - Camera hardware that digitizes the signal
- We typically rely on the *relative* digital values in the image and not the absolute digital values

# Flat-field correction



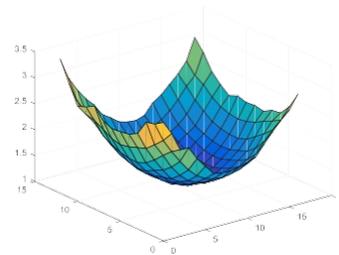
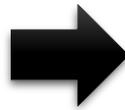
Uniform light falling on the sensor may not appear uniform in the raw-RGB image. This can be caused by the lens, sensor position in the camera housing, etc.



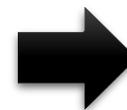
We want to correct this problem such that we get a "flat" output.



Before correction

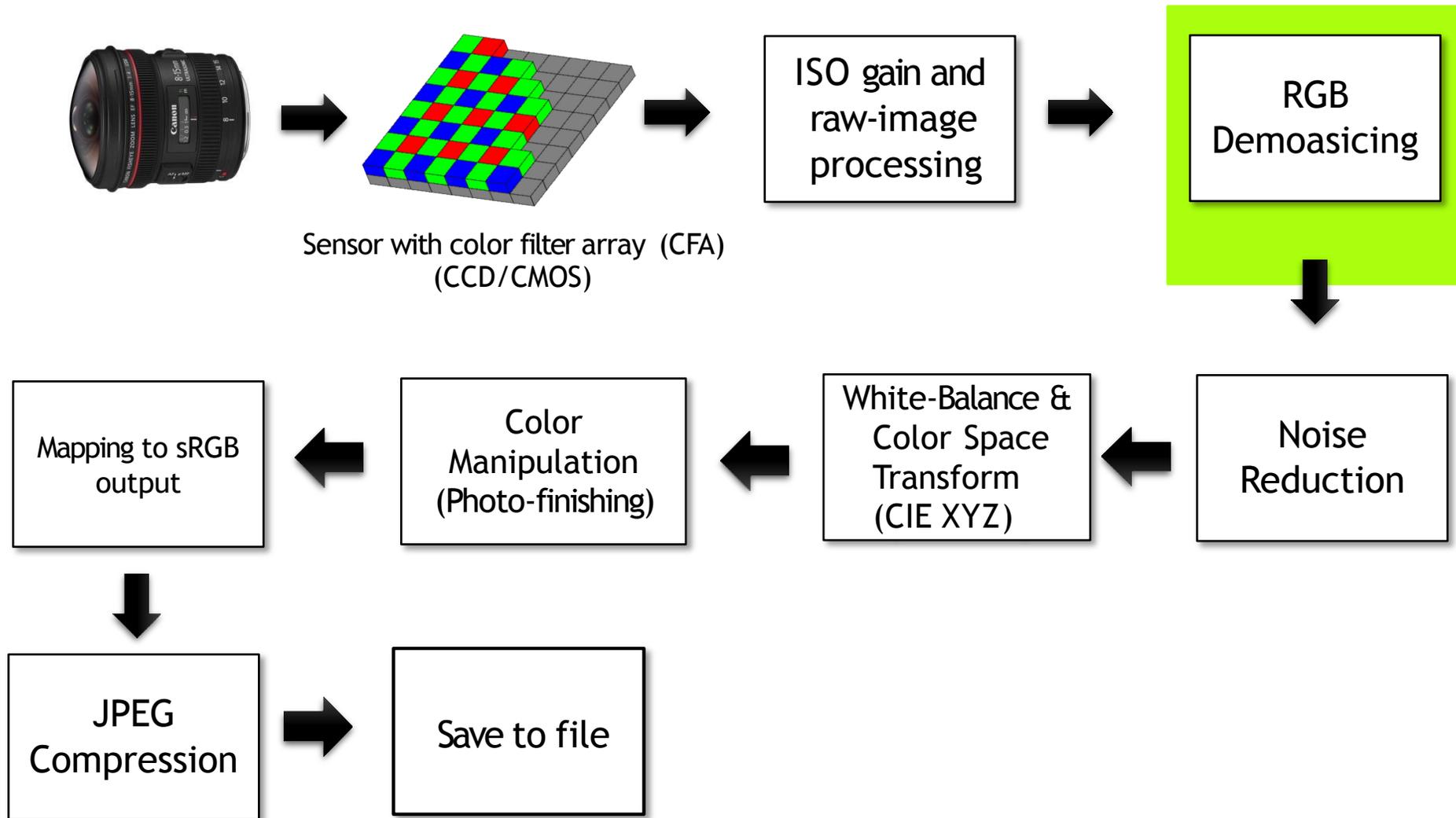


Apply a correction gain over the sensor values.



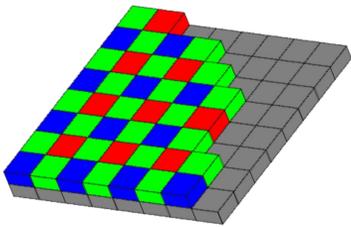
After correction

# A typical color imaging pipeline

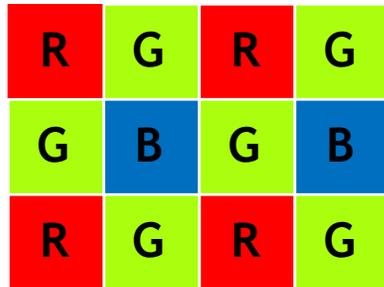


# CFA/Bayer pattern demosaicing

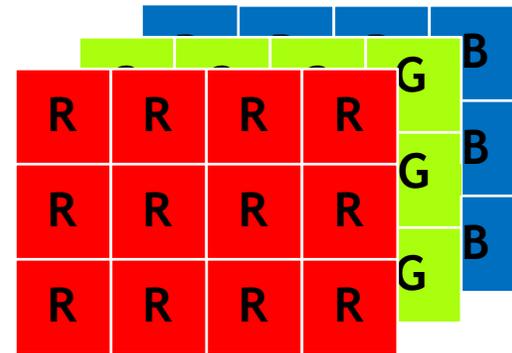
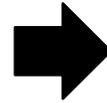
- Color filter array/Bayer pattern placed over pixel sensors
- We want an RGB value at each pixel, so we need to perform interpolation



Sensor with color filter array  
(CMOS)

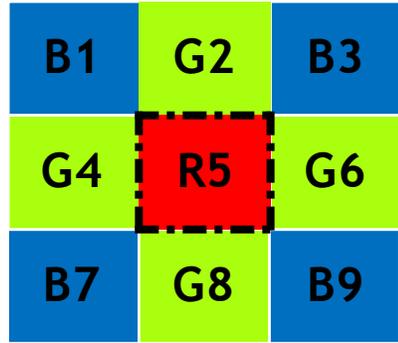


Sensor RGB layout



Desired output with RGB per  
pixel.

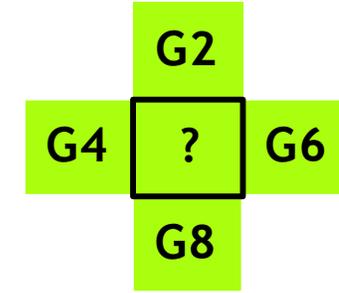
This is a zoomed up version of the Bayer pattern.



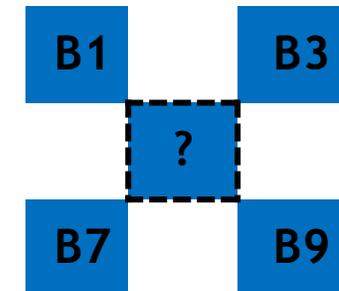
# Simple interpolation

At location R5, we have a red pixel value, but no Green or Blue pixel.

We need to estimate the G5 & B5 values at location R5.



$$\begin{array}{l} \text{R5} \\ \text{G5 ?} \\ \text{B5 ?} \end{array} \begin{array}{l} \nearrow \\ \searrow \end{array} \begin{array}{l} \text{G5} = \frac{\text{G2} + \text{G4} + \text{G6} + \text{G8}}{4} \\ \text{B5} = \frac{\text{B1} + \text{B3} + \text{B7} + \text{B9}}{4} \end{array}$$

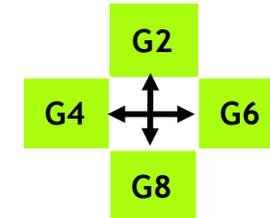


# Simple “edge aware” interpolation



If (  $|G2-G8| \ \&\& \ |G4-G6|$  both  $< \text{Thres}$  ):

$$G5 = \frac{G2 + G4 + G6 + G8}{4}$$

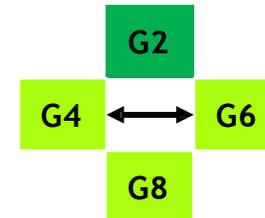


**Case 1**

All about the same.

elseif (  $|G2-G8| > \text{Thres}$  ):

$$G5 = \frac{G4 + G6}{2}$$

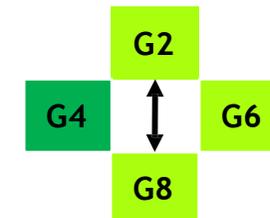


**Case 2**

G2 and G8 differ - ignore them in the interpolation

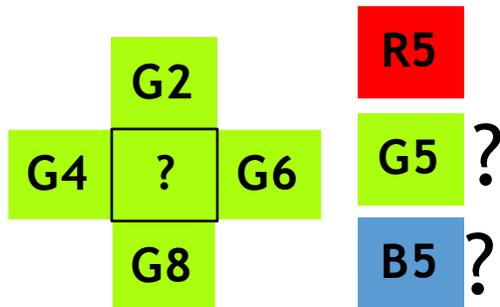
else:

$$G5 = \frac{G2 + G8}{2}$$



**Case 3**

G4 and G6 differ - ignore them in the interpolation



Do this procedure also for the blue pixel, B5.

# Demosaicing in practice

- The prior examples are illustrative algorithms only
- Camera IPs use more complex and proprietary algorithms.
- Demosaicing can be combined with additional processing
  - Highlight clipping
  - Sharpening
  - Noise reduction

# DNN for demosaicing (and denoising)

Gharbi et al SIGGRAPH Asia, 2016

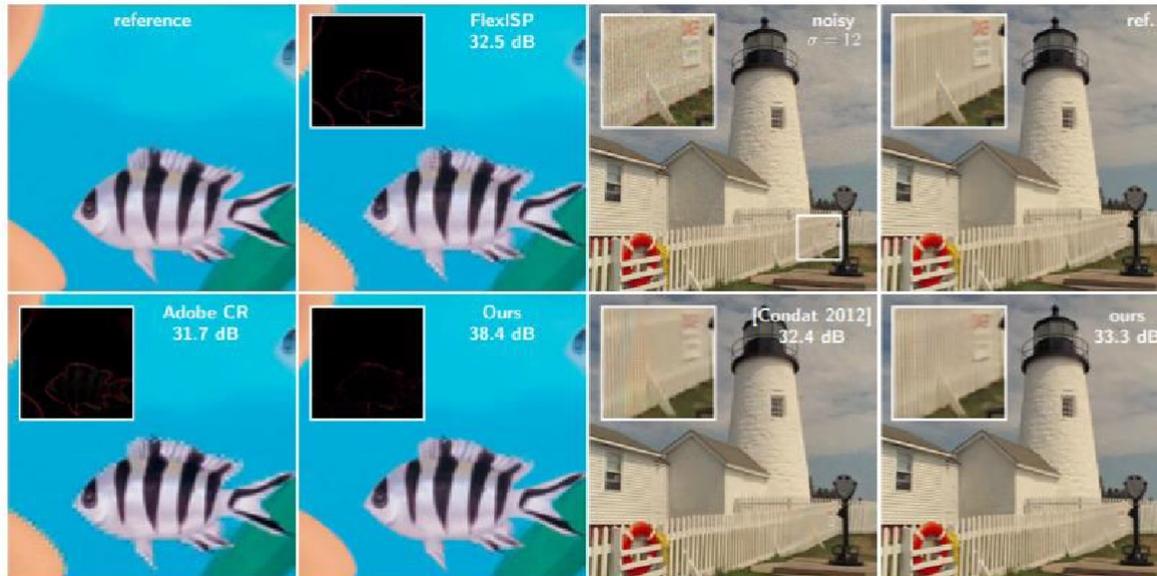
## Deep Joint Demosaicking and Denoising

Michaël Gharbi  
MIT CSAIL

Gaurav Chaurasia  
MIT CSAIL

Sylvain Paris  
Adobe

Frédéric Durand  
MIT CSAIL



**Figure 1:** We propose a data-driven approach for jointly solving denoising and demosaicking. By carefully designing a dataset made of rare but challenging image features, we train a neural network that outperforms both the state-of-the-art and commercial solutions on demosaicking alone (group of images on the left, insets show error maps), and on joint denoising–demosaicking (on the right, insets show close-ups). The benefit of our method is most noticeable on difficult image structures that lead to moiré or zippering of the edges.

### Abstract

Demosaicking and denoising are the key first stages of the digital imaging pipeline but they are also a severely ill-posed problem that infers three color values per pixel from a single noisy measurement. Existing methods rely on hand-crafted filters, or non-local and still exhibit

### 1 Introduction

Demosaicking and denoising are simultaneously the crucial first steps of most digital camera pipelines. They are quintessentially ill-posed reconstruction problems: at least two-thirds of the data is missing and the existing data is corrupted with noise. Furthermore,

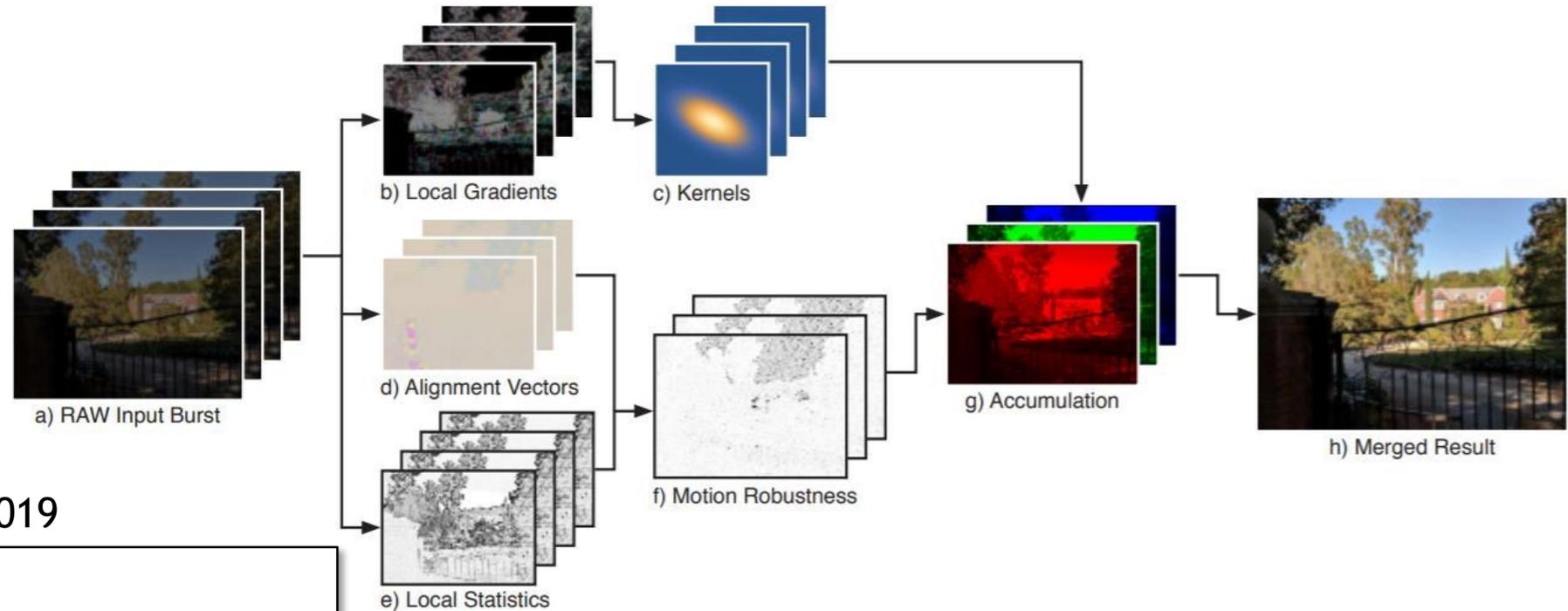
This is a very interesting paper that examines using a DNN to perform demosaicing.

Interestingly, the act of demosaicing to a clean ground truth image implicitly performs denoising.

The only drawback of this paper is that training/testing demosaiced images were synthetically generated by sampling sRGB images and adding noise.

A lack of a raw-RGB dataset is a problem and would be a nice addition in this area.

# Non-learning improvement for demosaicing



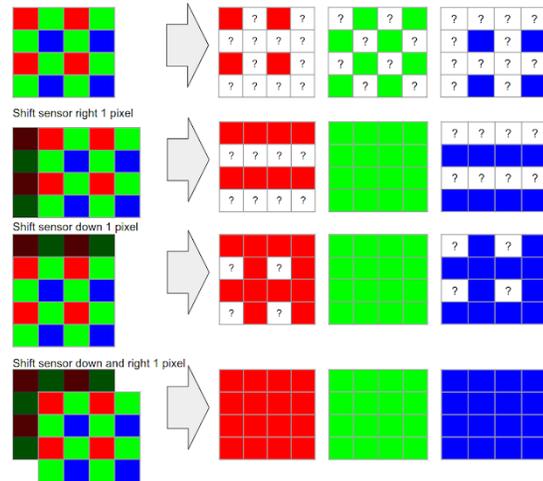
Wronski et al SIGGRAPH 2019

## Handheld Multi-Frame Super-Resolution

BARTLOMIEJ WRONSKI, IGNACIO GARCIA-DORADO, MANFRED ERNST, DAMIEN KELLY, MICHAEL KRAININ, CHIA-KAI LIANG, MARC LEVOY, and PEYMAN MILANFAR, Google Research

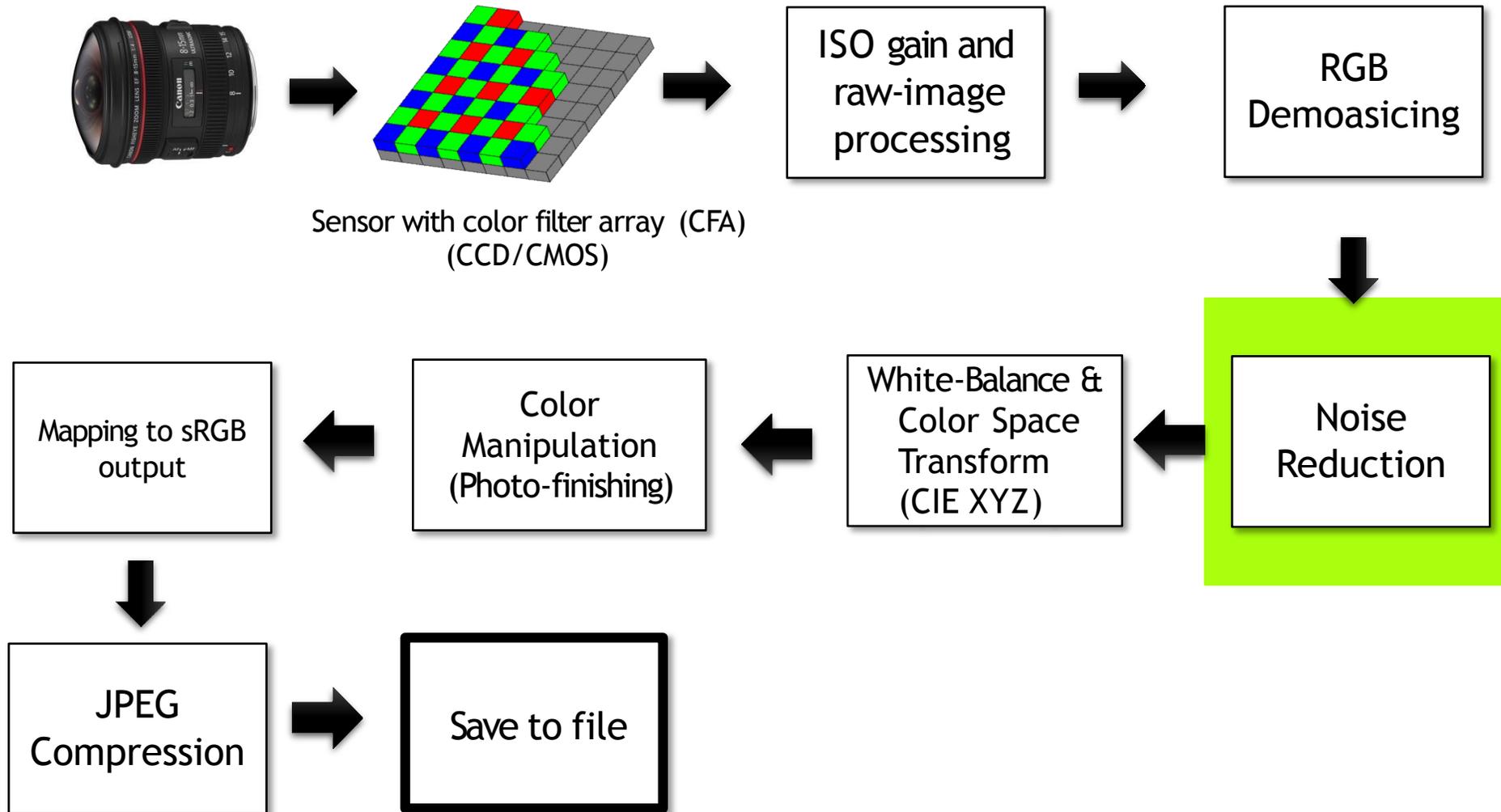


Fig. 1. We present a multi-frame super-resolution algorithm that supplants the need for demosaicing in a camera pipeline by merging a burst of raw images. We show a comparison to a method that merges frames containing the same-color channels together first, and is then followed by demosaicing (top). By contrast, our method (bottom) creates the full RGB directly from a burst of raw images. This burst was captured with a hand-held mobile phone and processed on device. Note in the third (red) inset that the demosaiced result exhibits aliasing (Moiré), while our result takes advantage of this aliasing, which changes on every frame in the burst, to produce a merged result in which the aliasing is gone but the cloth texture becomes visible.



This paper uses multiple frames and very small camera motion (from hand tremors) to perform demosaicing and super-resolution. By exploiting motion, they can fill in missing Bayer data.

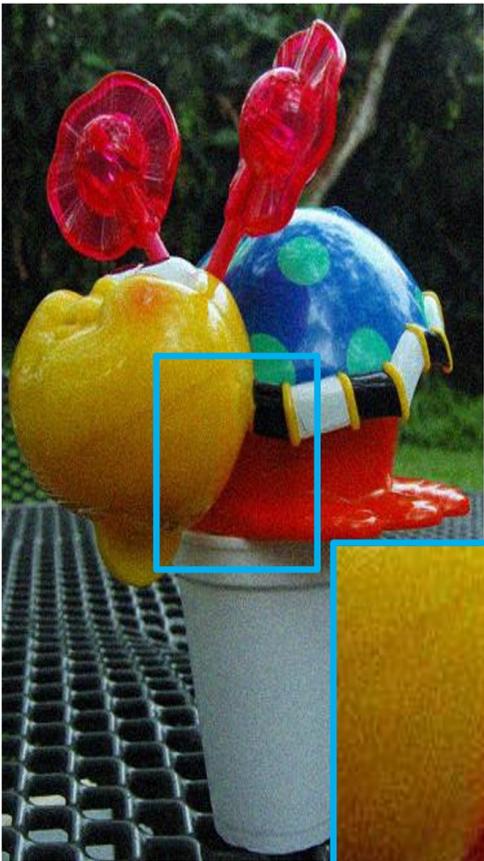
# A typical color imaging pipeline



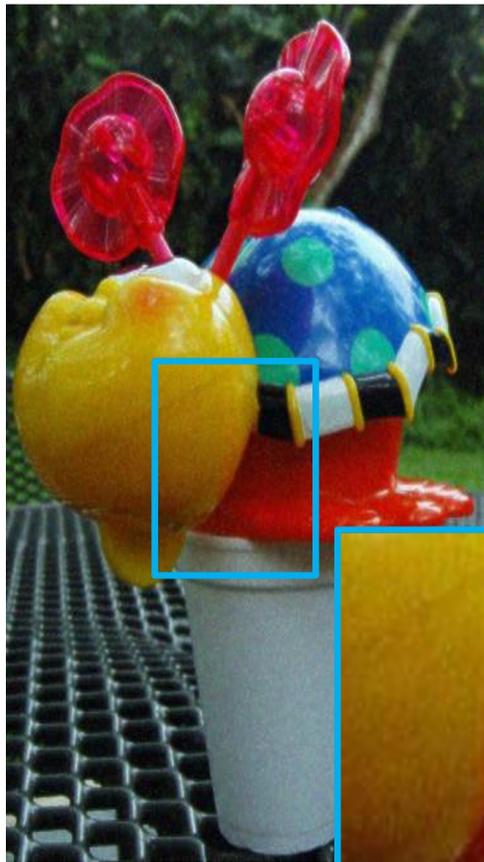
\* Note that steps can be optional (e.g. noise reduction) or applied in slightly different order.

# Noise reduction (NR)

- All sensors inherently have noise
- Most cameras apply additional NR after A/D conversion
- A simple method is described in the next slide
- For high-end cameras, it is likely that cameras apply different strategies depending on the ISO settings, e.g. high ISO will result in more noise, so a more aggressive NR could be used
- Smartphone cameras, because the sensor is small, apply aggressive noise reduction.



Input



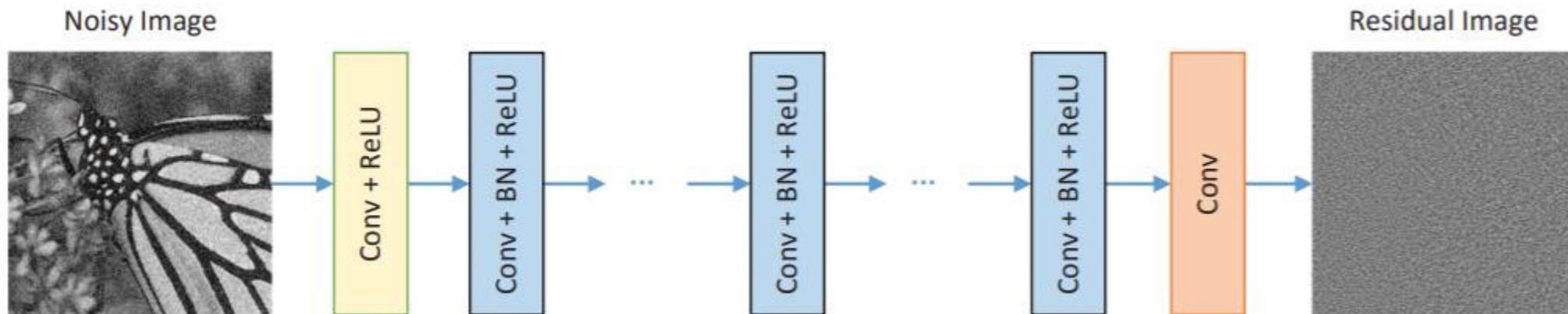
Noise reduced image



# DNNs for denoising

- Zhang et al "Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising", TIP 2017
- The following is a website with many denoising papers with code available:

<https://paperswithcode.com/task/image-denoising?page=2>



Many methods predict the noise image instead of the denoised image (DnNN from TIP 2017)

# Denoising contest at CVPR'19

## NTIRE denoising contest at CVPR'19

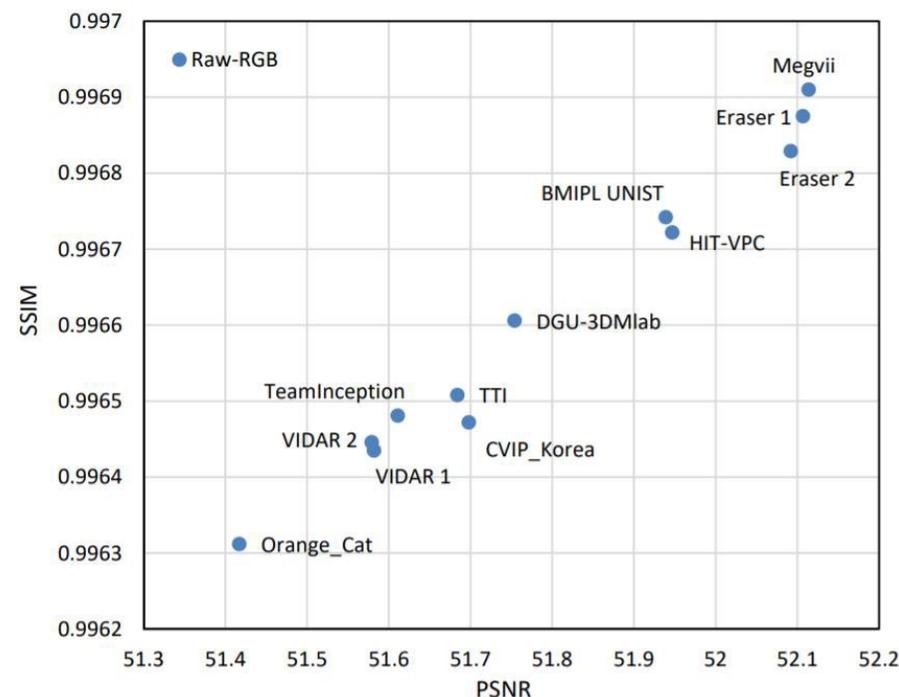
### NTIRE 2019 Challenge on Real Image Denoising: Methods and Results

Abdelrahman Abdelhamed	Radu Timofte	Michael S. Brown	Songhyun Yu		
Bumjun Park	Jechang Jeong	Seung-Won Jung	Dong-Wook Kim	Jae-Ryun Chung	
Jiaming Liu	Yuzhi Wang	Chi-Hao Wu	Qin Xu	Yuqian Zhou	Chuan Wang
Shaofan Cai	Yifan Ding	Haoqiang Fan	Jue Wang	Kai Zhang	Wangmeng Zuo
Magauyiya Zhussip	Dong Won Park	Shakarim Soltanayev	Se Young Chun		
Zhiwei Xiong	Chang Chen	Muhammad Haris	Kazutoshi Akita	Tomoki Yoshida	
Greg Shakhnarovich	Norimichi Ukita	Syed Waqas Zamir	Aditya Arora		
Salman Khan	Fahad Shahbaz Khan	Ling Shao	Sung-Jea Ko	Dong-Pan Lim	
Seung-Wook Kim	Seo-Won Ji	Sang-Won Lee	Wenyi Tang	Yuchen Fan	
Yuqian Zhou	Ding Liu	Thomas S. Huang	Deyu Meng	Lei Zhang	
Hongwei Yong	Yiyun Zhao	Pengliang Tang	Yue Lu	Raimondo Schettini	
Simone Bianco	Simone Zini	Chi Li	Yang Wang	Zhiguo Cao	

#### Abstract

*This paper reviews the NTIRE 2019 challenge on real image denoising with focus on the proposed methods and their results. The challenge has two tracks for quantitatively evaluating image denoising performance in (1) the Bayer-pattern raw-RGB and (2) the standard RGB (sRGB) color spaces. The tracks had 216 and 220 registered participants, respectively. A total of 15 teams, proposing 17 methods,*

*ing image denoisers, especially the additive white Gaussian noise (AWGN)—for example, [6, 9, 38]. Recently, more focus has been given to evaluating image denoisers on real noisy images [1, 25]. It was shown that the performance of learning-based image denoisers on real noisy images can be limited if trained using only synthetic noise. Also, hand-engineered and statistics-based methods have been shown to perform better on real noisy images. To this end, we have proposed this challenge as a means to evaluate and bench-*



More than 10 DNN - based submissions

# Training data for noise reduction (denoising)

## Training/validation/testing images

Real images captured under low-ISO and high-ISO.

Or, synthetic images with noise added.



Label/ground truth image

Low ISO setting

Noisy image

High ISO setting

Example from Darmstadt Noise Dataset

Most DNNs operate on small patches (32x32). So, a single image provides many training samples.

# Denoising dataset

## SIDD: Smartphone Image Denoising Dataset

Abdelhamed et al CVPR 2018

### A High-Quality Denoising Dataset for Smartphone Cameras

Abdelrahman Abdelhamed  
York University  
kamel@eecs.yorku.ca

Stephen Lin  
Microsoft Research  
stevelin@microsoft.com

Michael S. Brown  
York University  
mbrown@eecs.yorku.ca

#### Abstract

The last decade has seen an astronomical shift from imaging with DSLR and point-and-shoot cameras to imaging with smartphone cameras. Due to the small aperture and sensor size, smartphone images have notably more noise than their DSLR counterparts. While denoising for smartphone images is an active research area, the research community currently lacks a denoising image dataset representative of real noisy images from smartphone cameras with high-quality ground truth. We address this issue in this paper with the following contributions. We propose a systematic procedure for estimating ground truth for noisy images that can be used to benchmark denoising performance for smartphone cameras. Using this procedure, we have captured a dataset – the Smartphone Image Denoising Dataset (SIDD) – of ~30,000 noisy images from 10 scenes under different lighting conditions using five representative smartphone cameras and generated their ground truth images. We used this dataset to benchmark a number of denoising algorithms. We show that CNN-based methods perform better when trained on our high-quality dataset than when trained using alternative strategies, such as low-ISO images used as a proxy for ground truth data.

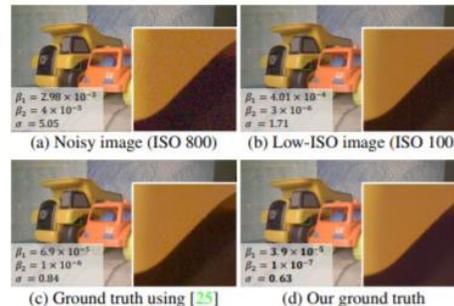
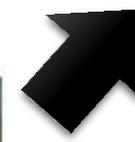
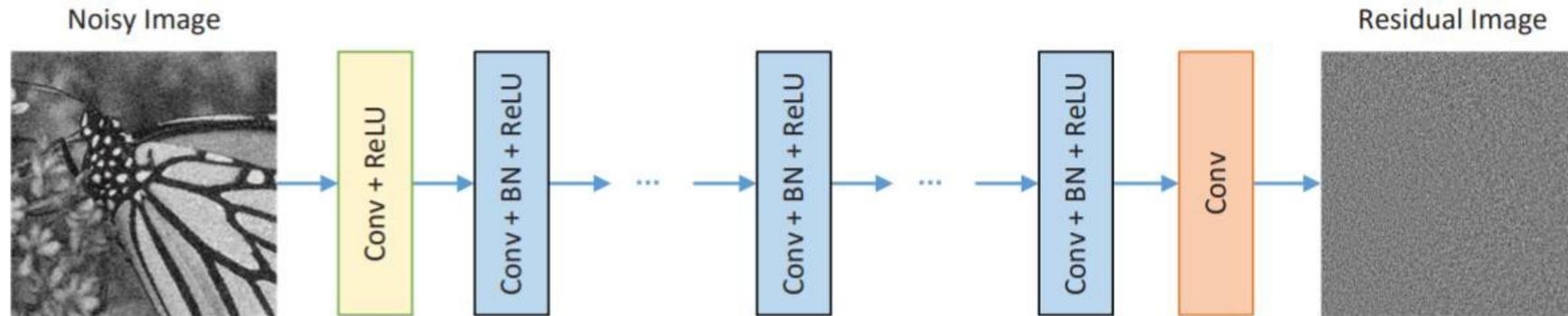


Figure 1: An example scene imaged with an LG G4 smartphone camera: (a) a high-ISO noisy image; (b) same scene captured with low ISO – this type of image is often used as ground truth for (a); (c) ground truth estimated by [25]; (d) our ground truth. Noise estimates ( $\beta_1$  and  $\beta_2$  for noise level function and  $\sigma$  for Gaussian noise – see Section 3.2) indicate that our ground truth has significantly less noise than both (b) and (c). Images shown are processed in raw-RGB, while sRGB images are shown here to aid visualization.

dataset is essential both to focus attention on denoising of

- 30,000 images
- 5 cameras
- 160 scene instances
- 15 ISO settings
- Direct current lighting
- Three illuminations

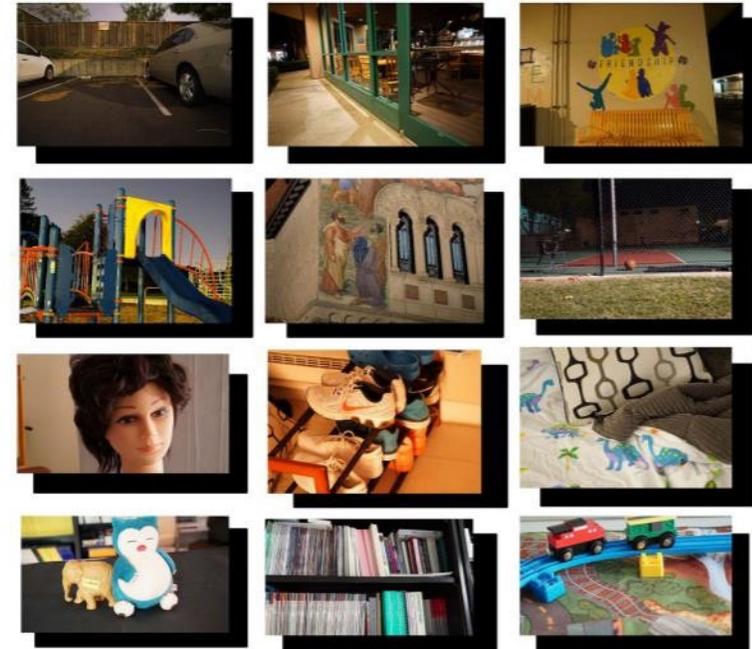
# Instant denoising improvements with better training data



Learning-based DNNs get better performance when trained on real noisy images.

# "Learning to see in the dark"

Sony $\alpha$ 7S II	Filter array	Exposure time (s)	# images
x300	Bayer	1/10, 1/30	1190
x250	Bayer	1/25	699
x100	Bayer	1/10	808
Fujifilm X-T2	Filter array	Exposure time (s)	# images
x300	X-Trans	1/30	630
x250	X-Trans	1/25	650
x100	X-Trans	1/10	1117



## Chen et al CVPR 2018

### Learning to See in the Dark

Chen Chen  
UIUC

Qifeng Chen  
Intel Labs

Jia Xu  
Intel Labs

Vladlen Koltun  
Intel Labs



(a) Camera output with ISO 8,000

(b) Camera output with ISO 409,600

(c) Our result from the raw data of (a)

Figure 1. Extreme low-light imaging with a convolutional network. Dark indoor environment. The illuminance at the camera is  $< 0.1$  lux. The Sony  $\alpha$ 7S II sensor is exposed for 1/30 second. (a) Image produced by the camera with ISO 8,000. (b) Image produced by the camera with ISO 409,600. The image suffers from noise and color bias. (c) Image produced by our convolutional network applied to the raw sensor data from (a).

#### Abstract

Imaging in low light is challenging due to low photon count and low SNR. Short-exposure images suffer from noise, while long exposure can induce blur and is often impractical. A variety of denoising, deblurring, and enhancement techniques have been proposed, but their effectiveness is limited in extreme conditions, such as video-rate imaging at night. To support the development of learning-based methods for low-light image processing, we introduce

cal means to increase SNR in low light, including opening the aperture, extending exposure time, and using flash. But each of these has its own characteristic drawbacks. For example, increasing exposure time can introduce blur due to camera shake or object motion.

The challenge of fast imaging in low light is well-known in the computational photography community, but remains open. Researchers have proposed techniques for denoising, deblurring, and enhancement of low-light im-

SID dark data generated 5000+ pairs of raw-RGB images captured with very short and very long exposures.

Given a short exposure image, the DNN predicts the long exposure image.

While note a denoising paper, this extended ISO feature is quite similar.

# “Burst photography for high dynamic range and low-light imaging on mobile cameras”, by Google Research. Part of Google Pixel’s Night mode camera

## Burst photography for high dynamic range and low-light imaging on mobile cameras

SIGGRAPH 2016

Samuel W. Hasinoff  
Jonathan T. Barron

Dillon Sharlet  
Florian Kainz  
Google Research

Ryan Geiss  
Jiawen Chen

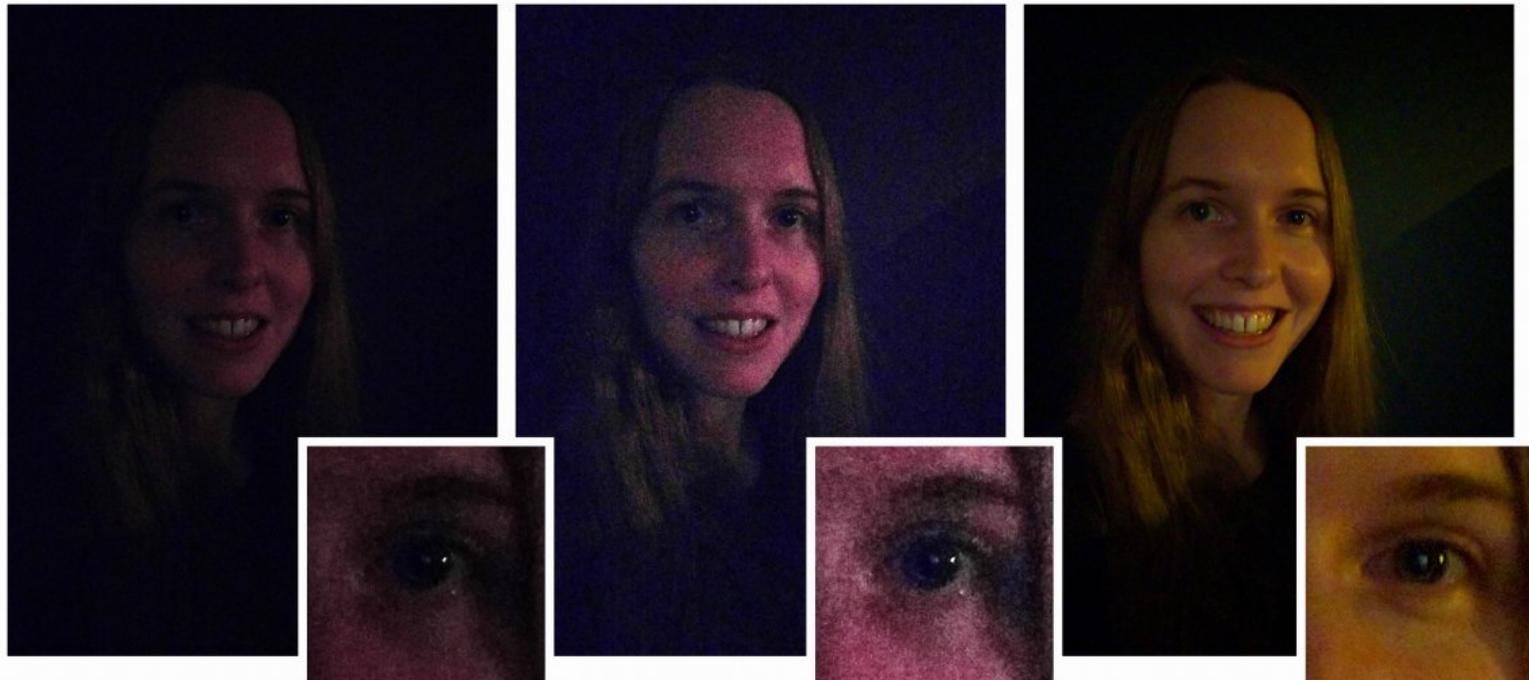
Andrew Adams  
Marc Levoy

Key Idea: Capture multiple images of fixed exposure. Merge and combine them in frequency domain for denoising.

See also: Bracketed Exposure (conventional way of denoising low-light imagery)

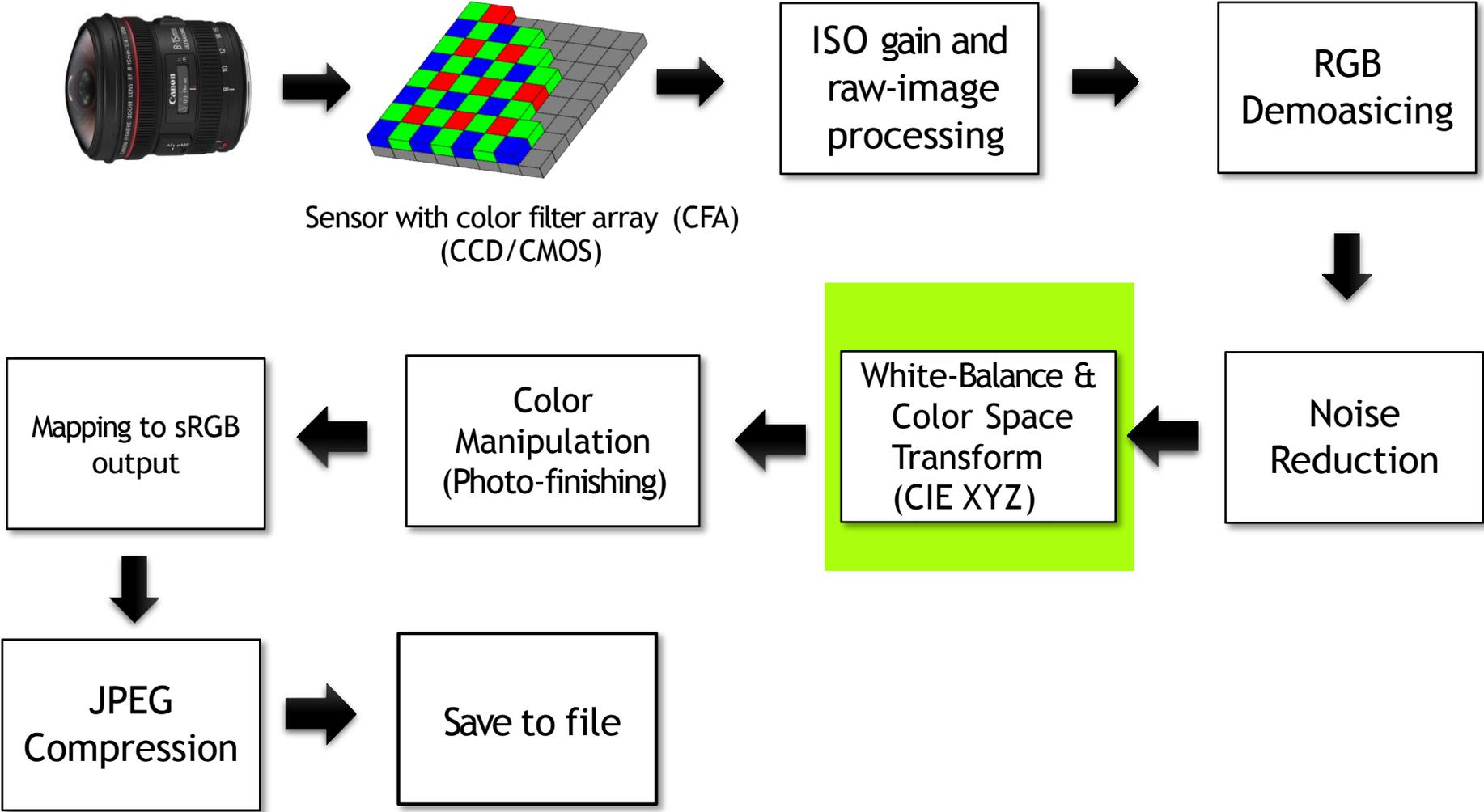
Conventional Camera

Burst Photography



This is why you have to hold your pose for few seconds while imaging in low-light, else you get blurred images!

# A typical color imaging pipeline



# Two step procedure

- (1) apply a white-balance correction to the raw-RGB values
- (2) map the white-balanced raw-RGB values to CIE XYZ

White balance

#		
	#	
		#

3x3 diagonal matrix

Color space transform (CST)

#	#	#
#	#	#
#	#	#

3x3 full matrix (or polynomial function)

raw-RGB values



#		
	#	
		#



white-balance raw-RGB



#	#	#
#	#	#
#	#	#



WB-raw-RGB mapped to CIE XYZ



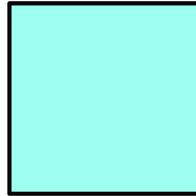
White-Balance &  
Color Space  
Transform  
(CIE XYZ)

# How does white balance (WB) work?



raw-RGB sensor image  
(pre-white-balance correction)

Sensor's  
response to  
illumination ( $\ell$ )



$$\begin{bmatrix} \ell_r \\ \ell_g \\ \ell_b \end{bmatrix} = \begin{bmatrix} 0.2 \\ 0.8 \\ 0.8 \end{bmatrix}$$

Color of 'true' white patch  
under this illumination.

White-balance  
diagonal matrix



"White-balanced"  
raw-RGB image

$$\begin{bmatrix} r_{wb} \\ g_{wb} \\ b_{wb} \end{bmatrix} = \begin{bmatrix} 1/\ell_r & 0 & 0 \\ 0 & 1/\ell_g & 0 \\ 0 & 0 & 1/\ell_b \end{bmatrix} \begin{bmatrix} r \\ g \\ b \end{bmatrix}$$

White-Balance &  
Color Space  
Transform  
(CIE XYZ)

# White balance (computational color constancy)

- **The challenging part for white-balance is determining the proper white-balance setting!**
- Users can manually set the white balance
  - Camera specific white-balance matrices for common illuminations
  - These can be manually selected by the user
- Otherwise auto white balance (AWB) is performed
  - In computer vision, we often refer to AWB as "illumination estimation"
  - Since the hard part is trying to determine what the illumination in the scene is.

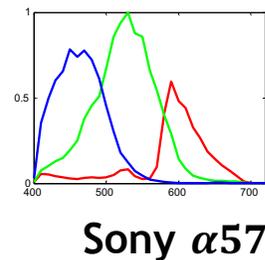
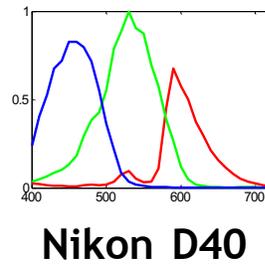
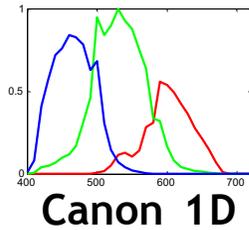
AWB is active research area

# Color mapping/colorimetric stage

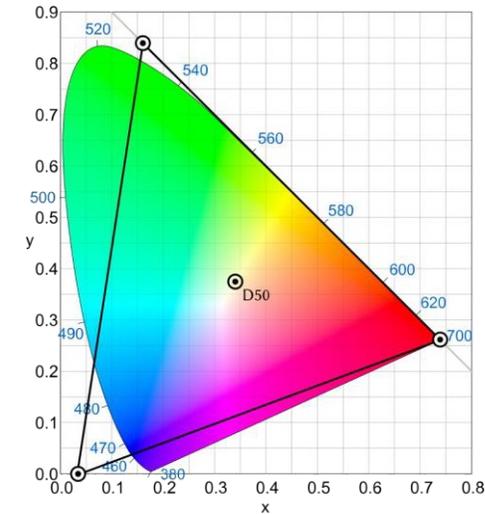
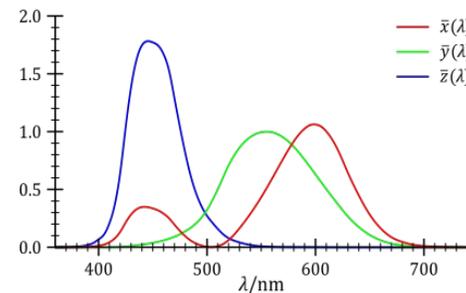
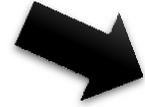
- This step in the IPS converts the sensor raw-RGB values to a device independent color space

Camera sensors have their own spectral response.

We need to map it into a standard response (CIE XYZ).



raw-RGB



We will use CIE XYZ in this lecture, but most cameras use a related space called ProPhoto.

White-Balance & Color Space Transform (CIE XYZ)

# Auto white balance (AWB)

- If manual white balance is not used, then an AWB algorithm is performed.
- AWB needs to determine the sensor's raw-RGB response to the scene illumination from an arbitrary image
- This is surprisingly hard and AWB still fails from time to time

# AWB is not easy



raw-RGB input image before white-balance

Given an arbitrary raw-RGB image, determine what is the camera's response to the illumination.

The idea is that something that is *white*\* is a natural reflector of the scene's illuminations SPD.

So, if we can identify what is "white" in the raw-RGB image, we are observing the sensor's RGB response to the illumination.

\* It doesn't have to be "white", but grey - sometimes we call these scene points "achromatic" or "neutral" regions.

# AWB example

2 commonly used classical AWB algorithms



Input



Gray World

Assumes avg. reflectance of a scene is achromatic (i.e. gray), image average should have  $R=G=B$ .



White Patch

Assumes the brightest spot in the image are from white objects, so max  $R,G,B$  values are estimate of white point.

White-Balance &  
Color Space  
Transform  
(CIE XYZ)

# Improving AWB with Deep Learning

- Lou et al "Color Constancy by Deep Learning", BMVC 2015
- Hu et al "FC4: Fully Convolutional Color Constancy with Confidence-weighted Pooling", CVPR 2017
- Barron "Convolutional Color Constancy", ICCV 2015
- Oh et al "Approaching the computational color constancy as a classification problem through deep learning", Pattern Recognition, 2017
- Afifi et al "Auto White-Balance Correction for Mixed-Illuminant Scenes", WACV 2022.
- **Many many more ...**

# Training data for AWB

## Training images

raw-RGB with a neutral or achromatic object inserted.  
The label/ground-truth for the training data is the raw-RGB value in the region of the neutral object.



Neutral object in the scene. The raw-RGB values at this pixel location is considered to be the entire scenes illumination.  
In the case of the color chart, only the white-patches are used.

## Testing/validation

The neutral object is masked out.



Testing image

Neutral object masked out.



Original image

# Two step procedure

- (1) apply a white-balance correction to the raw-RGB values
- (2) map the white-balanced raw-RGB values to CIE XYZ

White balance

#		
	#	
		#

3x3 diagonal matrix

Color space transform (CST)

#	#	#
#	#	#
#	#	#

3x3 full matrix (or polynomial function)

raw-RGB values



#		
	#	
		#



white-balance raw-RGB



#	#	#
#	#	#
#	#	#



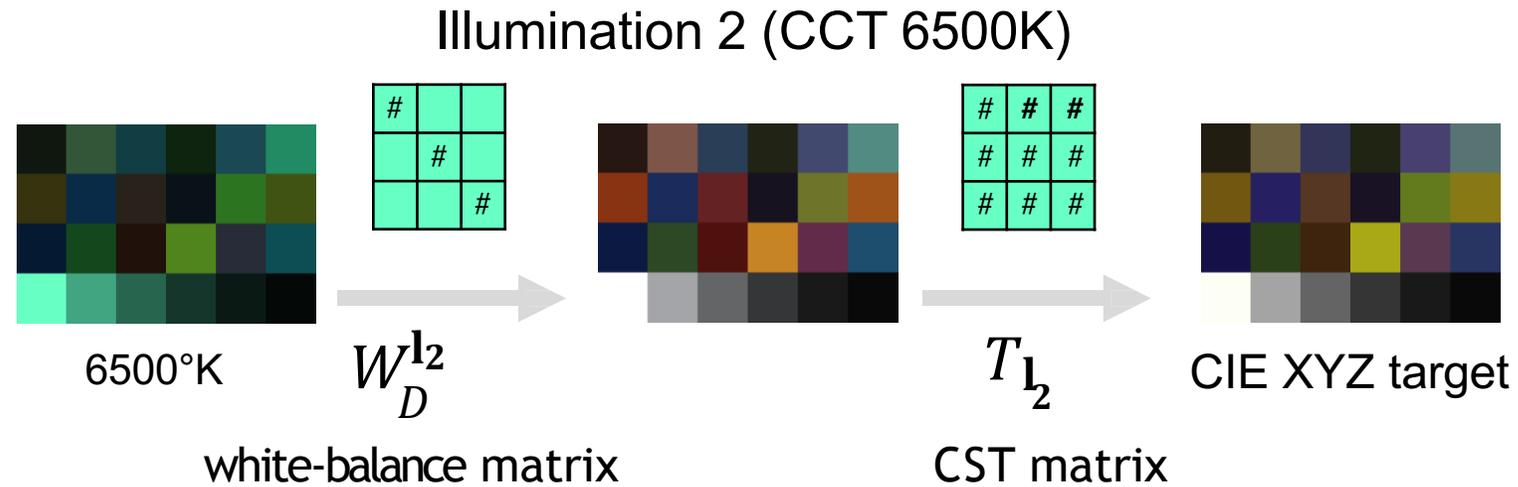
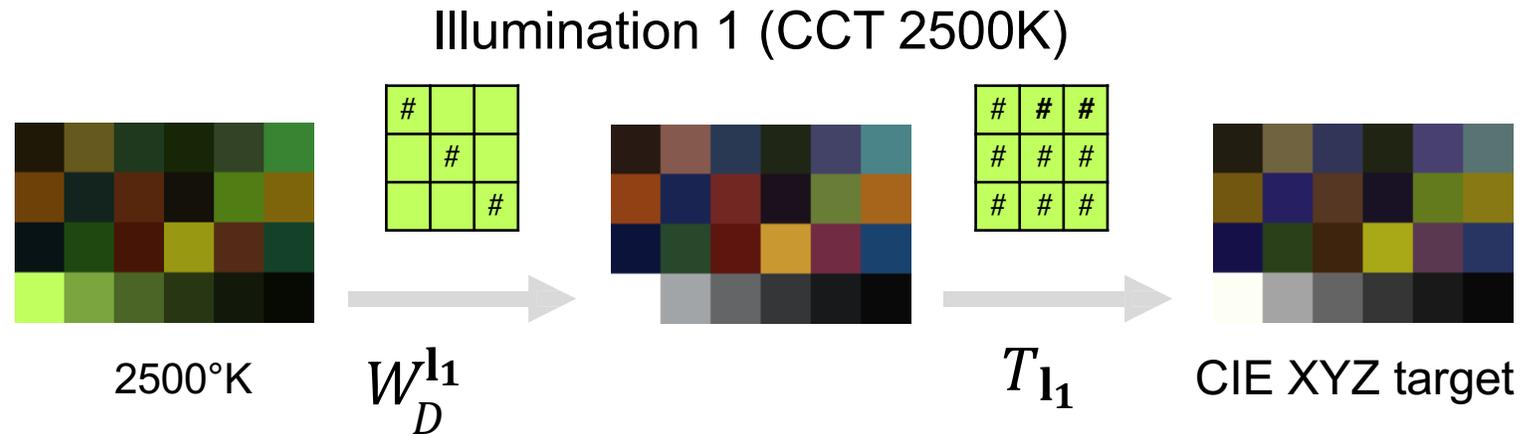
WB-raw-RGB mapped to CIE XYZ



White-Balance &  
Color Space  
Transform  
(CIE XYZ)

# Color space transform

Factory pre-calibration

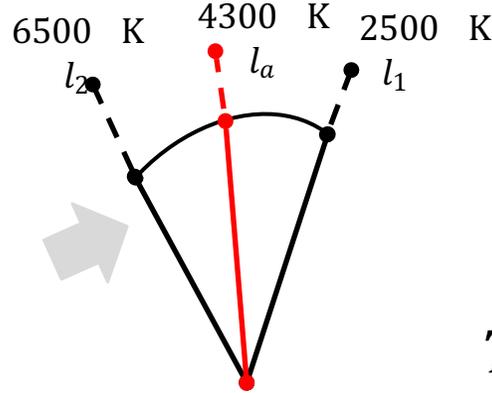
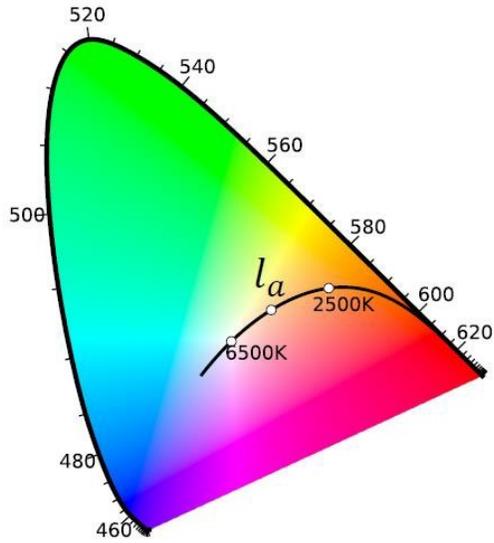


White-Balance &  
Color Space  
Transform  
(CIE XYZ)

CST matrices ( $T_{I_1}$  and  $T_{I_2}$ ) are calibrated for two different illuminations (I1 and I2). Depending on the temperature of the white-balance, we use the corresponding CST.

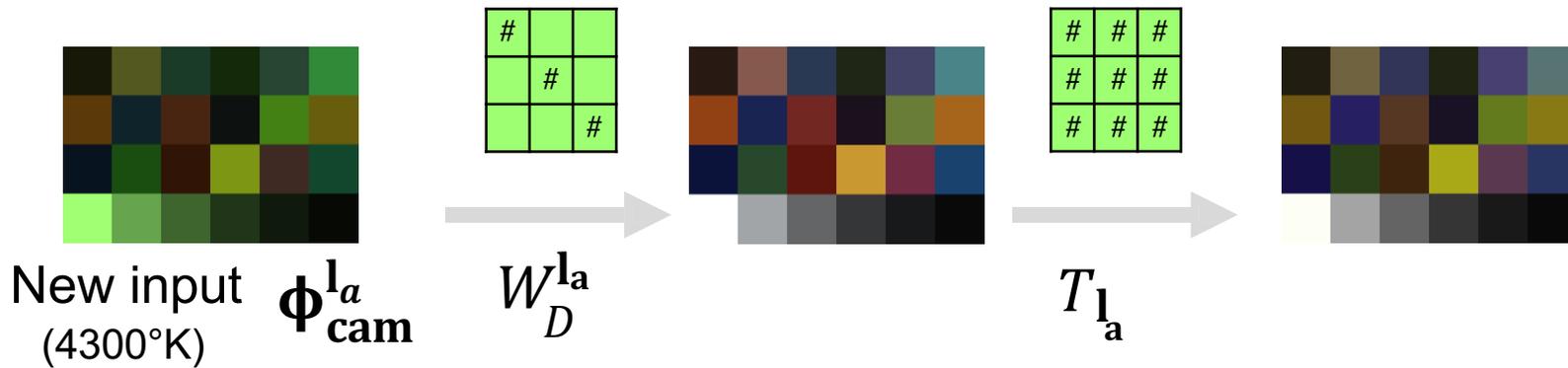
# Color space transform

Interpolation process



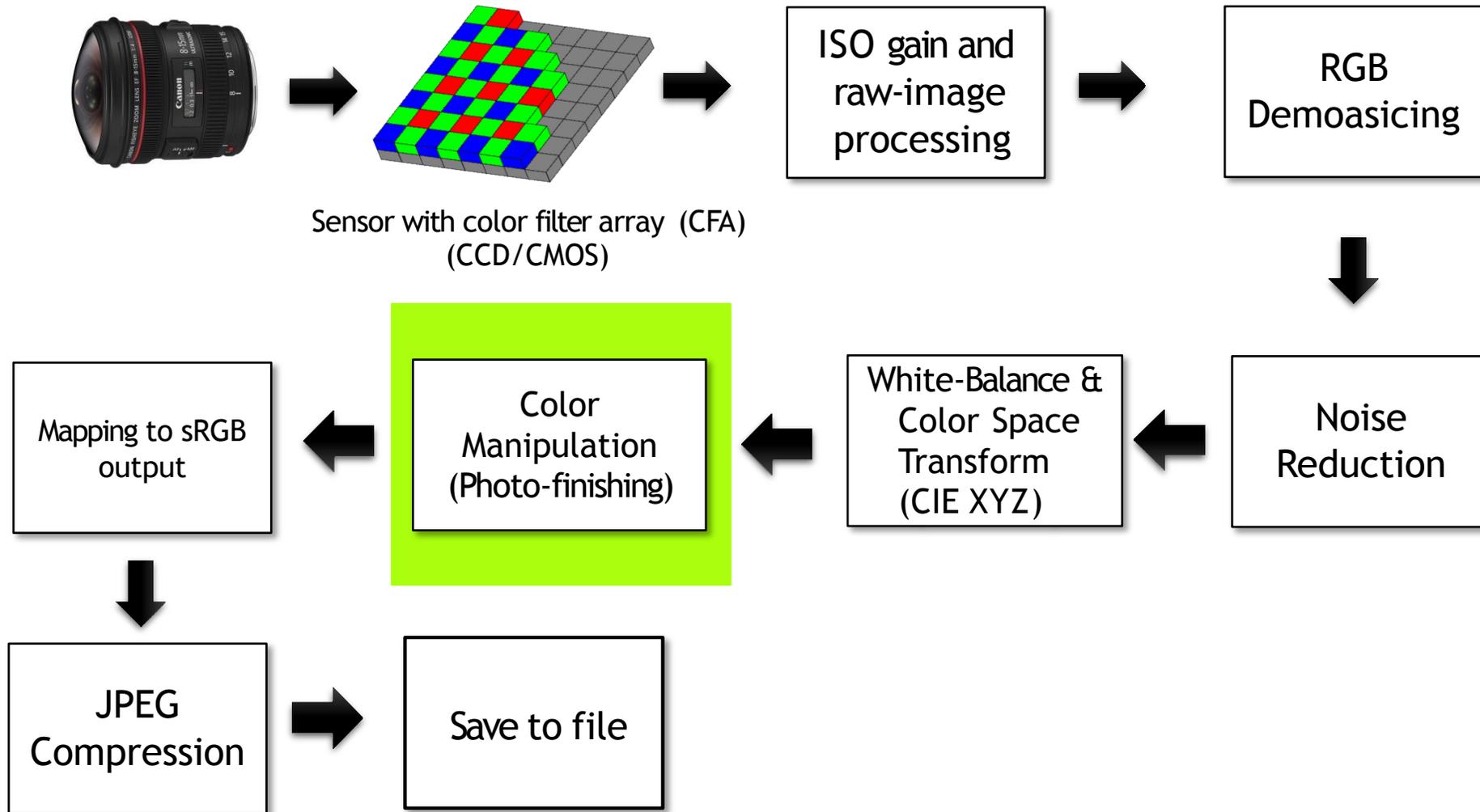
$$g = \frac{CCT_{l_a}^{-1} - CCT_{l_2}^{-1}}{CCT_{l_1}^{-1} - CCT_{l_2}^{-1}}$$

$$T_{l_a} = gT_{l_1} + (1 - g)T_{l_2}$$



Given a new illumination ( $l_a$ ) and its estimated correlated color temperature (CCT), we construct a CST matrix by blending the two factory pre-calibrated matrices.

# Typical color imaging pipeline



# Color manipulation

- This is the stage where a camera applies its "secret sauce" to make the images look good
- This procedure can be called by many names:
  - Color manipulation
  - Photo-finishing
  - Color rendering or selective color rendering
  - Yuv processing engine
- DSLR will often allow the user to select various photo-finishing styles
- Smartphones often compute this per-image
- Photo-finishing may also be tied to geographical regions!

# DSLR "picture" styles

## Standard



Glowing prints with crisp finishes.  
It is the basic color of EOS DIGITAL.

## Portrait



For transparent, healthy skin for women and children

## Landscape



Crisp and impressive reproduction of blue skies and green trees in deep, vivid color

## Neutral



Subjects are recorded in rich detail, giving the greatest latitude for image processing

## Faithful



Accurate recording of the subject's color, close to the actual image seen with the naked eye

## Monochrome



Filter work and sepia tone with the freedom of digital monochrome

From Canon's user manual

# Picture styles



Color  
Manipulation  
(Photo-finishing)

Example of four different picture styles from Nikon  
This image is the **same** raw-RGB image processed in four different ways.

# Photorealistic Stylization



(a) Inputs



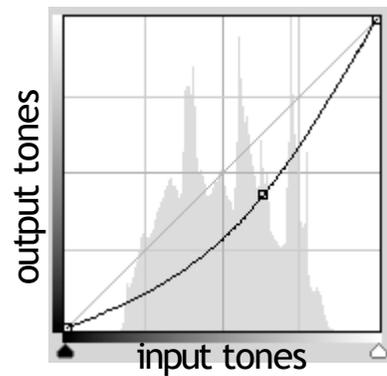
(d) Ours (WCT<sup>2</sup>)

“Photorealistic Style Transfer via Wavelet Transforms”, ICCV 2019.

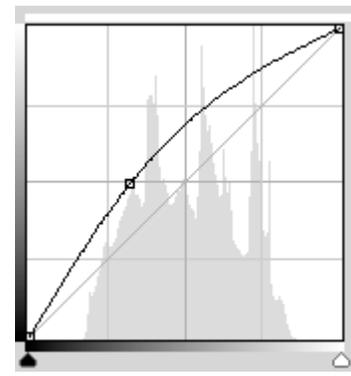
# Global tone map example (1D LUT)



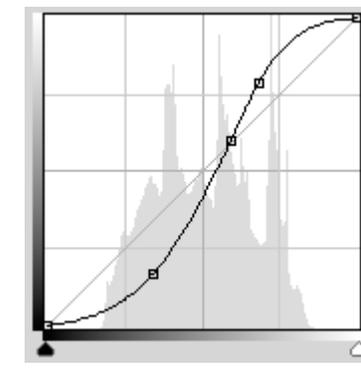
Input



Darkening the image



Brightening the image



Enhancing contrast (called an S-curve)

# Local tone mapping (LTM)

Global tone-mapping  
Camera mode - Manual

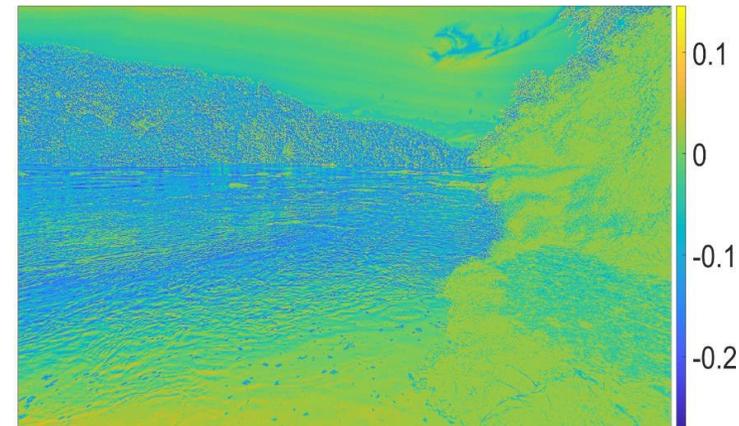


Local tone-mapping  
Camera mode - Auto



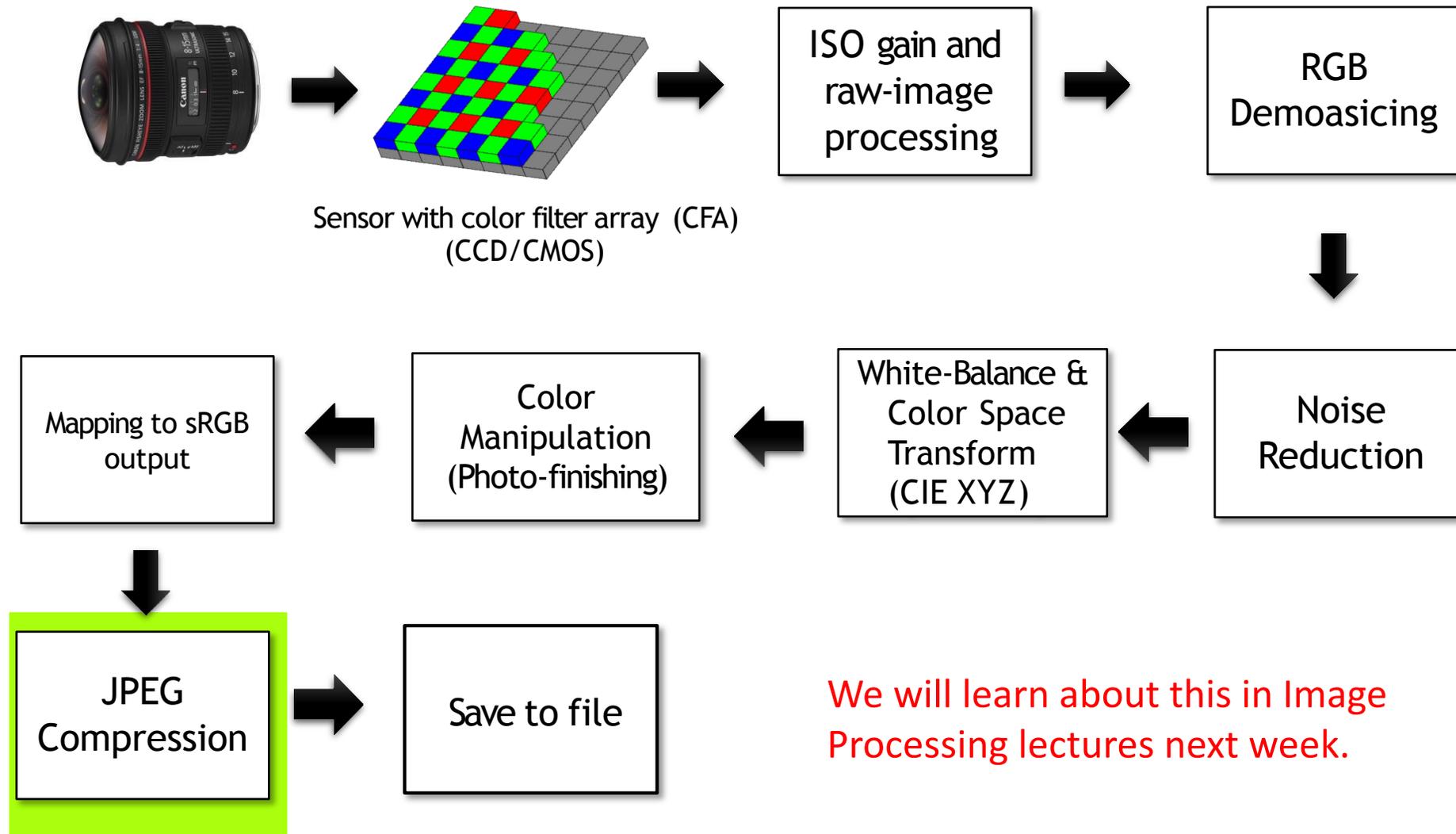
**NOTE:** On many cameras, esp smartphones, a local tone map is applied as part of the photo-finishing. This helps bring out highlights in the image.

Color  
Manipulation  
(Photo-finishing)

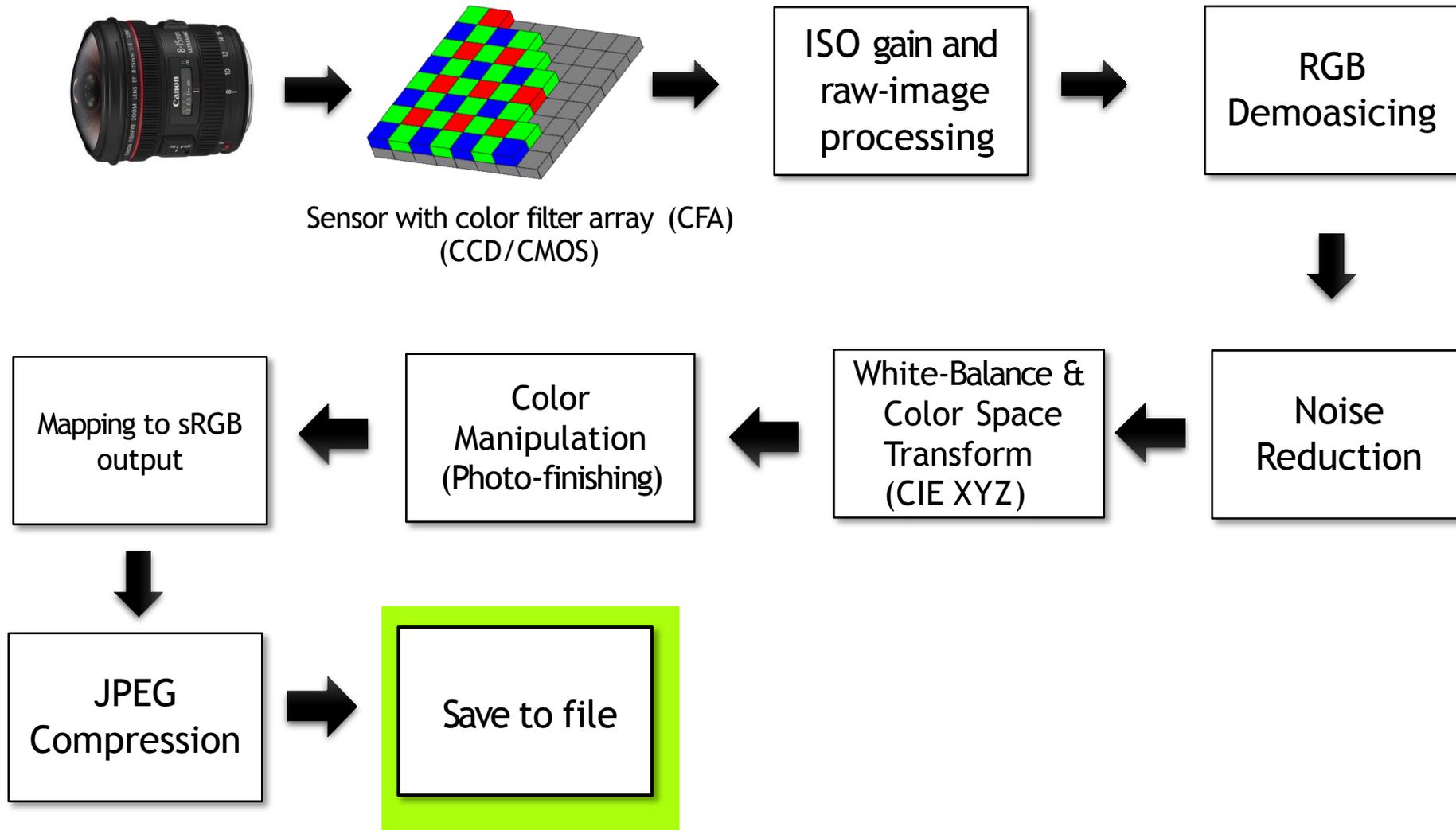


Difference map between image before and after LTM

# Typical color imaging pipeline



# Save to storage and we are done!



# Exif metadata

- Exchangeable image file format (Exif)
- Associates meta data with images
  - Date/time
  - Camera settings (basic)
    - Image size, aperture, shutter speed, focal length, ISO speed, metering mode (how exposure was estimated)
  - Additional info (from in some Exif files)
    - White-balance settings, even matrix coefficients of white-balance
    - Picture style (e.g. landscape, vivid, standard, portrait)
    - Output color space (e.g. sRGB, Adobe RGB, RAW)
    - GPS info
    - More ...

# EXIF as Language: Learning Cross-Modal Associations Between Images and Camera Metadata

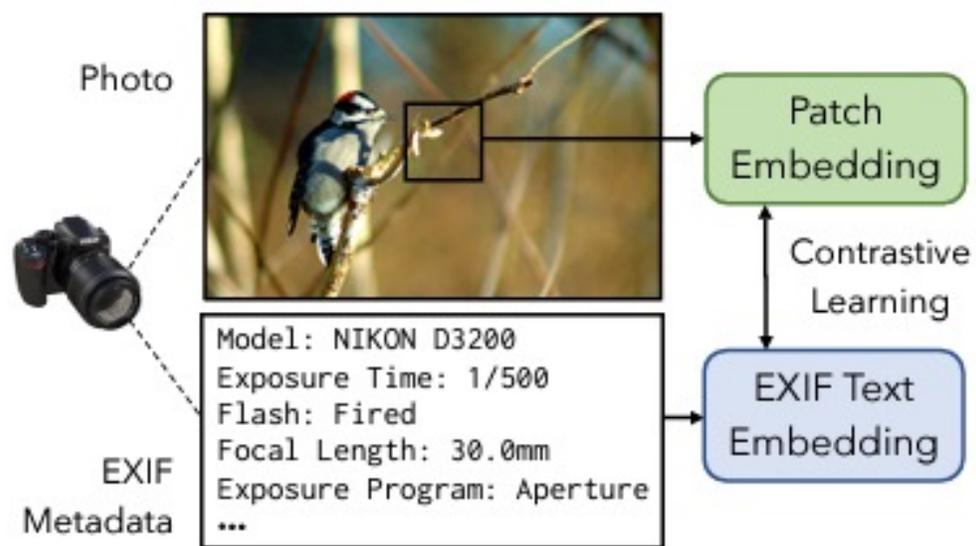
Chenhao Zheng

Ayush Shrivastava

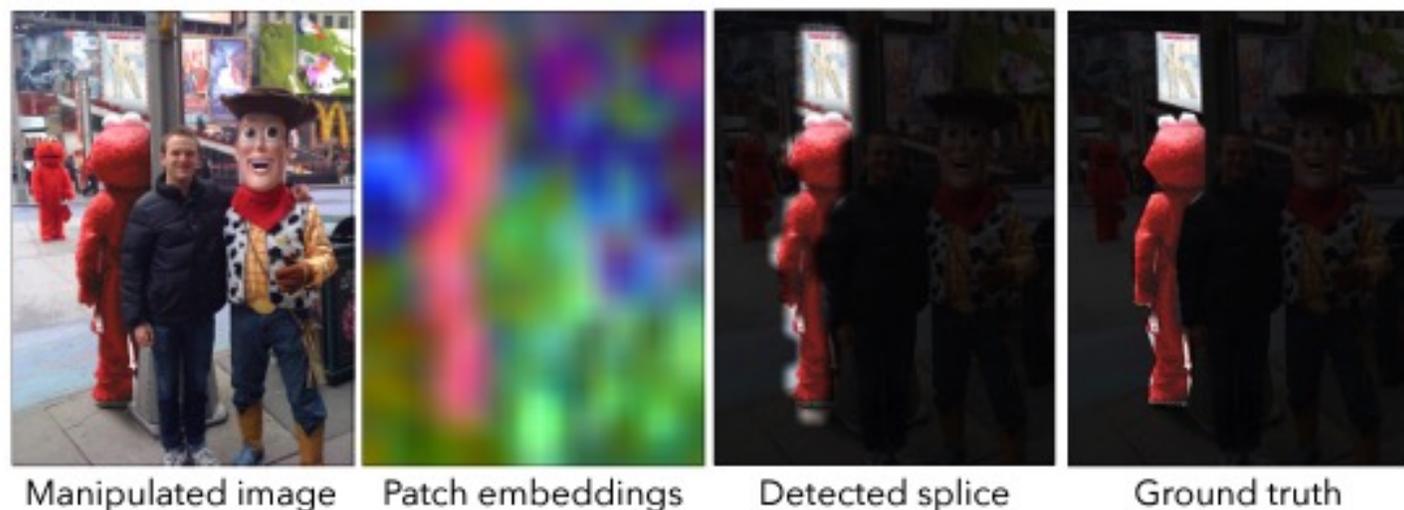
Andrew Owens

University of Michigan

<https://hellomuffin.github.io/exif-as-language>



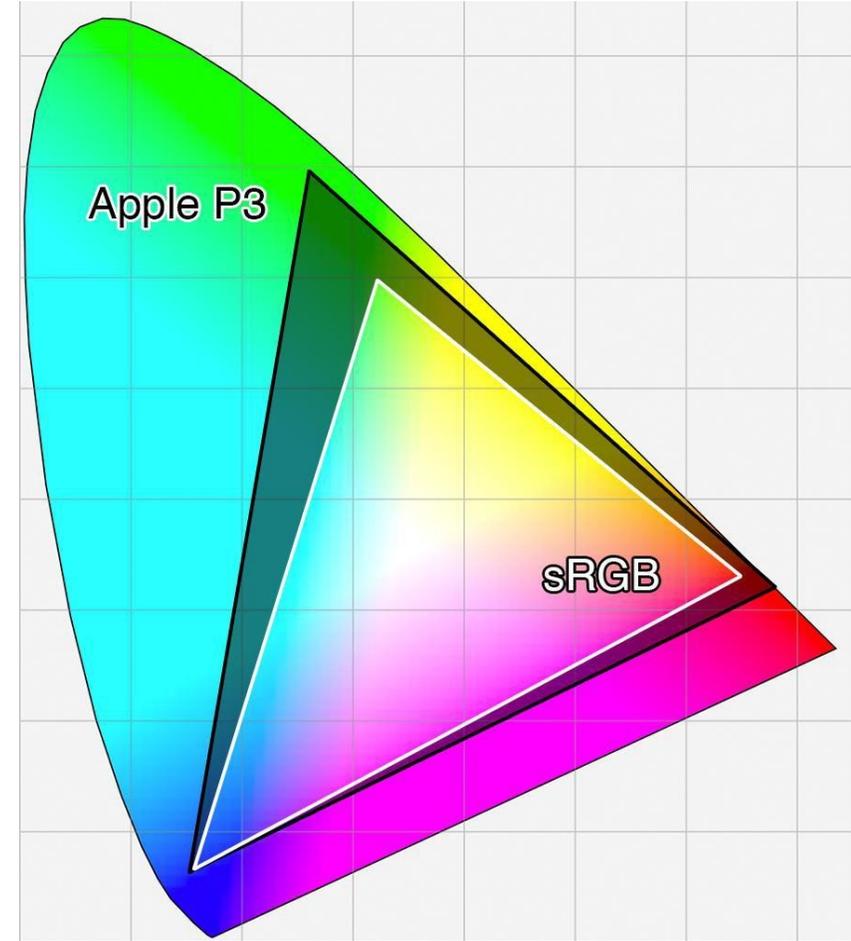
(a) Multimodal image-metadata embeddings



(b) Detecting spliced images by spotting inconsistent patch embeddings

# Note: sRGB/JPEG is slowly being replaced

- sRGB was developed for monitors in the 1990s - it is an old standard.
- **High Efficient Image Encoding (HEIC)**
  - Better compression than JPEG
- **Apple iPhone has started to use HEIC to replace JPEG**
- HEIC supports multiple color spaces. Apple uses Display P3 - a variation on a Digital Cinema Initiative P3 space.
- The P3 gamut is 25% wider than sRGB
- Pixel 4 and other Android devices are will support this color space soon.

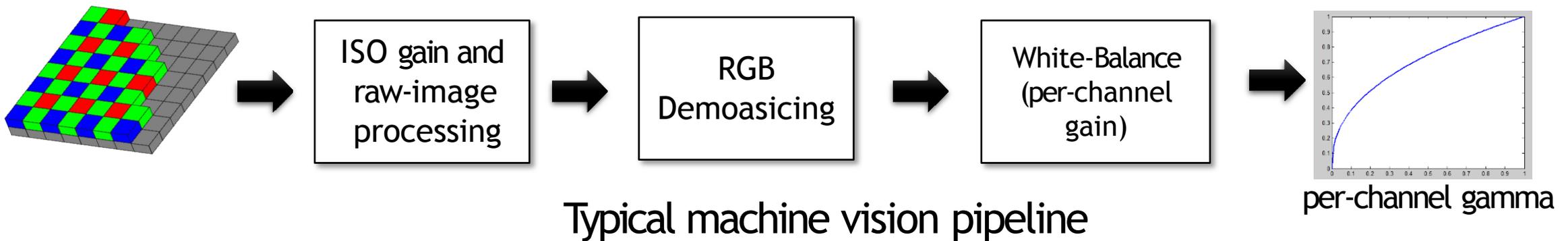


# Pipeline comments

- Again, important to stress that the exact steps mentioned in these notes only serve as a guide of what takes place in a camera
- Modern pipelines are more complex, however, you will find steps similar to what was described
- Note: for different camera makes/models, the operations could be performed in different order (e.g. white-balance after demosaicing) and in different ways (e.g. combining sharpening with demosaicing)

# What about machine vision cameras?

- Some industrial/machine vision cameras provide minimal ISP processing
- For example, some will only perform white-balance and apply a gamma to the raw-RGB values.
- This means the output is in a camera-specific color space

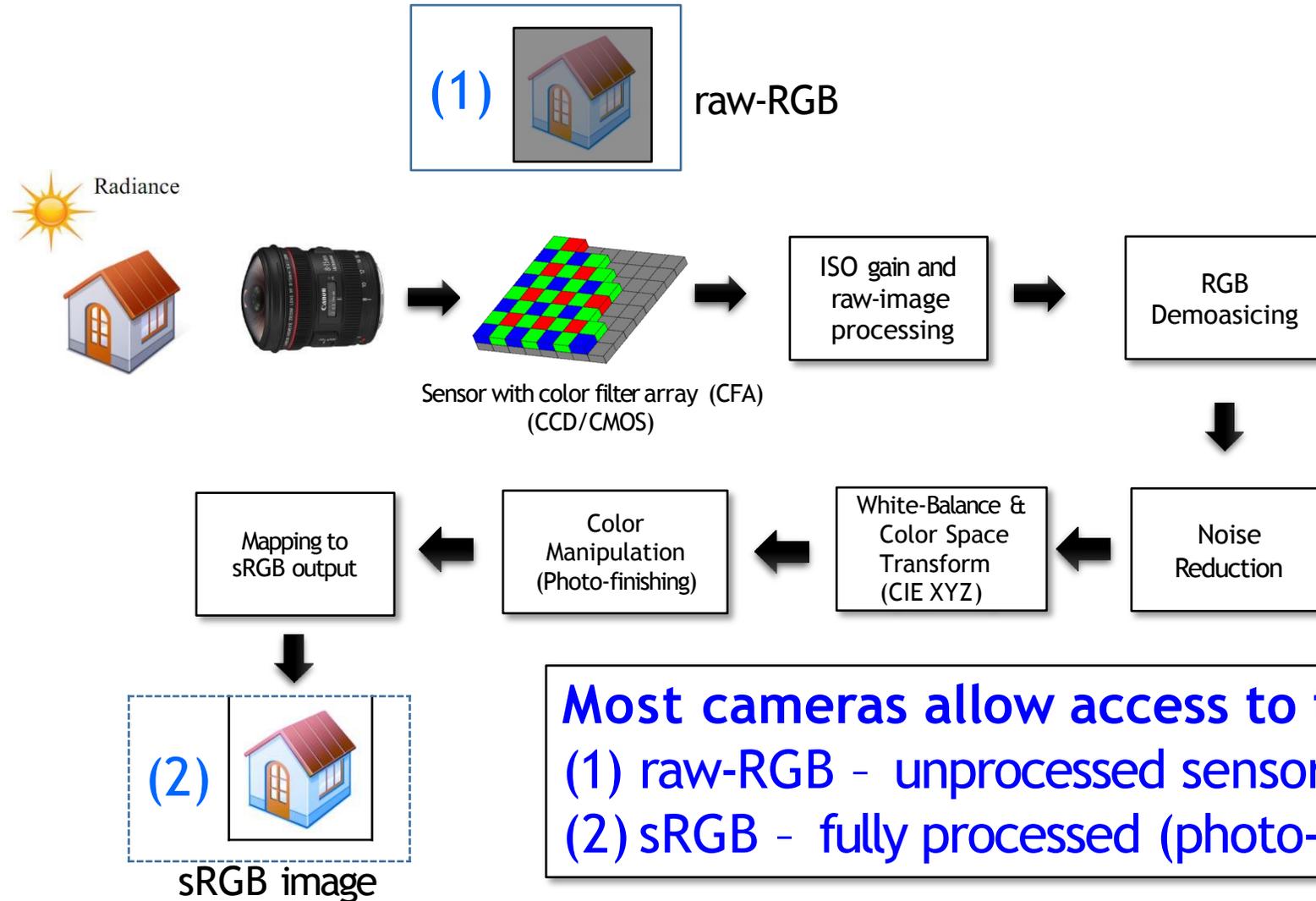


Point grey grasshopper camera.

# Mistake: Working in the wrong color space

- This problem stems from a lack of understanding of color spaces
- You can't just say "RGB", you need to specify **which** RGB (e.g. , raw-RGB, sRGB, NTSC, P3, etc). If you got the image off the web, it is highly likely to be sRGB.
- Certain color spaces are more suitable than others depending on the application.

# Current access to different image states



**Most cameras allow access to two image states.**  
**(1) raw-RGB - unprocessed sensor image**  
**(2) sRGB - fully processed (photo-finished) image**

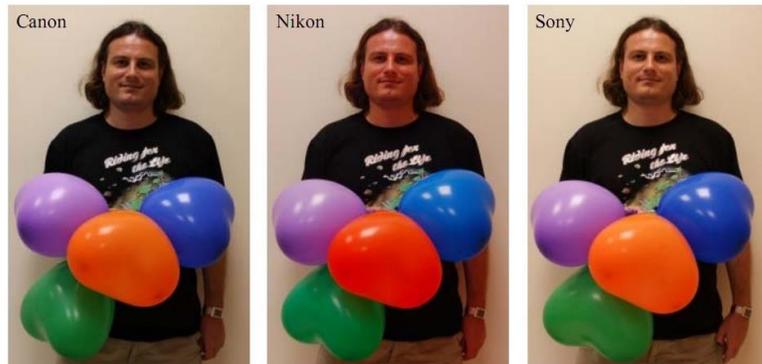
# Advantage of raw-RGB

- It is linear with respect to the scene's physical light. This means the pixel values are 'sort-of' related to the scene.
- Certain Computer Vision problems make this assumption
  - Shape from shading (photometric stereo)
  - Intrinsic image decomposition
  - Image deblurring
  - However, many times these problems are attempted in the sRGB space!

# sRGB images

- When you are working with an sRGB image, it represents the photo-finished image
- The colors represent the colors of the "photograph", not the colors of the "scene"
- These colors will be different for different cameras or even same camera with different picture styles!

sRGB output on different cameras.



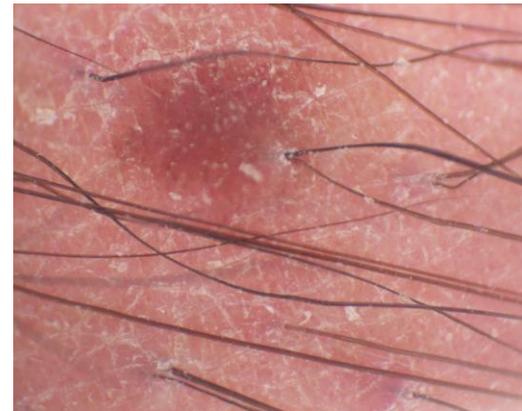
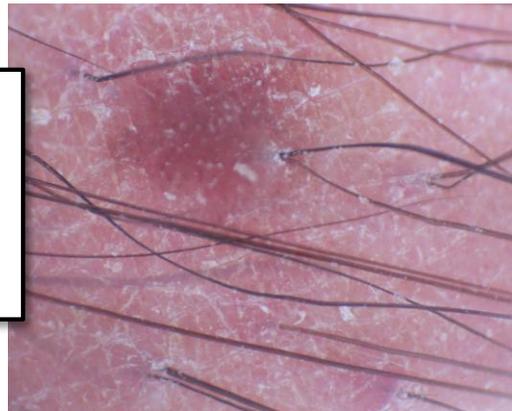
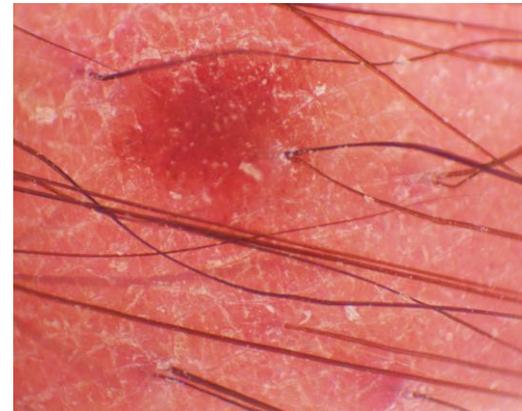
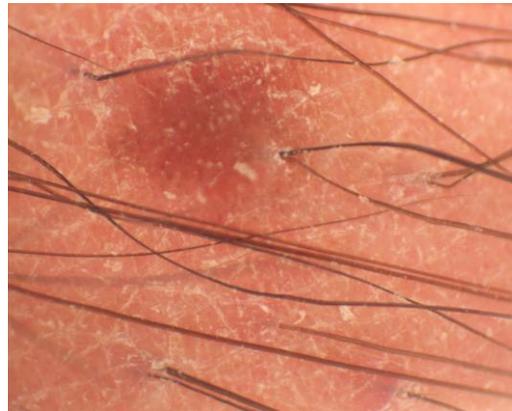
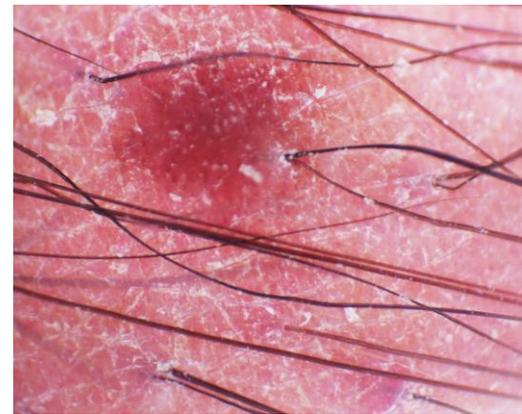
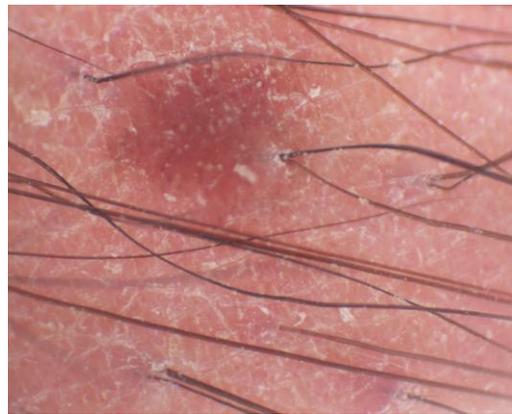
sRGB output on the same camera, w/ different color manipulation.



This assumption can lead to serious mistakes in a computer vision application.



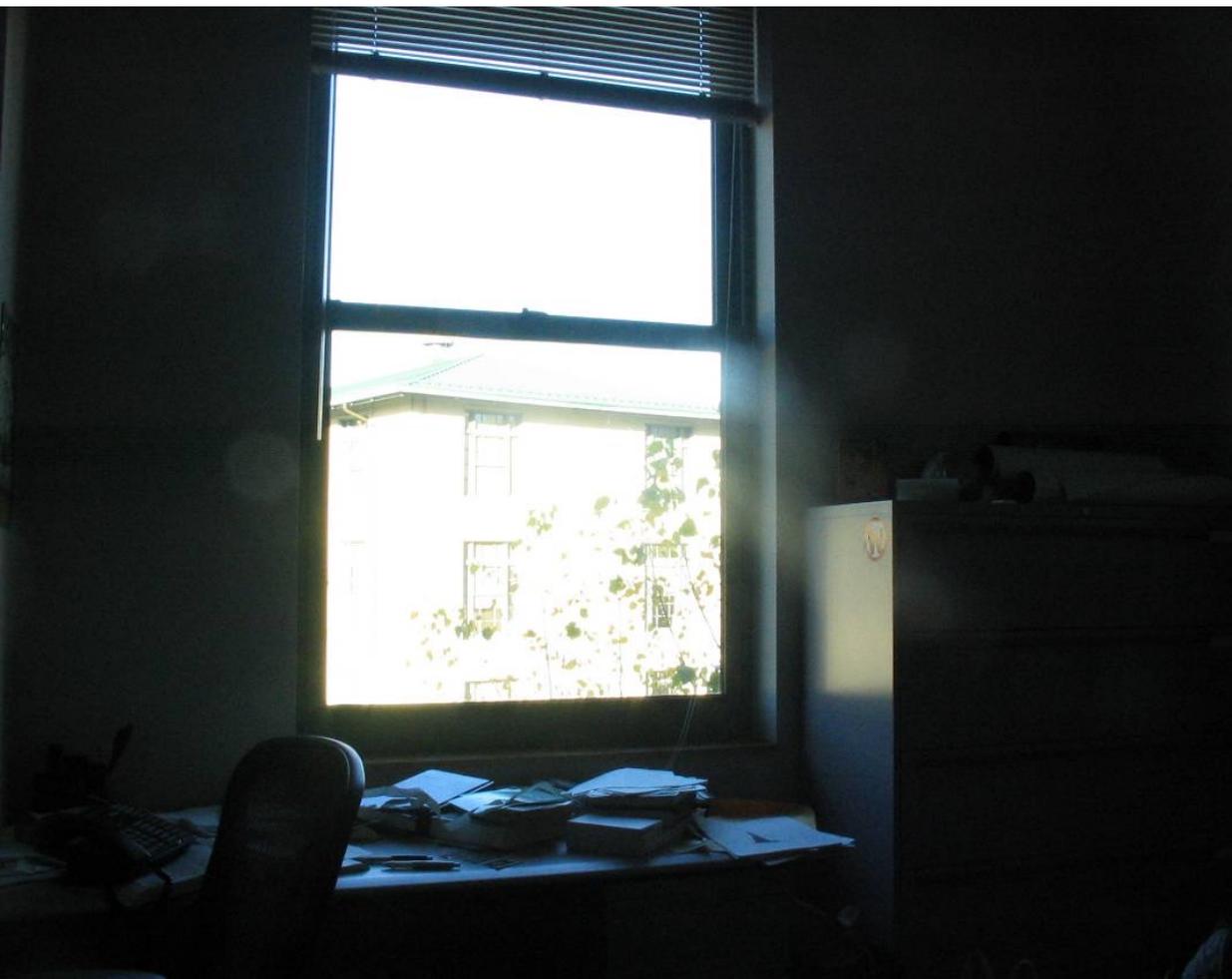
**This type of processing is not suitable for scientific applications!**



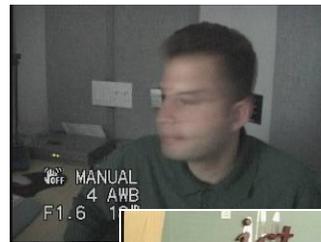
**Which one is correct?**

# HDR Imaging

# Why HDR?



# Problem: Dynamic Range



1



1500



25,000



400,000



2,000,000,000

The real world is high dynamic range.

High dynamic range imaging



HYATT

GARDNER PLAZA





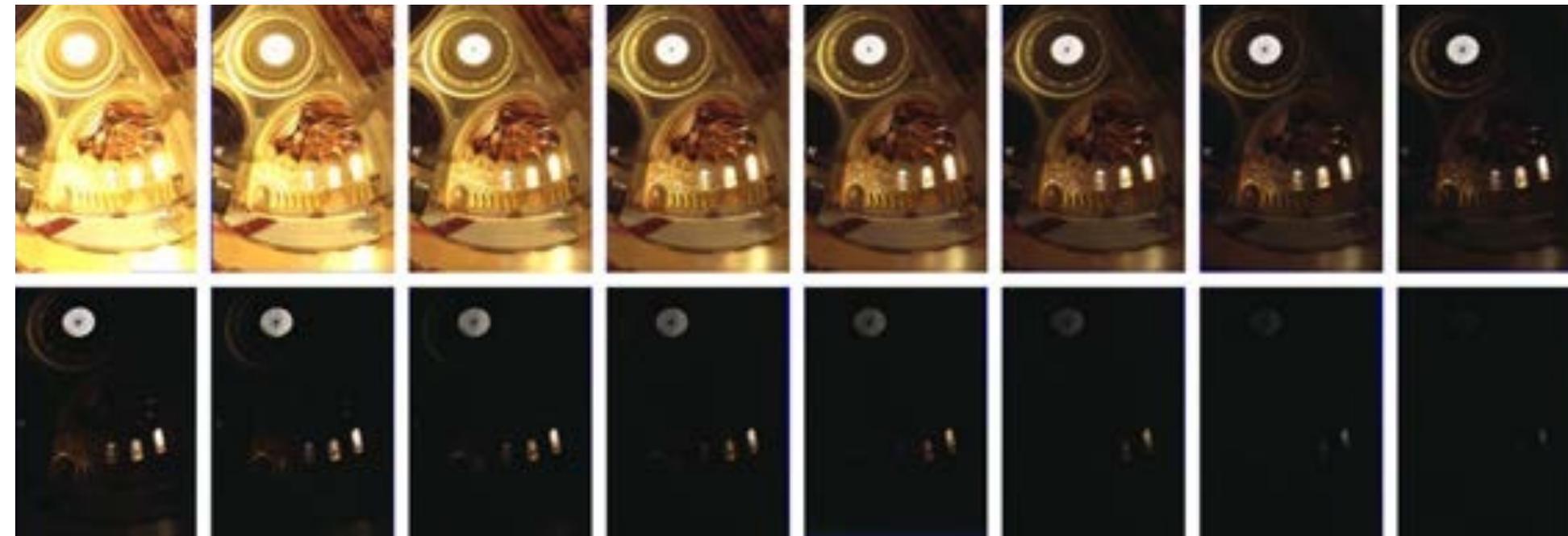




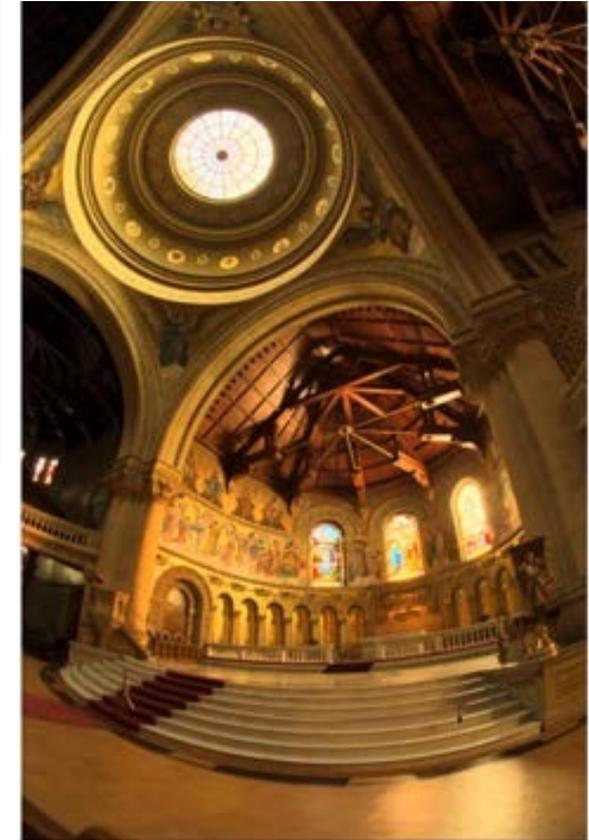


# Key idea

1. Exposure bracketing: Capture multiple LDR images at different exposures

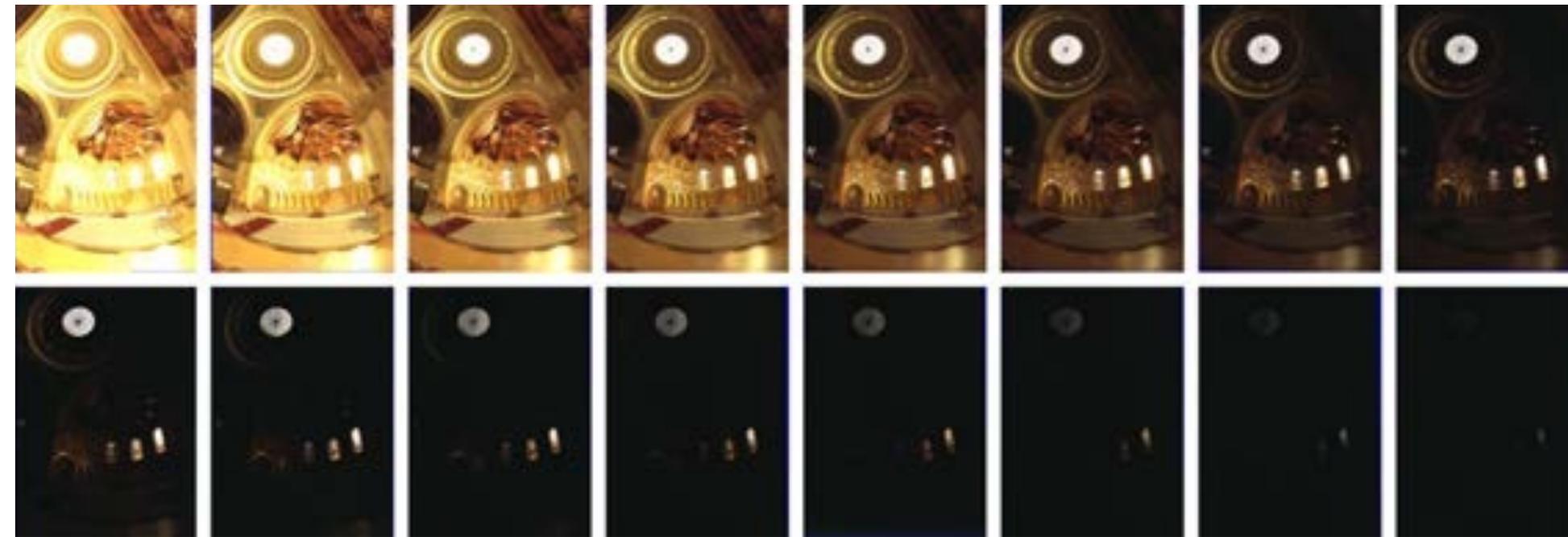


2. Merging: Combine them into a single HDR image

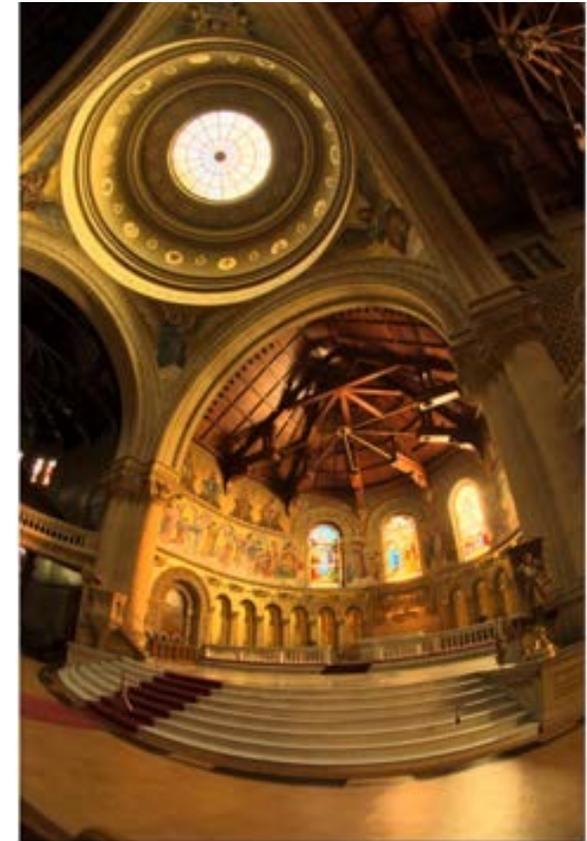


# Key idea

1. Exposure bracketing: Capture multiple LDR images at different exposures

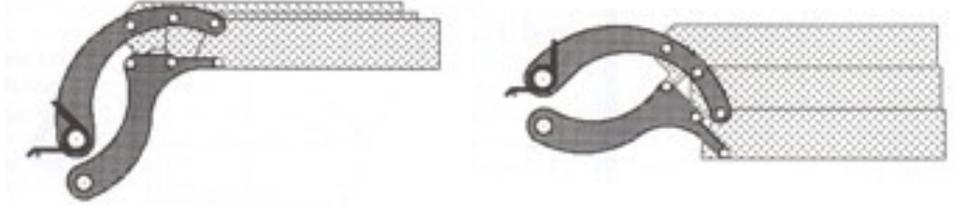


2. Merging: Combine them into a single HDR image

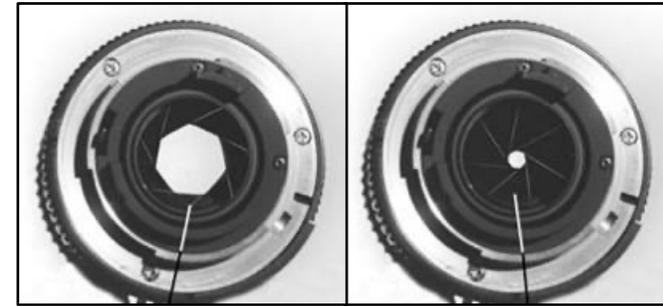


# Ways to vary exposure

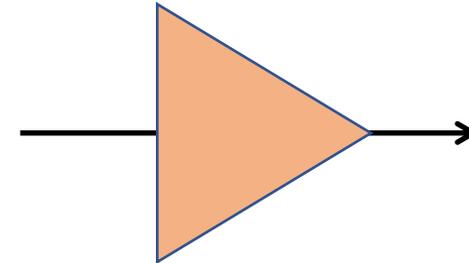
1. Shutter speed



2. F-stop (aperture, iris)



3. ISO



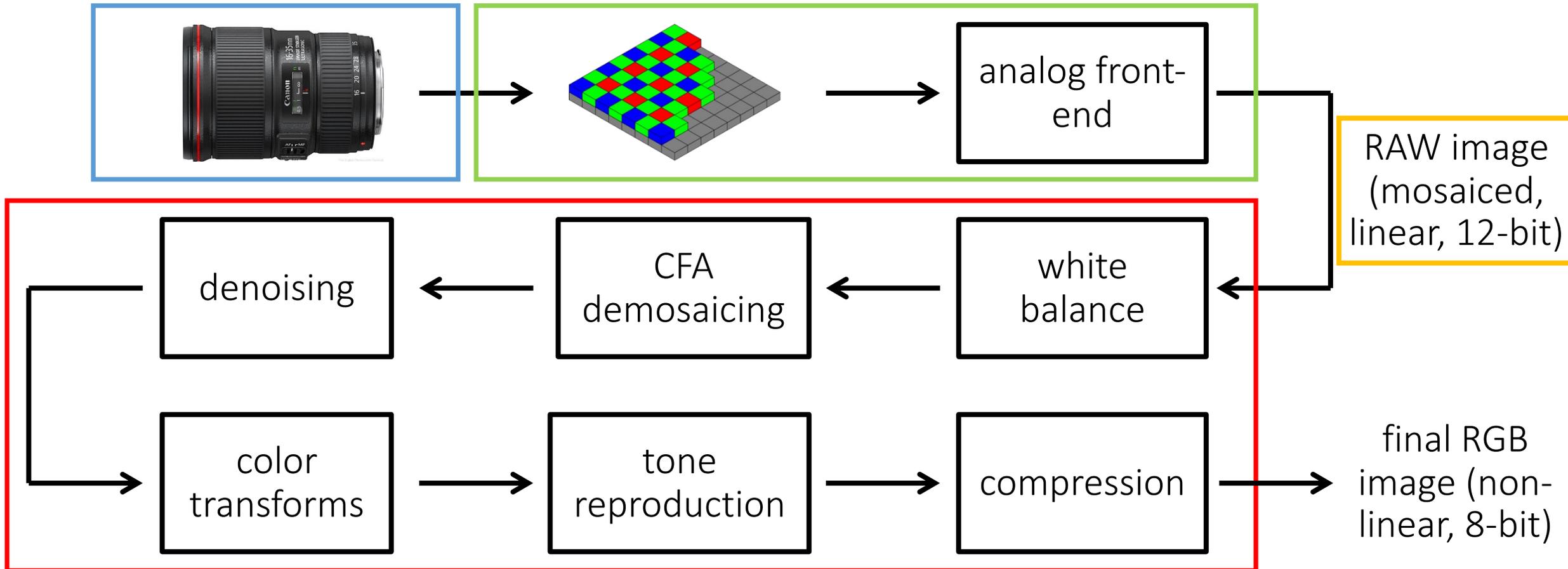
4. Neutral density (ND) filters



Pros and cons of each for HDR?

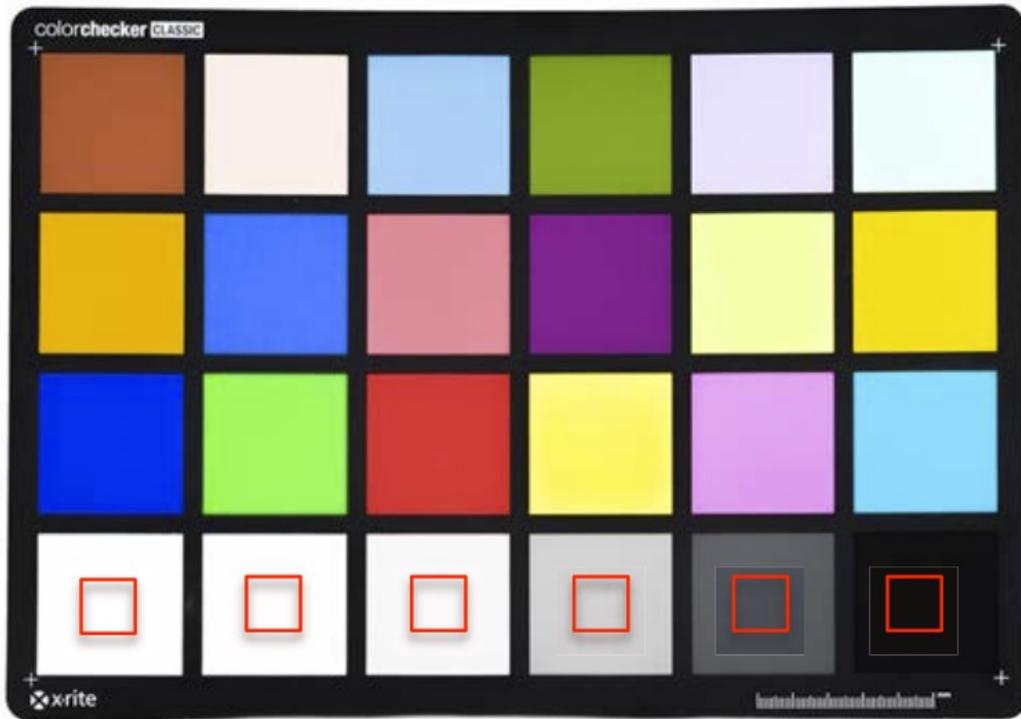
# The image processing pipeline

The sequence of image processing operations applied by the camera's image signal processor (ISP) to convert a RAW image into a "conventional" image.



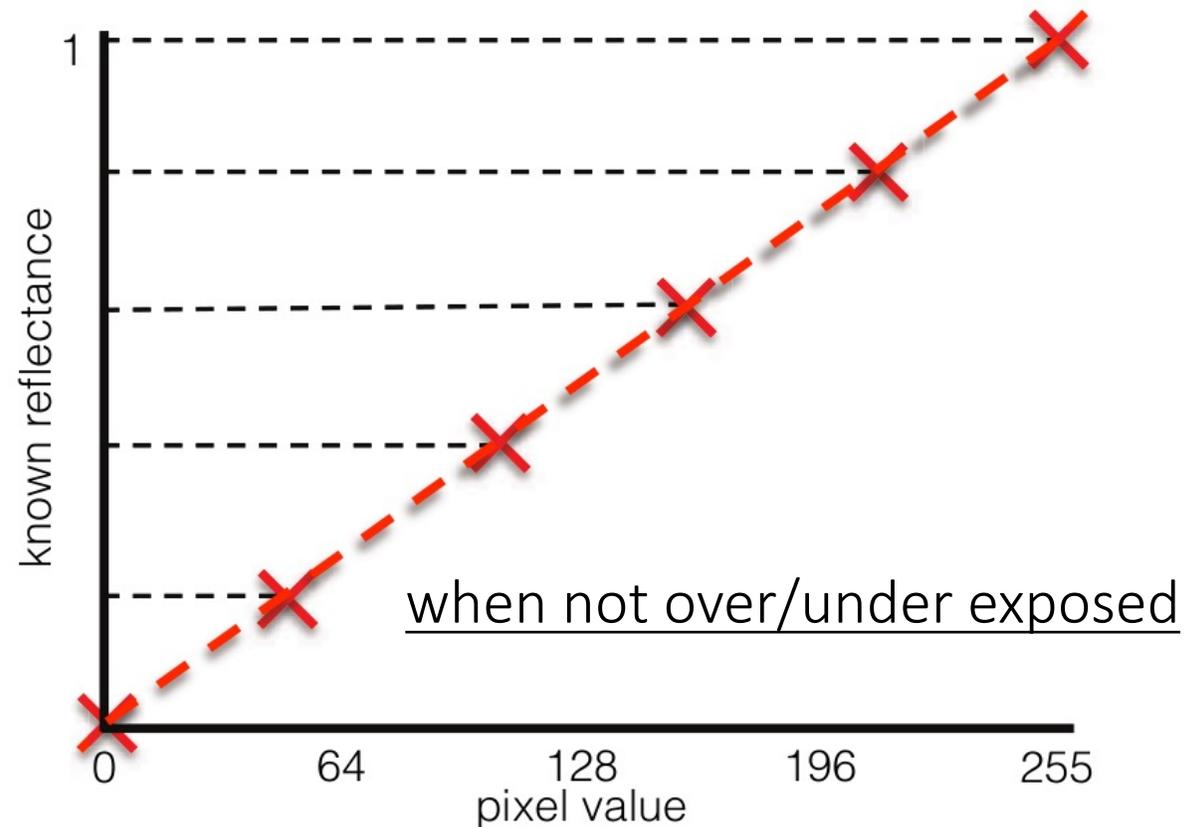
# RAW images have a linear response curve

Colorchecker: Great tool for radiometric and color calibration.



Patches at bottom row have reflectance that increases linearly.

linear RAW



# RAW (linear) image formation model

Real scene radiance for image pixel  $(x,y)$ :  $L(x, y)$

Exposure time:

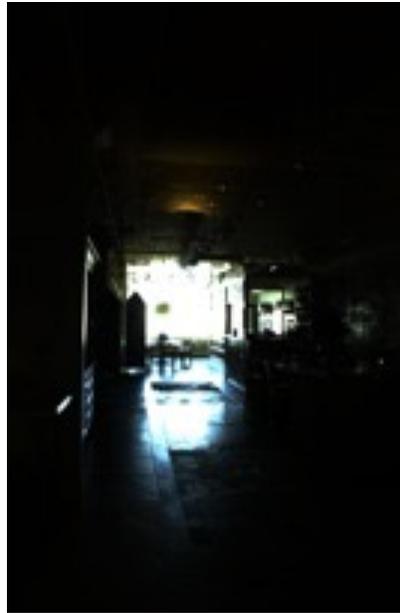
$t_5$



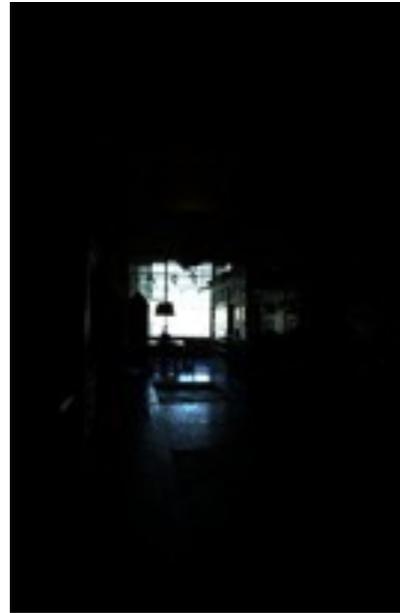
$t_4$



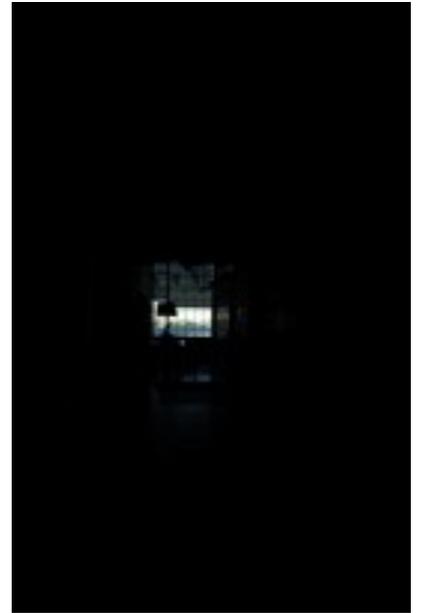
$t_3$



$t_2$



$t_1$



What is an expression for the image  $I_{\text{linear}}(x,y)$  as a function of  $L(x,y)$ ?

# RAW (linear) image formation model

Real scene radiance for image pixel  $(x,y)$ :  $L(x, y)$

Exposure time:

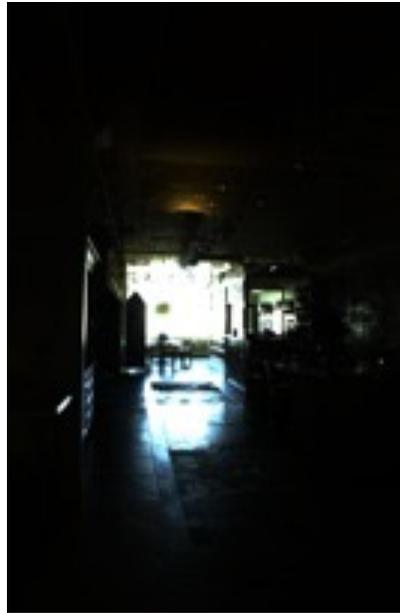
$t_5$



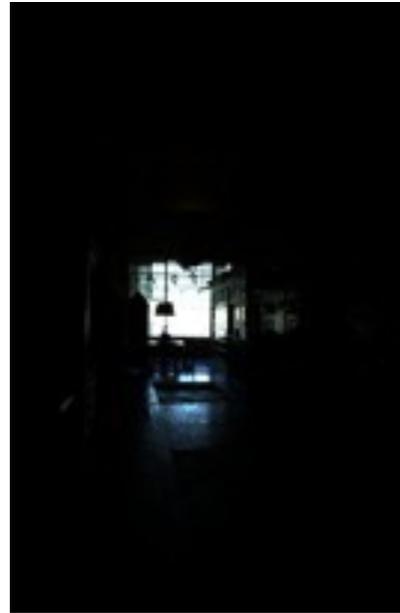
$t_4$



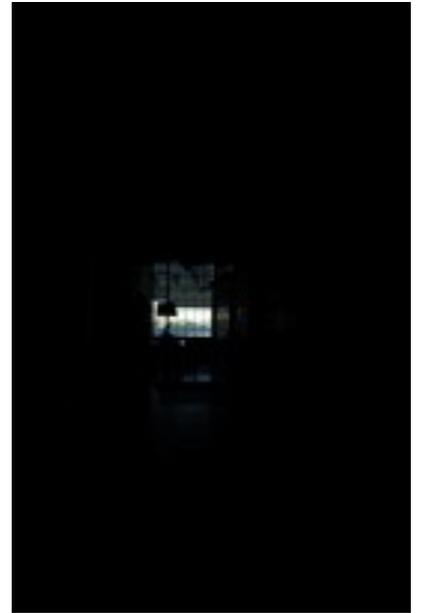
$t_3$



$t_2$



$t_1$



What is an expression for the image  $I_{\text{linear}}(x,y)$  as a function of  $L(x,y)$ ?

$$I_{\text{linear}}(x,y) = \text{clip}[ t_i \cdot L(x,y) + \text{noise} ]$$

How would you merge these images into an HDR one?

# Merging RAW (linear) exposure stacks

For each pixel:

1. Find “valid” images
2. Weight valid pixel values appropriately
3. Form a new pixel value as the weighted average of valid pixel values

How would you implement steps 1-2?

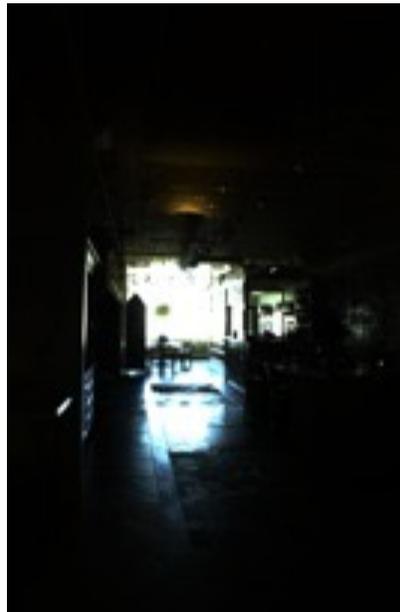
$t_5$



$t_4$



$t_3$



$t_2$



$t_1$



# Merging RAW (linear) exposure stacks

For each pixel:

1. Find “valid” images

← (noise)  $0.05 < \text{pixel} < 0.95$  (clipping)

2. Weight valid pixel values appropriately

● noise

● valid

3. Form a new pixel value as the weighted average of valid pixel values

● clipped

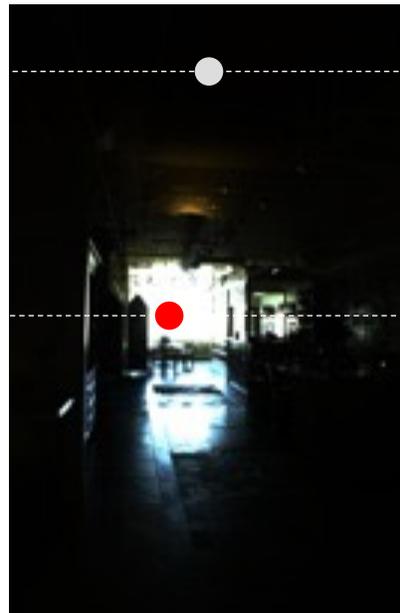
$t_5$



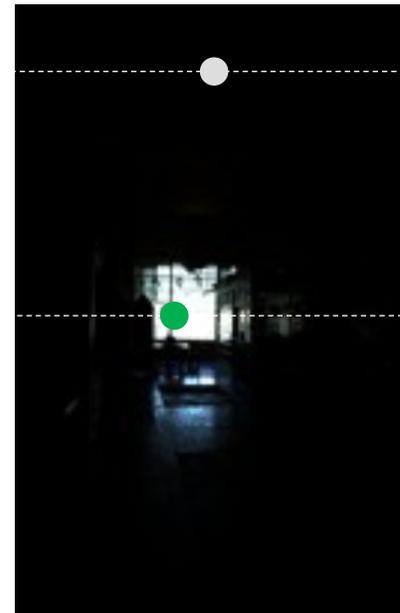
$t_4$



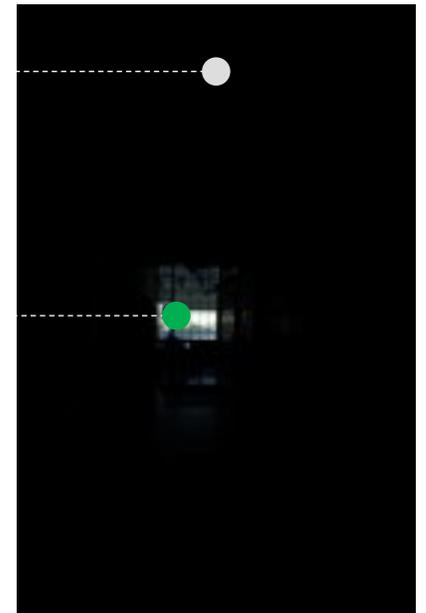
$t_3$



$t_2$



$t_1$



# Merging RAW (linear) exposure stacks

For each pixel:

1. Find “valid” images ← (noise)  $0.05 < \text{pixel} < 0.95$  (clipping)
2. Weight valid pixel values appropriately ← (pixel value) /  $t_i$
3. Form a new pixel value as the weighted average of valid pixel values

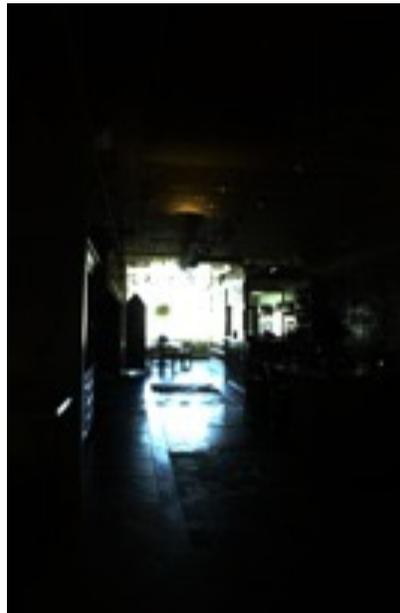
$t_5$



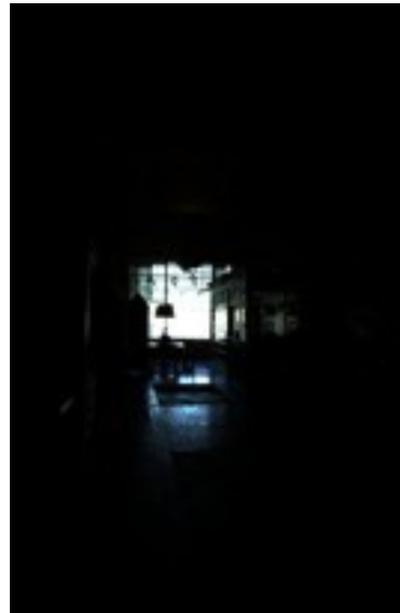
$t_4$



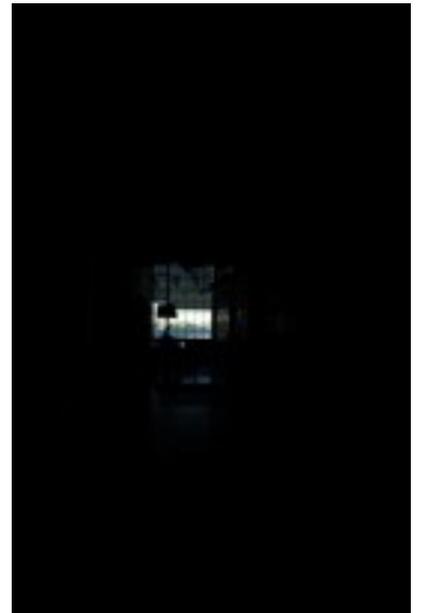
$t_3$



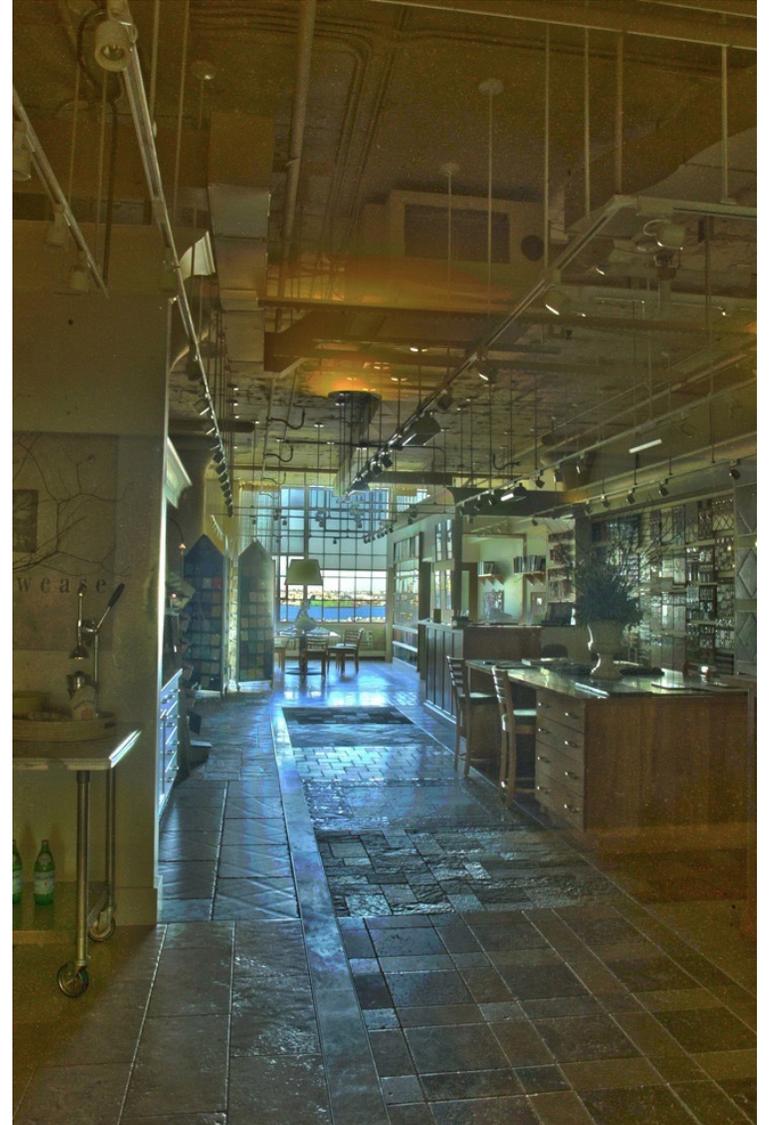
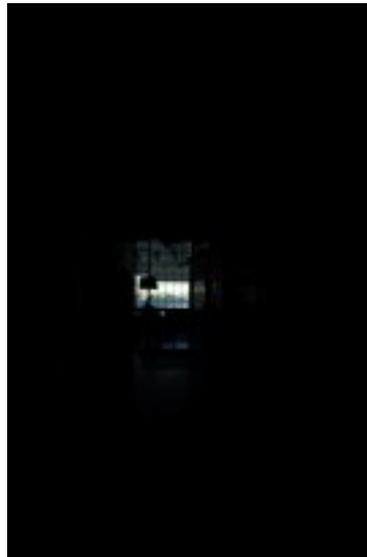
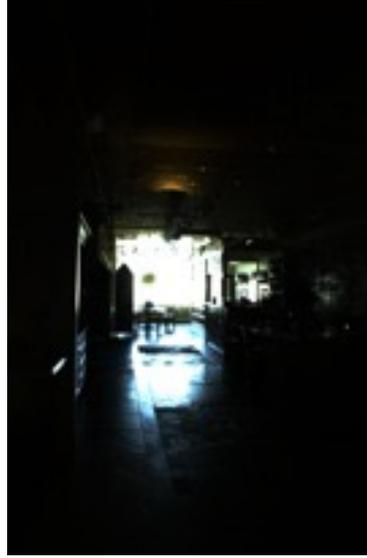
$t_2$



$t_1$

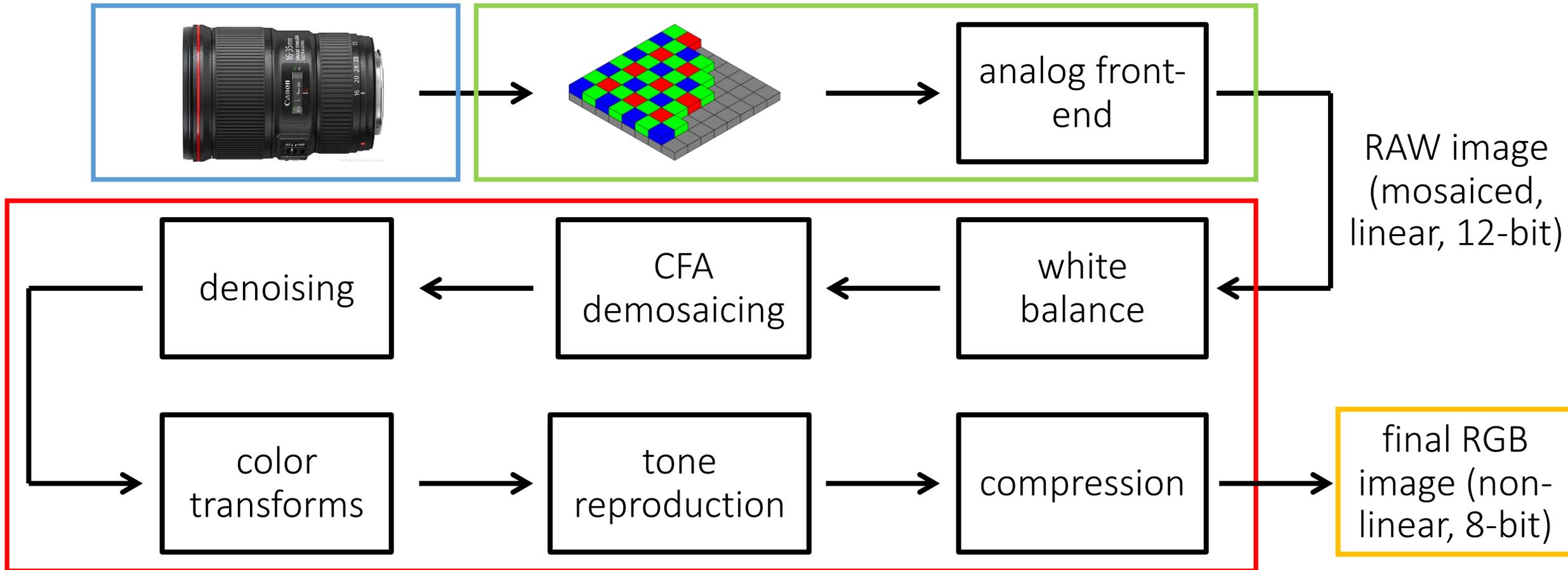


# Merging result (after tonemapping)



# The image processing pipeline

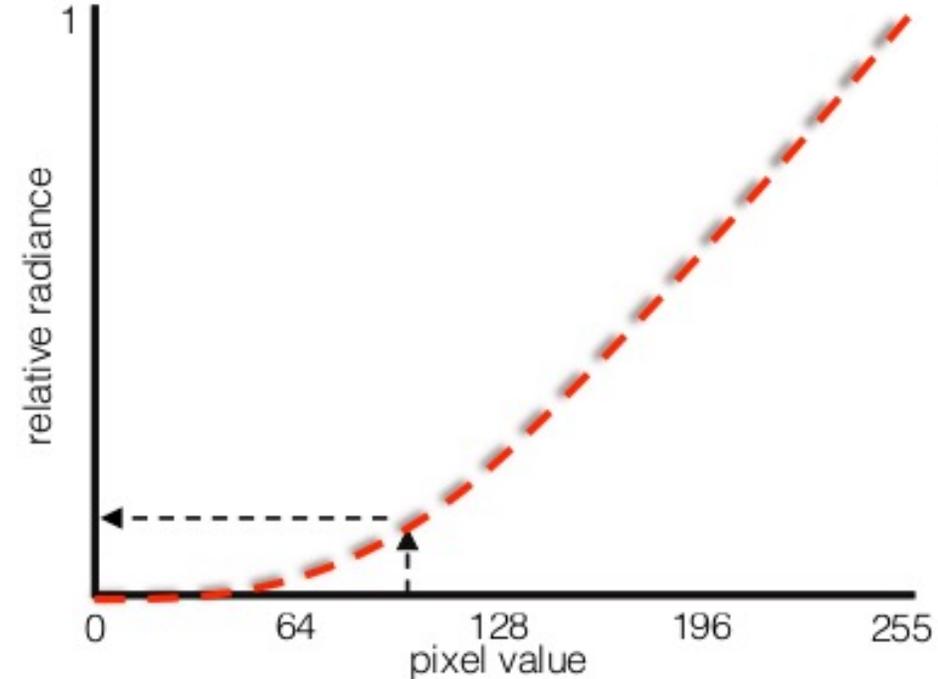
- Can you foresee any problem when we switch from RAW to rendered images?
- How do we deal with the nonlinearities?



# Non-linear image formation model

Real scene radiance for image pixel  $(x,y)$ :  $L(x, y)$

Exposure time:  $t_i$



$$I_{\text{linear}}(x,y) = \text{clip}[ t_i \cdot L(x,y) + \text{noise} ]$$

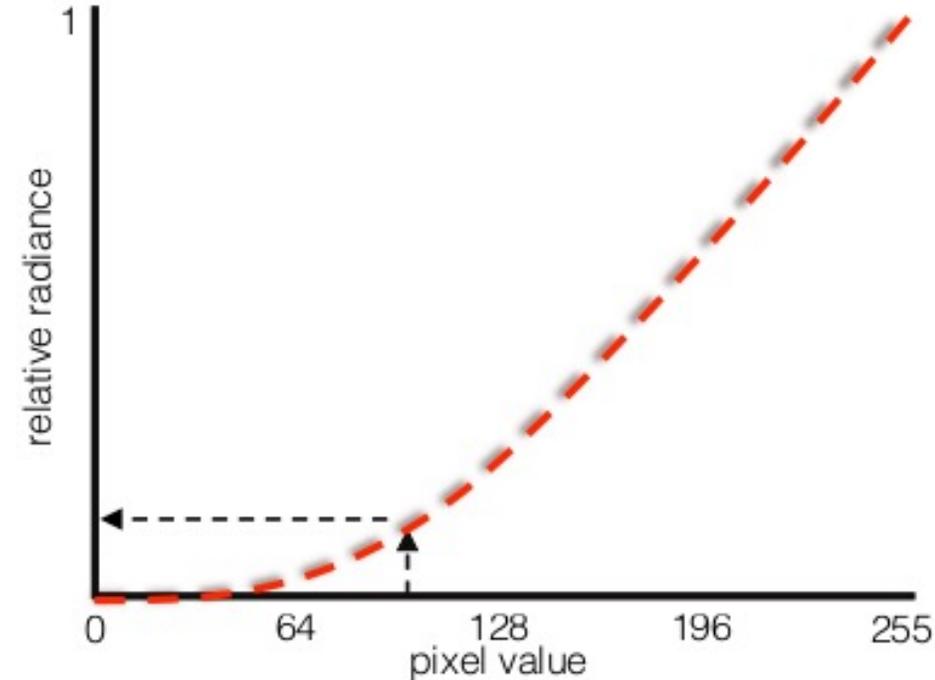
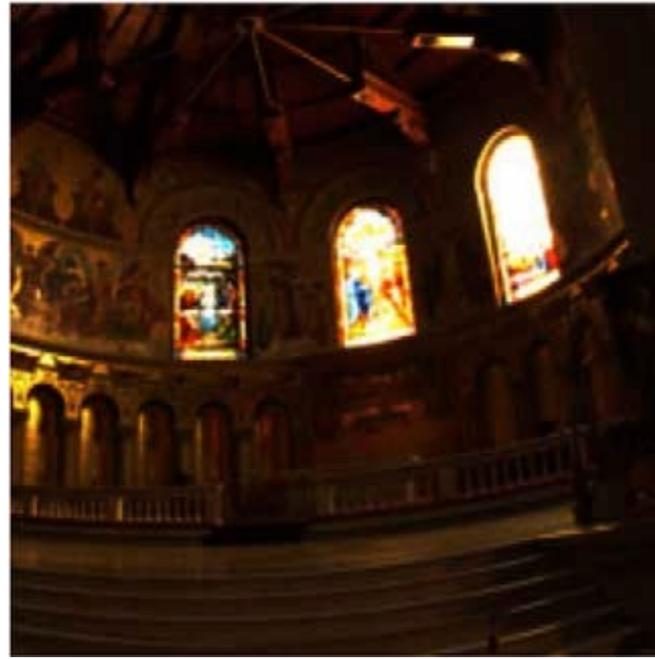
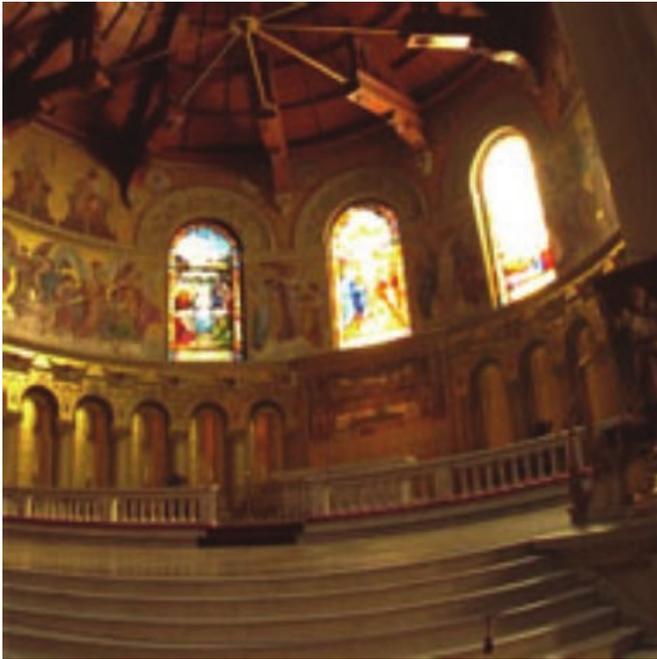
$$I_{\text{non-linear}}(x,y) = f[ I_{\text{linear}}(x,y) ]$$

How would you merge the non-linear images into an HDR one?

# Non-linear image formation model

Real scene radiance for image pixel  $(x,y)$ :  $L(x, y)$

Exposure time:  $t_i$



$$I_{\text{linear}}(x,y) = \text{clip}[ t_i \cdot L(x,y) + \text{noise} ]$$

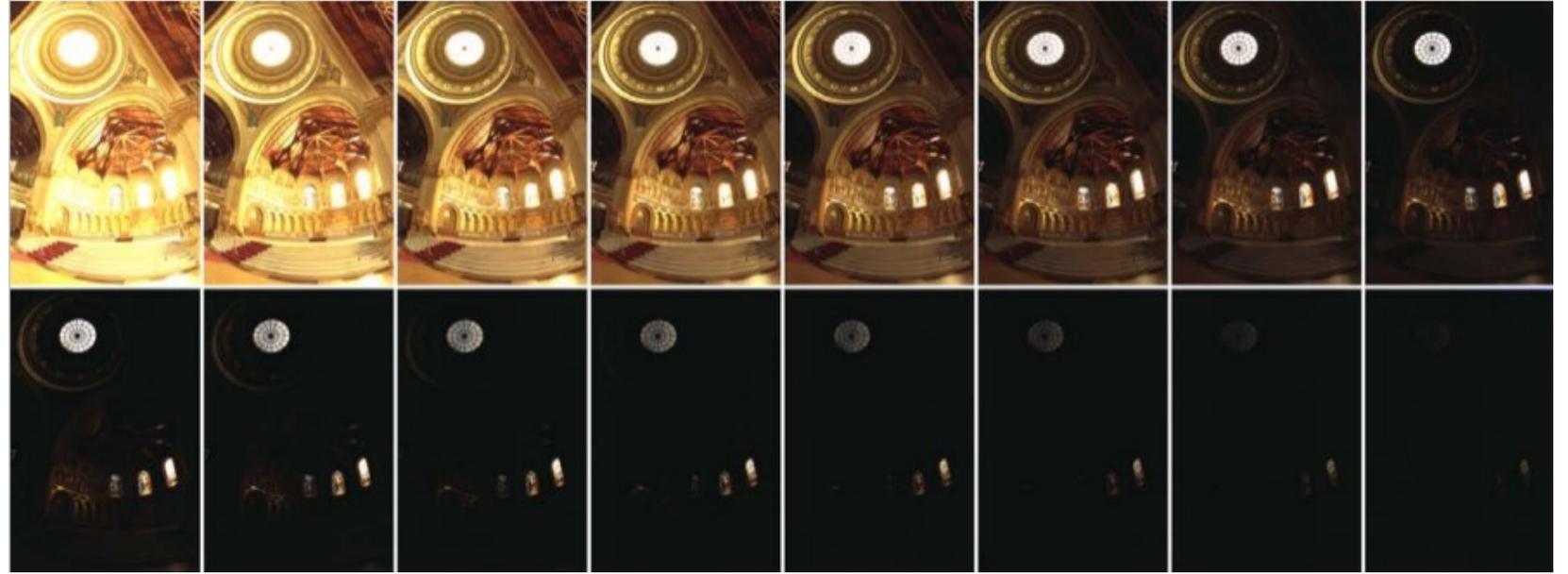
$$I_{\text{non-linear}}(x,y) = f[ I_{\text{linear}}(x,y) ]$$

$$I_{\text{est}}(x,y) = f^{-1}[ I_{\text{non-linear}}(x,y) ]$$

Use inverse transform to estimate linear image, then proceed as before

# Linearization

$$I_{\text{non-linear}}(x,y) = f[ I_{\text{linear}}(x,y) ]$$



$$I_{\text{est}}(x,y) = f^{-1}[ I_{\text{non-linear}}(x,y) ]$$



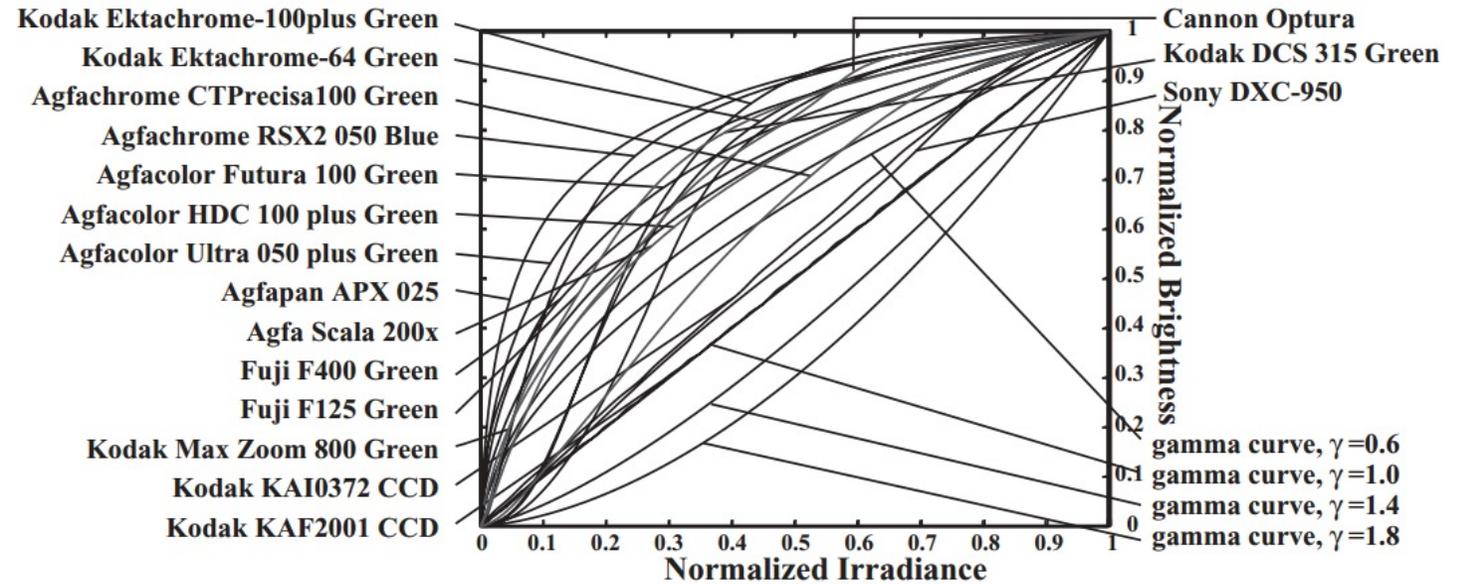
# Tone reproduction curves



linear



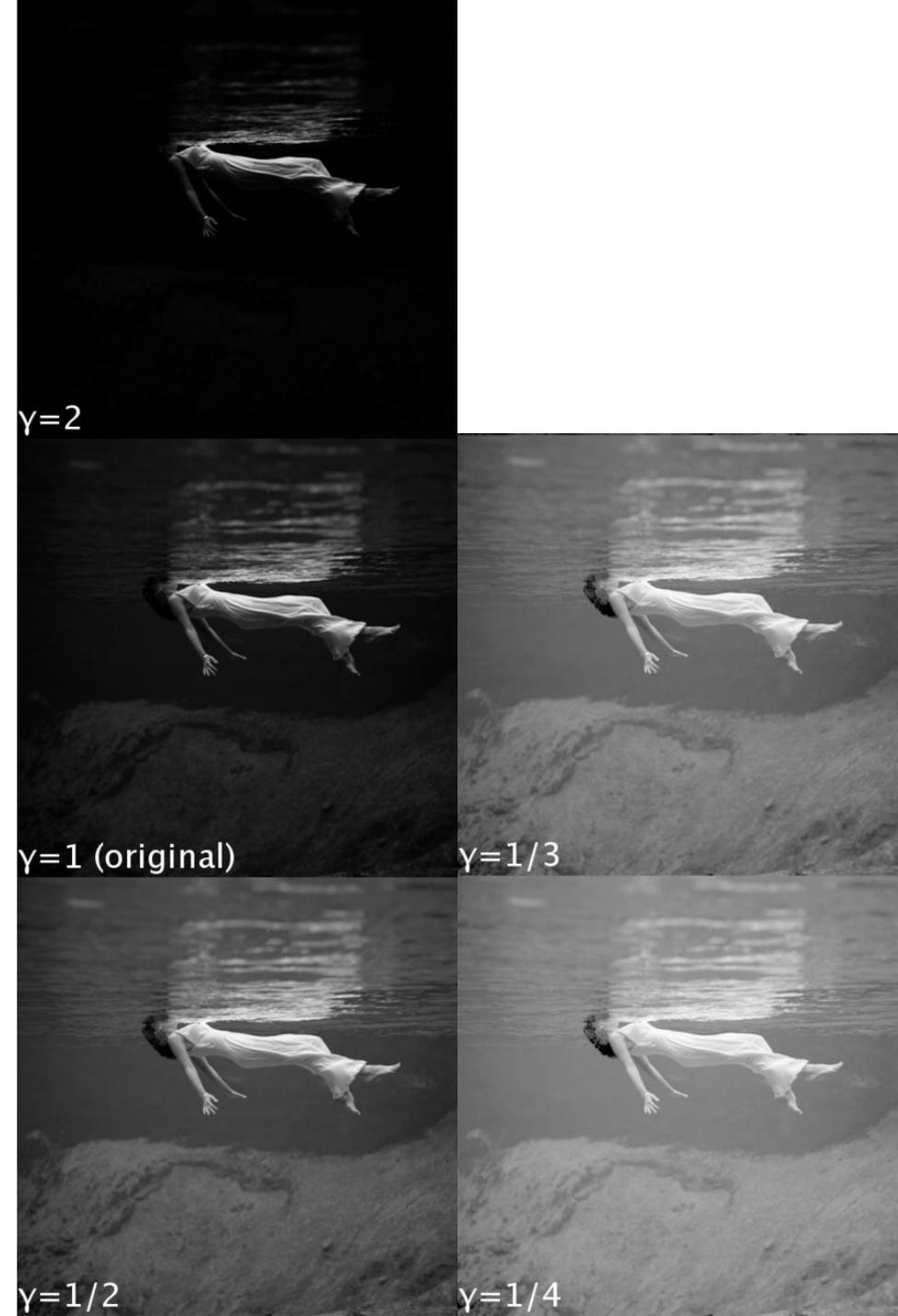
non-linear



# Gamma Correction $V_{\text{out}} = AV_{\text{in}}^\gamma$ ,

Linear raw-RGB images are tone mapped into non-linear sRGB space.

- Most images we work with are in sRGB space, i.e., already tone-mapped.
- Many Vision algorithms expects the image to be in linear space.
- sRGB -> Linear space conversion requires explicit knowledge of camera processing pipeline. Requires knowledge of tone-production curve.
- In absence of it it is very common to use  $\gamma = 2.2$ .
- Note: This is not accurate, just a cheap approximation that works for most Vision tasks.



# Merging non-linear exposure stacks

1. Calibrate response curve

2. Linearize images

For each pixel:

3. Find “valid” images

← (noise)  $0.05 < \text{pixel} < 0.95$  (clipping)

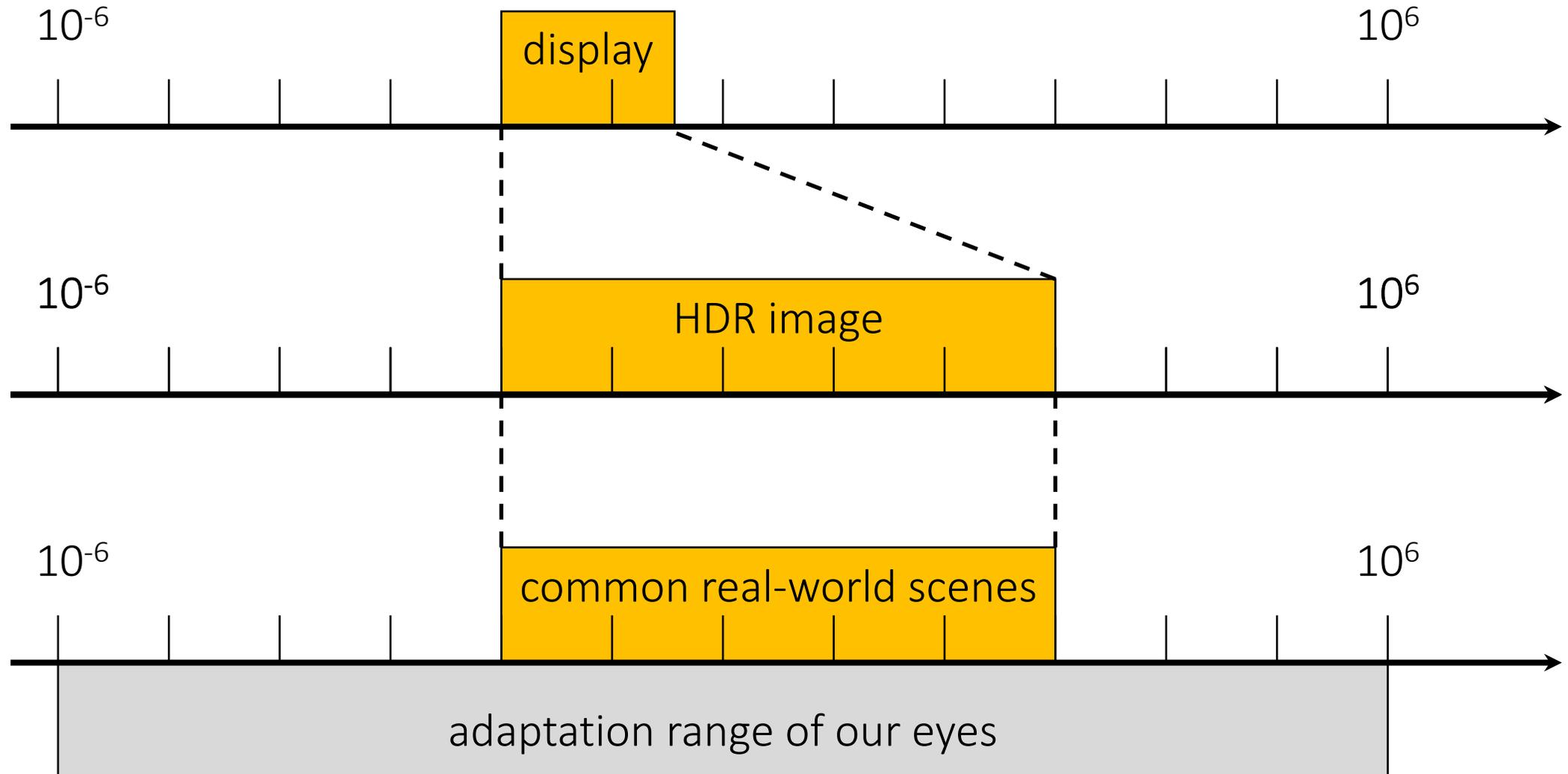
4. Weight valid pixel values appropriately

← (pixel value) /  $t_i$

5. Form a new pixel value as the weighted average of valid pixel values

→ Same steps as in the RAW case.

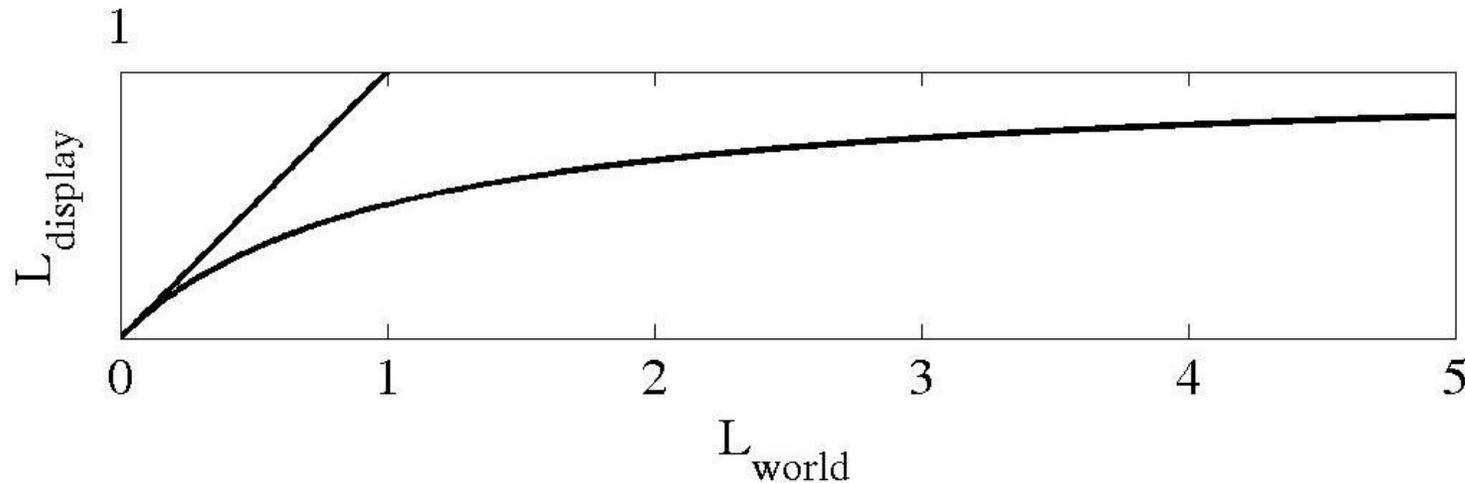
# How do we display our HDR images?



# Photographic tonemapping

Apply the same non-linear scaling to all pixels in the image so that:

- Bring everything within range  $\rightarrow$  asymptote to 1
- Leave dark areas alone  $\rightarrow$  slope = 1 near 0



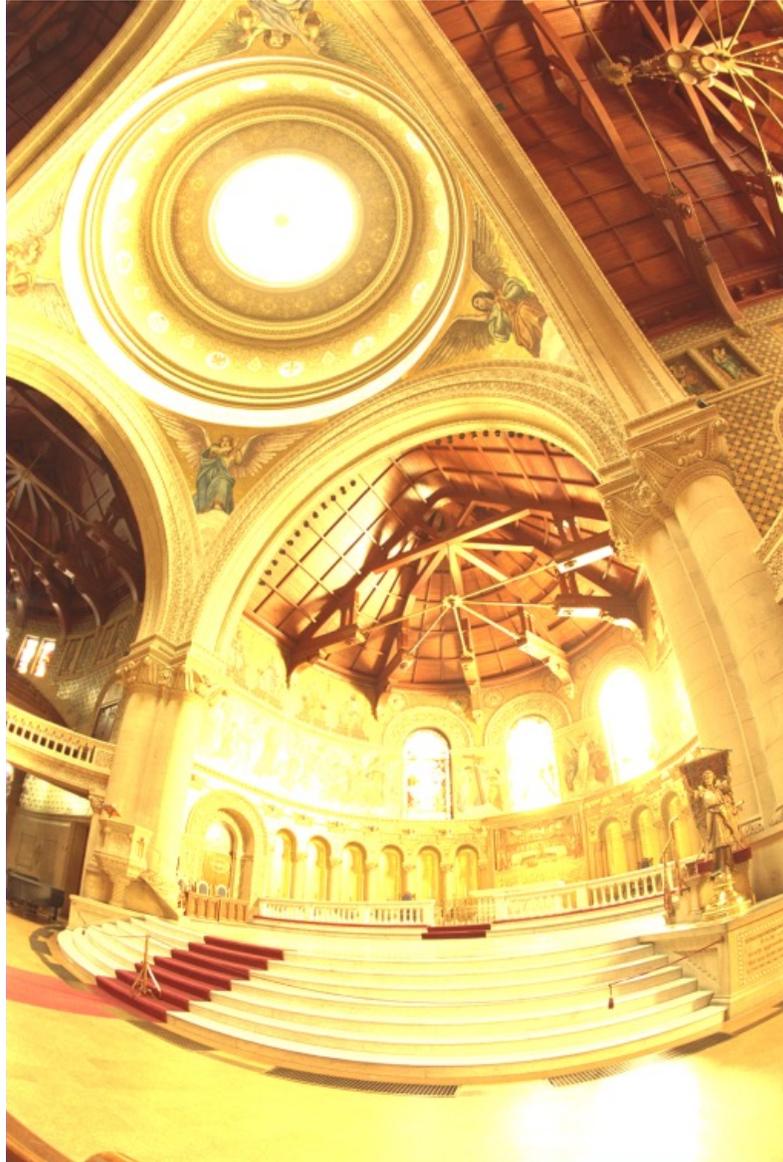
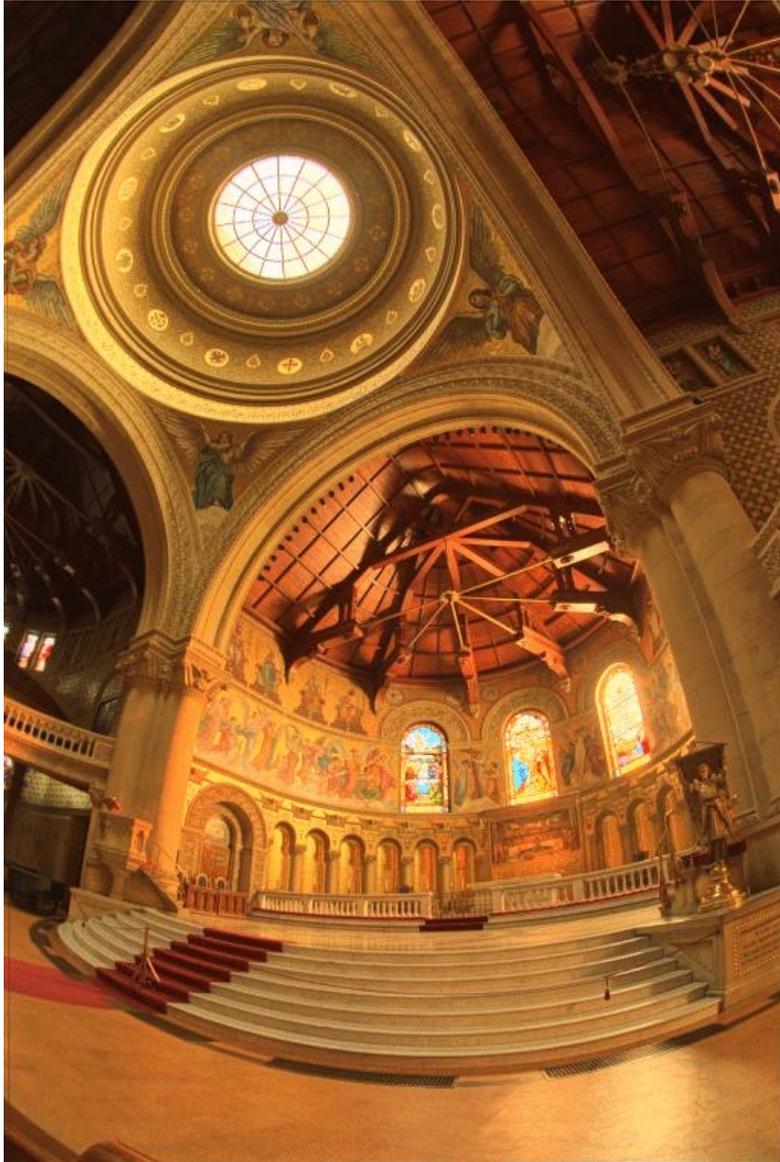
$$I_{display} = \frac{I_{HDR}}{1 + I_{HDR}}$$

(exact formula more complicated)

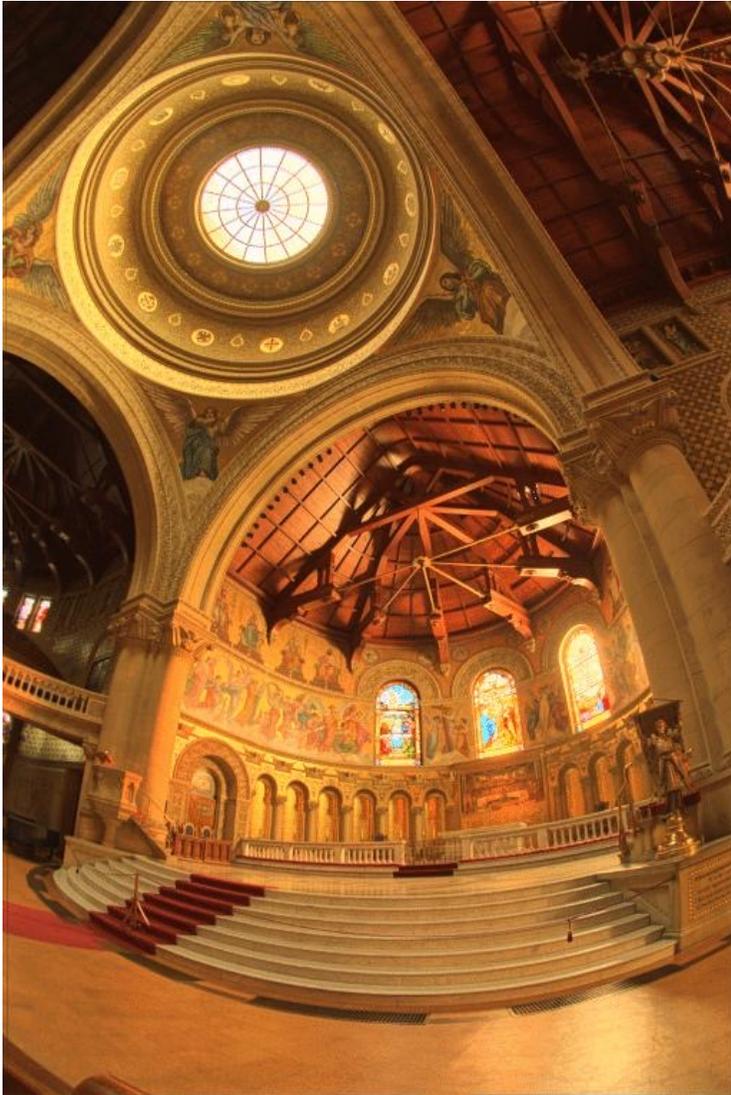
- Photographic because designed to approximate film zone system.
- Perceptually motivated, as it approximates our eye's response curve.

# Compare with LDR images

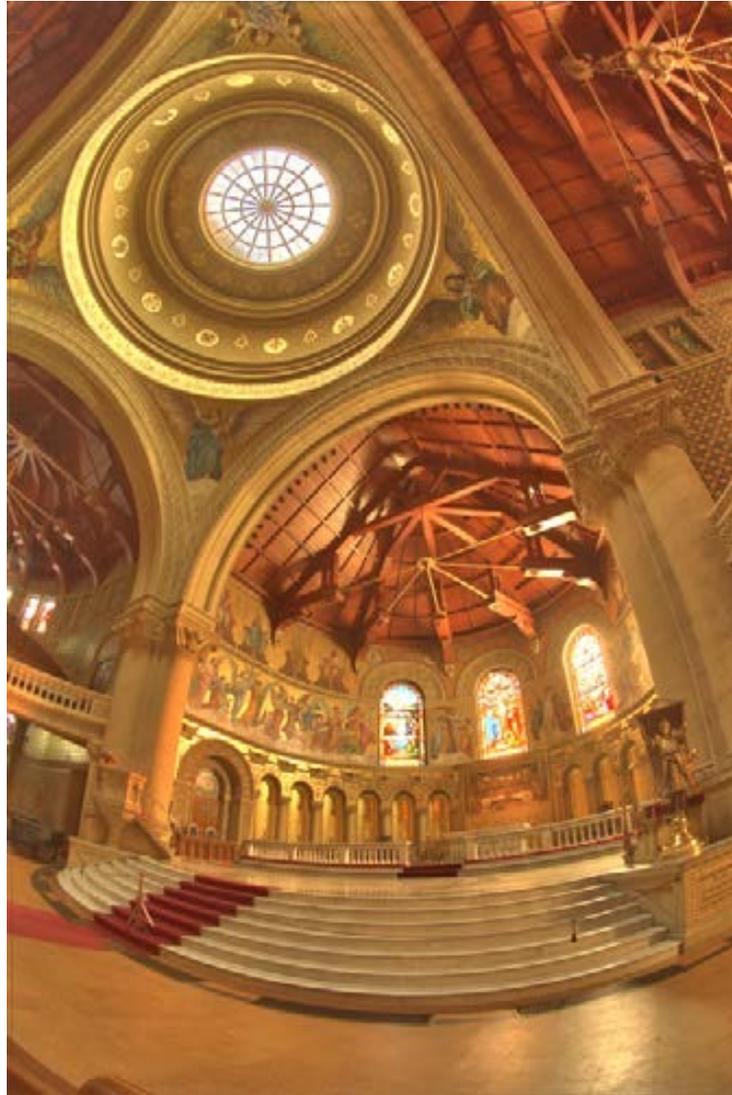
photographic tonemapping



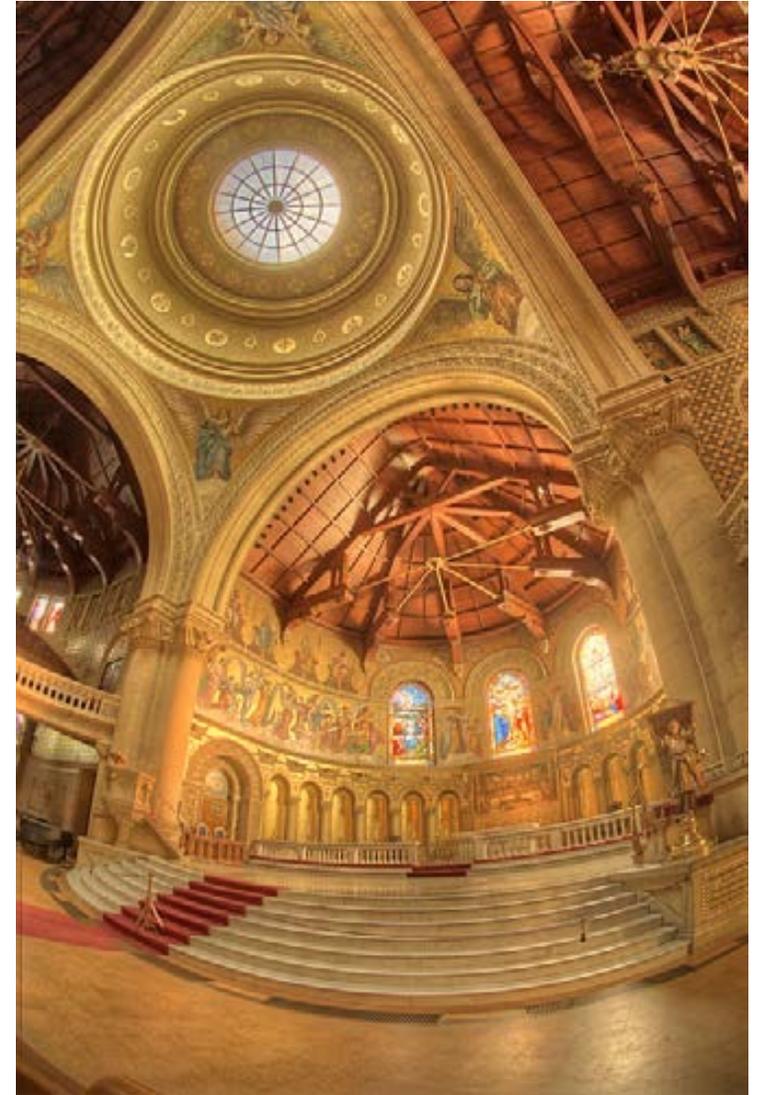
# Comparison (which one do you like better?)



photographic



bilateral filtering



gradient-domain

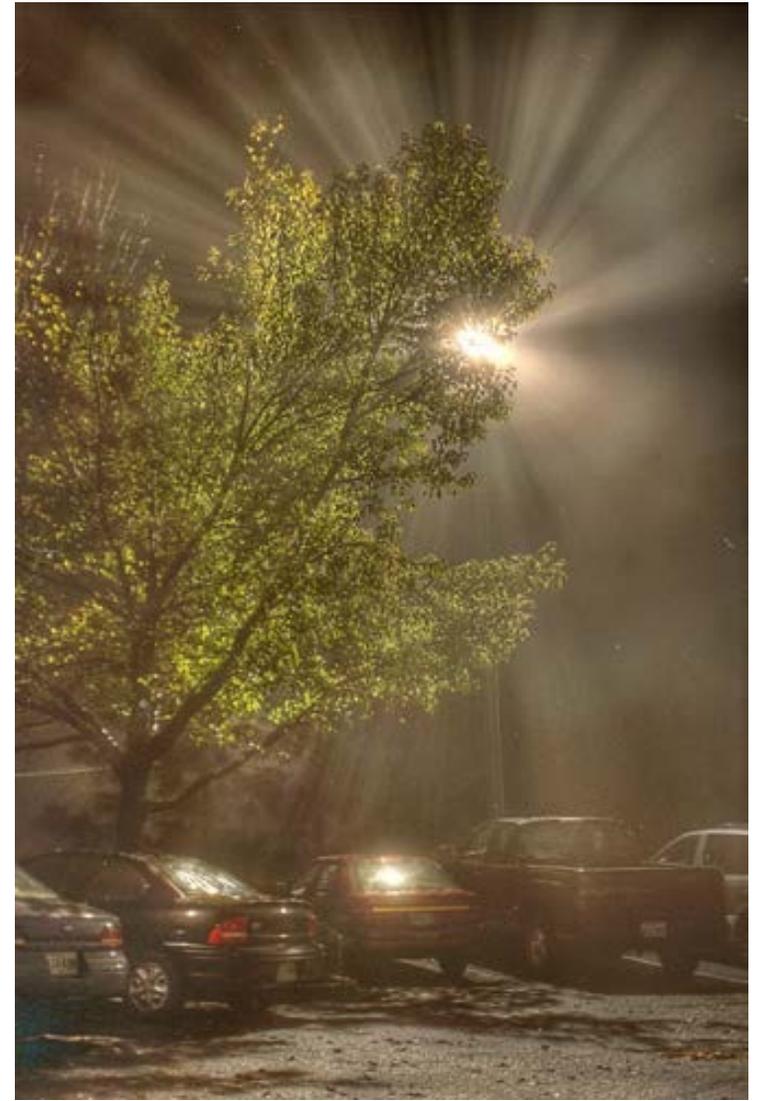
# Comparison (which one do you like better?)



photographic



bilateral filtering



gradient-domain

# Comparison (which one do you like better?)



There is no ground-truth: which one looks better is entirely subjective



photographic

bilateral filtering

gradient-domain







# "Take away" messages

1) Consumer cameras do not attempt to reproduce accurate color!



# "Take away" messages

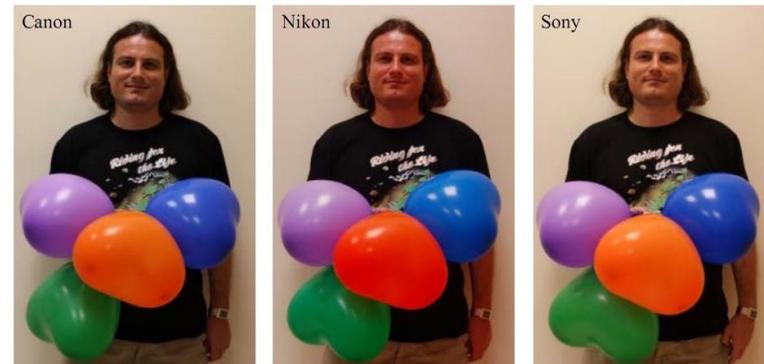
2) sRGB is a standard color space, but colors are not accurate with respect to the physical world.

Remember, sRGB values have been manipulated by your camera hardware and do not represent the physical world, they represent the "photo"!

sRGB output on the same camera,  
different photo options!



sRGB output on different cameras.

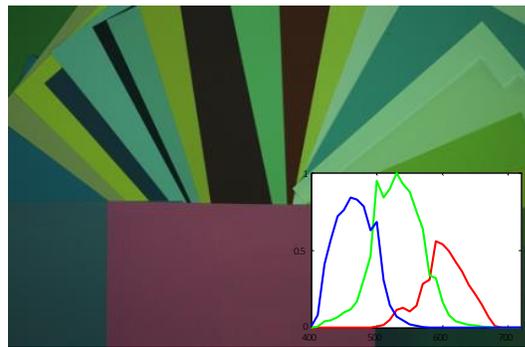


# "Take away" messages

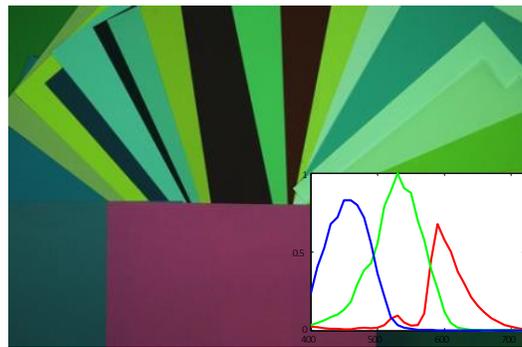
## 3) raw-RGB images are linear with respect to irradiance

If you need to measure the physical world, capture raw-RGB images.

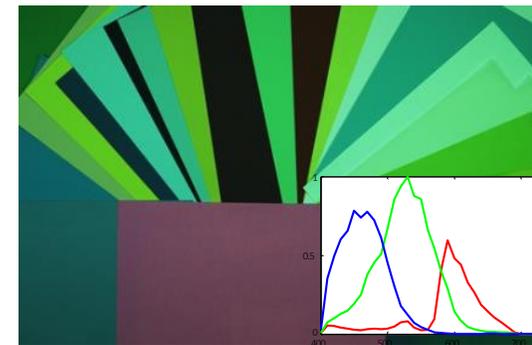
However, keep in mind that raw-RGB images are not standard between devices!  
The raw-RGB is device specific.



Canon 1D

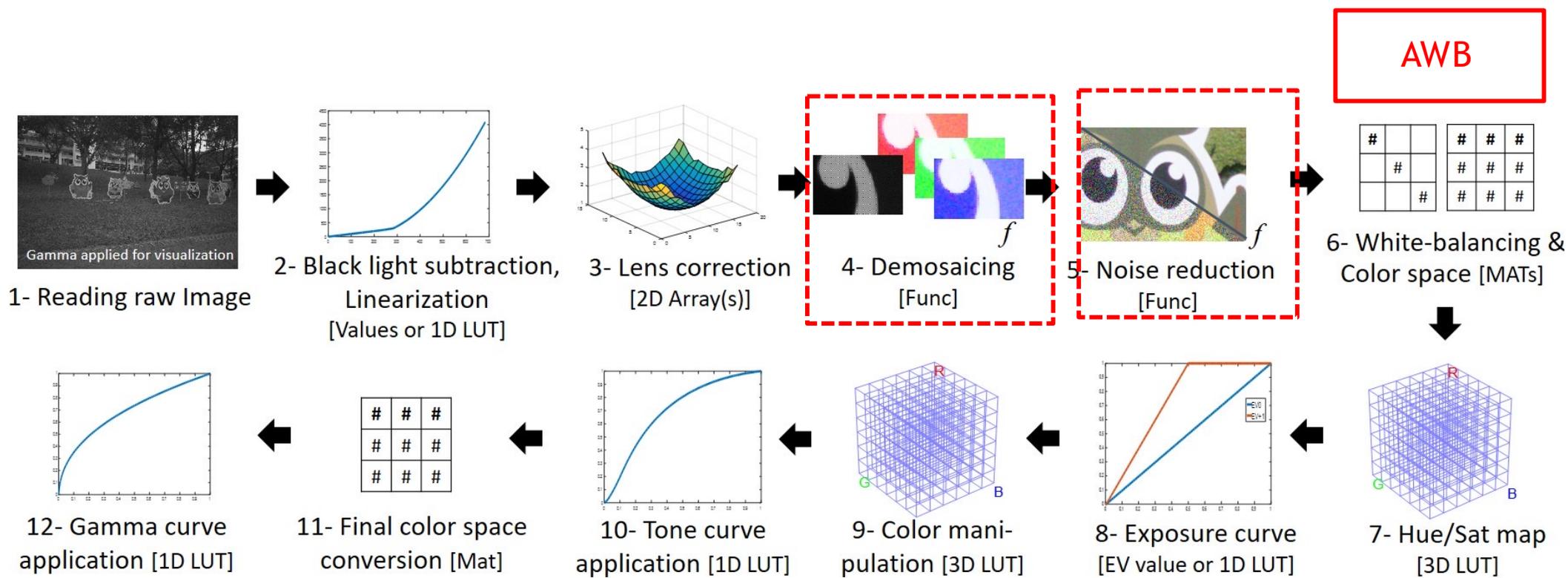


Nikon D40



Sony *α*57

# Many opportunities to improve the camera pipeline



# Color reproduction notes

To properly reproduce the color of an image file, you need to convert it from the color space it was stored in, to a reference color space, and then to the color space of your display.

On the camera side:

- If the file is RAW, it *often* has EXIF tags with information about the RGB color space corresponding to the camera's color sensitivity functions.
- If the file is not RAW, you *may* be lucky and still find accurate information in the EXIF tags about what color space the image was converted in during processing.
- If there is no such information and you own the camera that shot the image, then you can do color calibration for the camera.
- If all of the above fails, assume sRGB.

On the display side:

- If you own a high-end display, it likely has accurate color profiles provided by the manufacturer.
- If not, you can use a spectrometer to do color profiling (not color calibration).
- Make sure your viewer does not automatically do color transformations.

Be careful to account for any gamma correction!

Amazing resource for color management and photography: <https://ninedegreesbelow.com/>

# How do you convert an image to grayscale?

First, you need to answer two questions:

1) Is your image linear or non-linear?

- If the image is linear (RAW, HDR, or otherwise radiometrically calibrated), skip this step.
- If the image is nonlinear (PNG, JPEG, etc.), you must undo the tone reproduction curve.
  - i. If you can afford to do radiometric calibration, do that.
  - ii. If your image has EXIF tags, check there about the tone reproduction curve.
  - iii. If your image is tagged as non-linear sRGB, use the inverse of the sRGB tone reproduction curve.
  - iv. If none of the above, assume sRGB and do as in (iii).

2) What is the color space of your image?

- If it came from an original RAW file, read the color transform matrix from there (e.g., dcrw).
- If not, you need to figure out the color space.
  - i. If you can afford to do color calibration, use that.
  - ii. If your image has EXIF tags, check there about the color space.
  - iii. If your image is tagged as non-linear sRGB, use the color transform matrix for linear sRGB.
  - iv. If none of the above, assume sRGB and do (iii).

With this information in hand:

- Transform your image into the XYZ color space.
- Extract the Y channel.
- If you want brightness instead of luminance, apply the Lab brightness non-linearity.

# Slide Credits

- “Understanding Color and the In-Camera Image Processing Pipeline for Computer Vision”, by Michael S. Brown, ICCV 2019 tutorial.
- [CS 15-463, 663, 862](#), CMU, by Computational Photography, Ioannis Gkioulekas.