

**Lecture 12:**

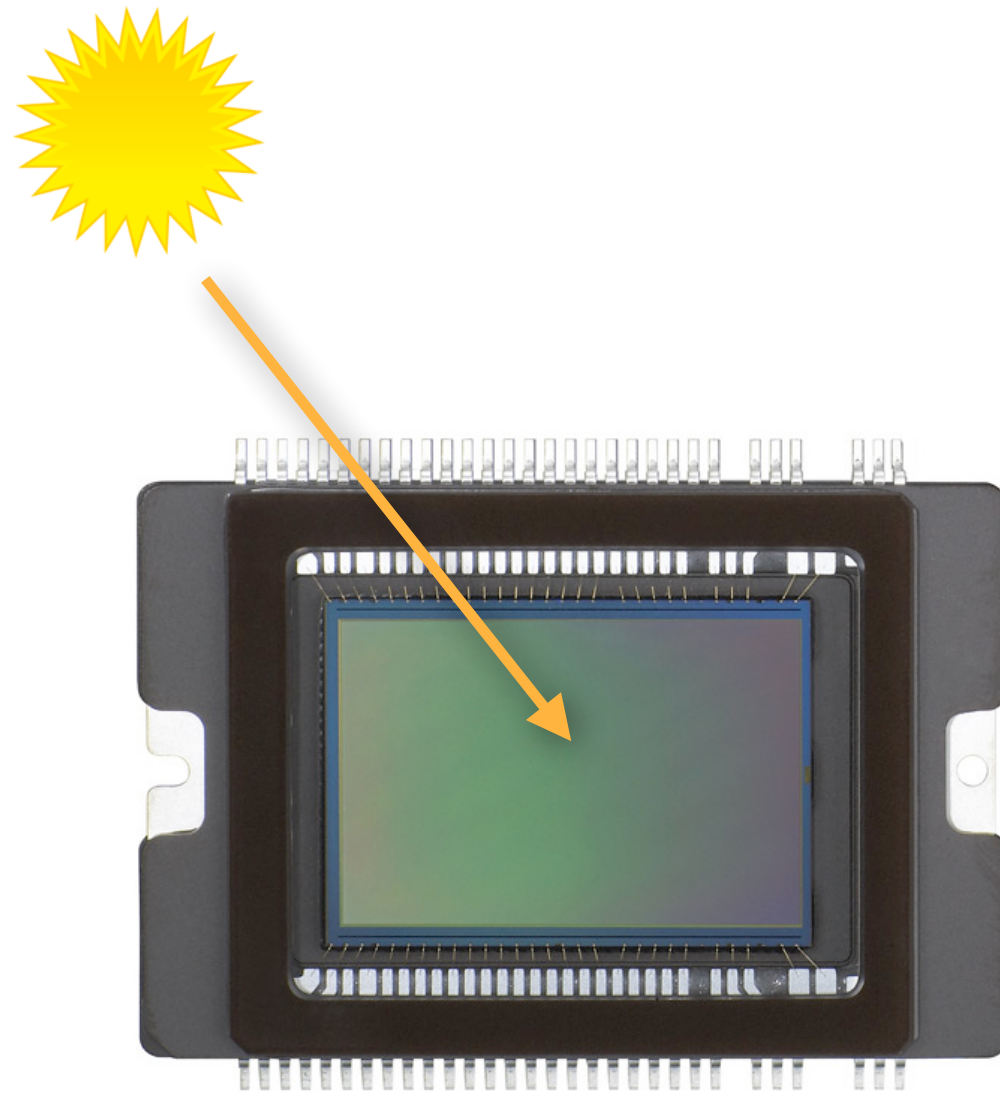
# **Camera Image Processing Pipeline**

---

**Visual Computing Systems  
CMU 15-869, Fall 2014**

# This week

**Sequence of operations that convert photons hitting a sensor to a high-quality image**  
**Processing systems used to efficiently implement these operations**

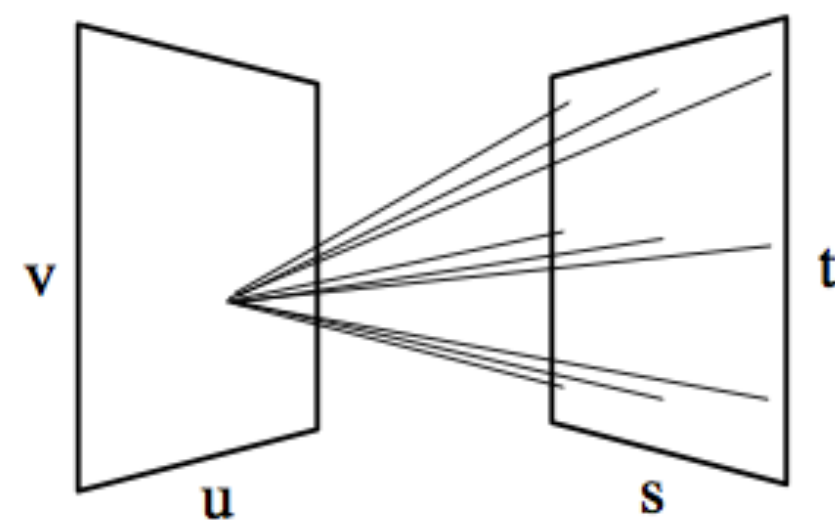


**Canon 14 MP CMOS Sensor**  
**(14 bits per pixel)**

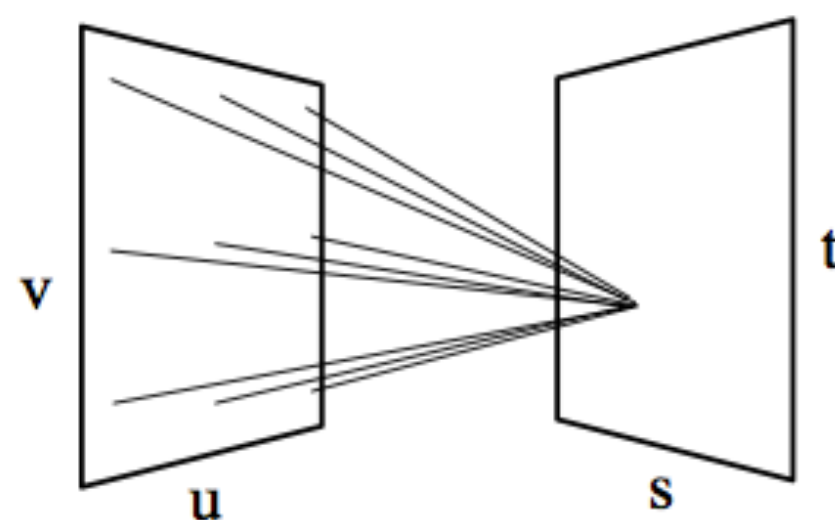
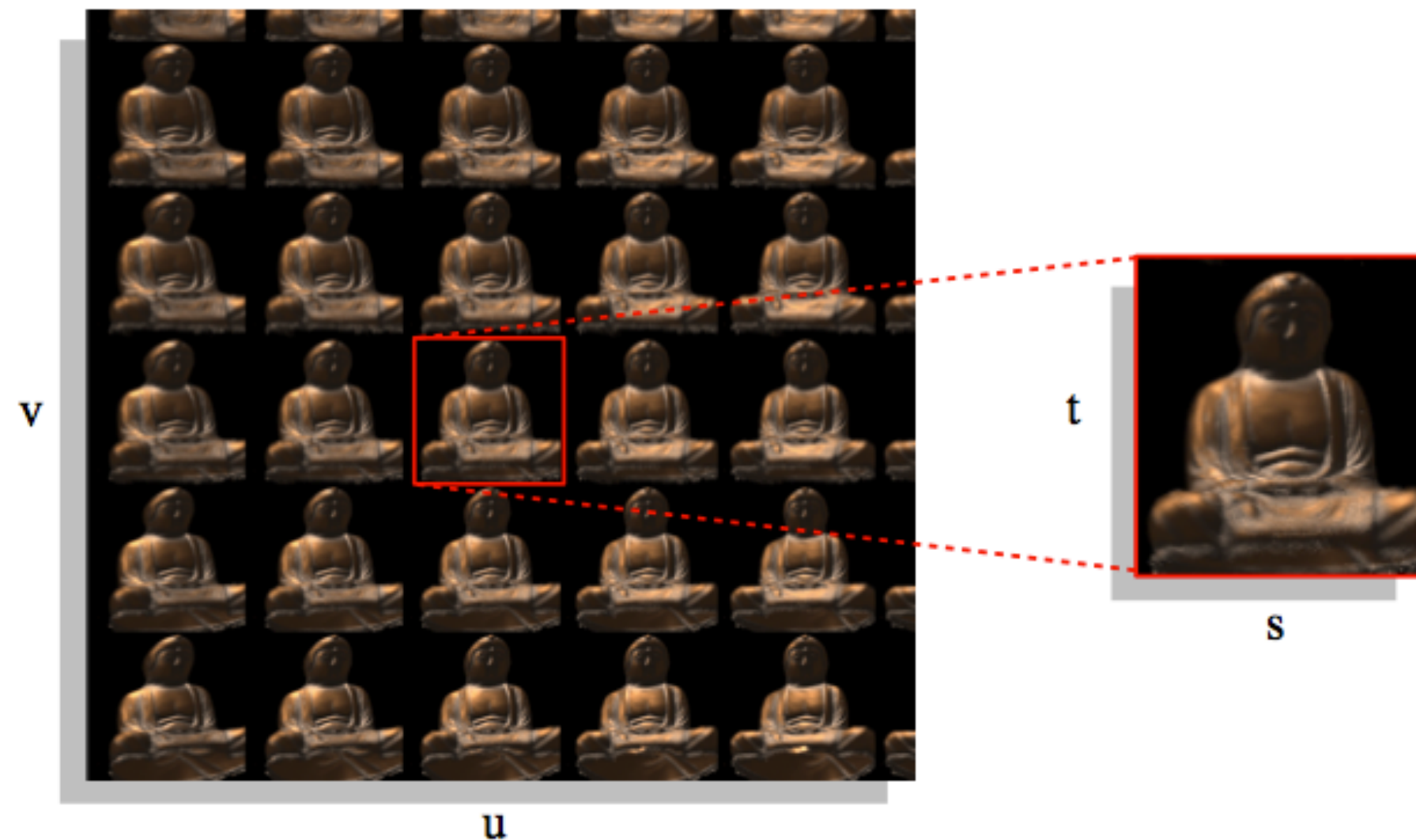


**Final image representation**  
**(e.g., JPG file)**

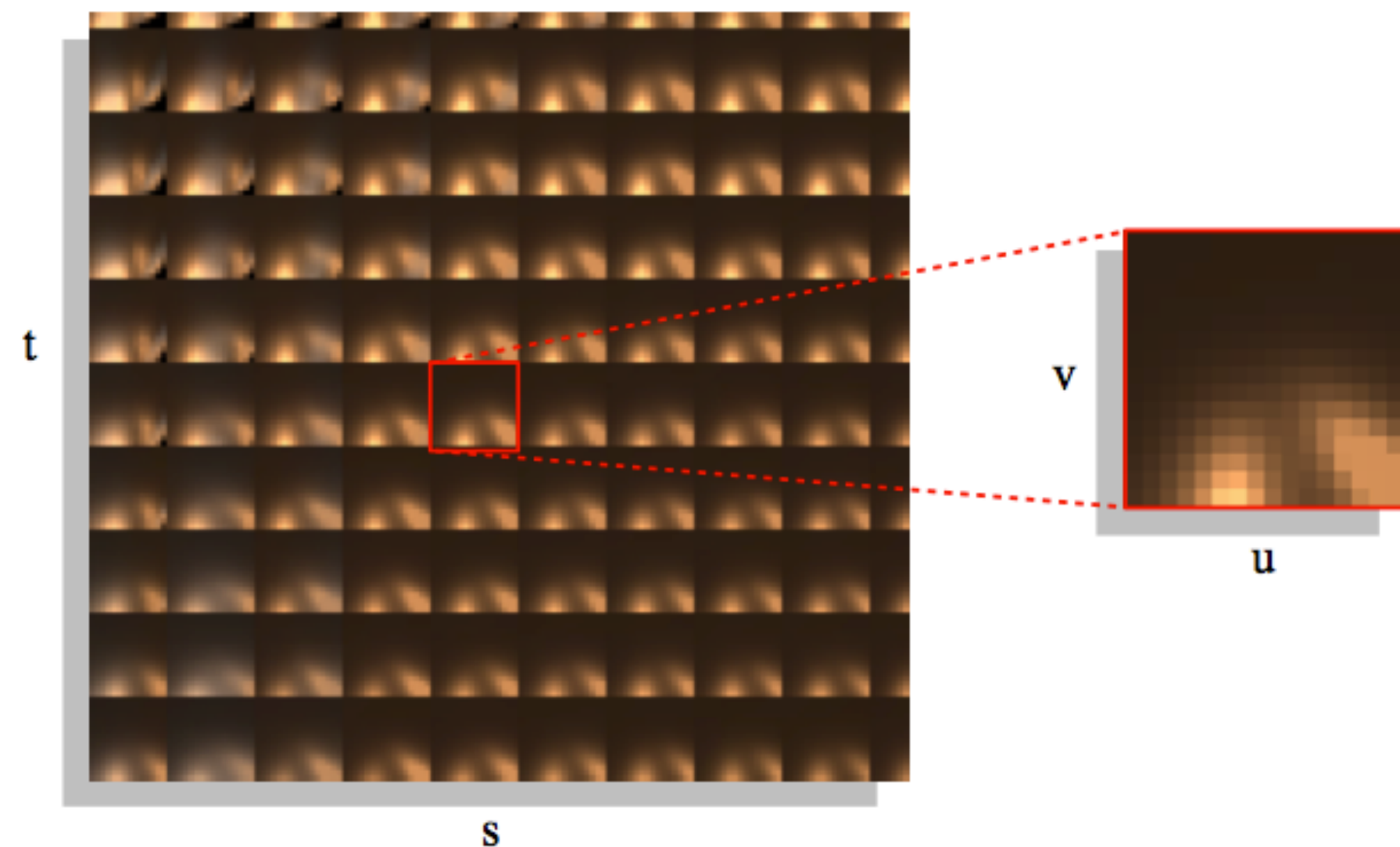
# Review: who can describe this figure?



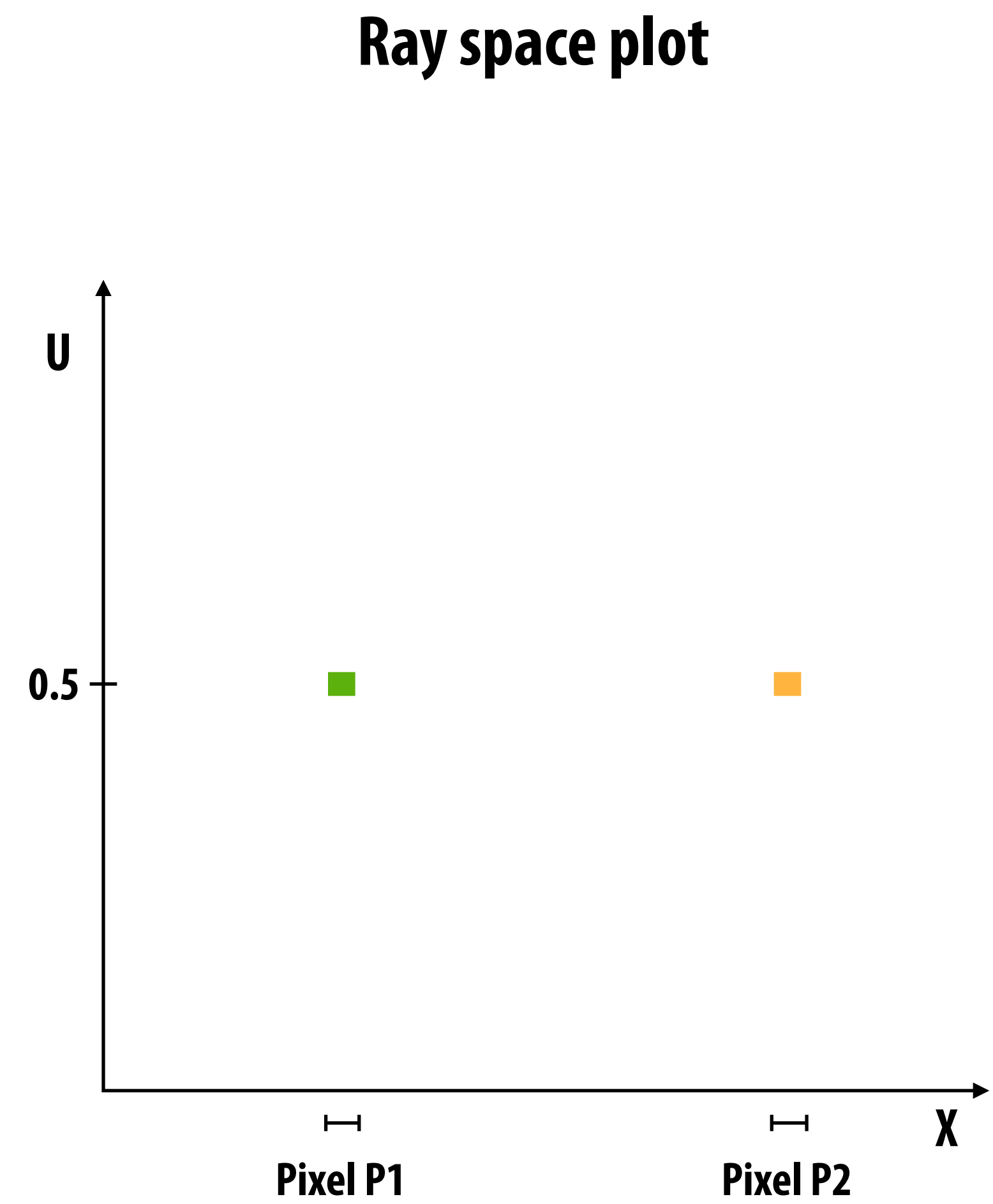
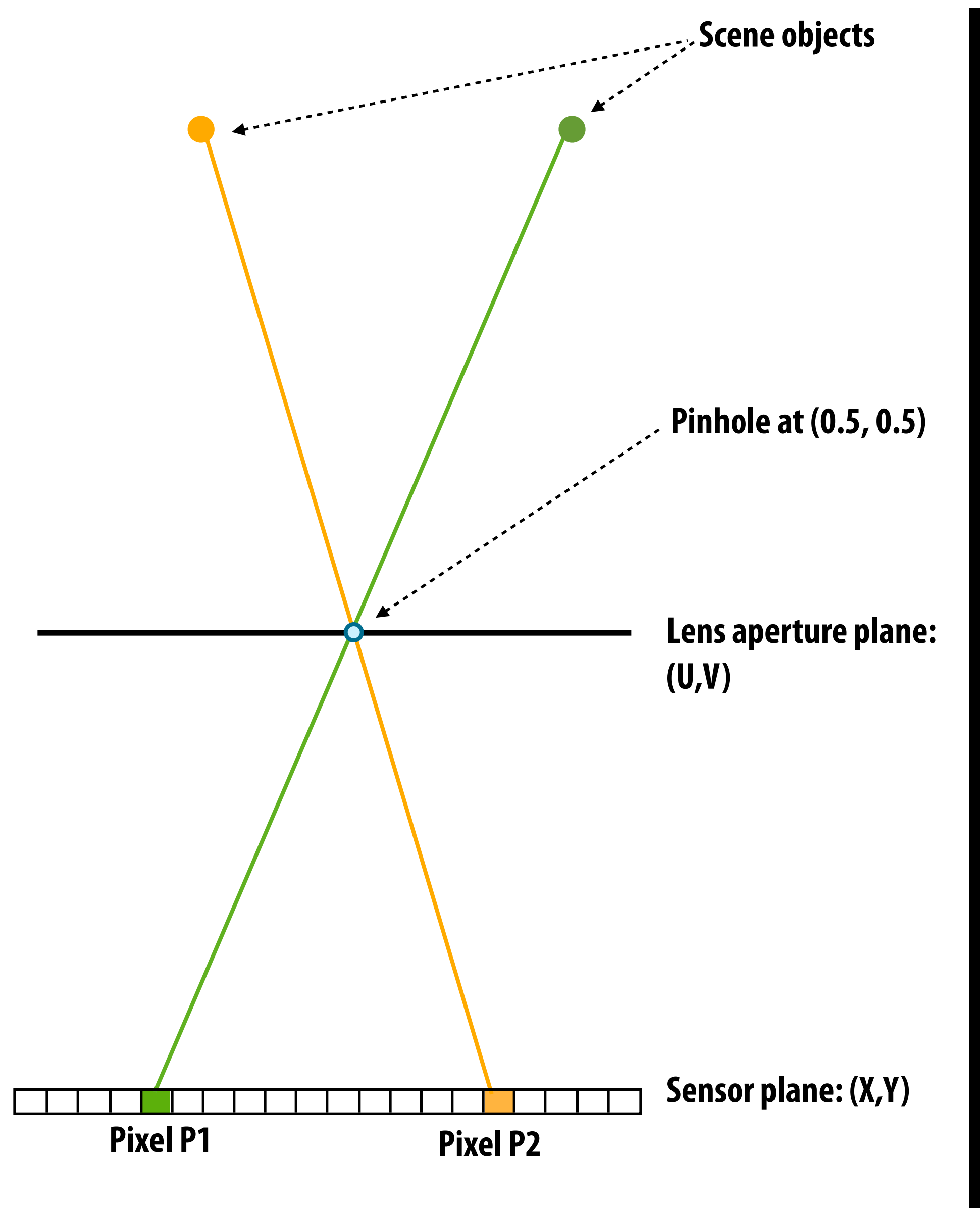
(a)



(b)

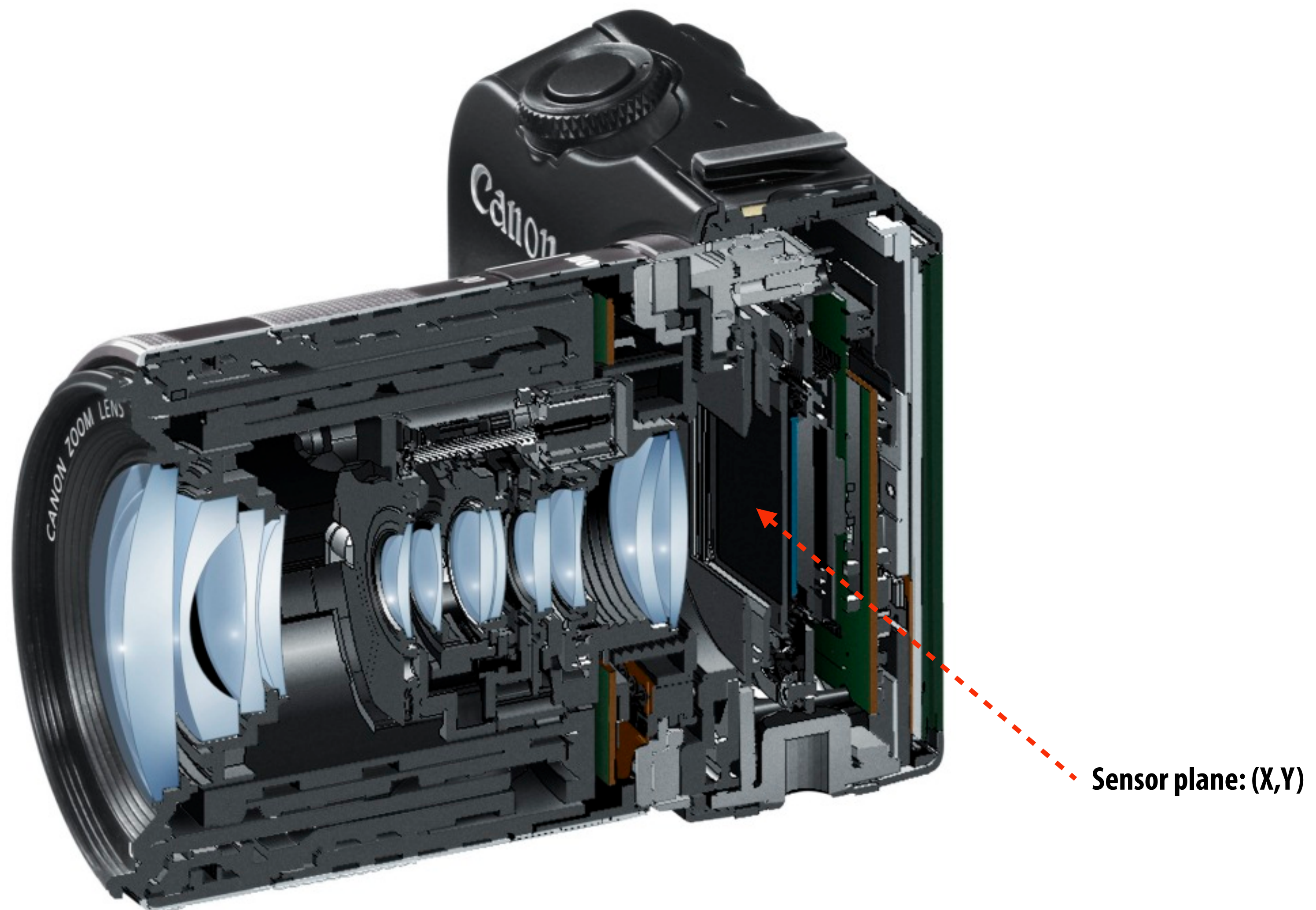


# Review: light field inside a pinhole camera

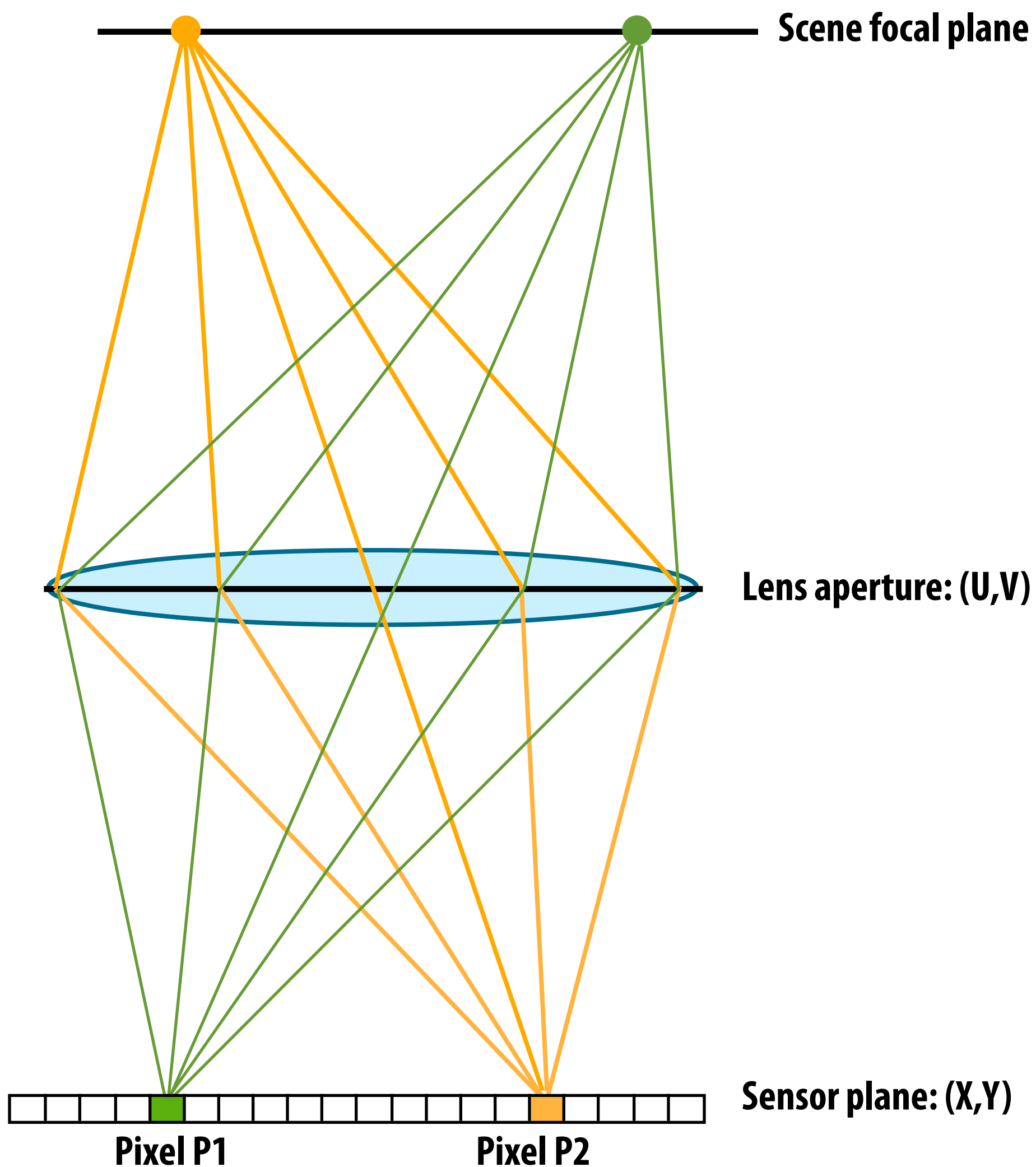




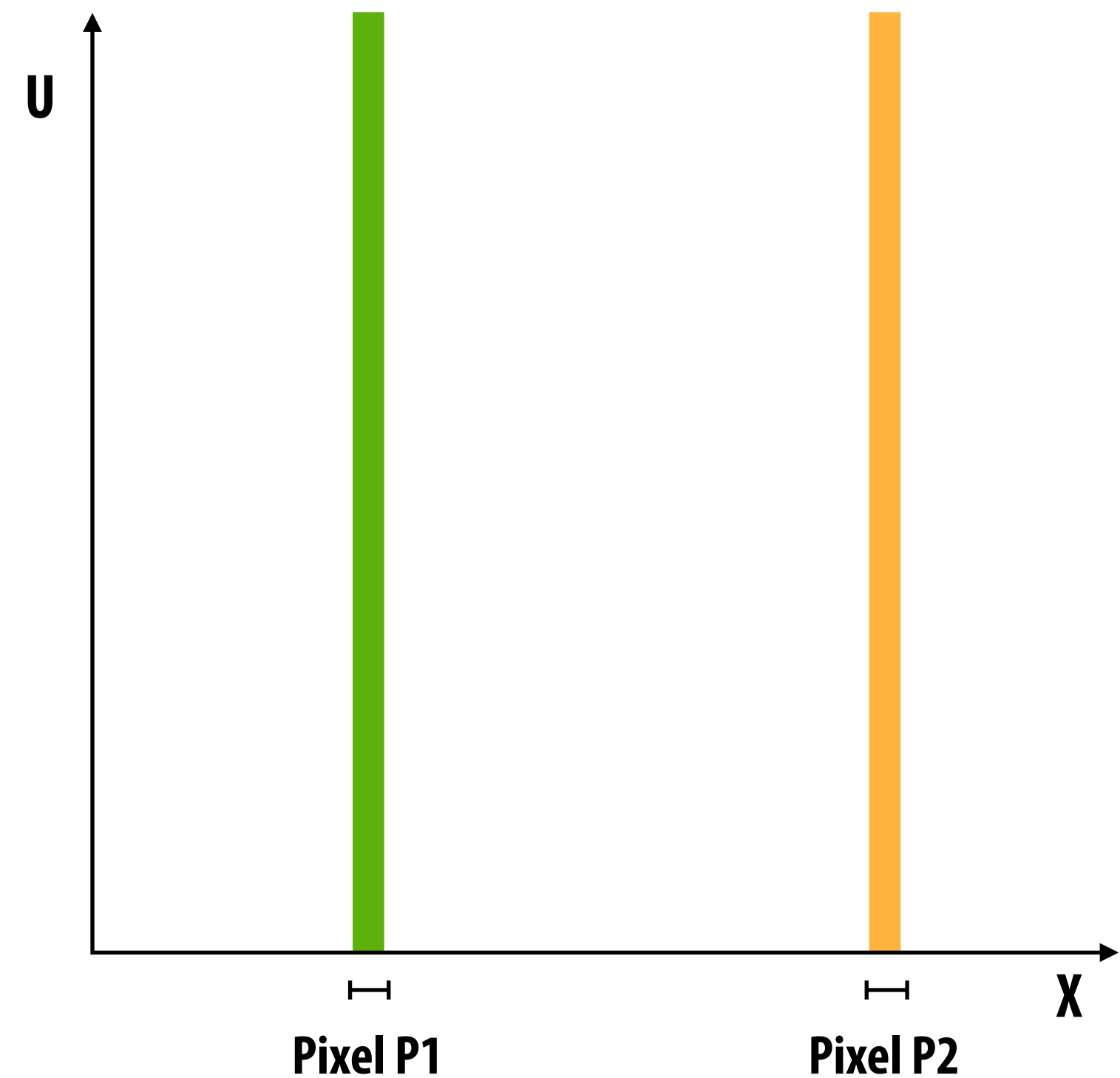
# Camera cross section



# Review: light field inside a camera



Ray space plot



Sensor pixels measure integral of energy due to all rays of light passing through points on the aperture and a pixel-sized area of the sensor.

# Where we are headed

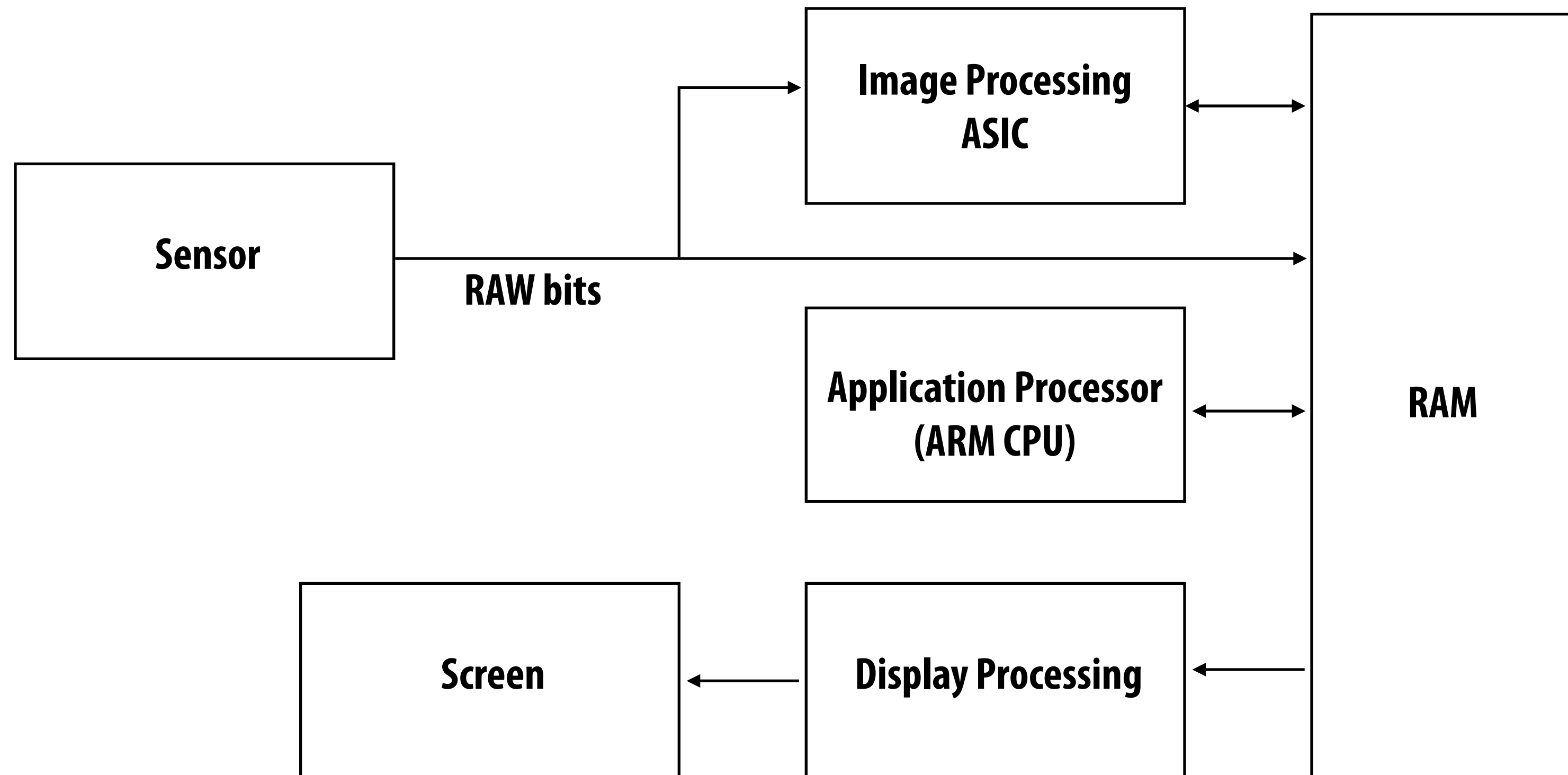
## ■ I'm about to describe the pipeline of operations that take raw image pixels from a sensor (measurements) to high-quality images

- Correct for sensor bias (using measurements of optically black pixels)
- Correct pixel defects
- Vignetting compensation
- Dark frame subtract (optional)
- White balance
- Demosaic
- Denoise / sharpen, etc.
- Color Space Conversion
- Gamma Correction
- Color Space Conversion (Y'CbCr)
- 4:4:4 to 4:2:2 chroma subsampling
- JPEG compress (lossy)

## ■ Today's pipelines are sophisticated, but they only scratch the surface of what future image processing pipelines might do

- Consider what a future image analysis pipeline might feature: person identification, action recognition, scene understanding (to automatically compose shot or automatically pick best picture) etc.

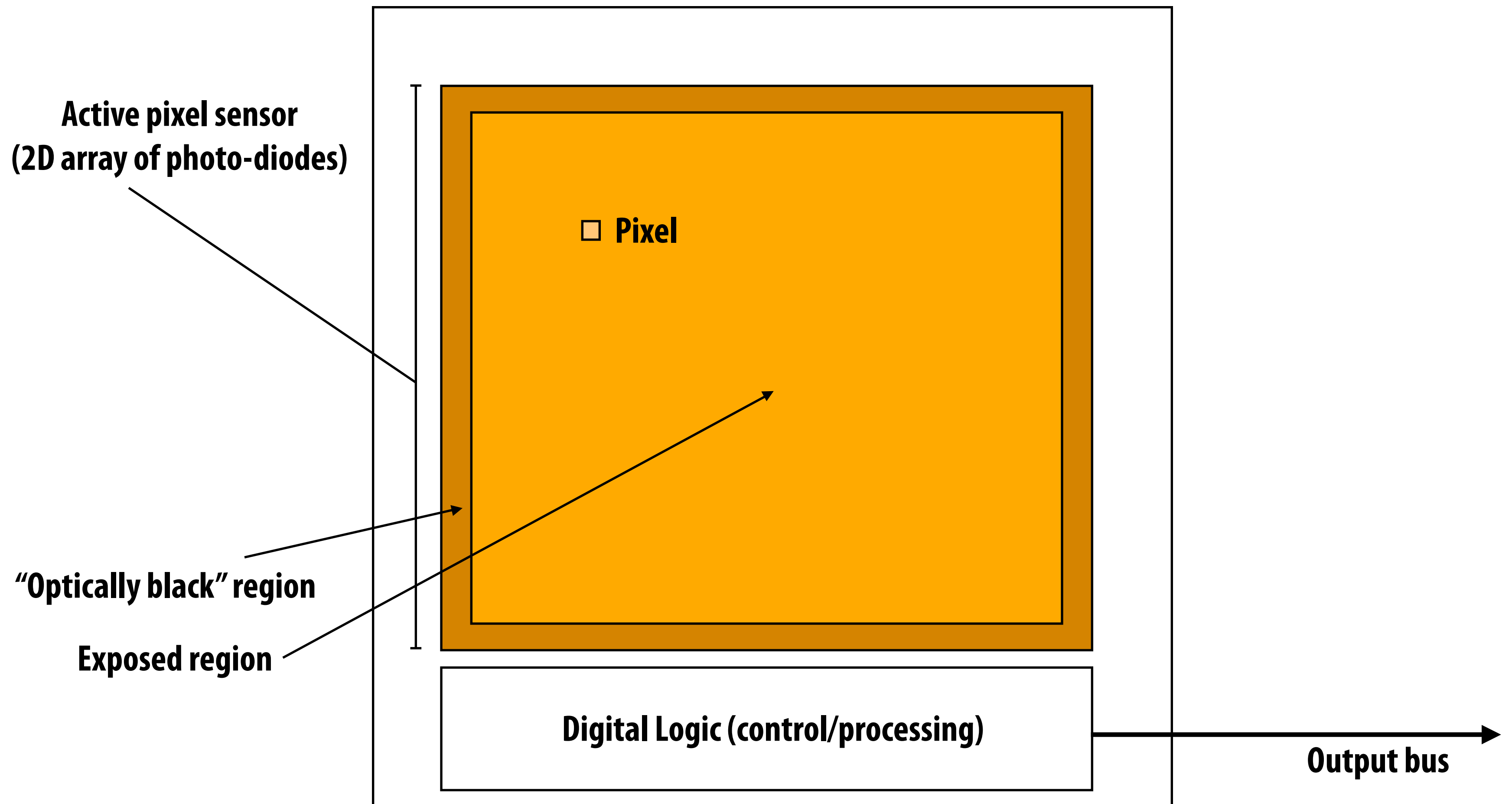
# Generic camera: system overview





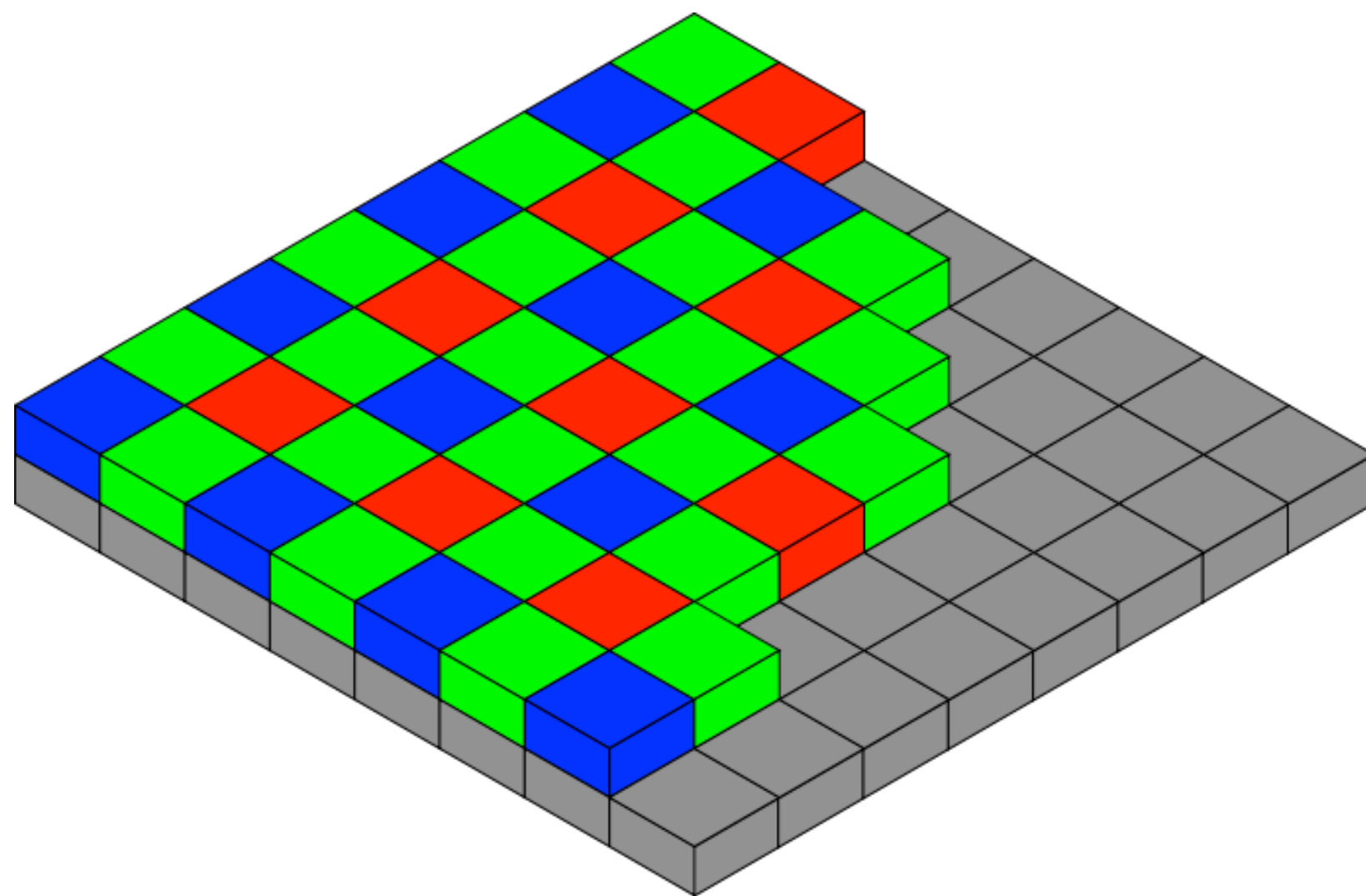
# The Sensor

# CMOS sensor

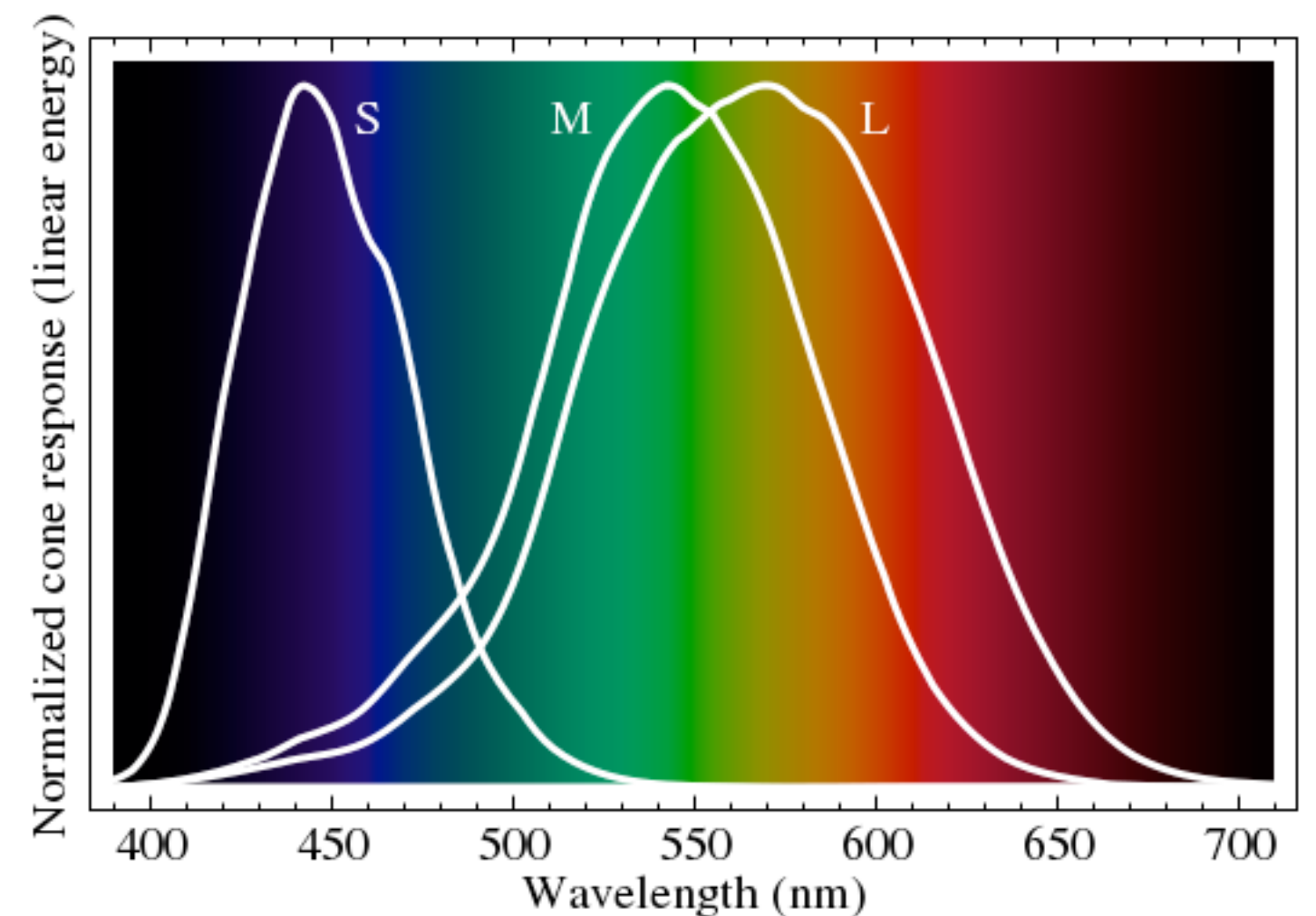


# Color filter array (Bayer mosaic)

- Color filter array placed over sensor
- Result: each pixel measures incident red, green, or blue light
- 50% of pixels are green pixels
  - Human visual perception most sensitive to green light (in normal light levels)



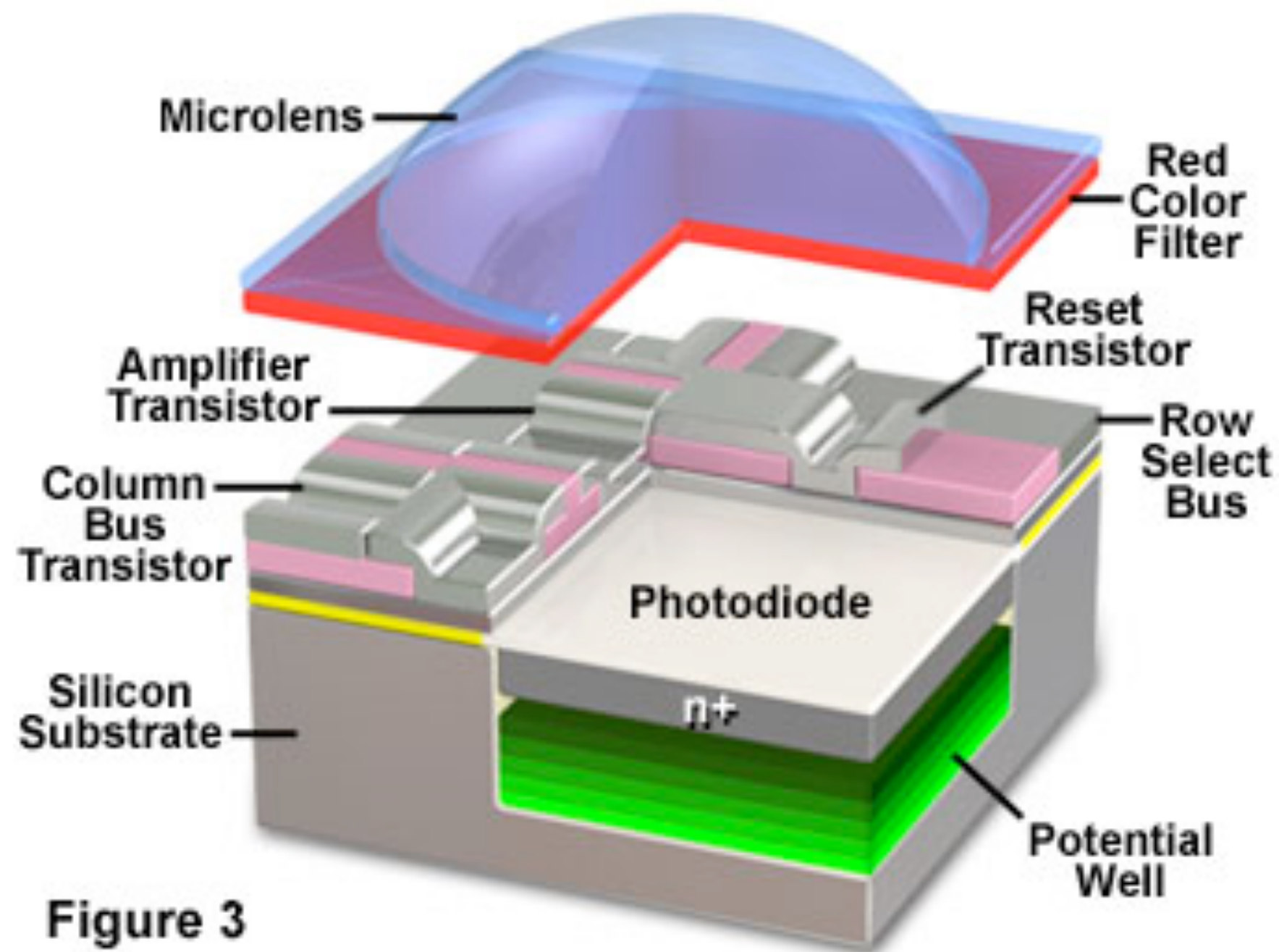
**Traditional Bayer mosaic**  
(other filter patterns exist: e.g., Sony's RGBE)



**Human eye: cone spectral response**

# CMOS sensor pixel

Anatomy of the Active Pixel Sensor Photodiode



**Color filter attenuates light**

**Fill factor: fraction of surface area used for light gathering**

**Microlens (a.k.a. lenslet) steers light toward photo-sensitive region (increases light-gathering capability)**

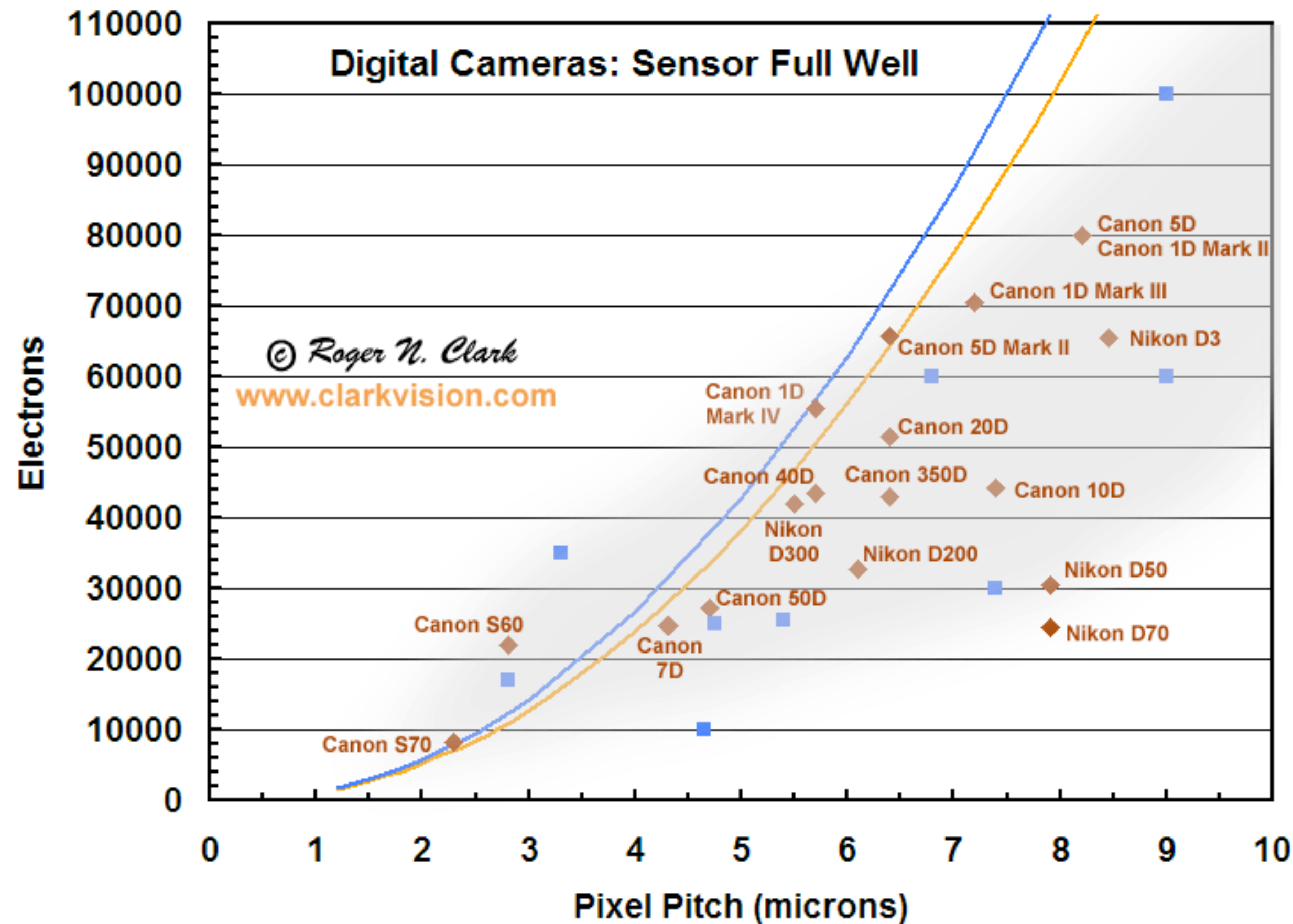
**Microlens also serves to prefilter signal. Why?**

**Quantum efficiency of photodiode in typical digital camera ~ 50%**

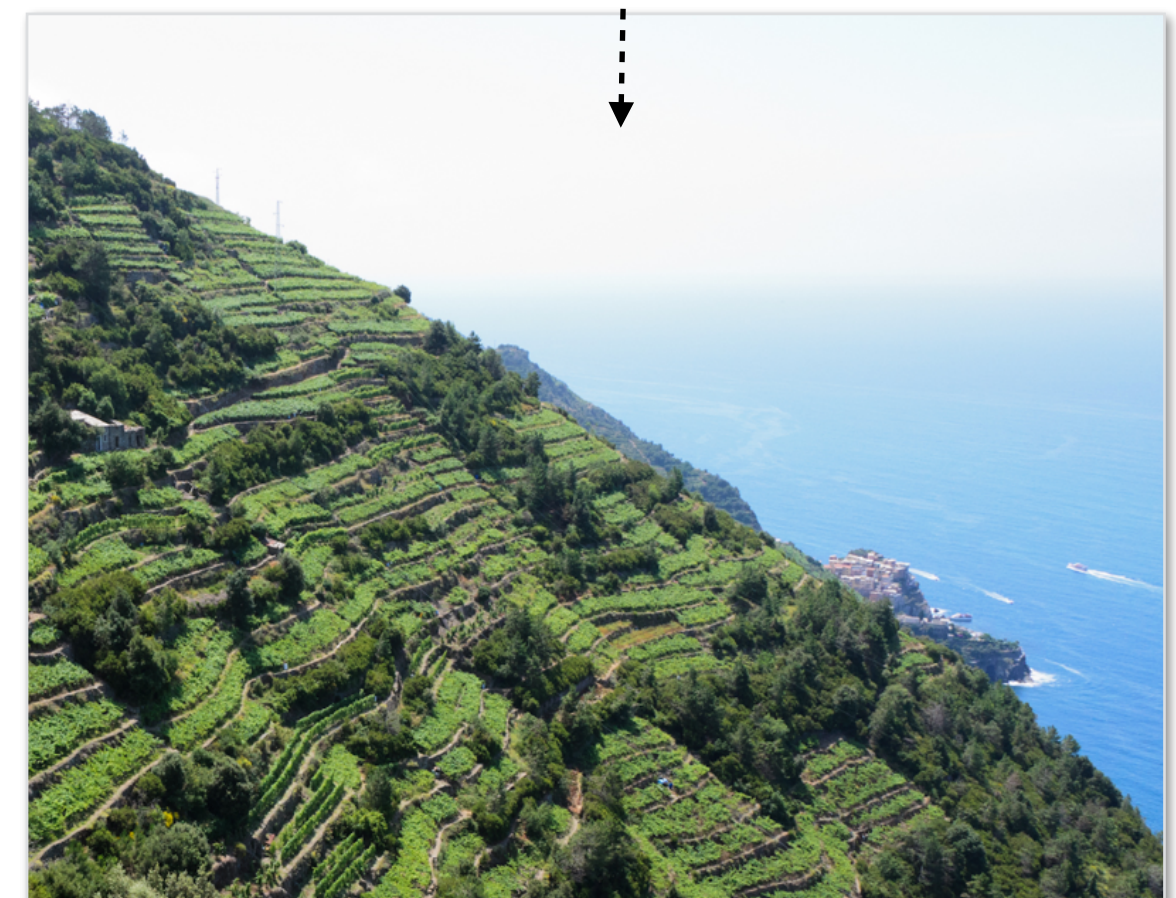


# Full-well capacity

Pixel saturates when capacity is exceeded

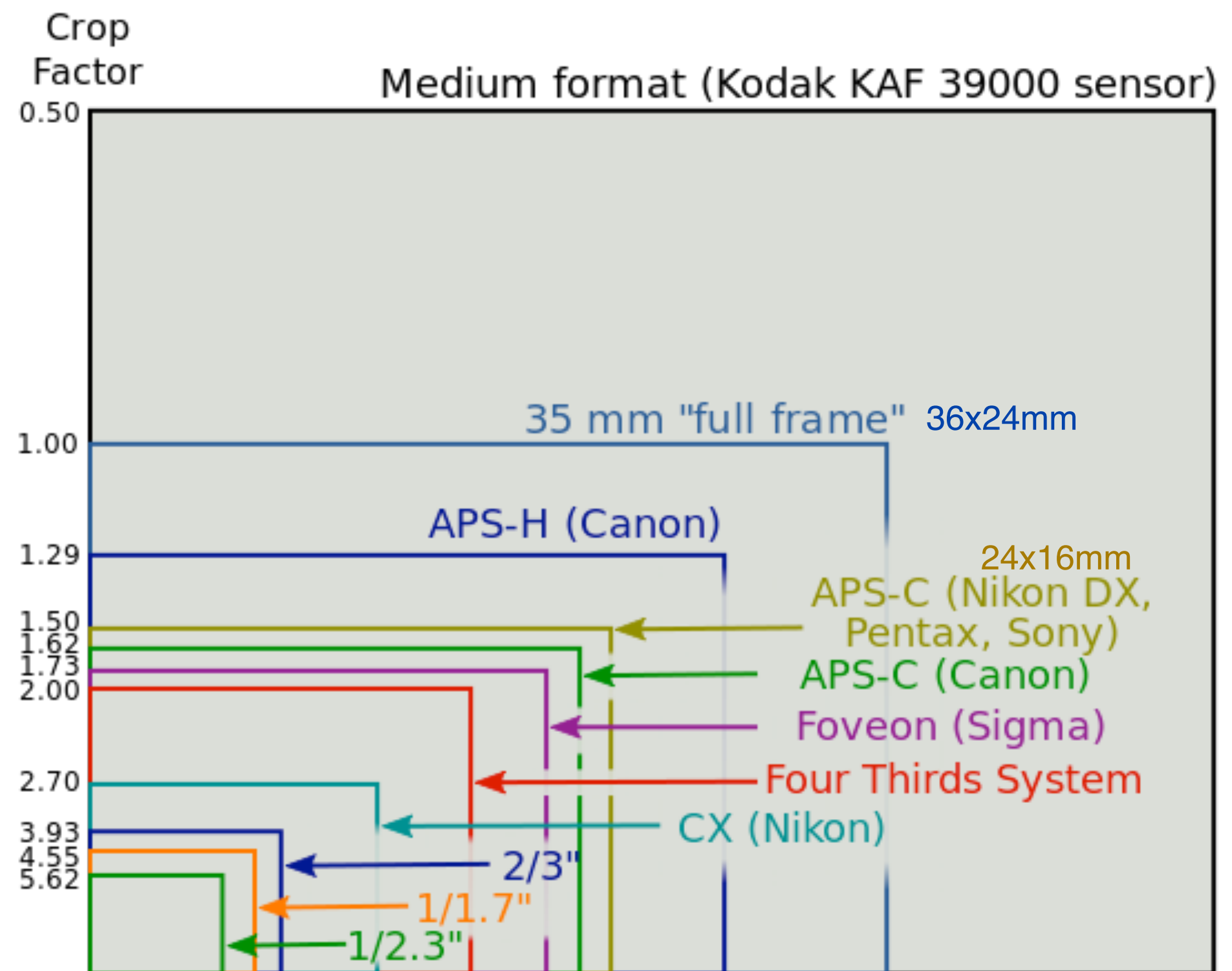


Oversaturated pixels



# Bigger sensors = bigger pixels (or more pixels?)

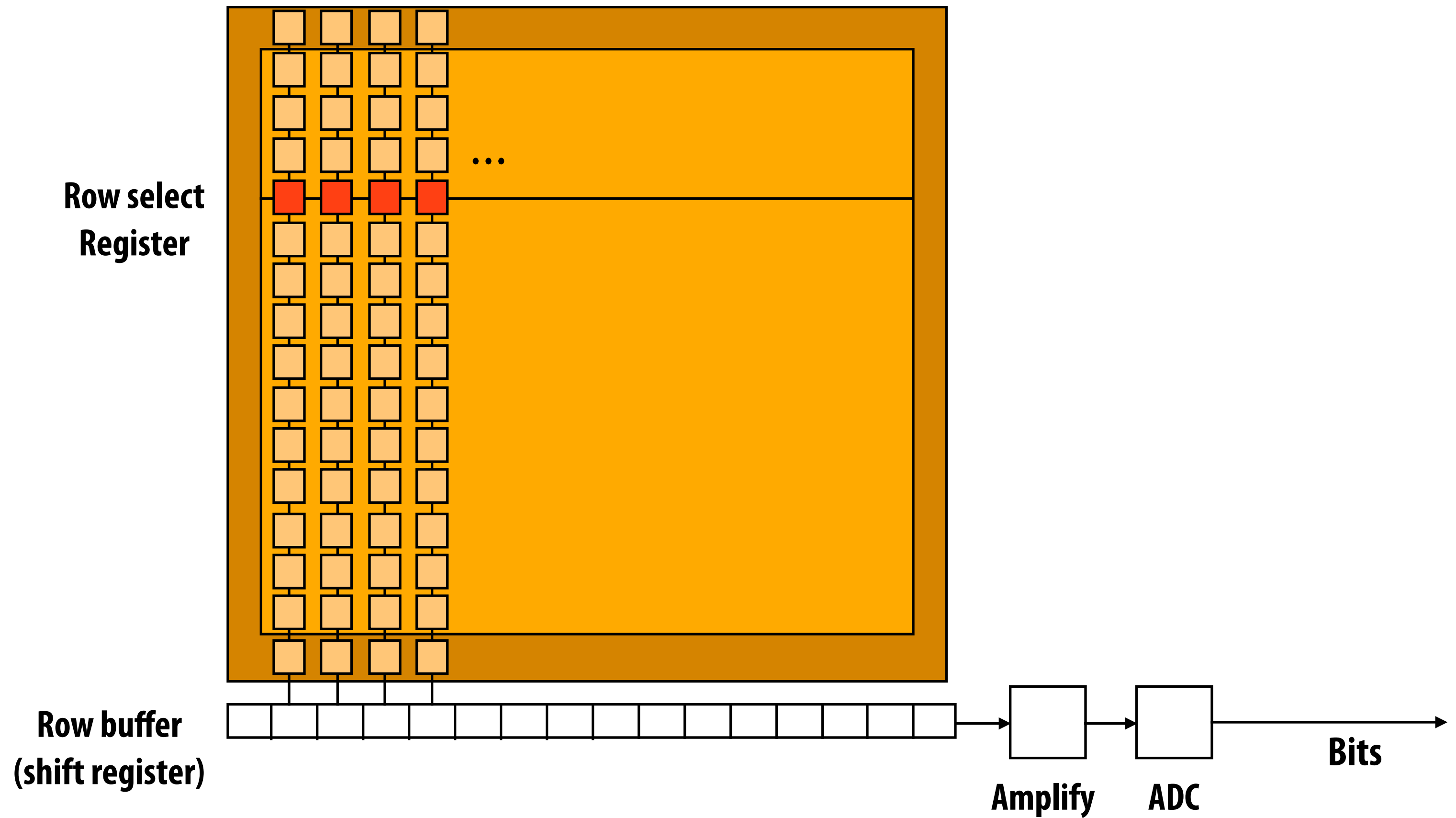
- iPhone 5s (1.5 micron pixels, 8 MP)
- My Nikon D7000 (APS-C) (4.8 micron pixels, 16 MP)
- Nikon D4 (full frame) (7.3 micron pixels, 16 MP)
- Implication: very high pixel count sensors can be built with current CMOS technology
  - Full frame sensor with iPhone 5s pixel size = 380 MP sensor



Nokia Lumia  
(41 MP)



# Reading sensed signal

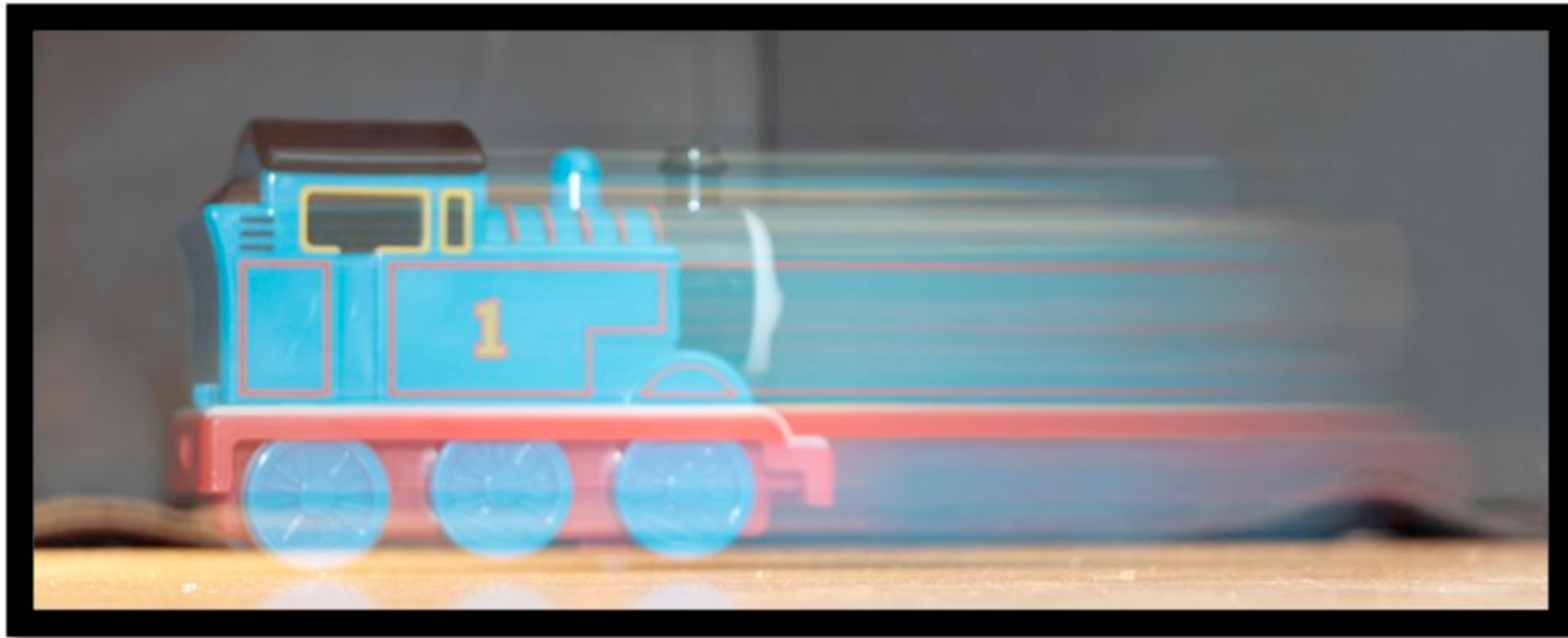


# Capturing an image

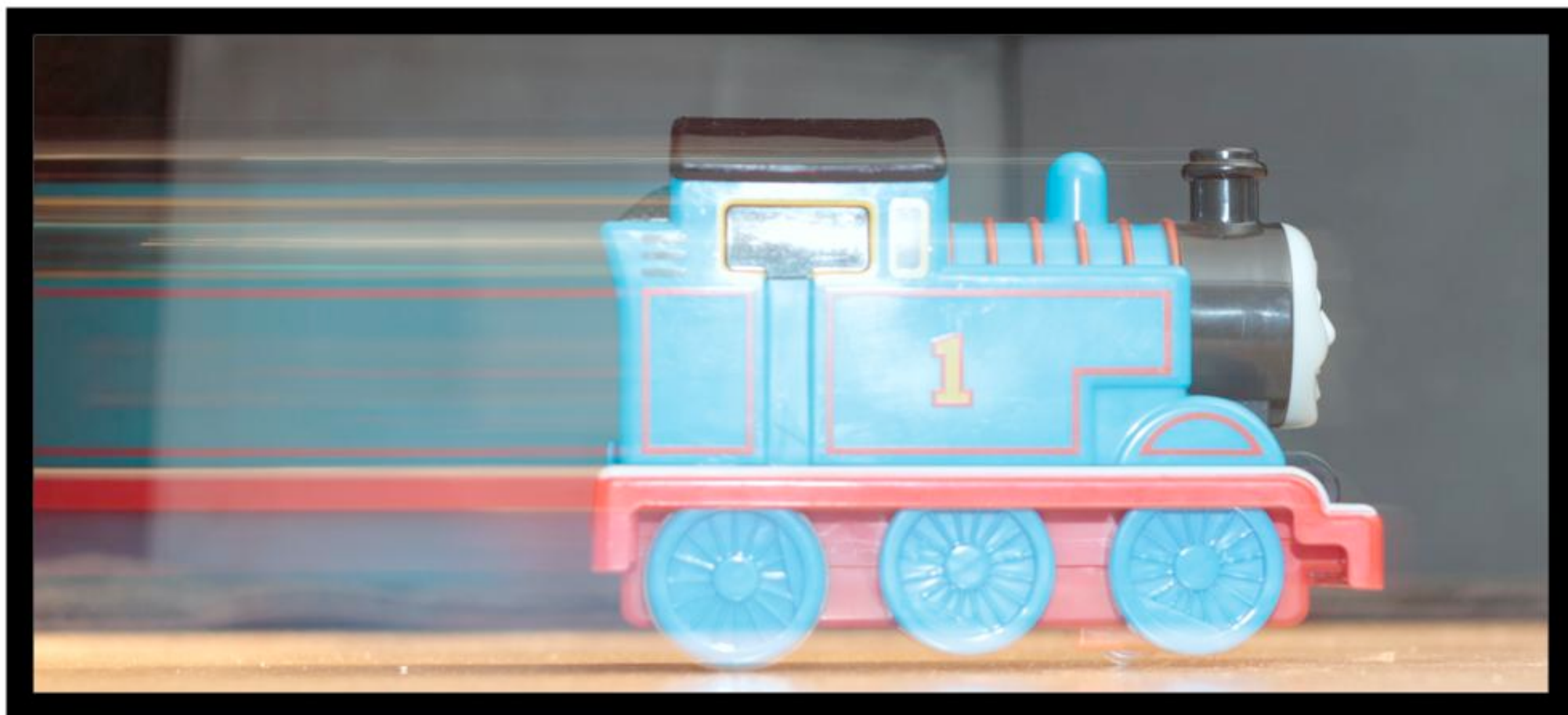
- 1. Clear sensor pixels**
- 2. Open camera mechanical shutter (exposure begins)**
- 3. Optional: fire flash**
- 4. Close camera mechanical shutter (exposure ends)**
- 5. Read results**
  - For each row:**
    - Read pixel for all columns in parallel**
    - Pass data stream through amplifier and DAC**



# Aside: when to fire flash?



First curtain sync



Second curtain sync

# Electronic rolling shutter

Many cameras do not have a mechanical shutter  
(e.g., smart-phone cameras)

- Exposure
1. Clear sensor pixels for row  $i$  (exposure begins)
  2. Clear sensor pixels for row  $i+1$  (exposure begins)
  - ...
  3. Read row  $i$  (exposure ends)
  4. Read row  $i+1$  (exposure ends)

Photo of red square, moving to right



Each image row exposed for the same amount of time (same exposure)

Each image row exposed over different interval of time  
(time offset determined by row read speed)

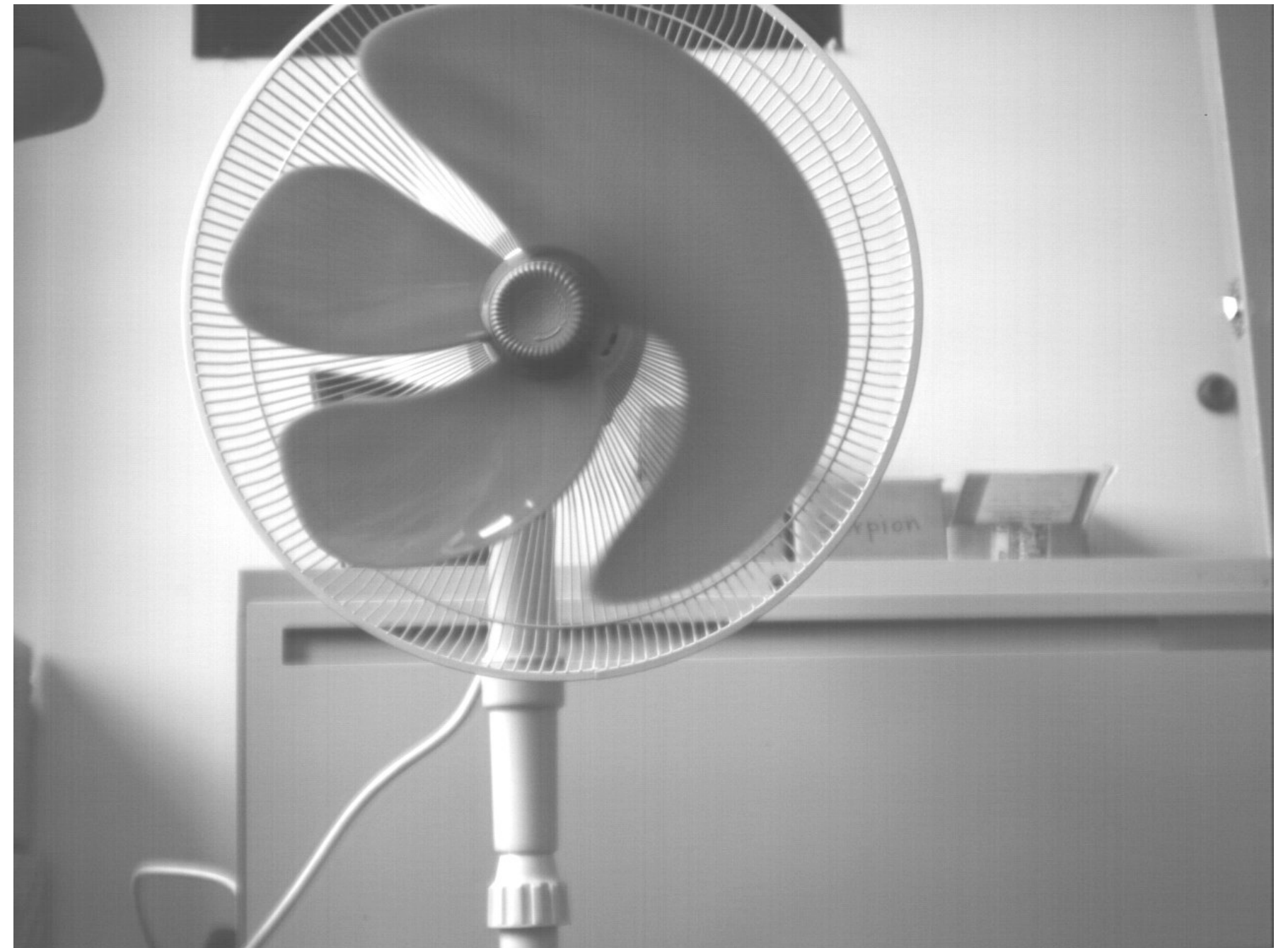


# Rolling shutter effects

**Demo: everyone take out camera phones**



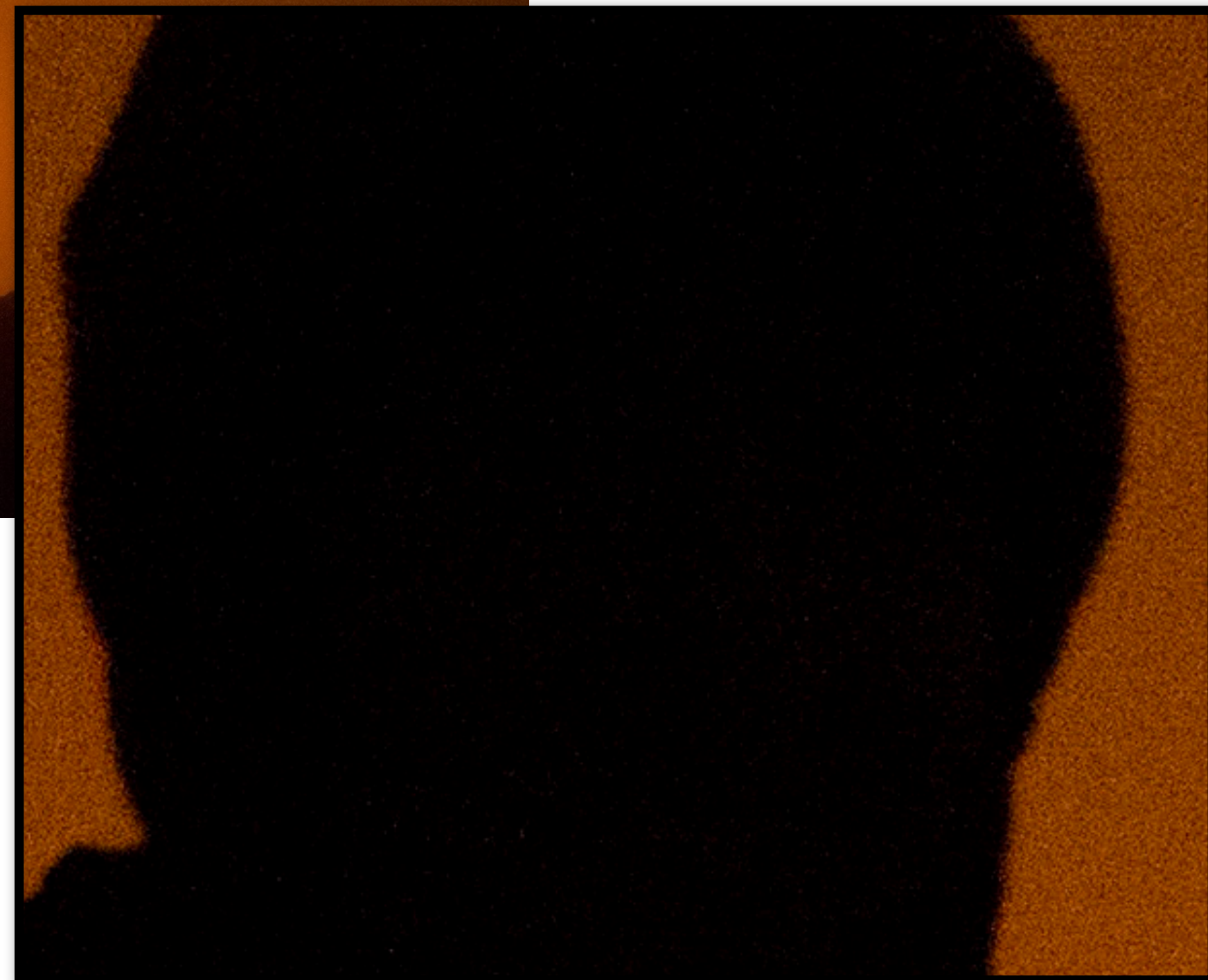
**Image credit: Wikipedia**



**Image credit: Point Grey Research**

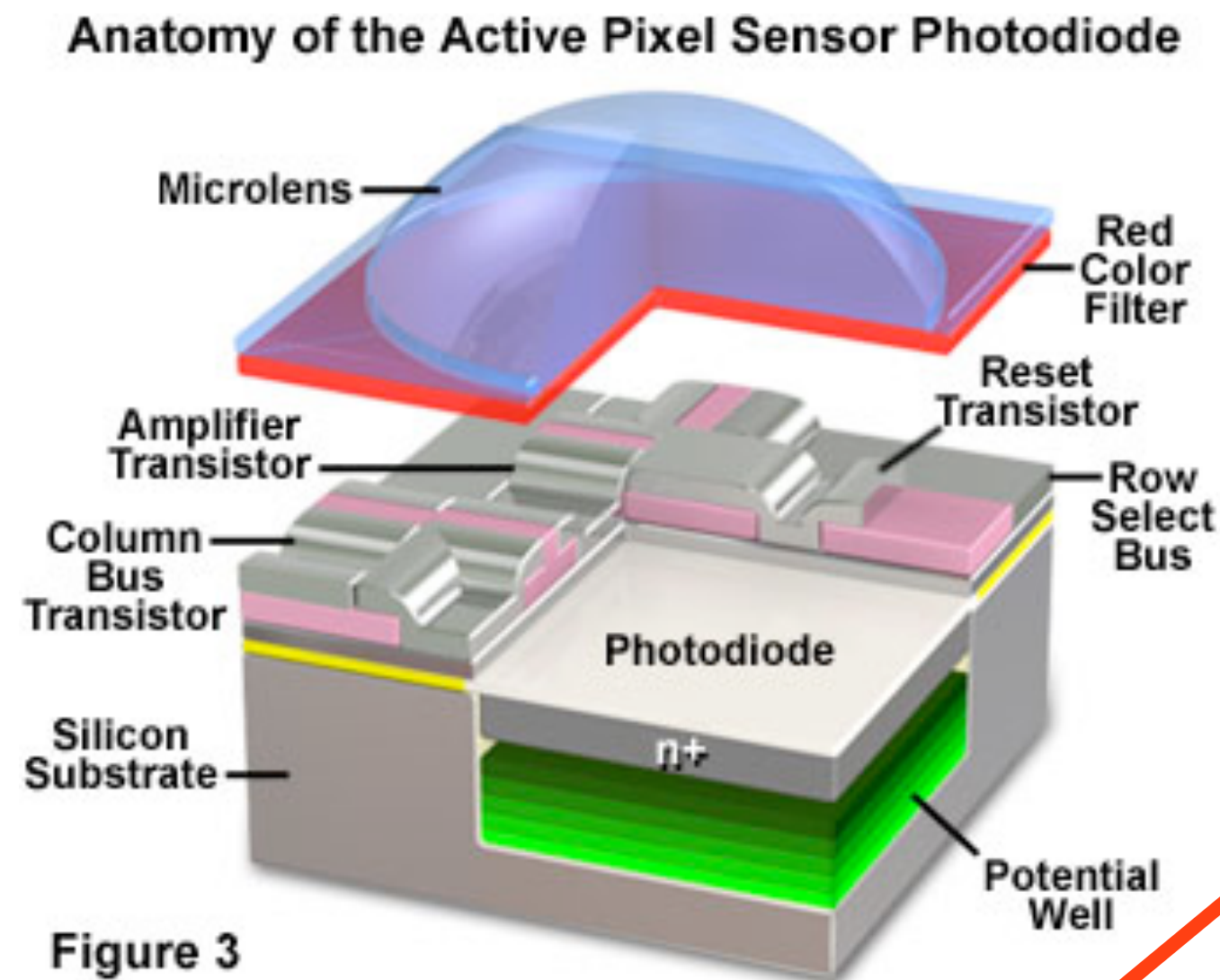


# Measurement noise





# Measurement noise



Addressed by: subtract dark image

Addressed by: flat field image

## ■ Photon shot noise:

- Photon arrival rates feature poisson distribution
- Standard deviation =  $\sqrt{N}$
- Signal-to-noise ratio:  $N/\sqrt{N}$

## ■ Dark-shot noise

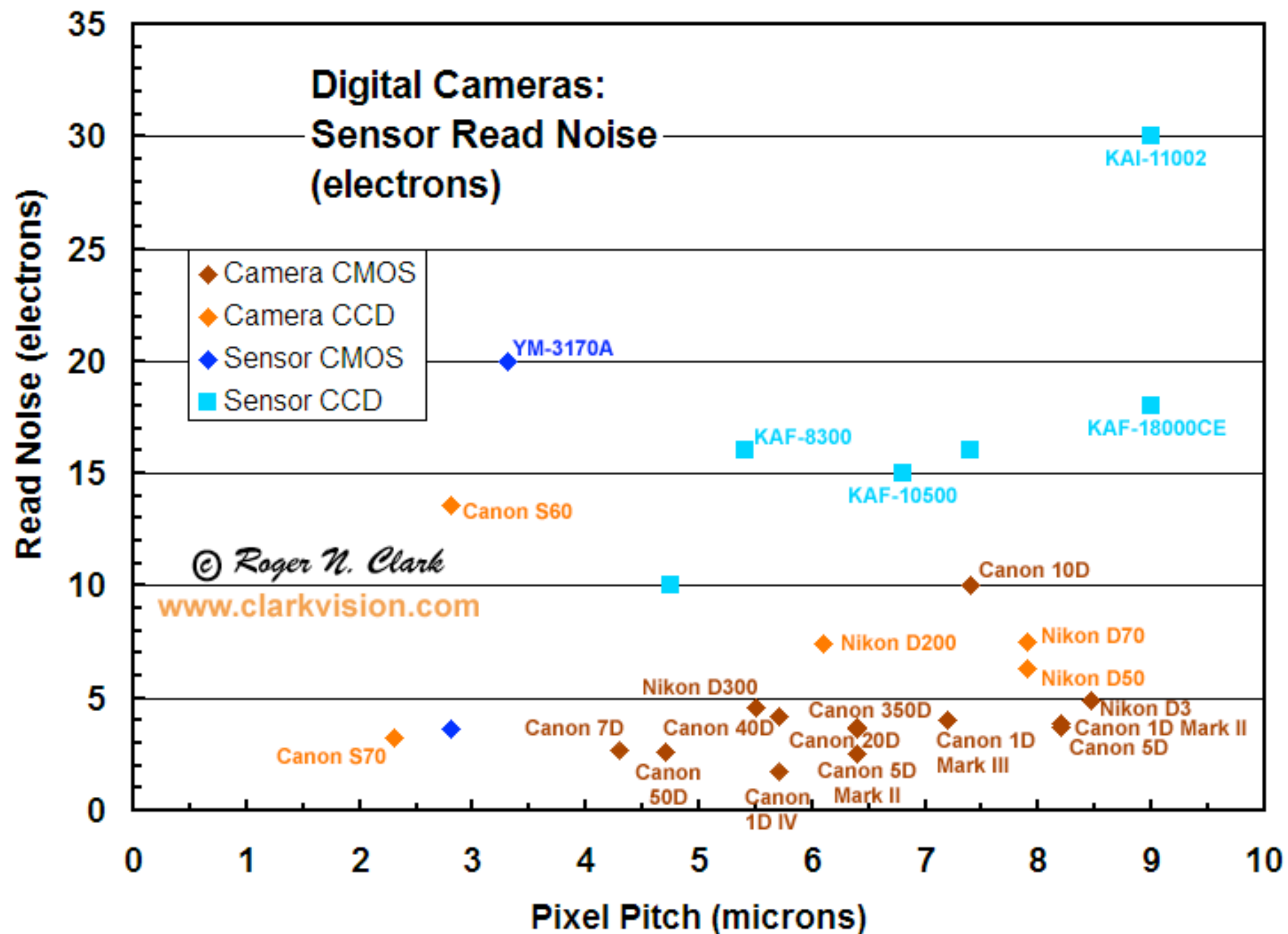
- Due to leakage current

## ■ Non-uniformity of pixel sensitivity

## ■ Read noise

- e.g., due to amplification

# Read noise



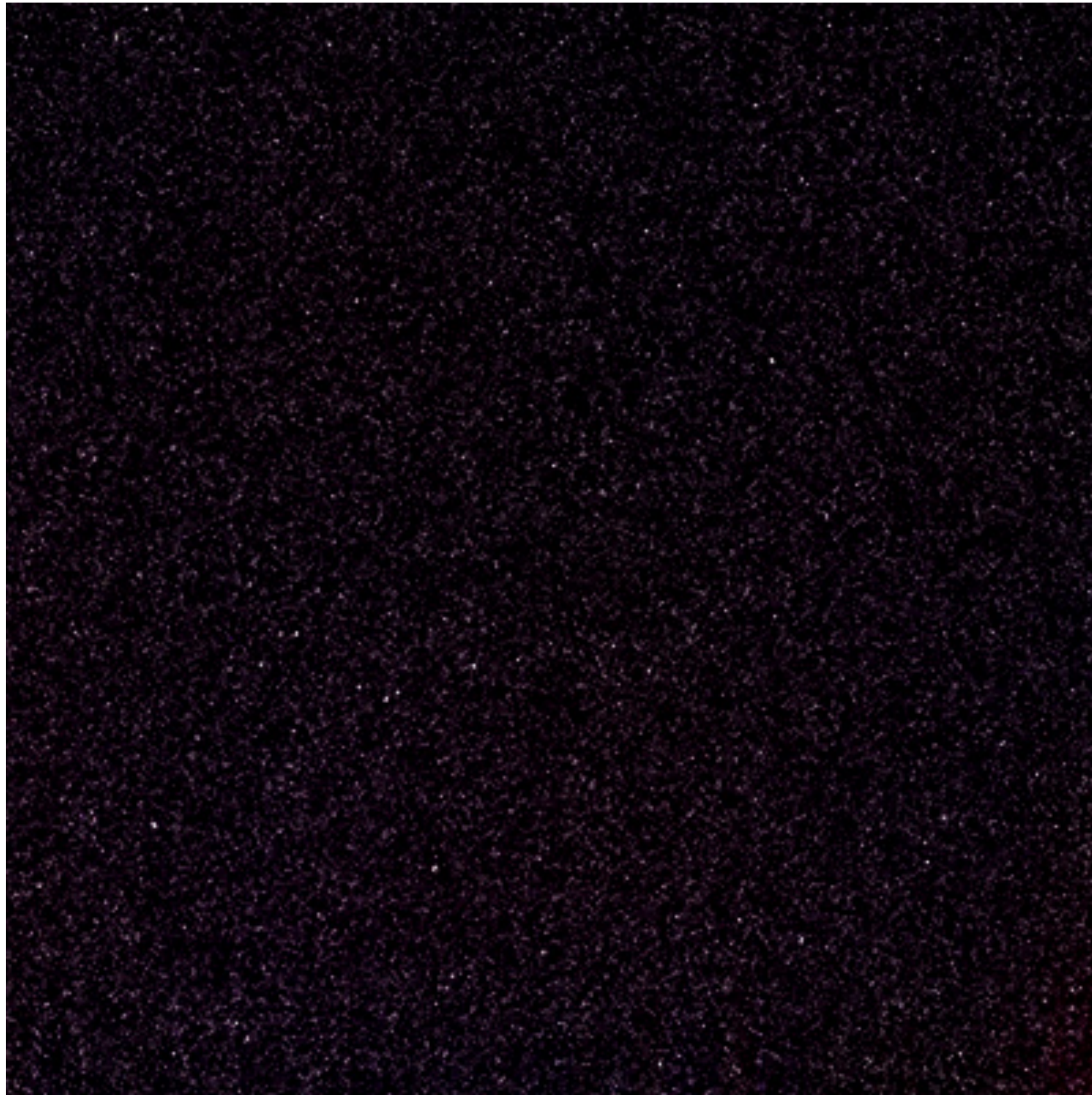
**Read noise is largely independent of pixel size**

**Large pixels, bright scene: noise determined largely by photon shot noise**



# Noise

**Black image examples: Nikon D7000, High ISO**



**1/60 sec exposure**



**1 sec exposure**



# Maximize light gathering capability

## ■ Goal: increase signal-to-noise ratio

- Dynamic range of a pixel determined by noise floor (minimum signal) and full-well capacity (maximum signal)

## ■ Big pixels

- Nikon D4: 7.3  $\mu\text{m}$
- iPhone 5s: 1.5  $\mu\text{m}$

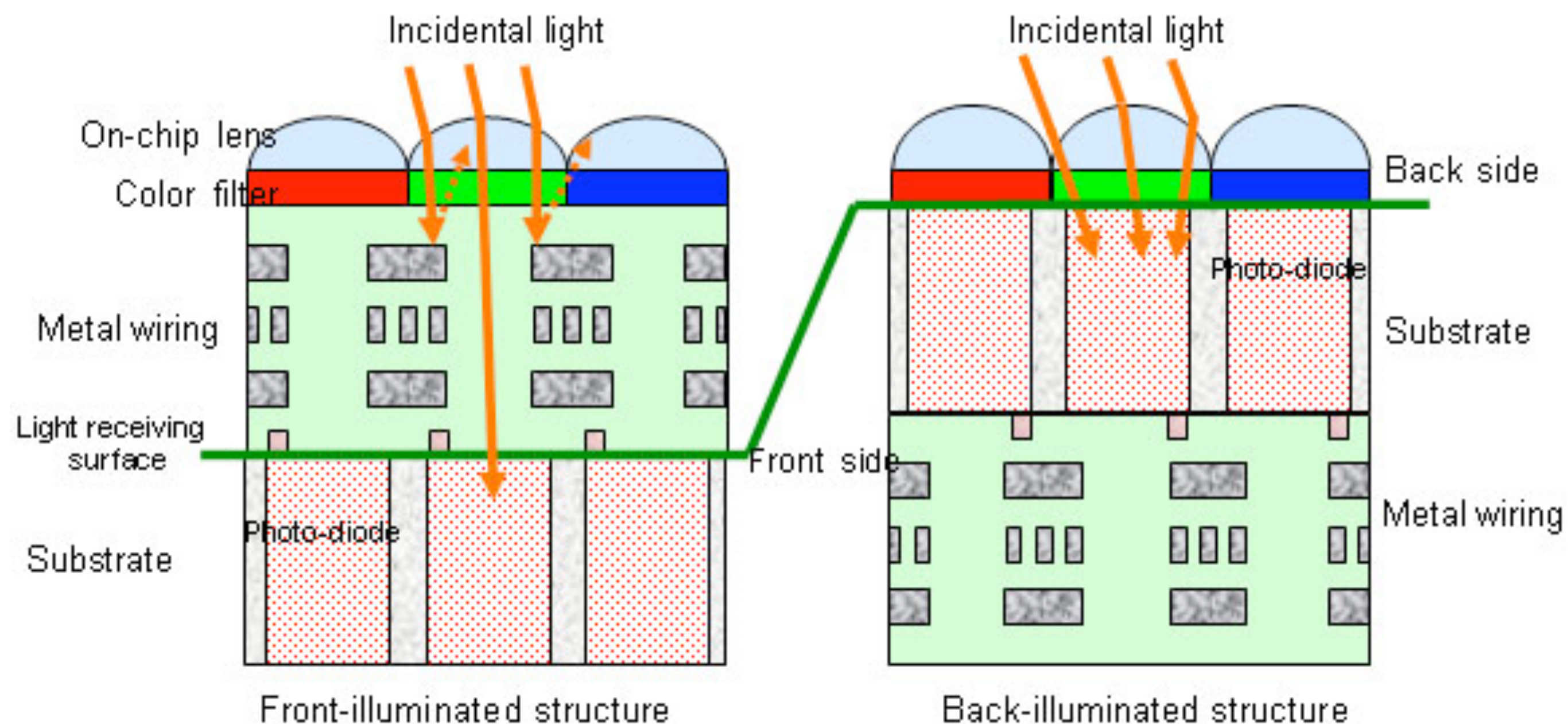
## ■ Sensitive pixels

- Good materials
- High fill factor



# Backside illumination sensor

- **Traditional CMOS: electronics block light**
- **Idea: move electronics underneath light gathering region**
  - **Increases fill factor**
  - **Implication 1: better light sensitivity at fixed sensor size**
  - **Implication 2: equal light sensitivity at smaller sensor size (shrink sensor)**



# Vignetting

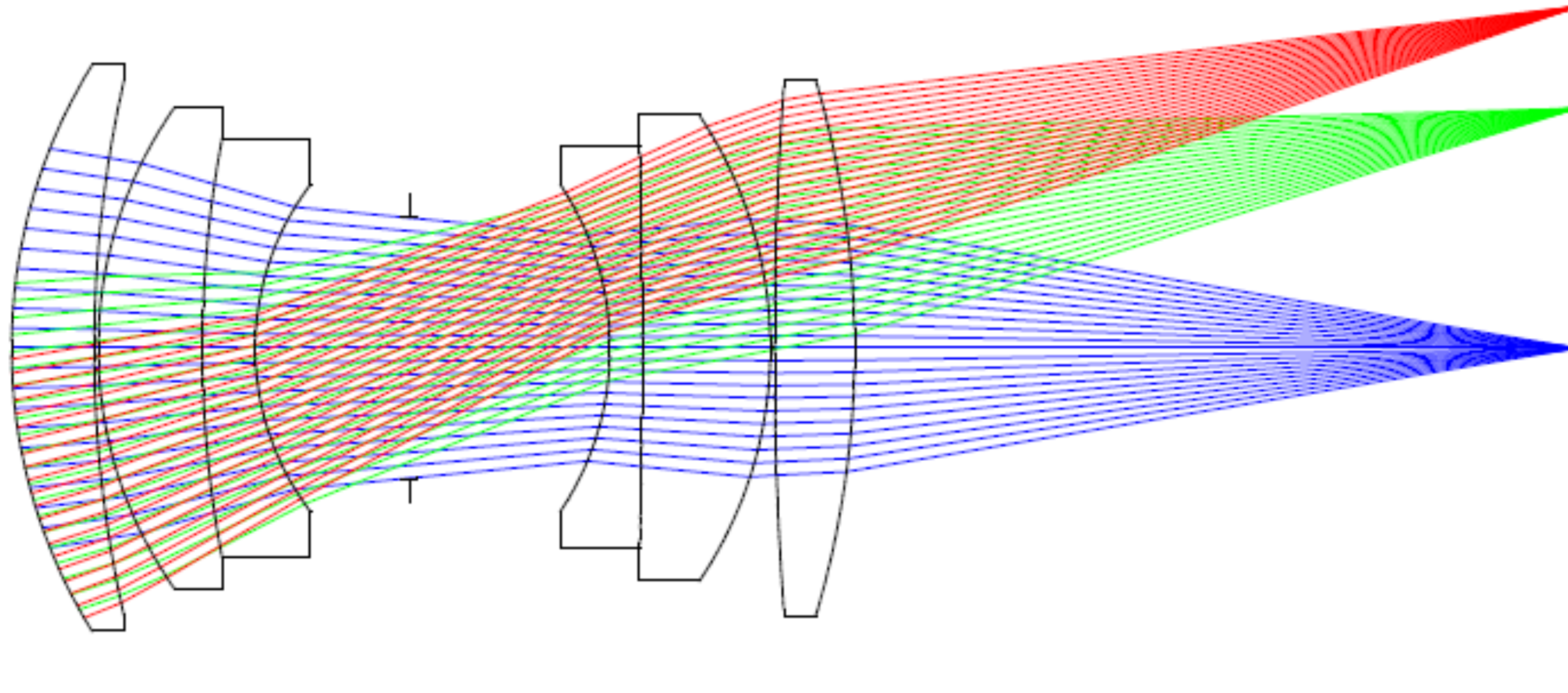
**Image of white wall (contrast enhanced to show effect)**





# Types of vignetting

**Optical vignetting: less light reaches edges of sensor due to physical obstruction in lens**



**Pixel vignetting: light reaching pixel at an oblique angle is less likely to hit photosensitive region than light incident from straight above (e.g., obscured by electronics)**

- Microlens reduces pixel vignetting

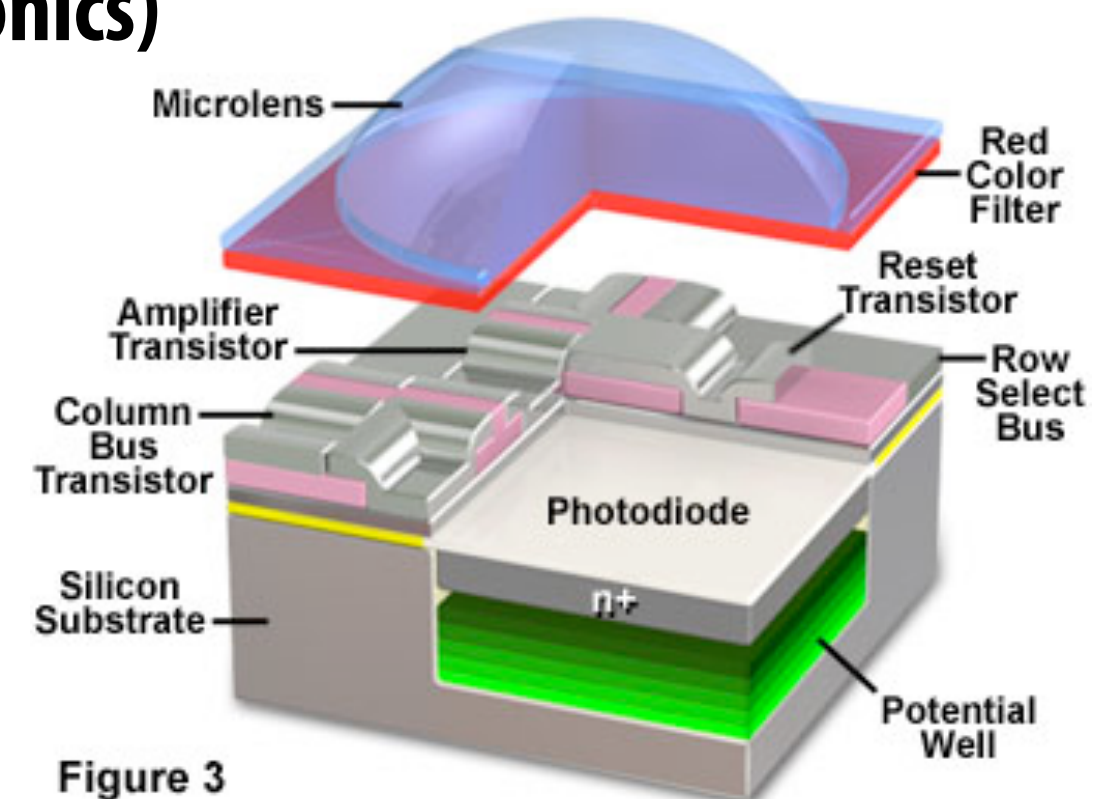


Figure 3



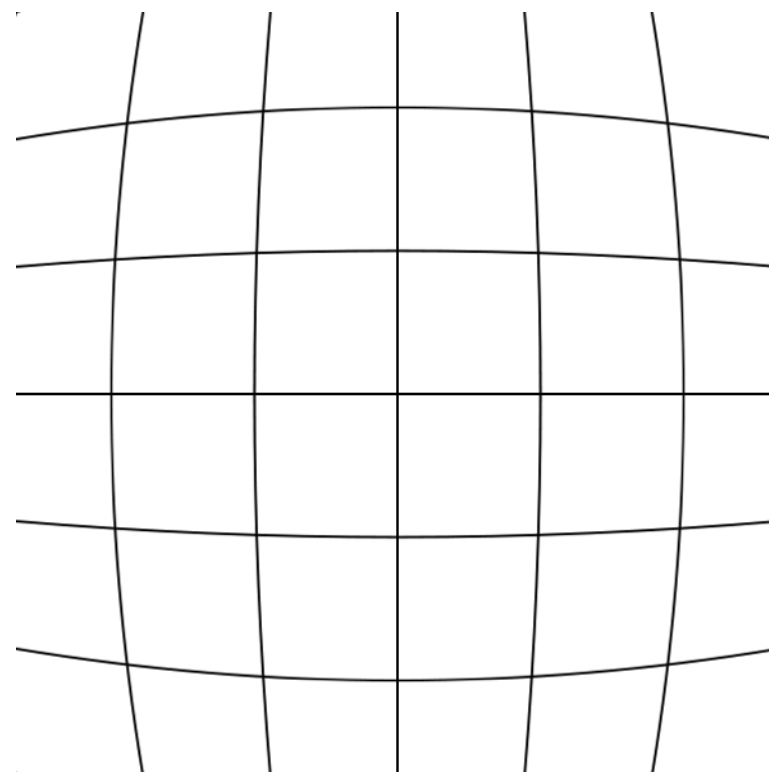
# More challenges

- **Chromatic shifts over sensor**

- Pixel light sensitivity changes over sensor due to interaction with microlens  
(index of refraction depends on wavelength)

- **Dead pixels (stuck at white or black)**

- **Lens distortion**



**Pincushion distortion**



**Captured Image**



**Corrected Image**

Image credit: PCWorld



**Theme so far: directly displaying bits off the sensor would  
not yield an acceptable photograph**

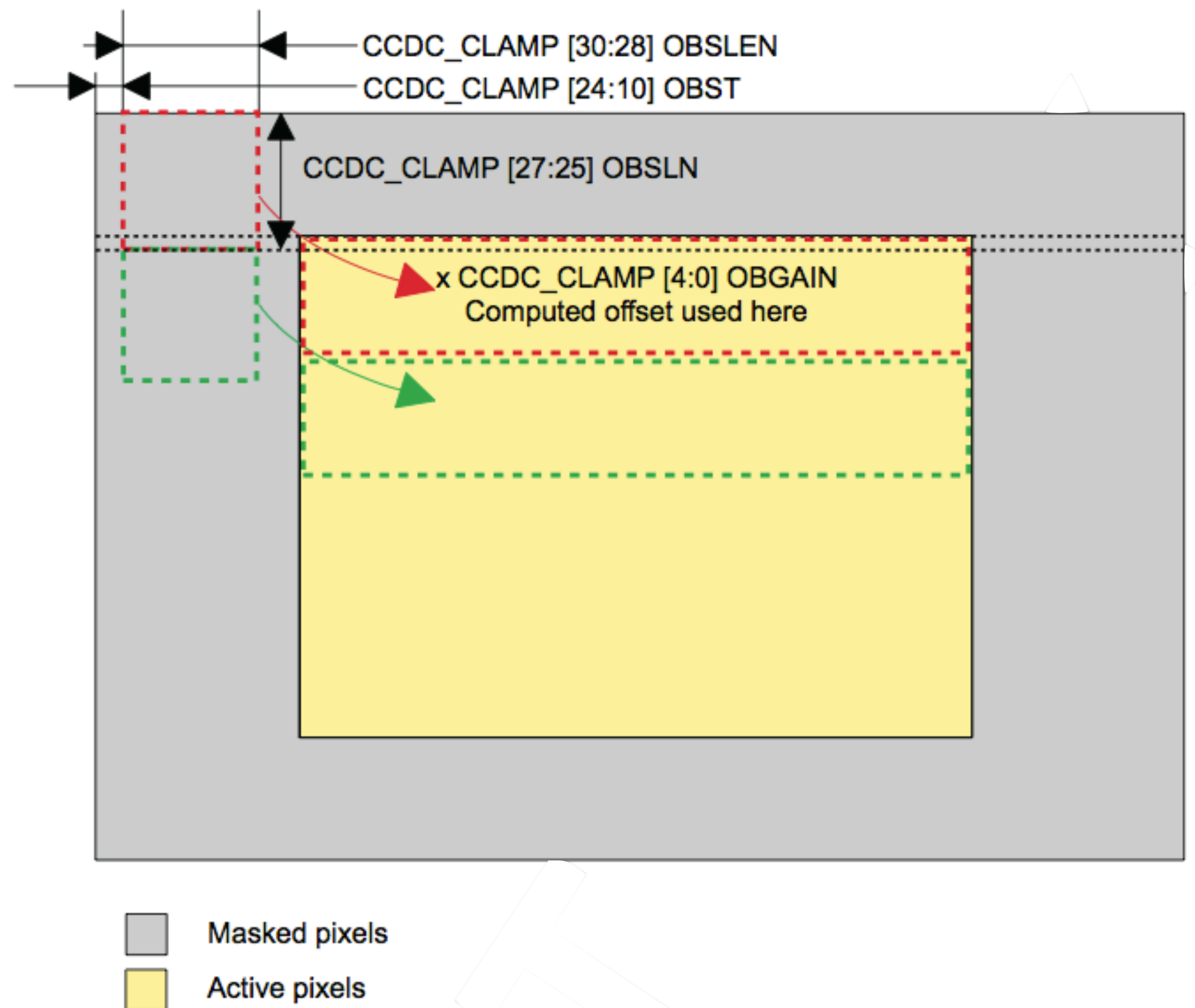
# **RAW image processing**

# Example image processing pipeline

- **Adopting terminology from Texas Instruments OMAP Image Signal Processor pipeline (public documentation exists)**
- **Assume: receiving 12 bits/pixel Bayer mosaiced data from sensor**

# Optical clamp: remove sensor offset bias

$\text{output\_pixel} = \text{input\_pixel} - [\text{average of pixels from optically black region}]$



**Remove bias due to sensor black level  
(from nearby sensor pixels at time of shot)**

# Step 2: correct for defective pixels

## ■ Store LUT with known defect pixels

- e.g., determined on manufacturing line, during sensor calibration and test

## ■ Example correction methods

- Replace defect with neighbor
- Replace defect with average of neighbors
- Correct defect by subtracting known bias for the defect

```
output_pixel = (isdefect(current_pixel_xy)) ?  
                average(previous_input_pixel, next_input_pixel) :  
                input_pixel;
```



# Lens shading compensation

- **Correct for vignetting**
- **Possible implementations:**
  - **Use 2D buffer stored in memory**
    - Lower resolution buffer, upsampled on-the-fly
  - **Use analytic function**

```
offset = upsample_compensation_offset_buffer(current_pixel_xy);  
gain = upsample_compensation_gain_buffer(current_pixel_xy);  
  
output_pixel = offset + gain * input_pixel;
```

# Optional dark-frame subtraction

- **Similar computation to lens shading compensation**

```
output_pixel = input_pixel - dark_frame[current_pixel_xy];
```

# White balance

- Adjust relative intensity of rgb values (so neutral tones appear neutral)

```
output_pixel = white_balance_coeff * input_pixel
```

```
// note: in this example, white_balance_coeff is vec3  
// (adjusts ratio of red-blue-green channels)
```

- Determine white balance coefficients based on analysis of image contents:

- Example simple auto-white balance algorithms
  - Gray world assumption: make average of all pixels gray
  - Find brightest region of image, make it white

- Modern cameras have sophisticated (heuristic-based) white-balance algorithms

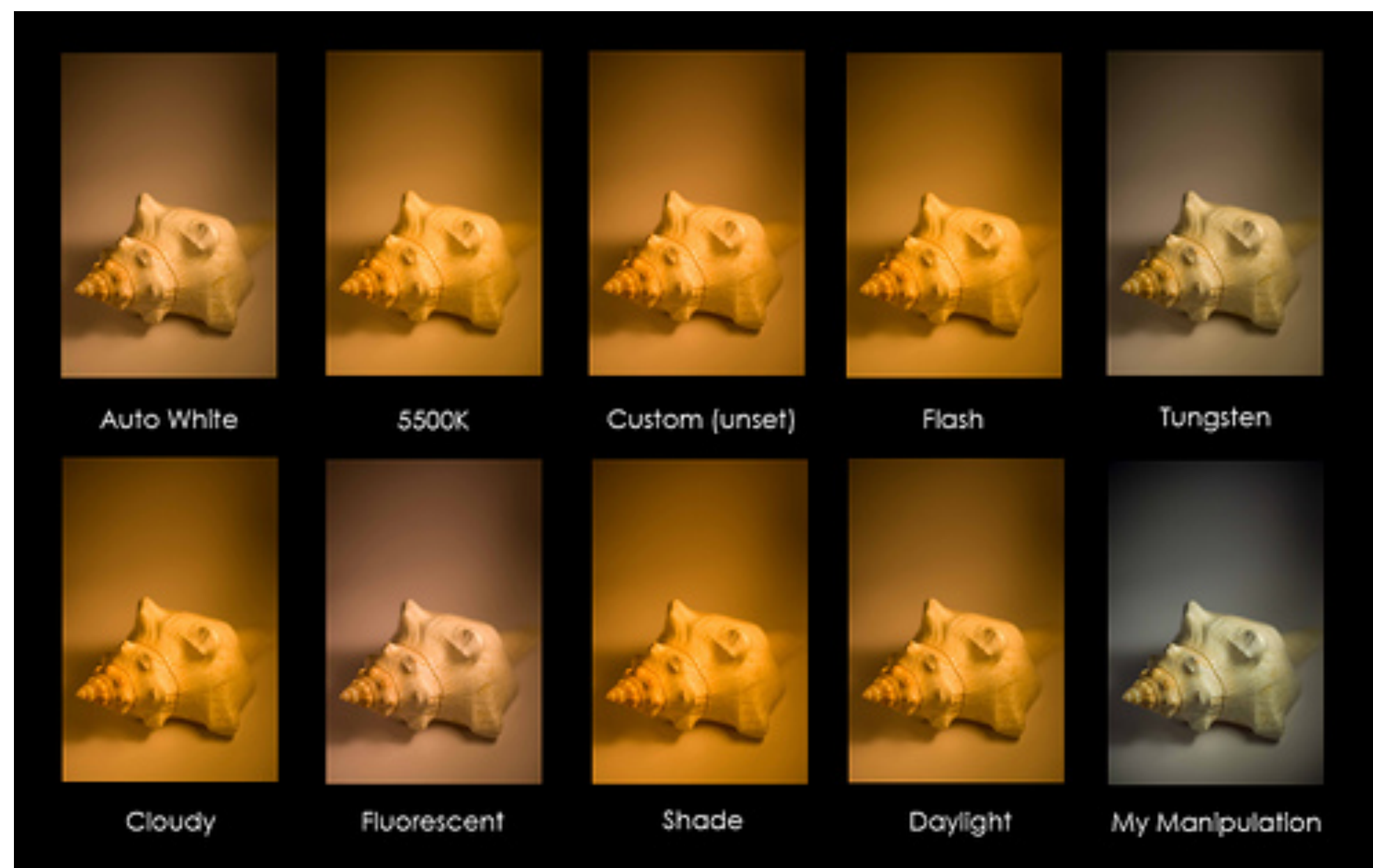
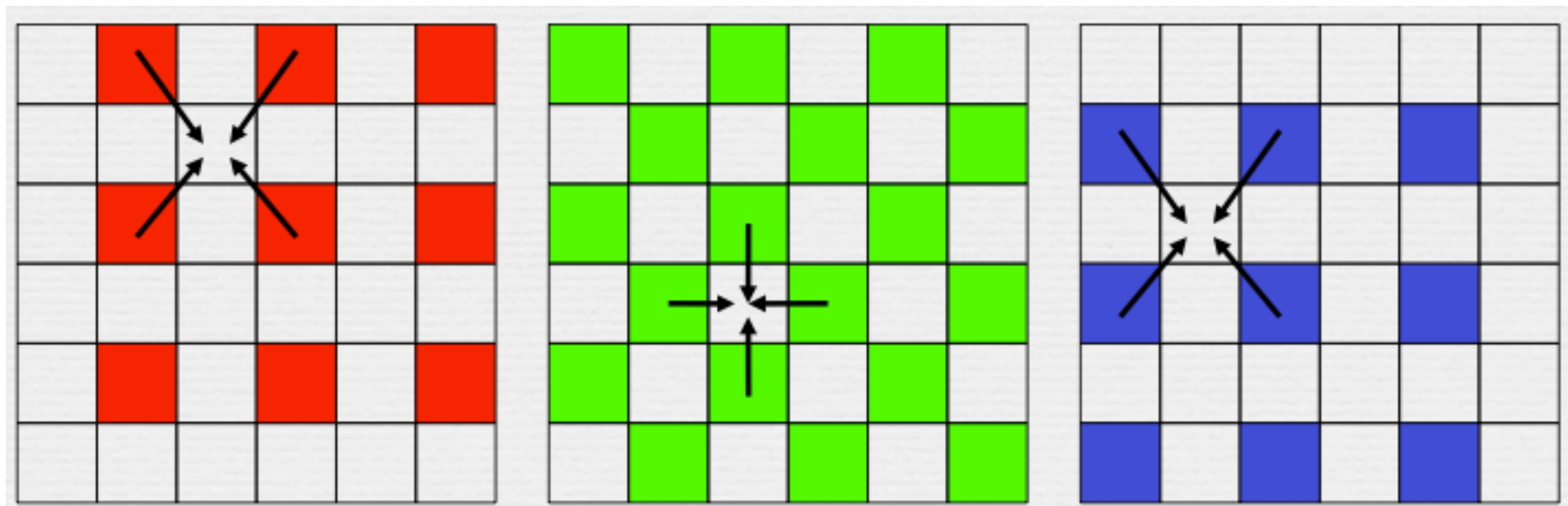


Image credit: basedigitalphotography.com



# Demosiatic

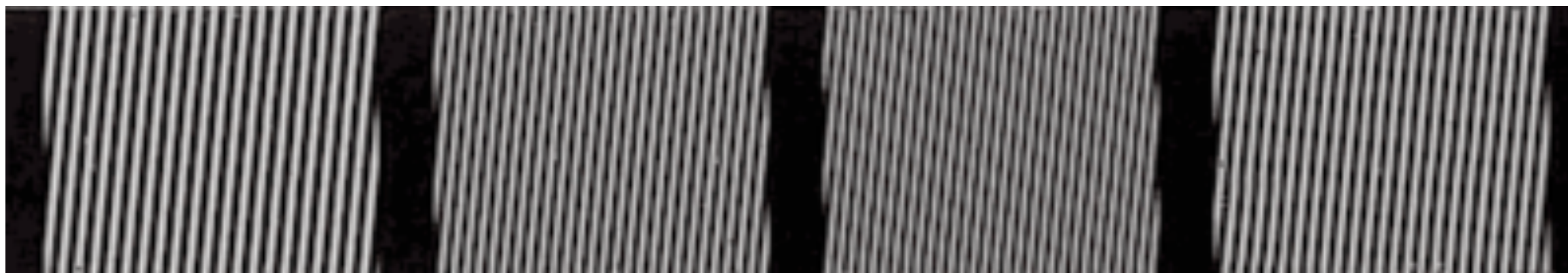
- Produce RGB image from mosaiced input image
- Basic algorithm: bilinear interpolation of mosaiced values (need 4 neighbors)
- More advanced algorithms:
  - Bibubic interpolation (wider filter support region... may overblur)
  - Good implementations attempt to find and preserve edges



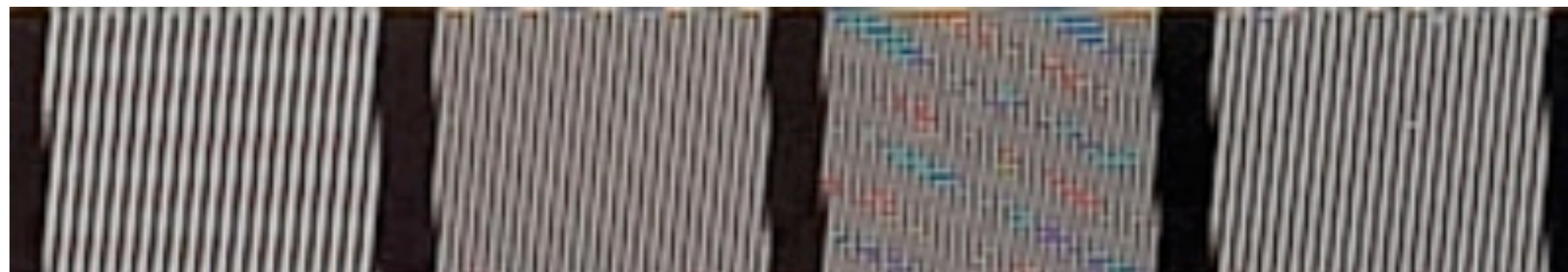
# Demosaicing errors

## ■ Moire pattern color artifacts

- Common trigger: fine diagonal black and white stripes
- Common solution:
  - Convert demosaiced value to YCbCr
  - Prefilter CbCr channels
  - Combine prefiltered CbCr with full resolution Y from sensor to get RGB



RAW data from sensor



Demosaiced



# Denoising

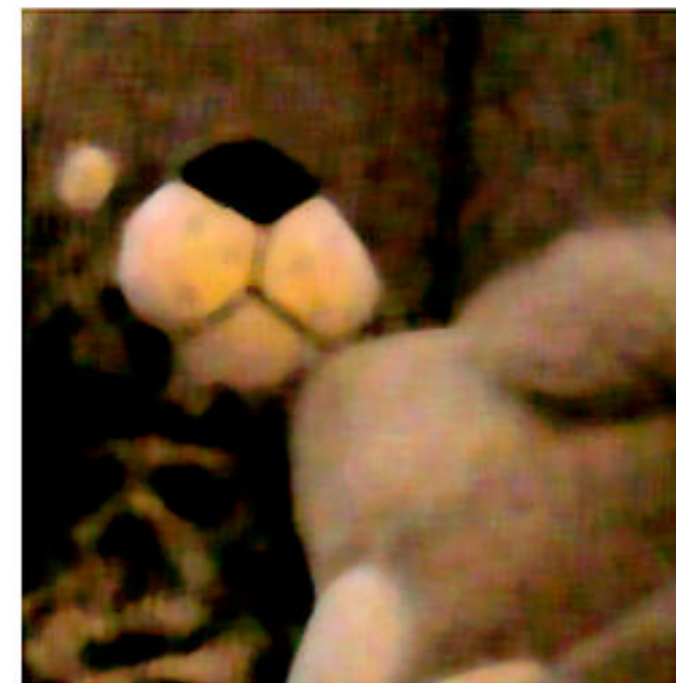
**Median Filter**  
 $\text{output\_pixel} = \text{median of neighboring pixels}$



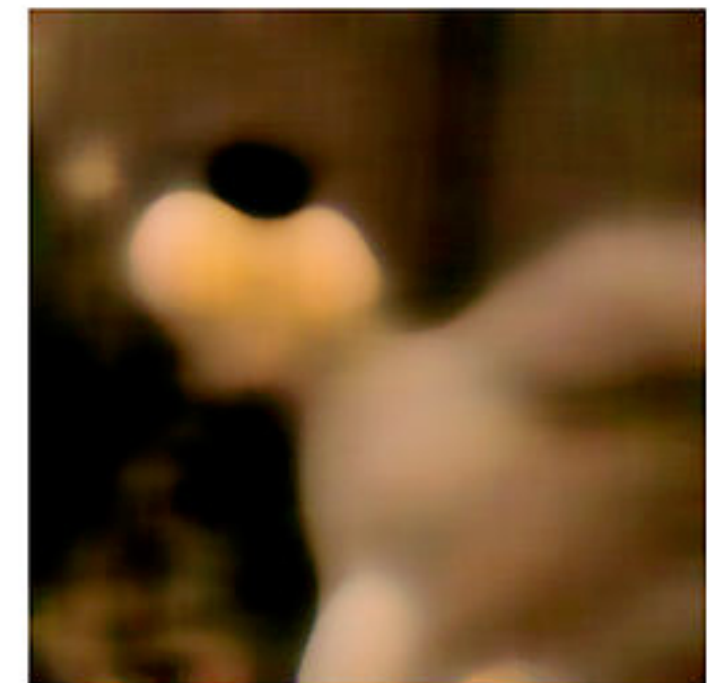
original image



1px median filter



3px median filter



10px median filter



**Bilateral filter: remove noise while preserving edges**



# Simplified image processing pipeline

- Correct for sensor bias (using measurements of optically black pixels)
- Correct pixel defects
- Vignetting compensation
- Dark frame subtract (optional)
- White balance
- Demosaic
- Denoise / sharpen, etc.

lossless compression

RAW file

**Today**

- Color Space Conversion
- Gamma Correction
- Color Space Conversion (Y'CbCr)
- 4:4:4 to 4:2:2 chroma subsampling
- JPEG compress (lossy)

JPEG file