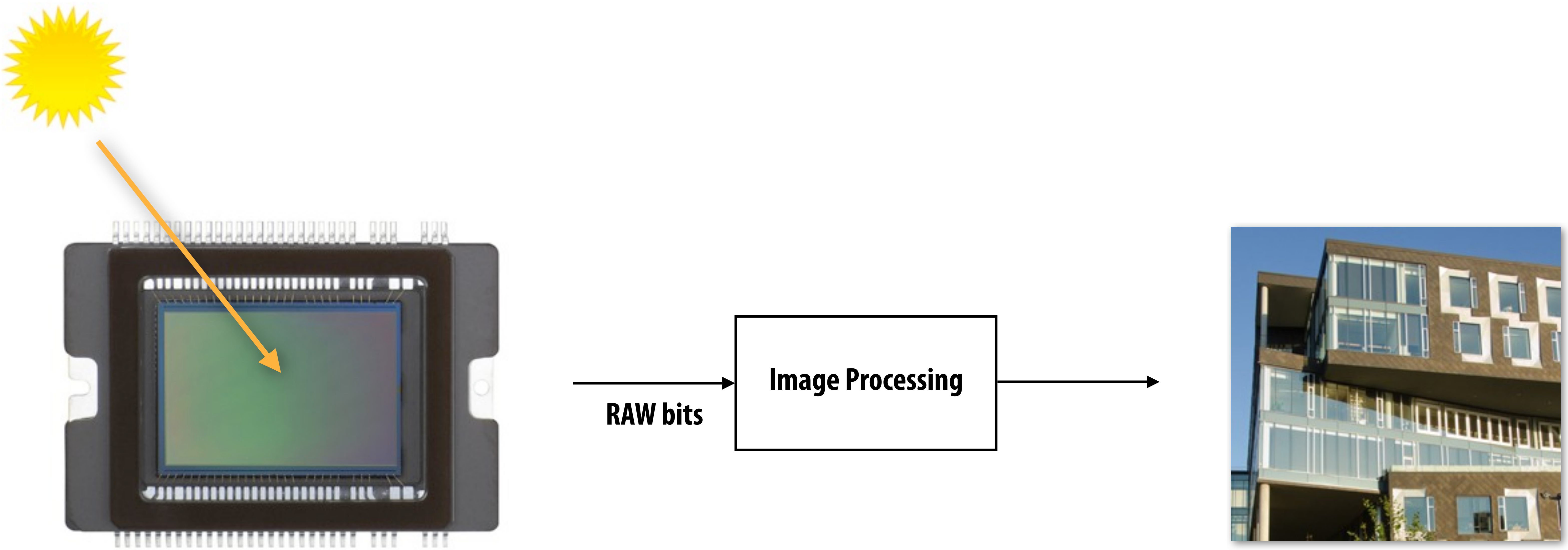
Lecture 16:
A Camera's Image Processing Pipeline
Part 1

Kayvon Fatahalian

CMU 15-869: Graphics and Imaging Architectures (Fall 2011)

Today (actually all week)

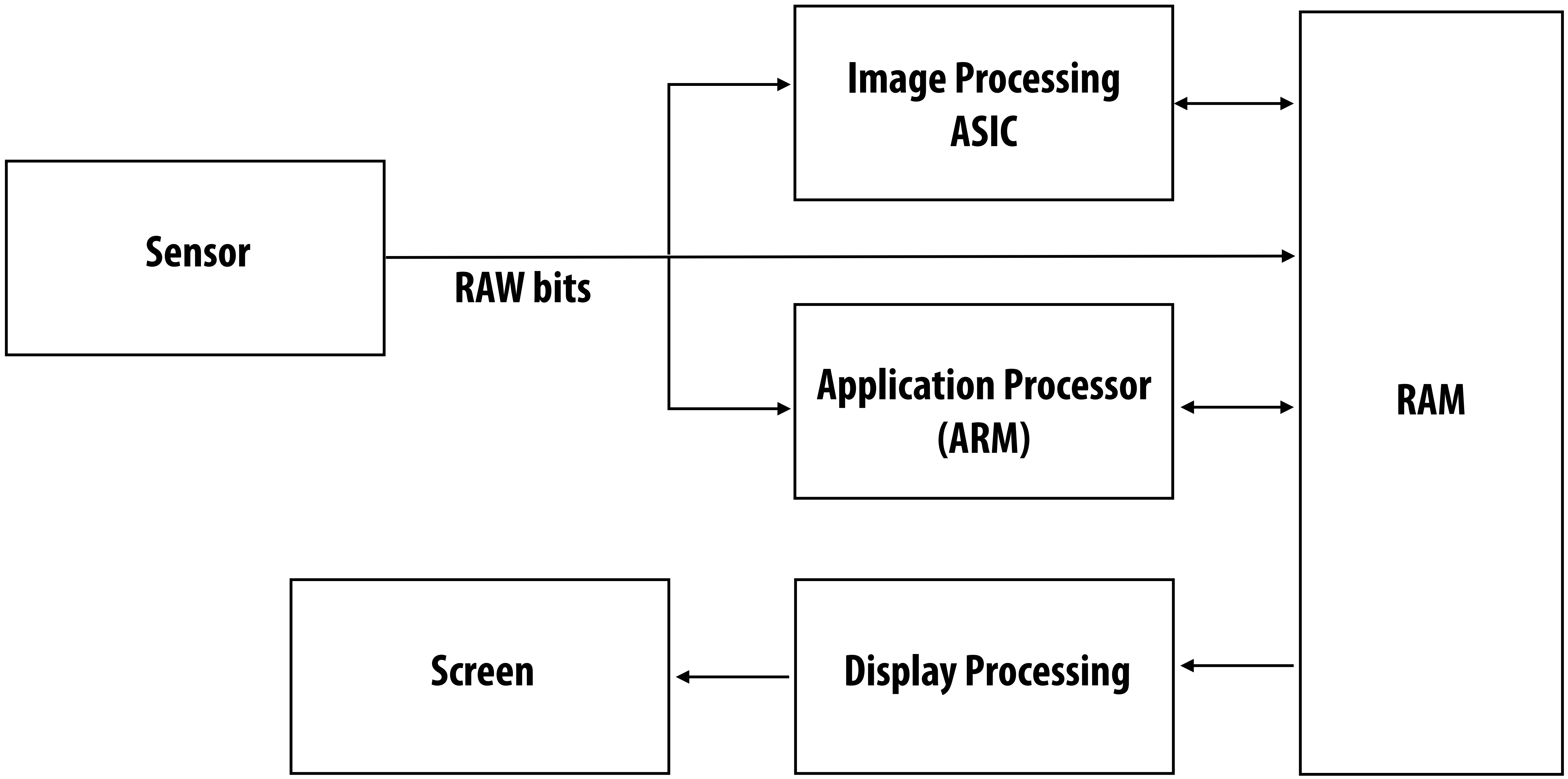
Operations that take photons to an image
Processing systems used to efficiently implement these operations



Canon 14 MP CMOS Sensor
(14 bits per pixel)

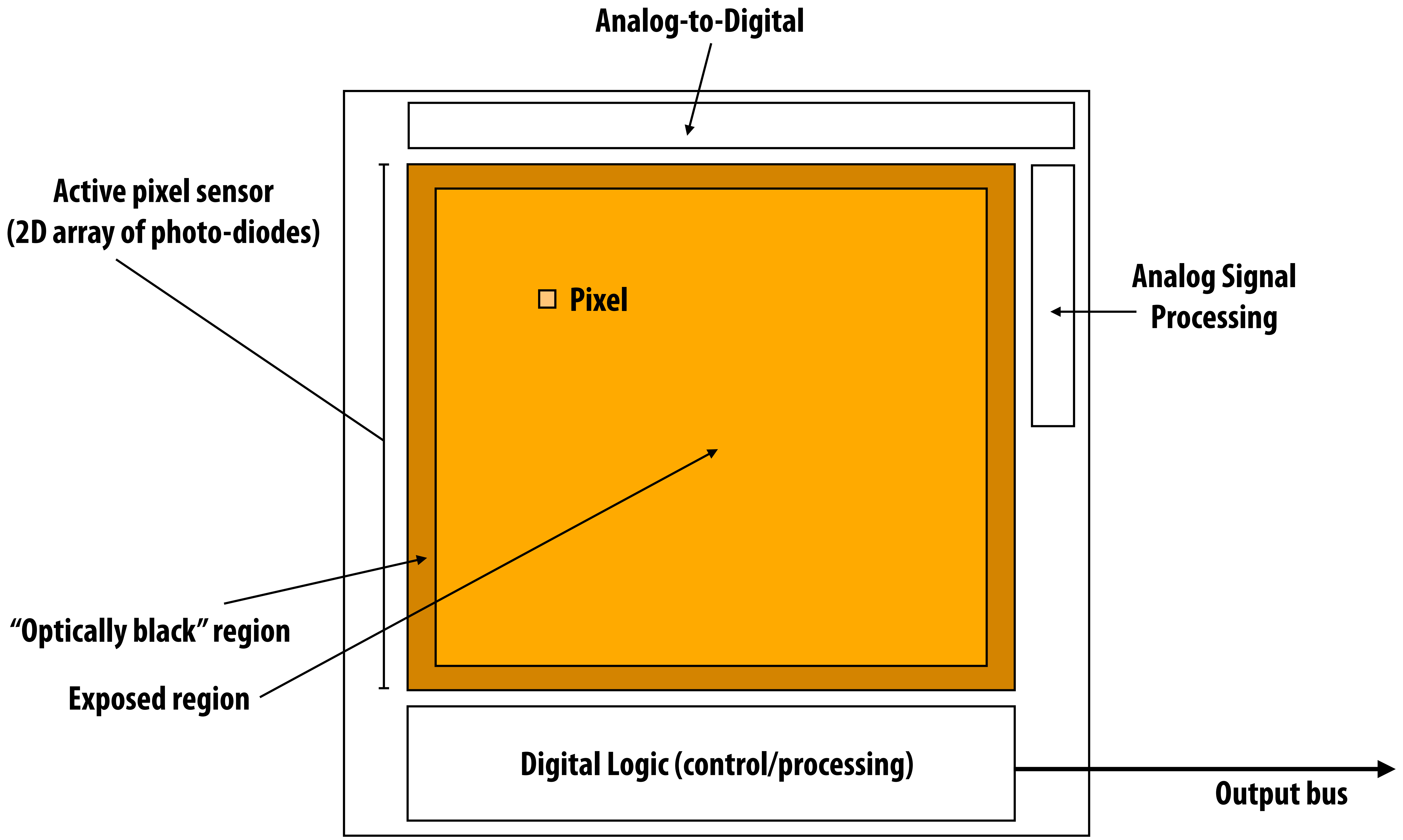
Final Image Representation
(e.g., JPG file)

Generic camera: system overview



The Sensor

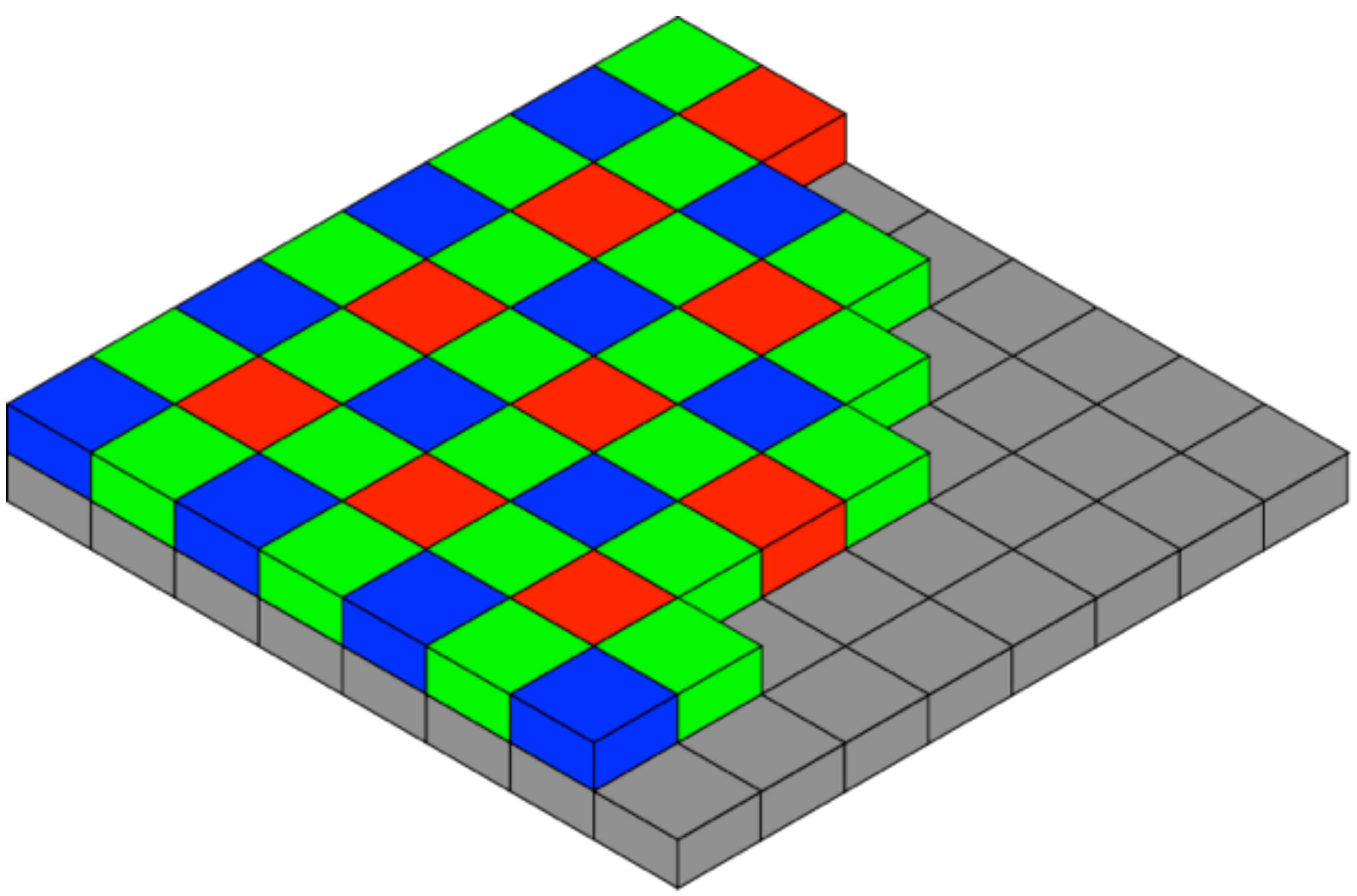
CMOS sensor



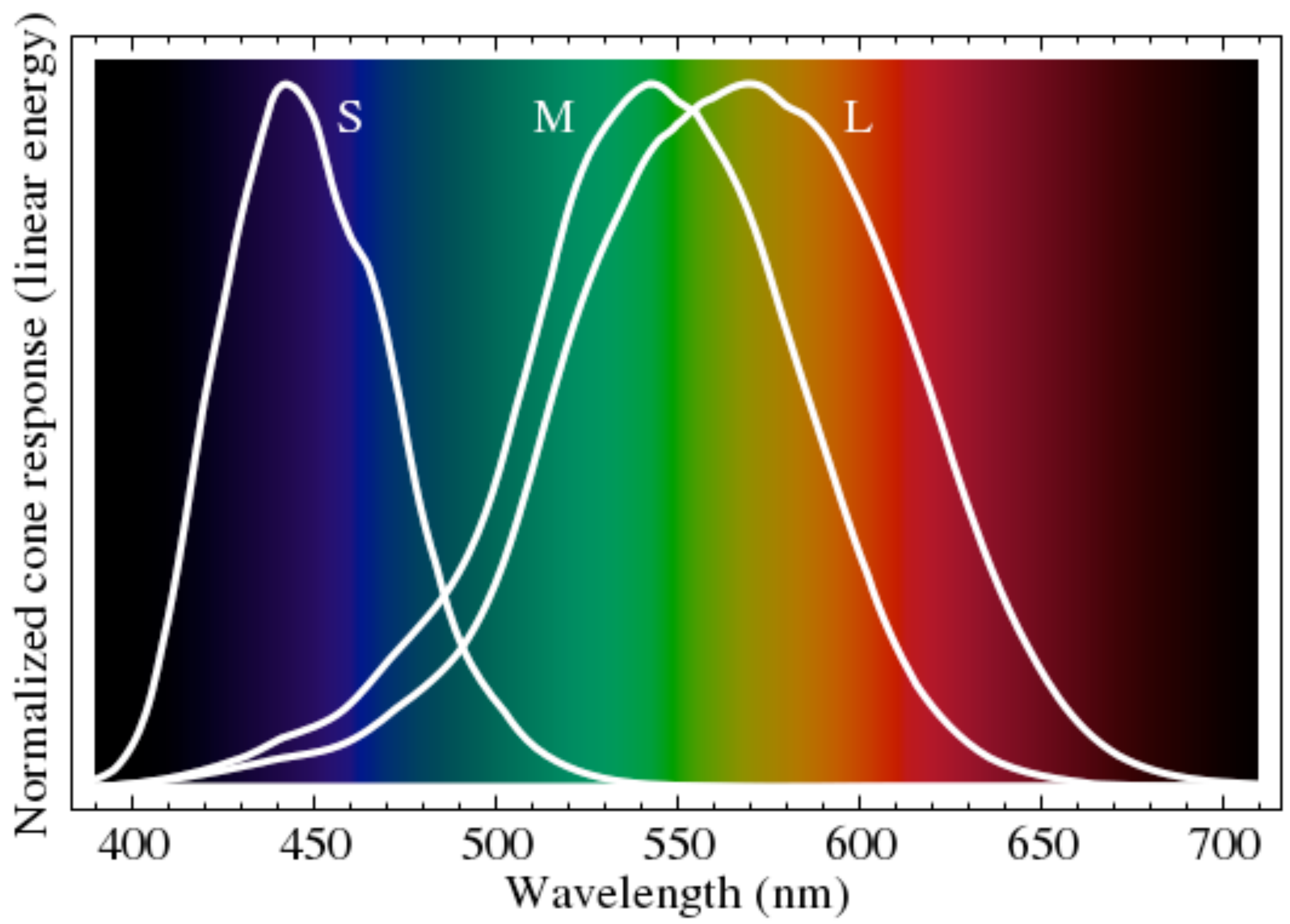
Analog-to-Digital
Active pixel sensor
(2D array of photo-diodes)
Pixel
Analog Signal
Processing
"Optically black" region
Exposed region
Digital Logic (control/processing)
Output bus

Bayer filter 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)



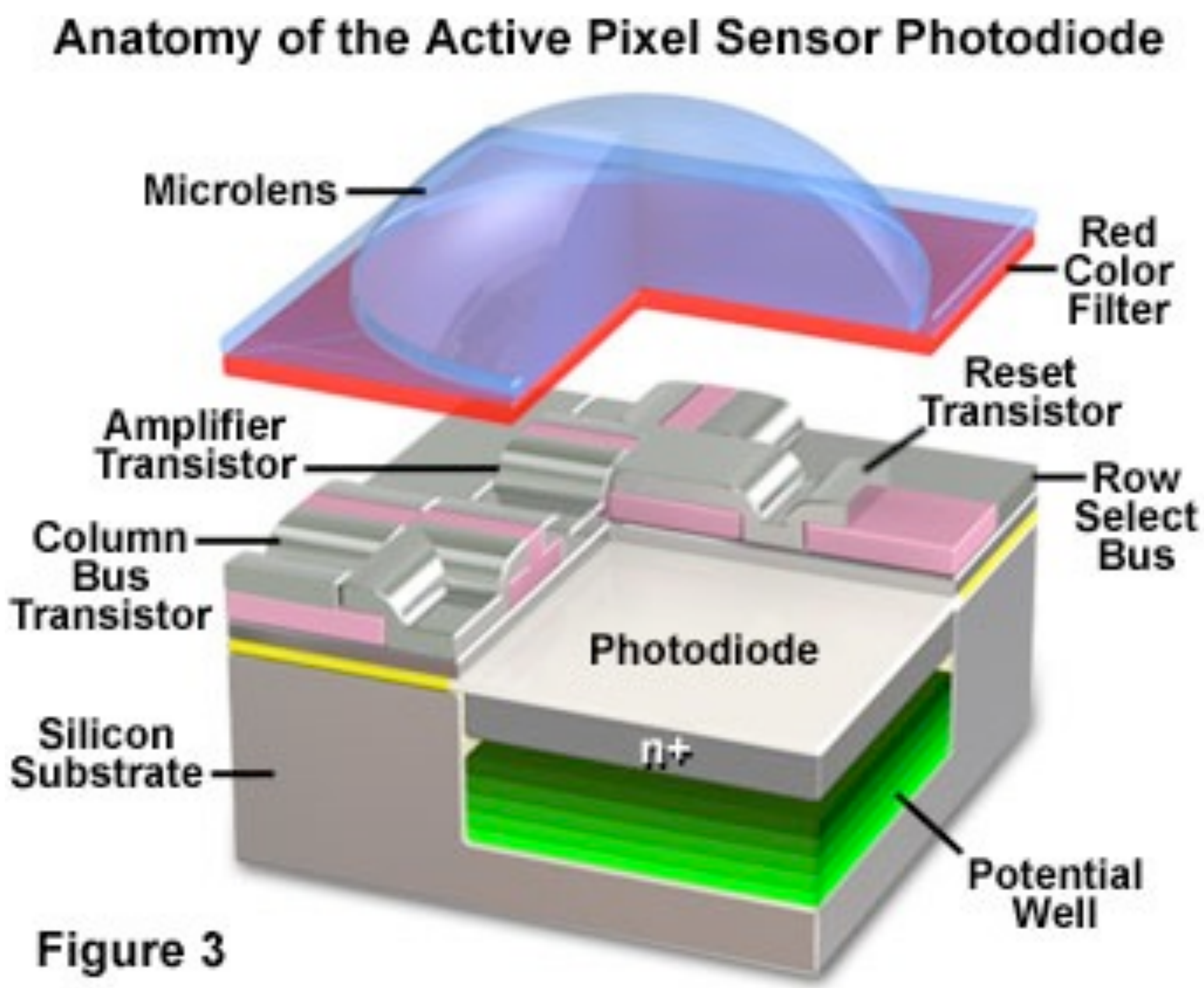
Traditional Bayer mosaic
(other filter patterns exist: e.g., Sony's RGBE)



Human eye: cone spectral response

Image credit: Wikipedia

CMOS sensor pixel



Color filter attenuates light

Fill factor: fraction of surface area used for light gathering

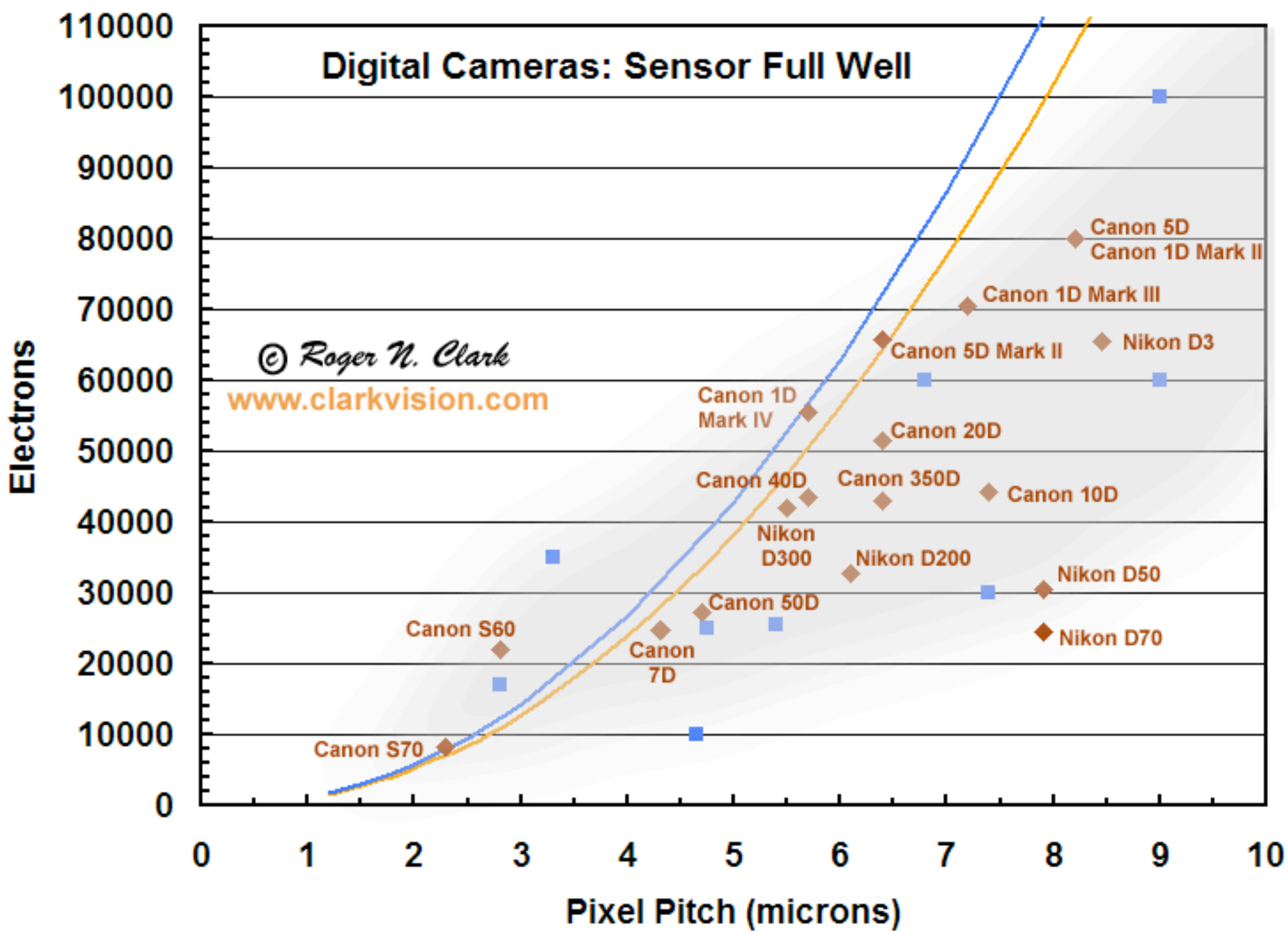
Microlens (a.k.a. lenslet) steers light toward photo-sensitive region (increases light-gathering capability)

Quantum efficiency of photodiode in typical digital camera ~ 50%

Illustration credit: Molecular Expressions (http://micro.magnet.fsu.edu/primer/digitalimaging/cmosimagesensors.html)

Full-well capacity

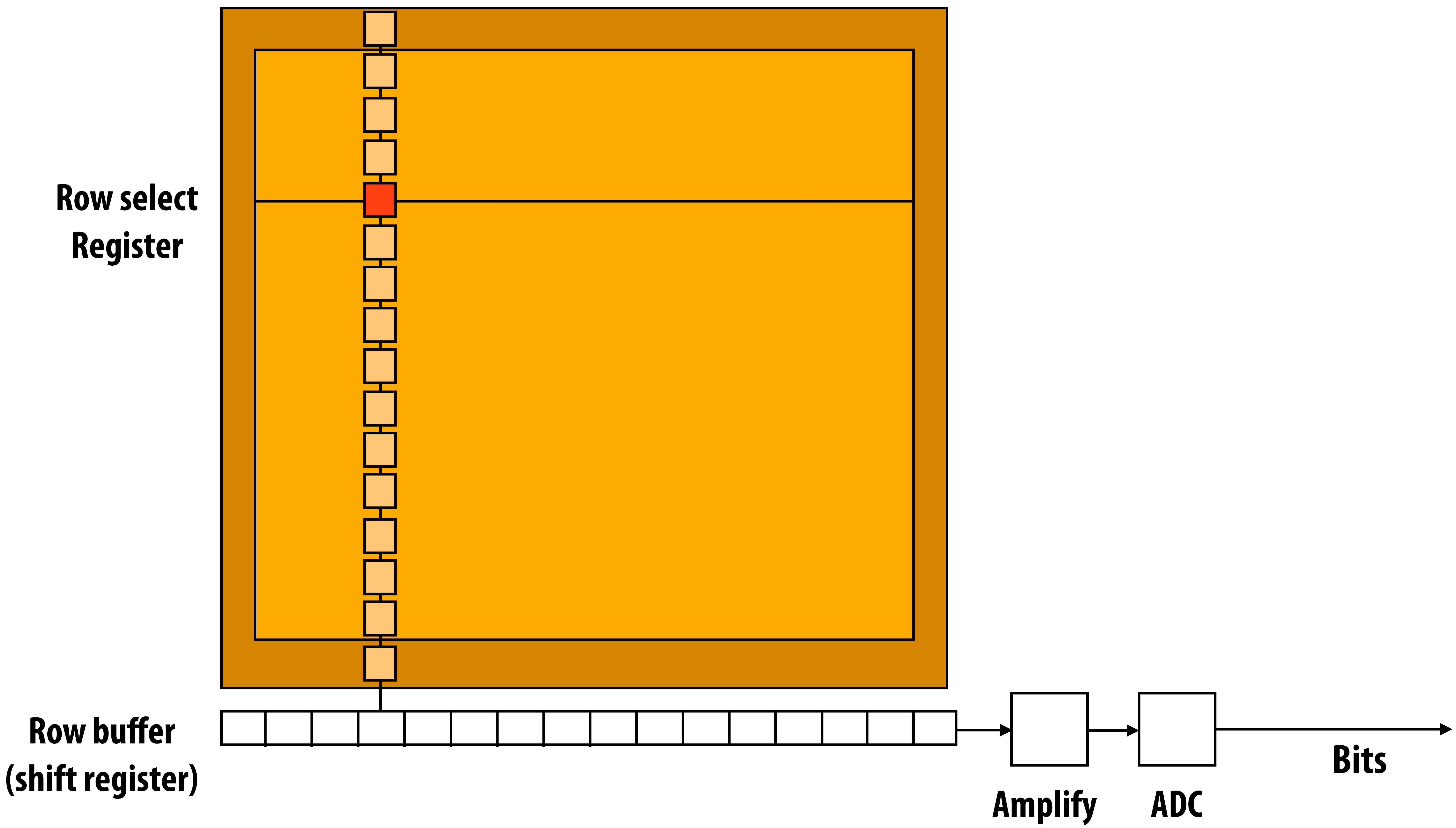
Pixel saturates when capacity is exceeded



Oversaturated pixels
(note surrounding "bloom")

Graph credit: clarkvision.com

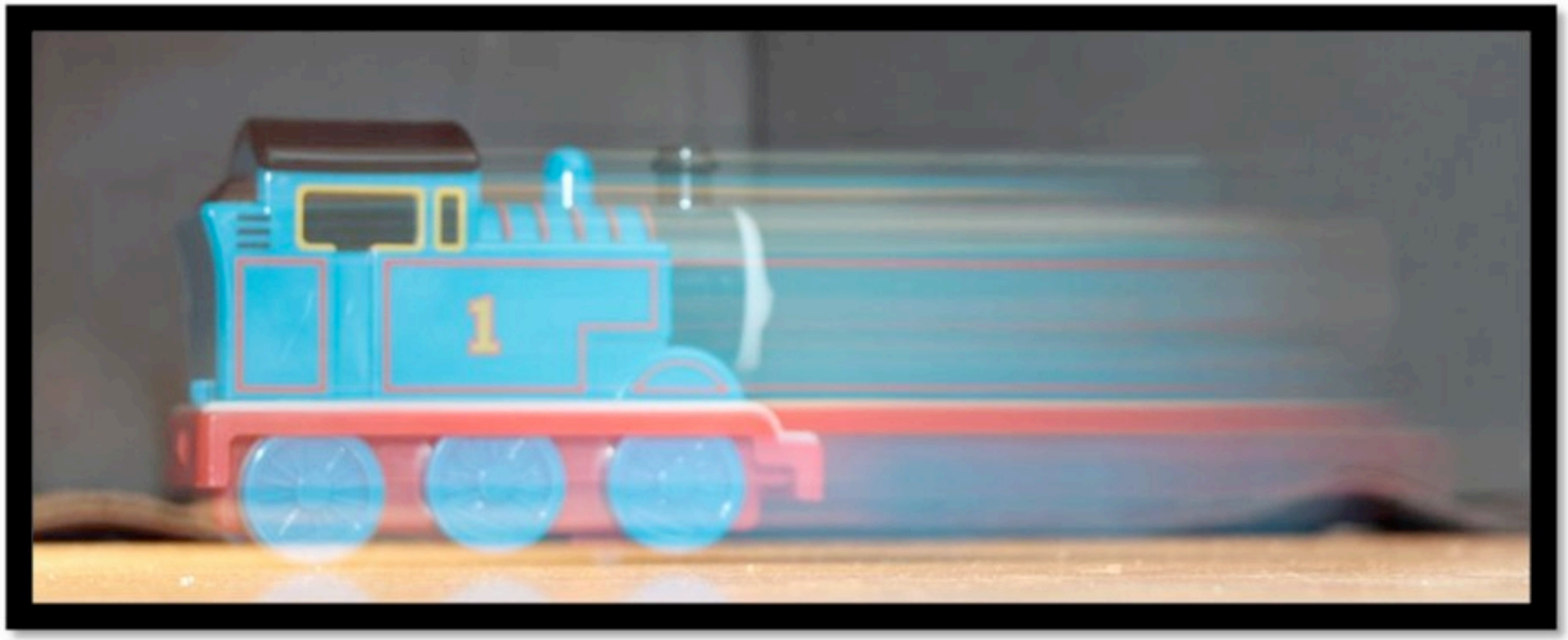
Reading sensed signal



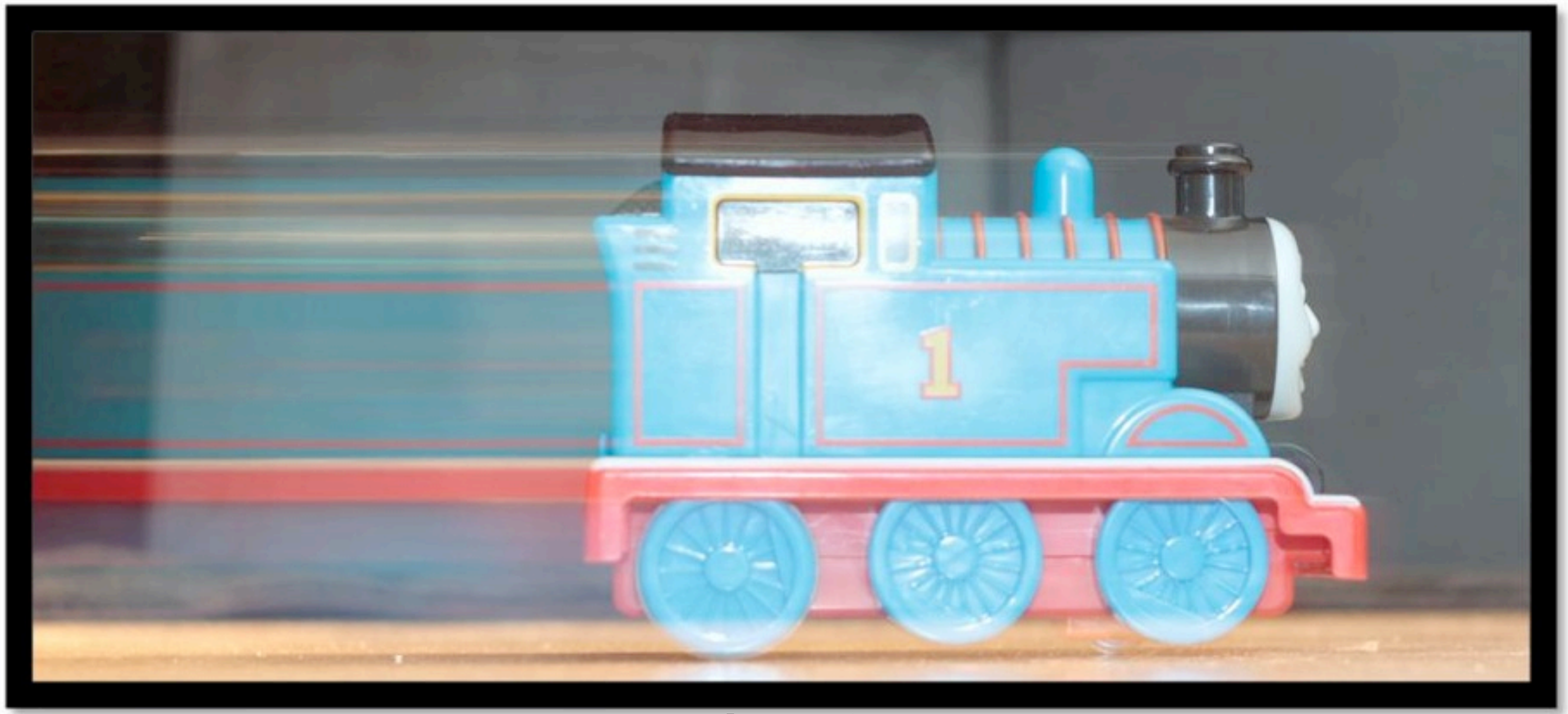
Capturing an image

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



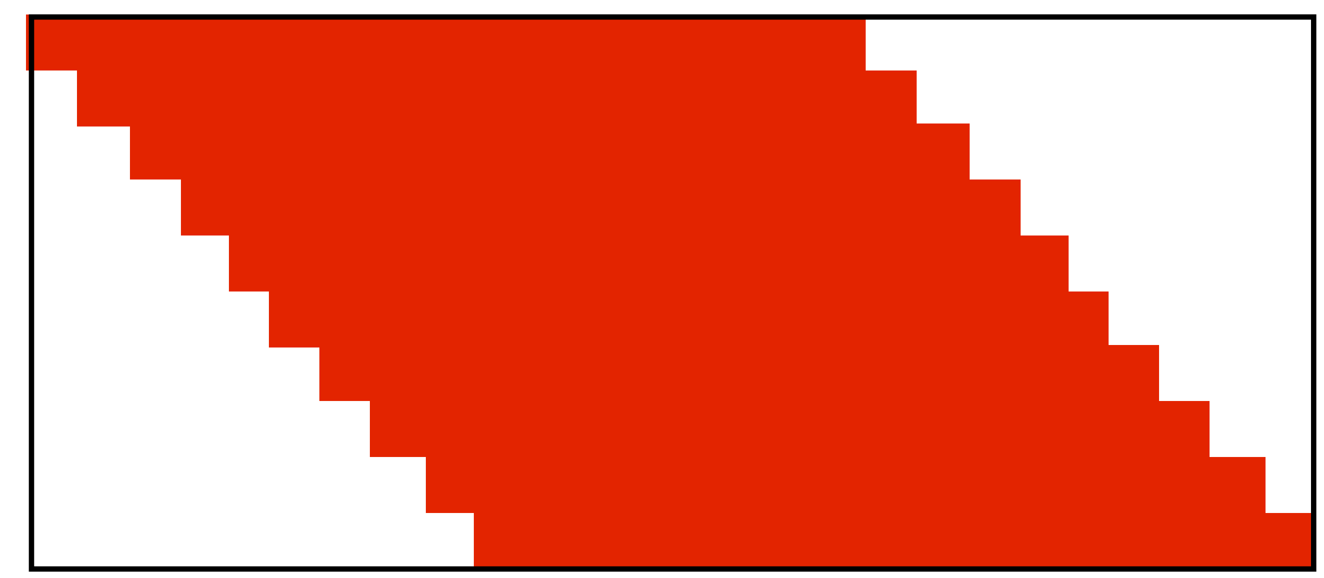
Second curtain sync

Image credit: Michael R. Beeman

Electronic rolling shutter

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

Photo of red square, moving to right



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)

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

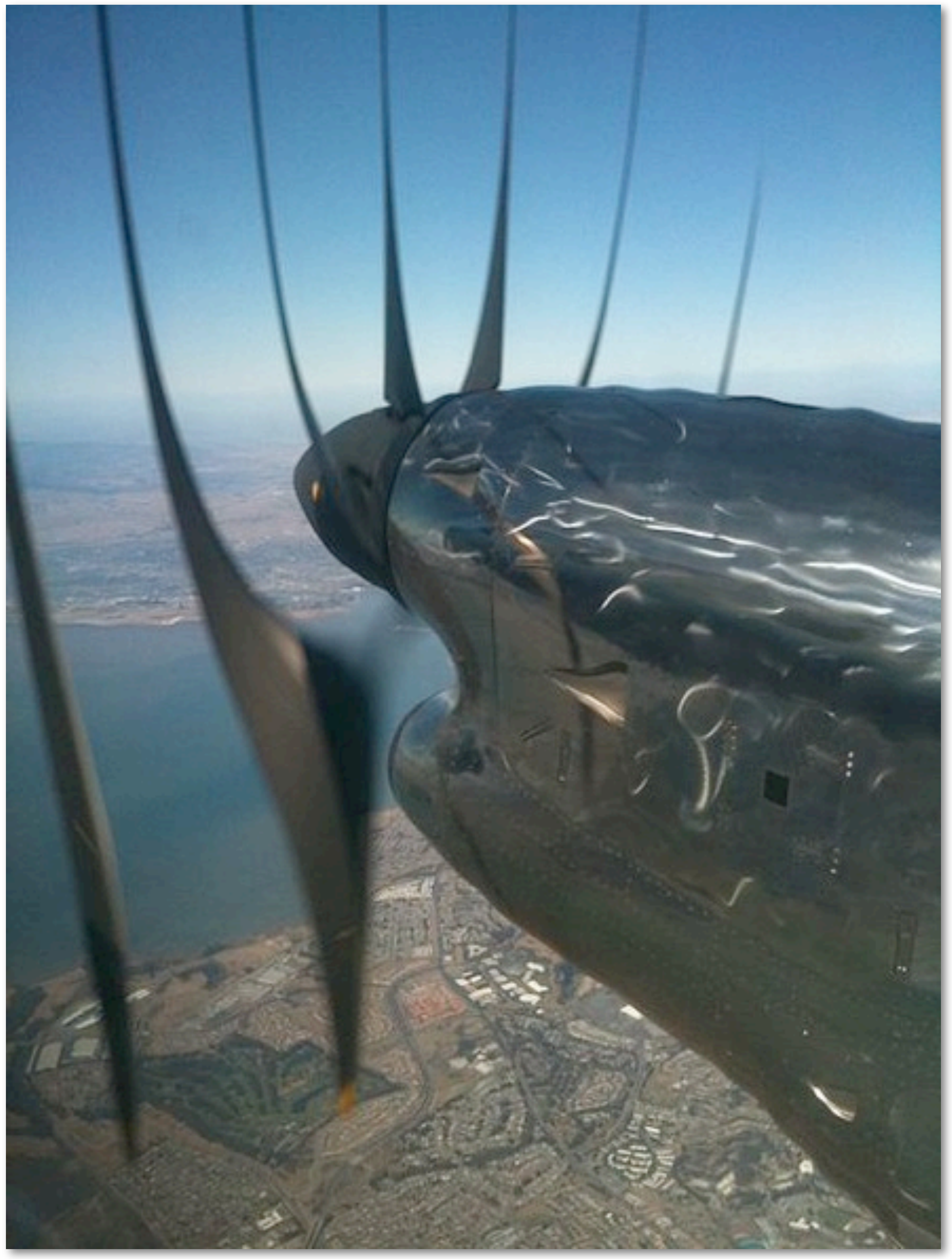


Image credit: Wikipedia

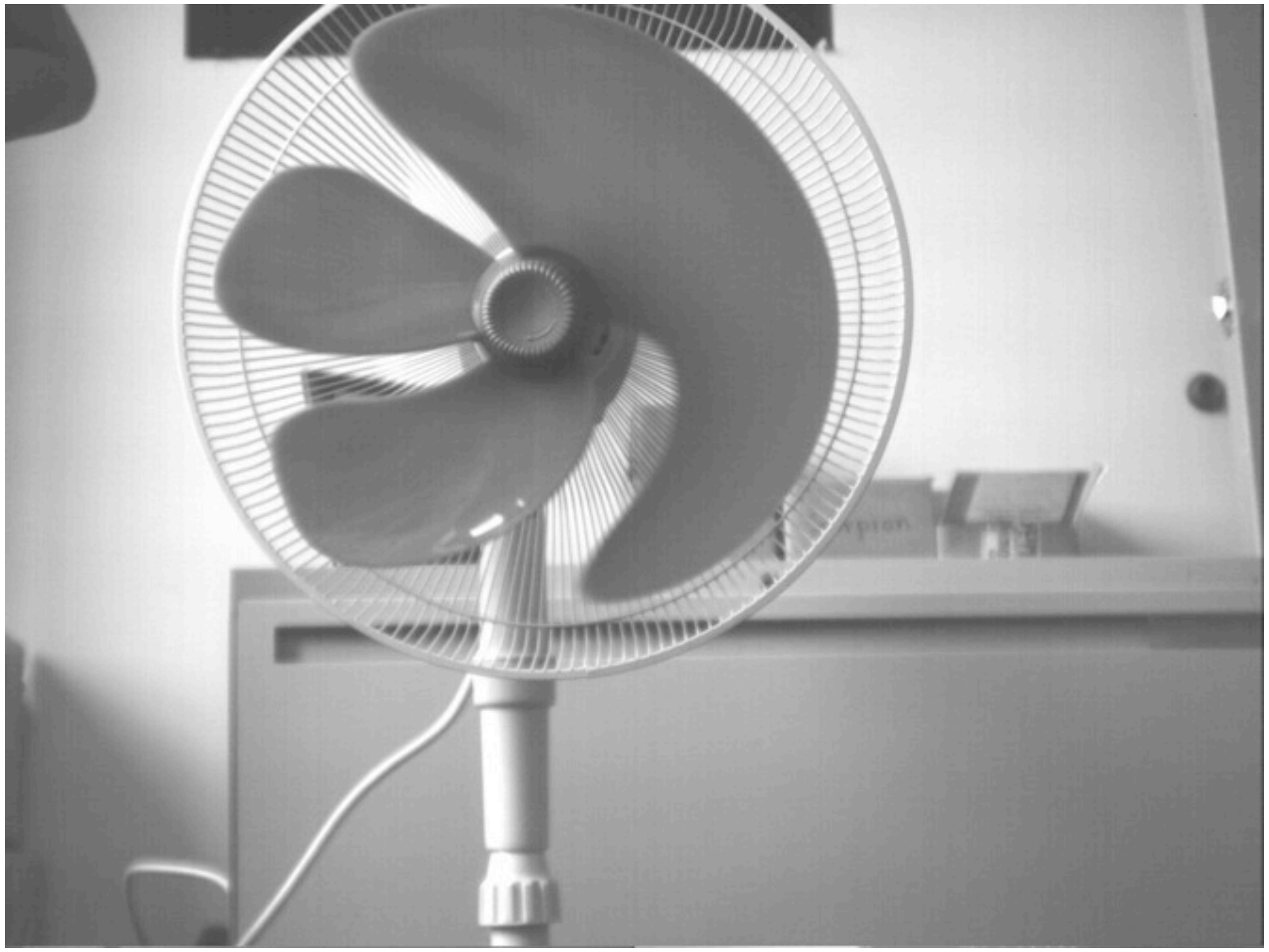
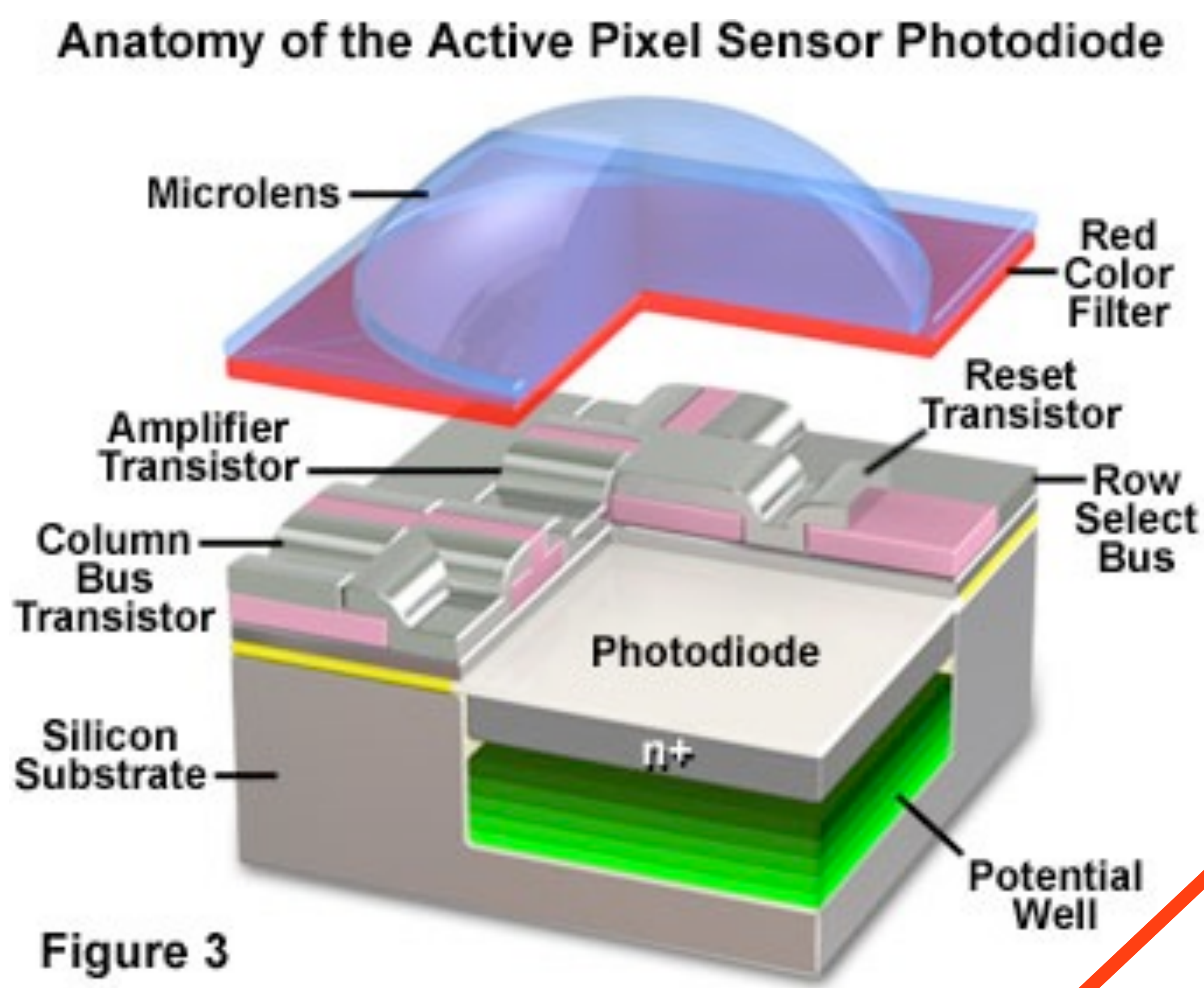


Image credit: Point Grey Research

Measurement noise



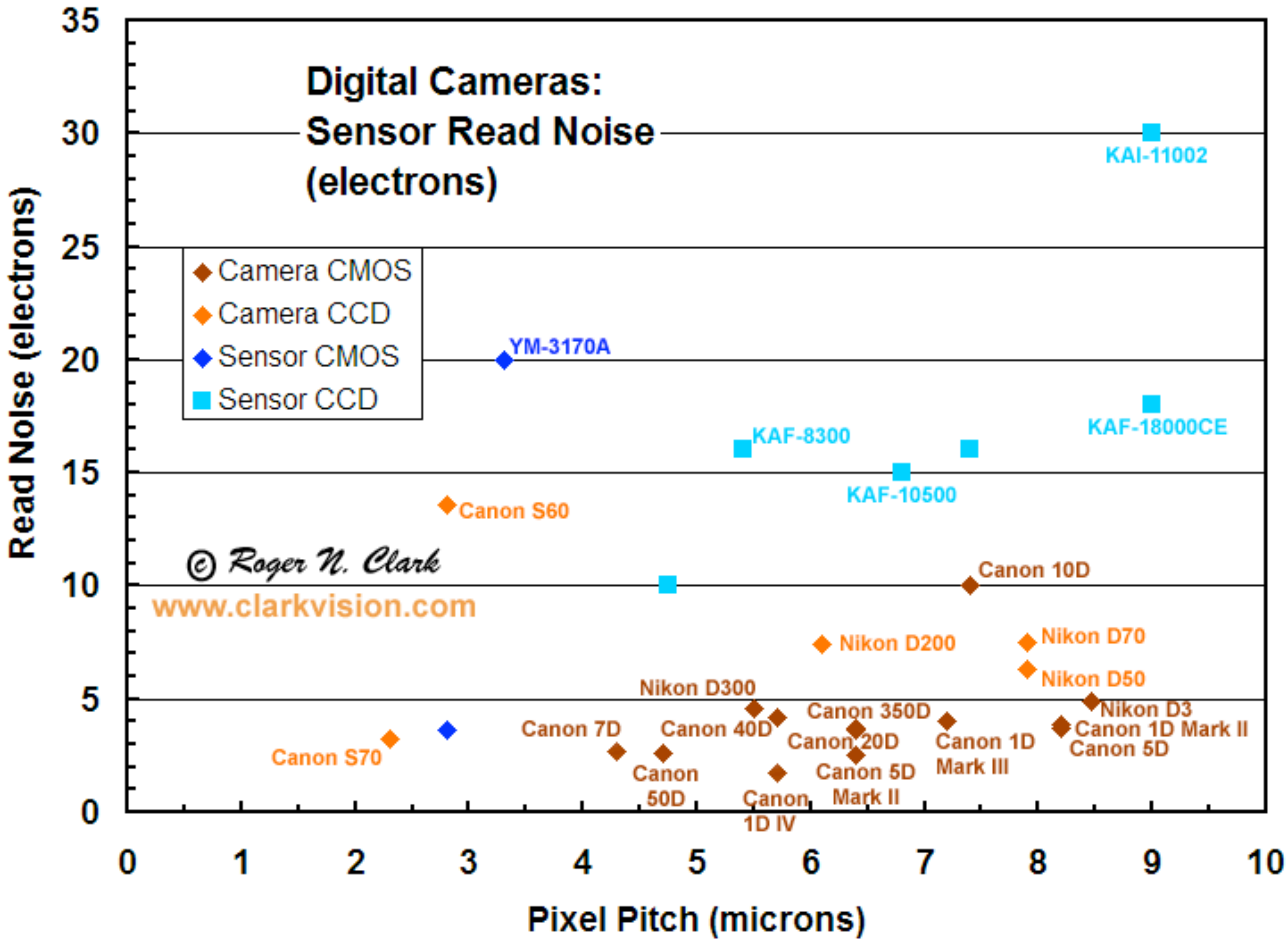
- Photon shot noise:
  - Photon arrival rates feature poisson distribution
  - Standard deviation = sqrt(N)
- Dark shot noise:
  - Due to leakage current
- Read noise
  - e.g., due to amplification
- Non-uniformity of pixel sensitivity

Subtract dark image

Flat field image

Illustration credit: Molecular Expressions (http://micro.magnet.fsu.edu/primer/digitalimaging/cmosimagesensors.html)

Read noise



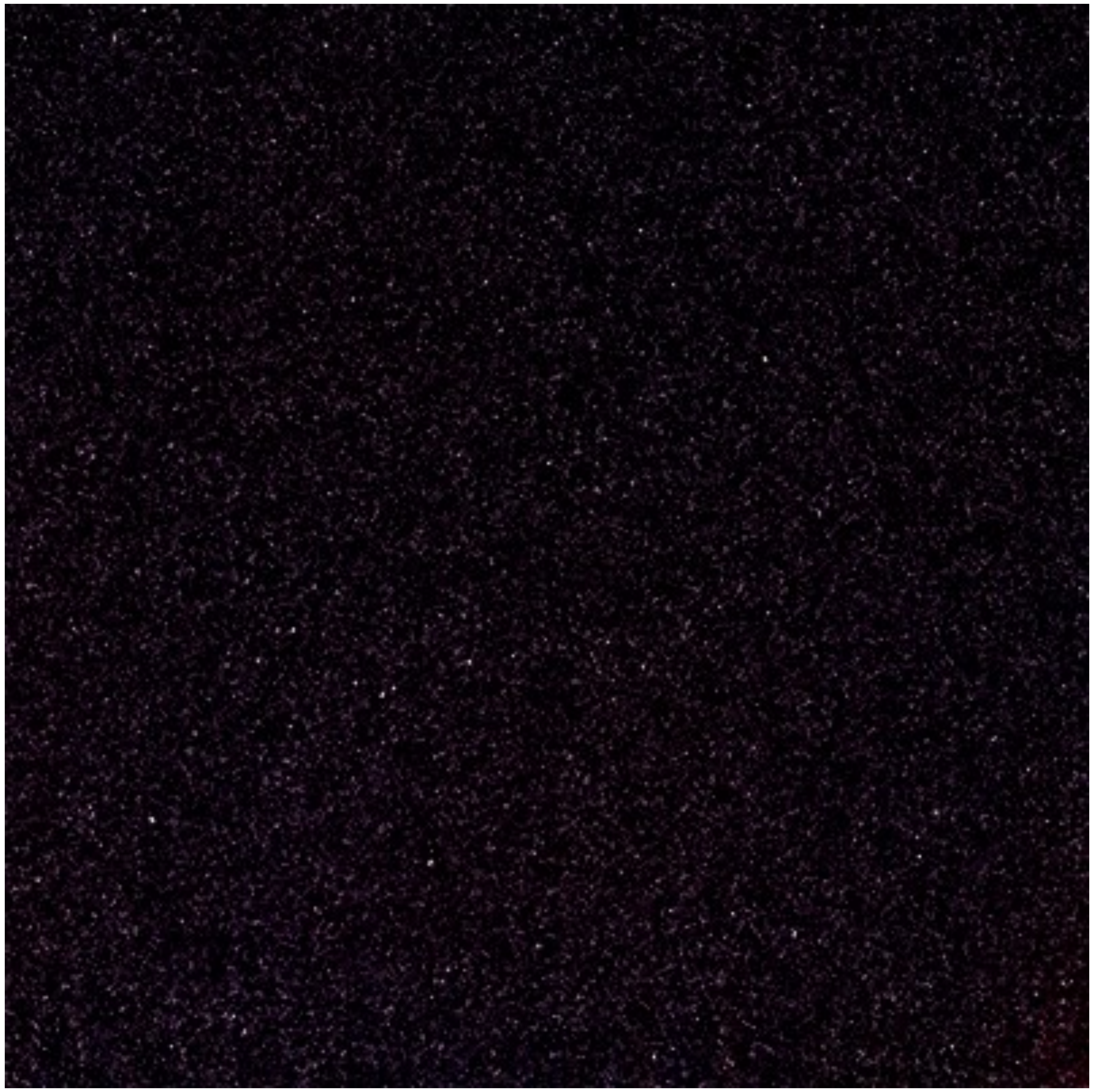
Read noise largely independent of pixel size

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

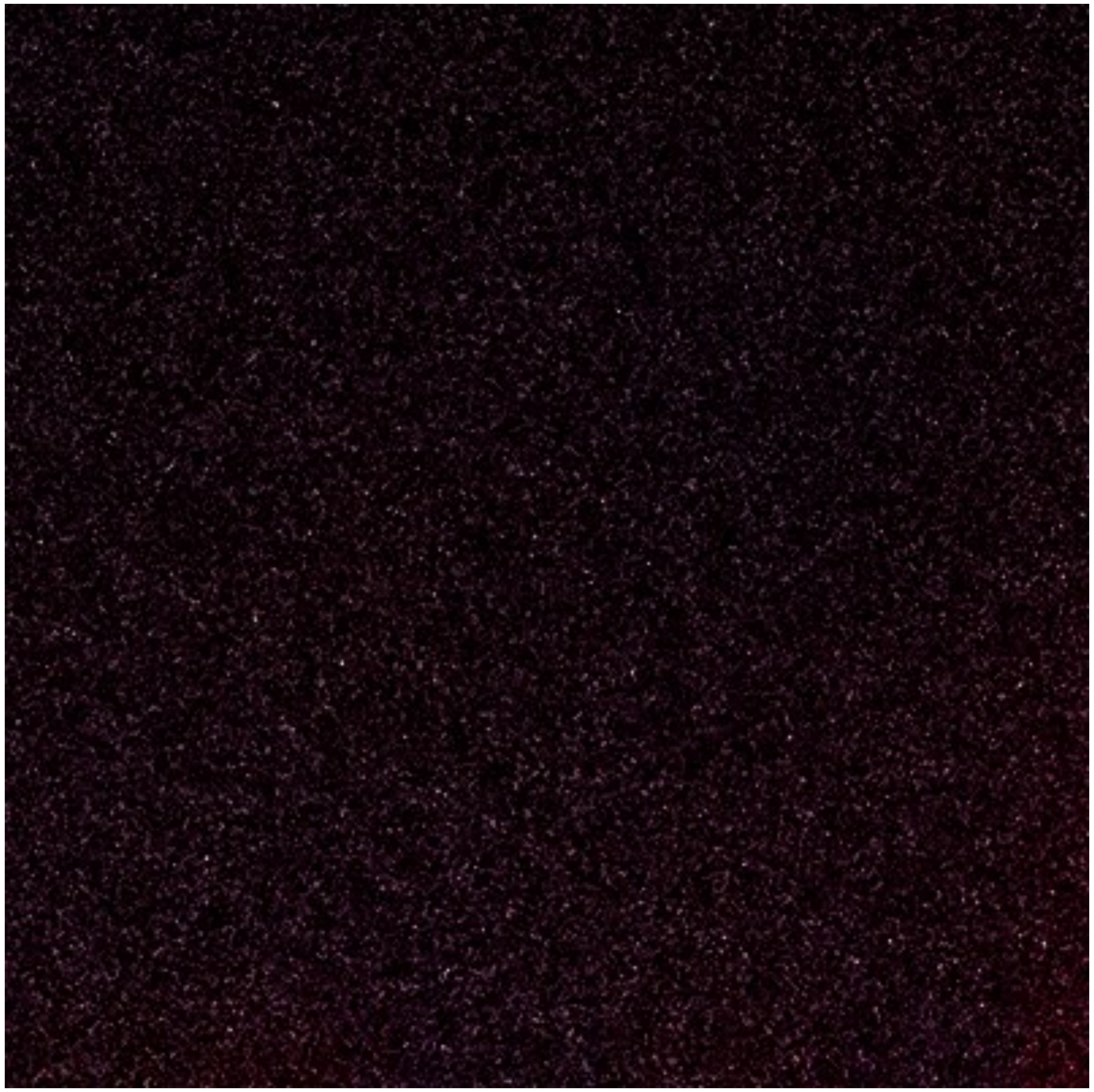
Image credit: clarkvision.com

Noise

Black image examples: Nikon D7000, High ISO



1/60 sec exposure



1 sec exposure

Maximize light gathering capability

- **Increase signal-to-noise ratio**
  - Dynamic range determined by noise floor and full-well capacity
- **Big pixels**
  - Nikon D3: 8.5 um
  - iPhone 4: 1.75 um
- **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)

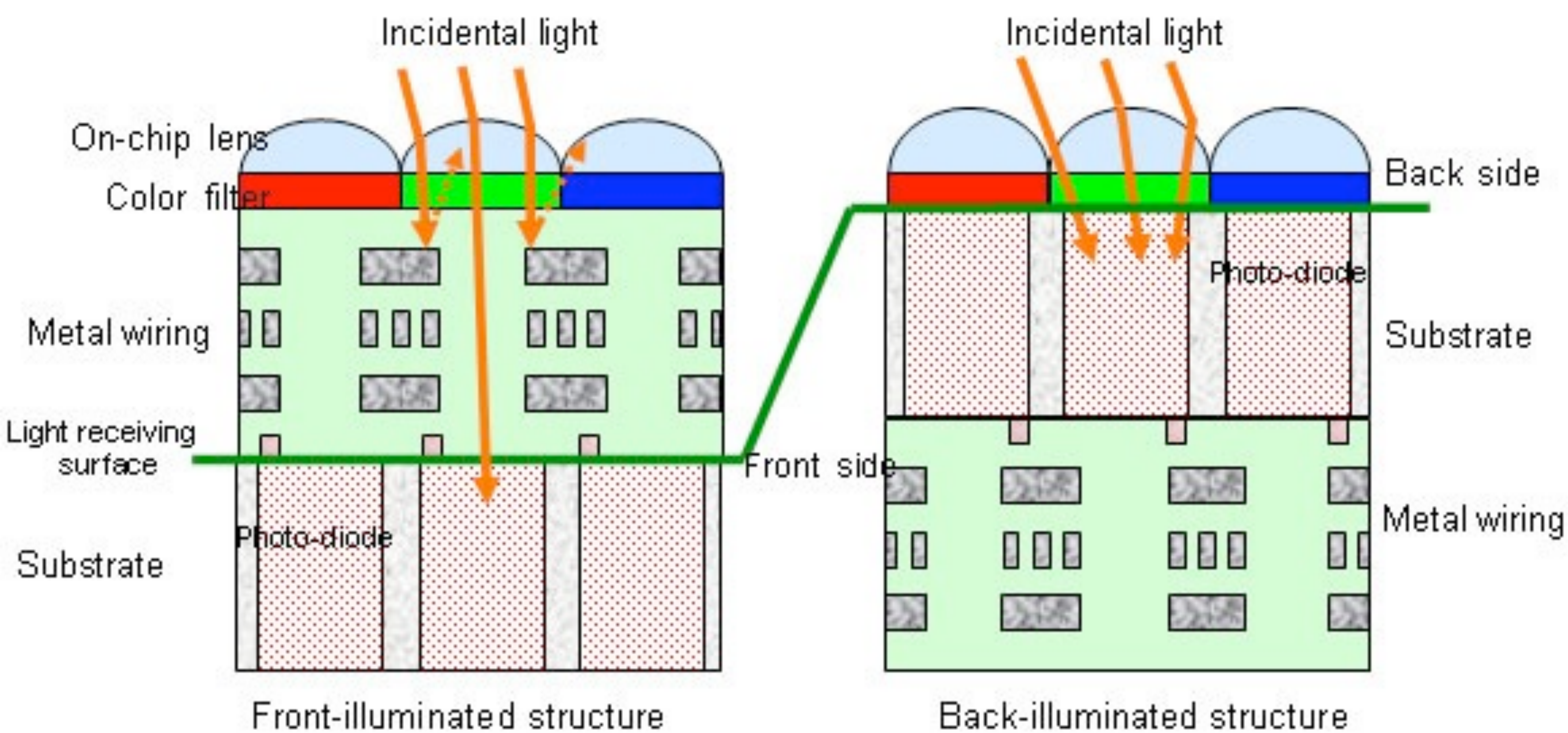


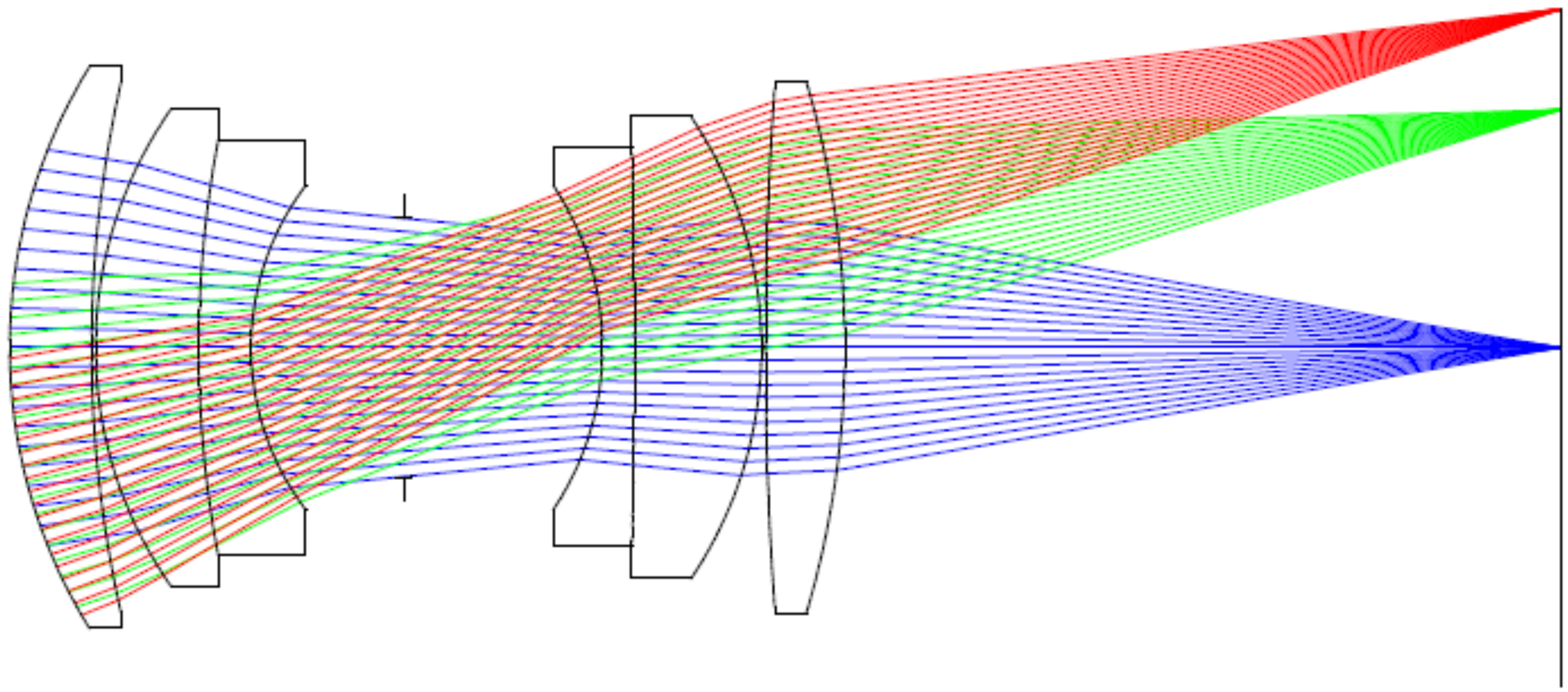
Illustration credit: Sony

Vignetting

Image of white wall:

Types of vignetting

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



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

- **Microlens reduces pixel vignetting**

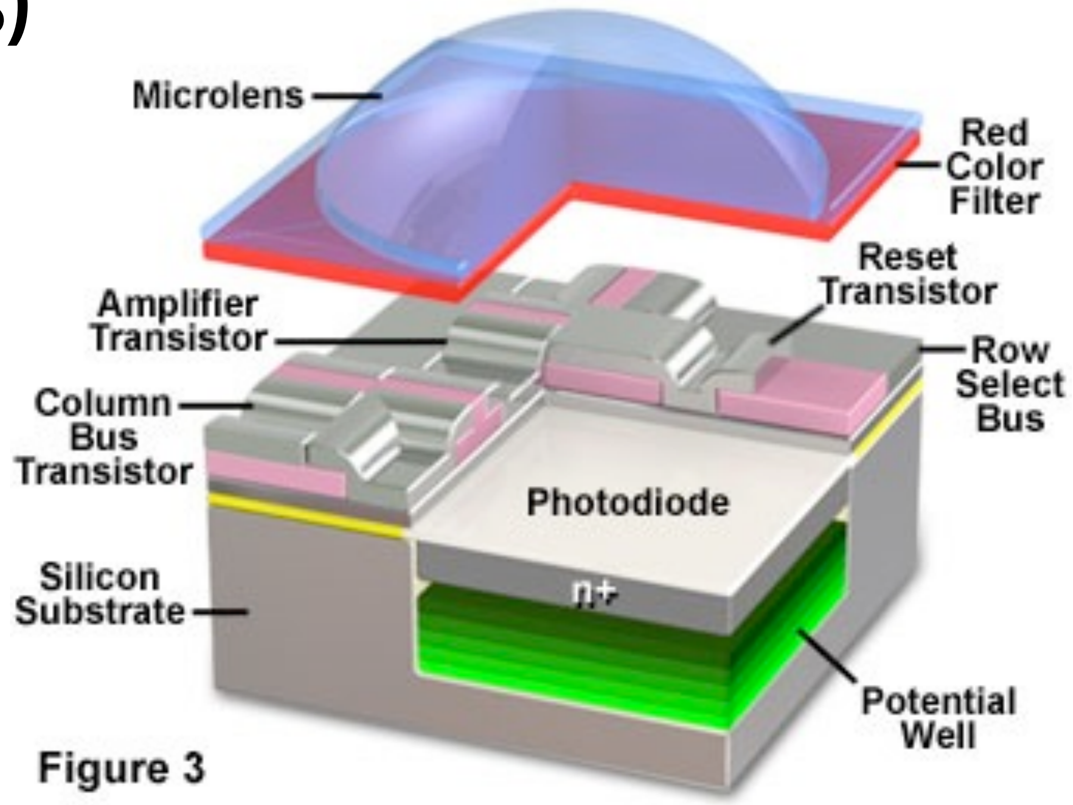
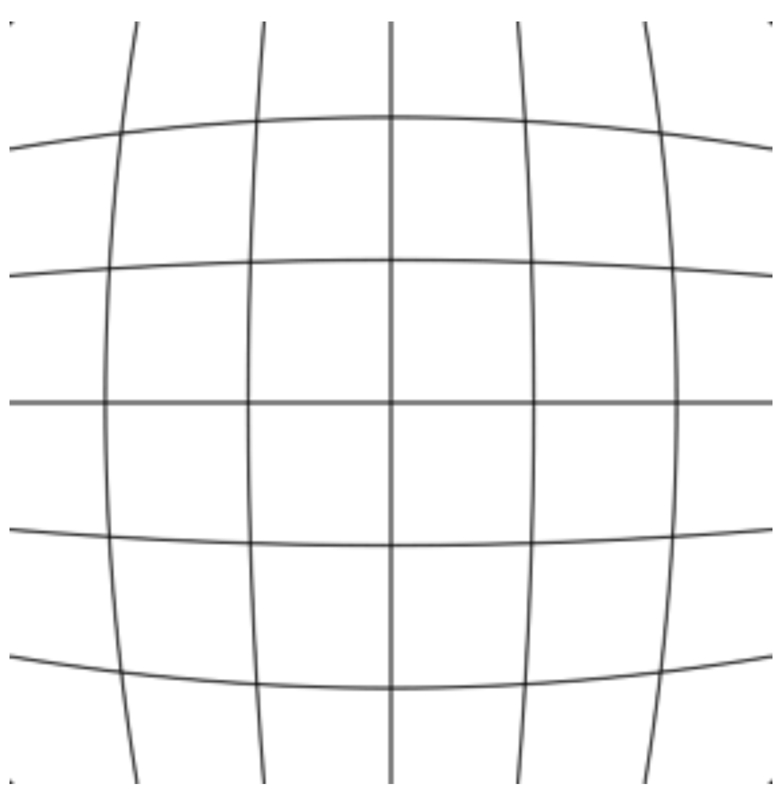


Figure 3

Image credit: Mark Butterworth

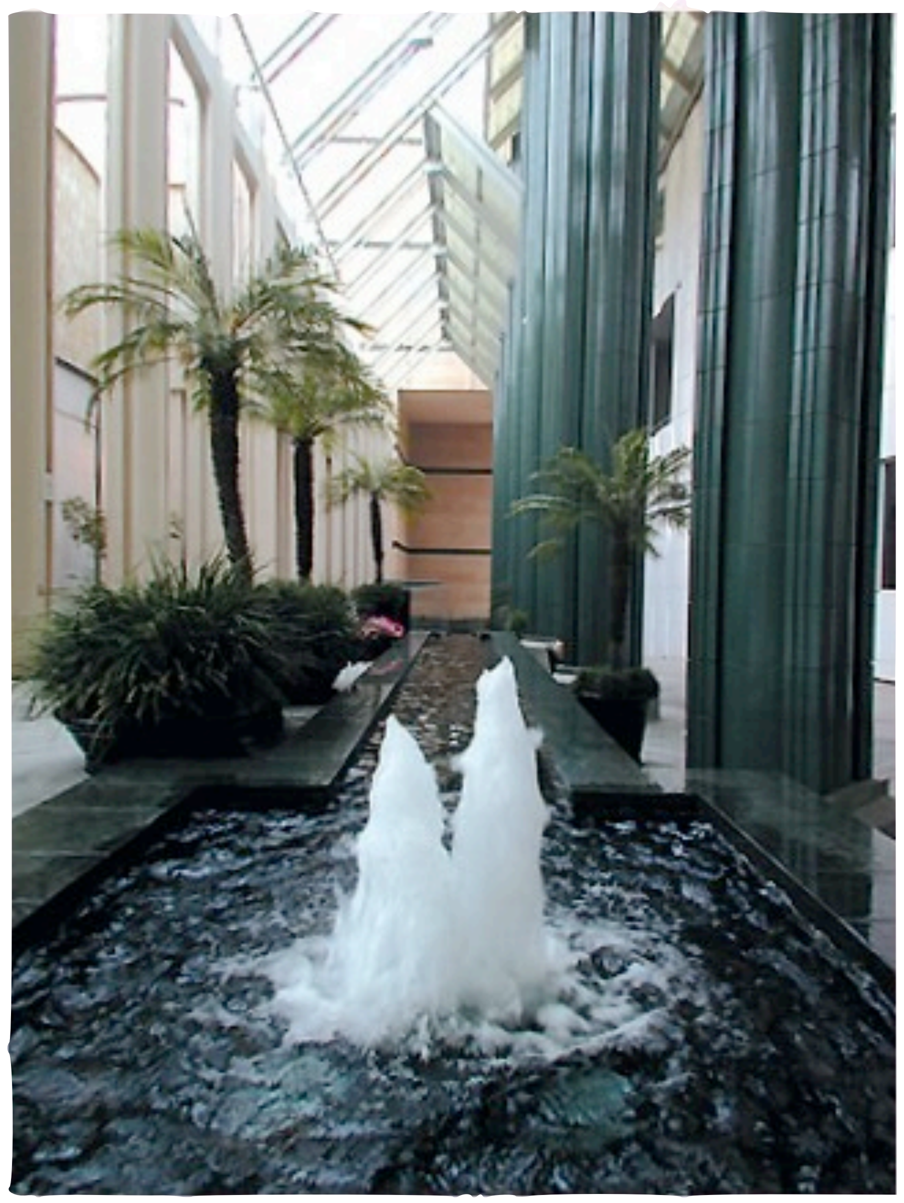
More challenges

- **Chromatic shifts over sensor**
  - Pixel light sensitivity changes over sensor due to interaction with microlens (index of refraction depends on wavelength)
- **Dead pixels**
- **Lens distortion**



Pincushion distortion

Captured Image



Corrected Image

Image credit: PCWorld

Theme so far: bits off the sensor do not form a displayable image

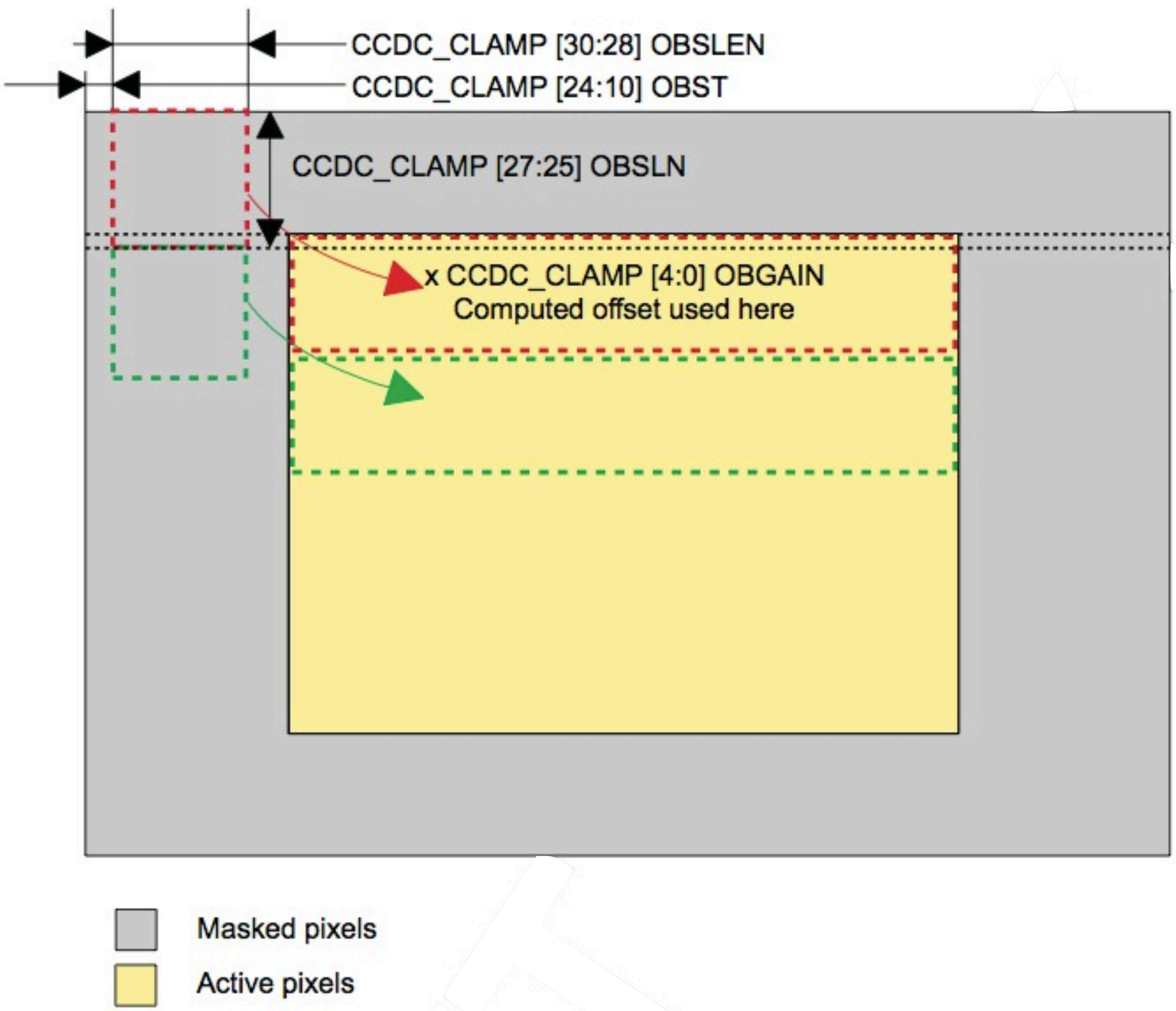
RAW image processing

Example image processing pipeline

- Adopting terminology from Texas Instruments OMAP Image Signal Processor pipeline
- Assume: receiving 12 bits/pixel Bayer mosaiced data from sensor

Optical clamp: remove sensor offset bias

```
output_pixel = input_pixel - [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 defect pixels

- Store LUT with known defect pixels
- 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**
- **Use 2D buffer stored in memory**
  - Lower res buffer, upsampled on-the-fly

```
offset = upsample_compensation_offset_buffer(current_pixel_xy);
gain = upsample_compensation_gain_buffer(current_pixel_xy);

output_pixel = gain + offset * input_pixel;
```

Optional dark frame subtract

- **Similar computation to lens shading compensation**

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

White balance

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

```
output_pixel = white_balance_coeff * input_pixel

note: white_balance_coeff depends on whether pixel is red, green, or blue pixel
```

- **Setting white balance coefficients:**
  - Example naive 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 white-balance algorithms (proprietary)**

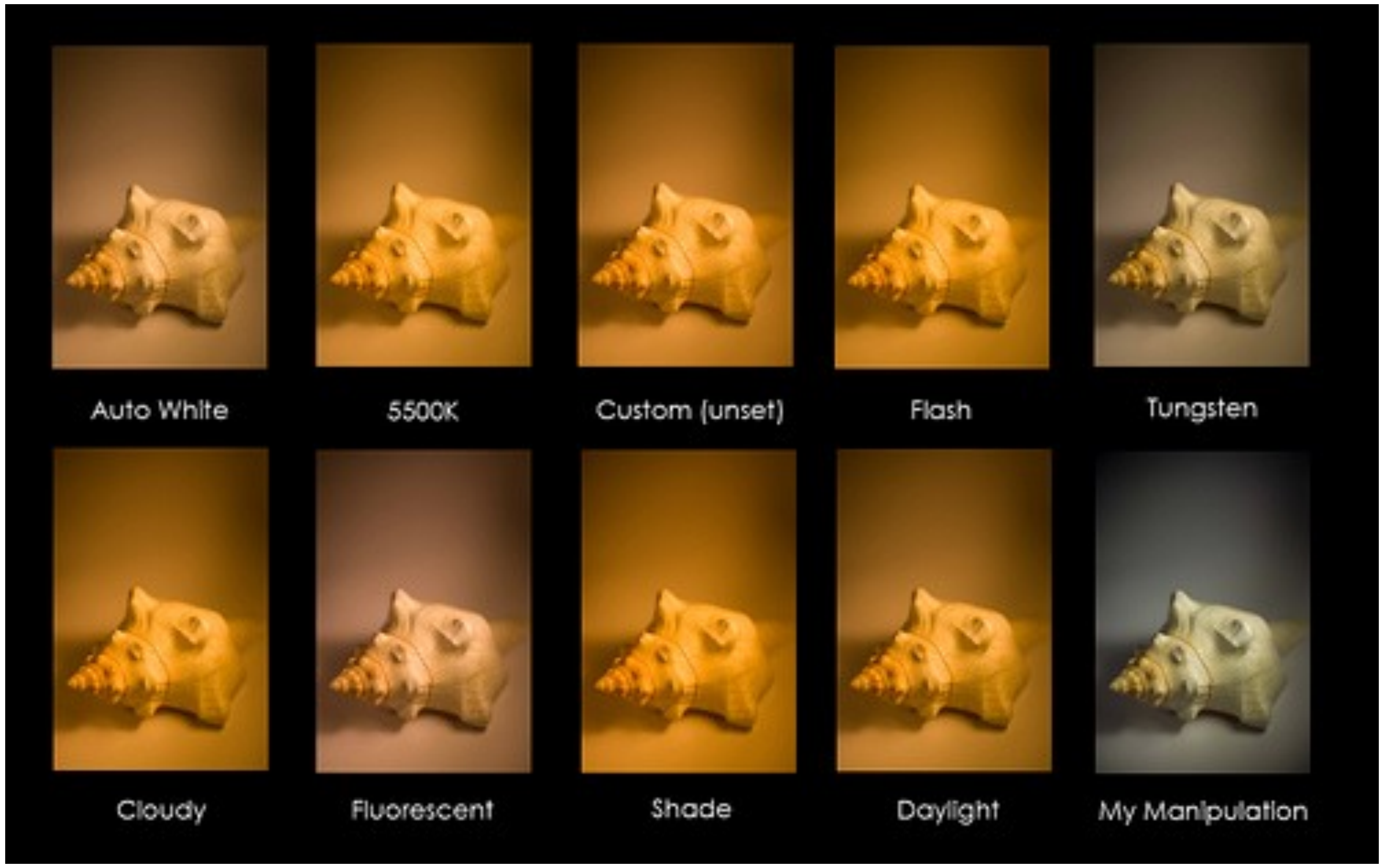


Image credit: basedigitalphotography.com

Demosiac

- Produce a RGB image from mosaiced input
- Basic algorithm: linear interpolation of mosaiced values
- More advanced algorithms: attempt to preserve edges

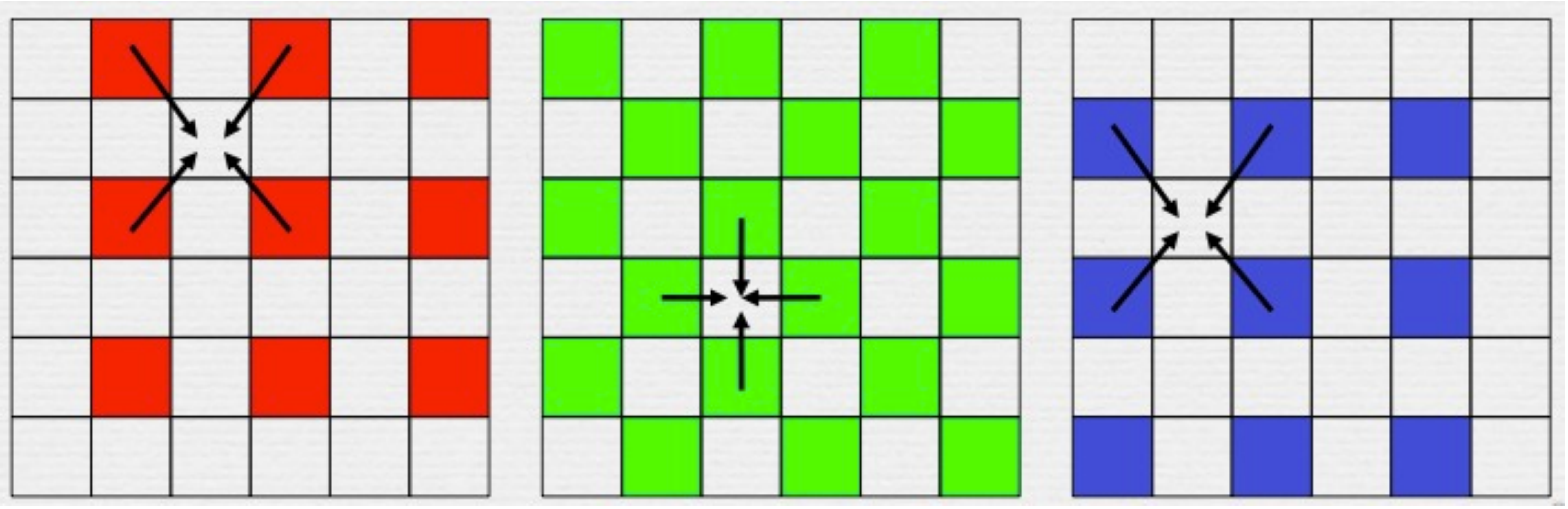
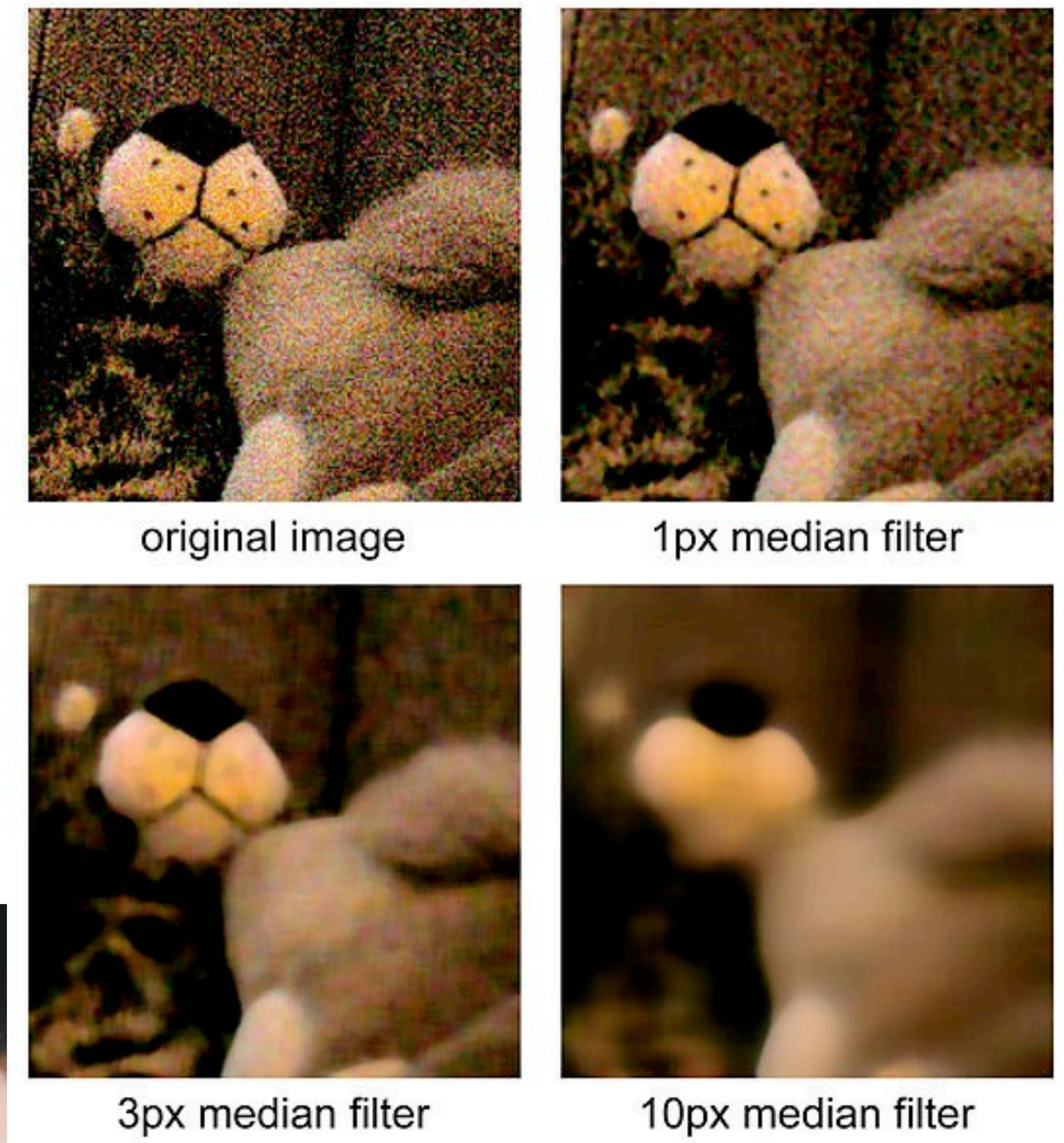


Image credit: Mark Levoy

Denoise



Median Filter

Bilateral filter: remove noise, preserve edges

Color conversion

- **Change of basis**
- **3 x 3 matrix multiplication**

```
output_rgb_pixel = CONVERSION_MATRIX * input_rgb_pixel
```

Simplified image processing pipeline

- Correct for sensor bias (using measurements of optically black pixels)
- Correct pixel defects
  - lossless compression → RAW file
- 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)
  - → JPEG file

Next time