CPSC 4040/6040 Computer Graphics Images

Joshua Levine
levinej@clemson.edu

Lecture 14
Sampling and Interpolation + High Dynamic Range Imaging

Oct. 6, 2015

Slide Credits:
Frédo Durand
Alexei Efros
Paul Debevec
Yung-Yu Chung

Agenda

- PA04 Questions?

Last Time

Jim Blinn's Corner

http://www.research.microsoft.com/~blinn/

What Is a Pixel?

James F. Blinn

Microsoft Research

I am not a major sports fan. But there is one story from the folklore of football that has always intrigued me. The story goes that legendary football coach Vince Lombardi was observing his new players, recruited from the best college teams and all presumably excellent players. He was, however, not pleased with their performance. So he called them all to a meeting, which he began by holding up the essential tool of their trade and saying, "This is a football." I was sufficiently impressed by this back-to-basics attitude that, when I taught computer graphics rendering classes, I used to start the first lecture of the term by going to the blackboard (boards were black then, not white) and drawing a little dot and saying, "This is a pixel."

But was I right? Most computer graphicists would agree that the pixel is the fundamental atomic element of imaging. But what is a pixel really? As I have played with various aspects of pixel mashing, it has occurred to me that the concept of the pixel is really multifaceted (to use a weird metaphor.) Perhaps a better metaphor that spirit I am going to list some possible meanings for a pixel that I will expand on in some later columns.

A pixel is a little square

Early 2D windowing systems considered a pixel a little square. This is perhaps the simplest possible definition, but it only works if you are mostly drawing horizontal and vertical lines and rectangles that are integral numbers of pixels in size. Anything at fractional pixel size or at an angle yields jaggies and other forms of aliasing.

A pixel is a point sample of a continuous function

A more enlightened signal processing approach thinks of a pixel as a point sample of a continuous function (see the dots in Figure 1). (This approach was actually taken by the rendering community long before pixel displays were used for user interfaces and windowing systems.) Applying linear filtering theory to pixel manip-

Pixels Are Image Samples

Image Samples

- Each pixel is a sample of what?
  - One interpretation: a pixel represents the intensity of light at a single (infinitely small point in space)
- The sample is displayed in such a way as to spread the point out across some spatial area (drawing a square of color)

Continuous vs. Discrete

- Key Idea: An image is either (both?) in the
  - Continuous domain: where light intensity is defined at every (infinitesimally small) point in some projection
  - Discrete domain, where intensity is defined only at a discretely sampled set of points.

Converting Between Image Domains

- When an image is acquired, an image is taken from some continuous domain to a discrete domain.
- **Reconstruction** converts digital back to continuous through interpolation.
- The reconstructed image can then be **resampled** and **quantized** back to the discrete domain.

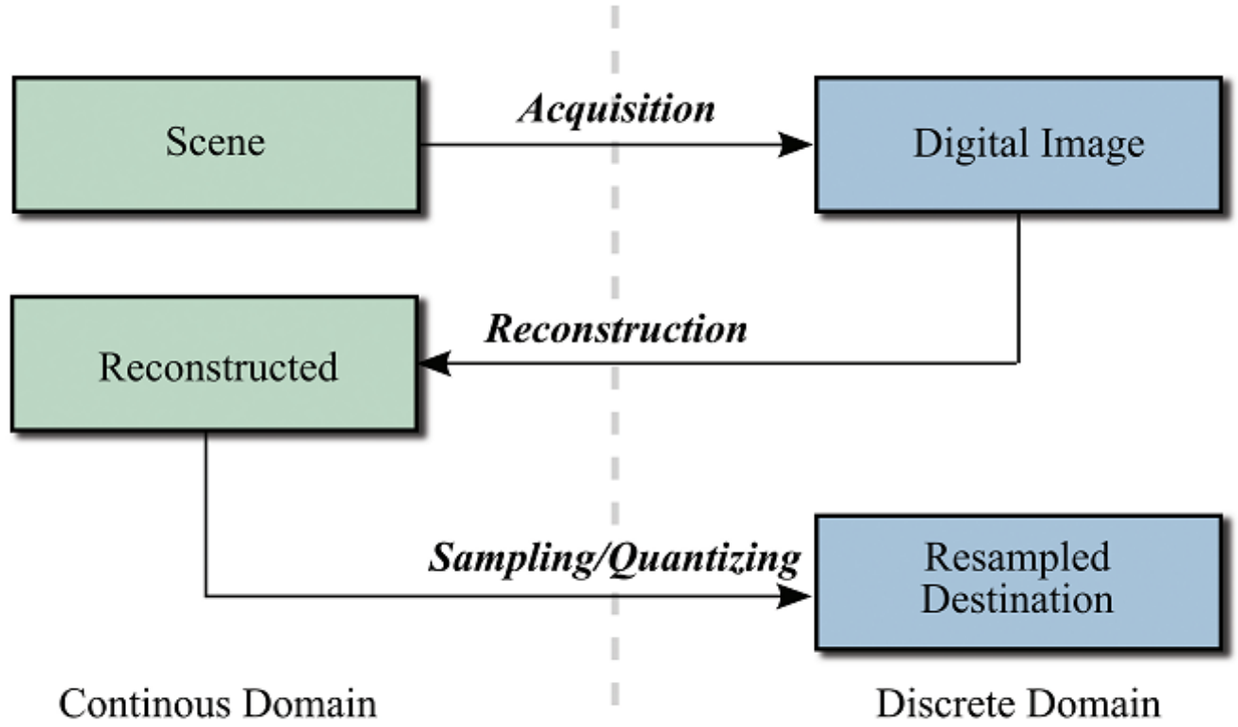


Figure 7.7. Resampling.

Interpolation

- **Interpolation** refers to the creation of a continuous representation from existing image samples.

- Using interpolation it is possible to define a new sample at any point in space

    - Enables the use of real-valued coordinates instead of pixel coordinates.

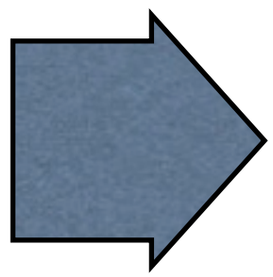
- For example, interpolation can be used to increase resolution of an image by adding more samples “in between” the current ones.

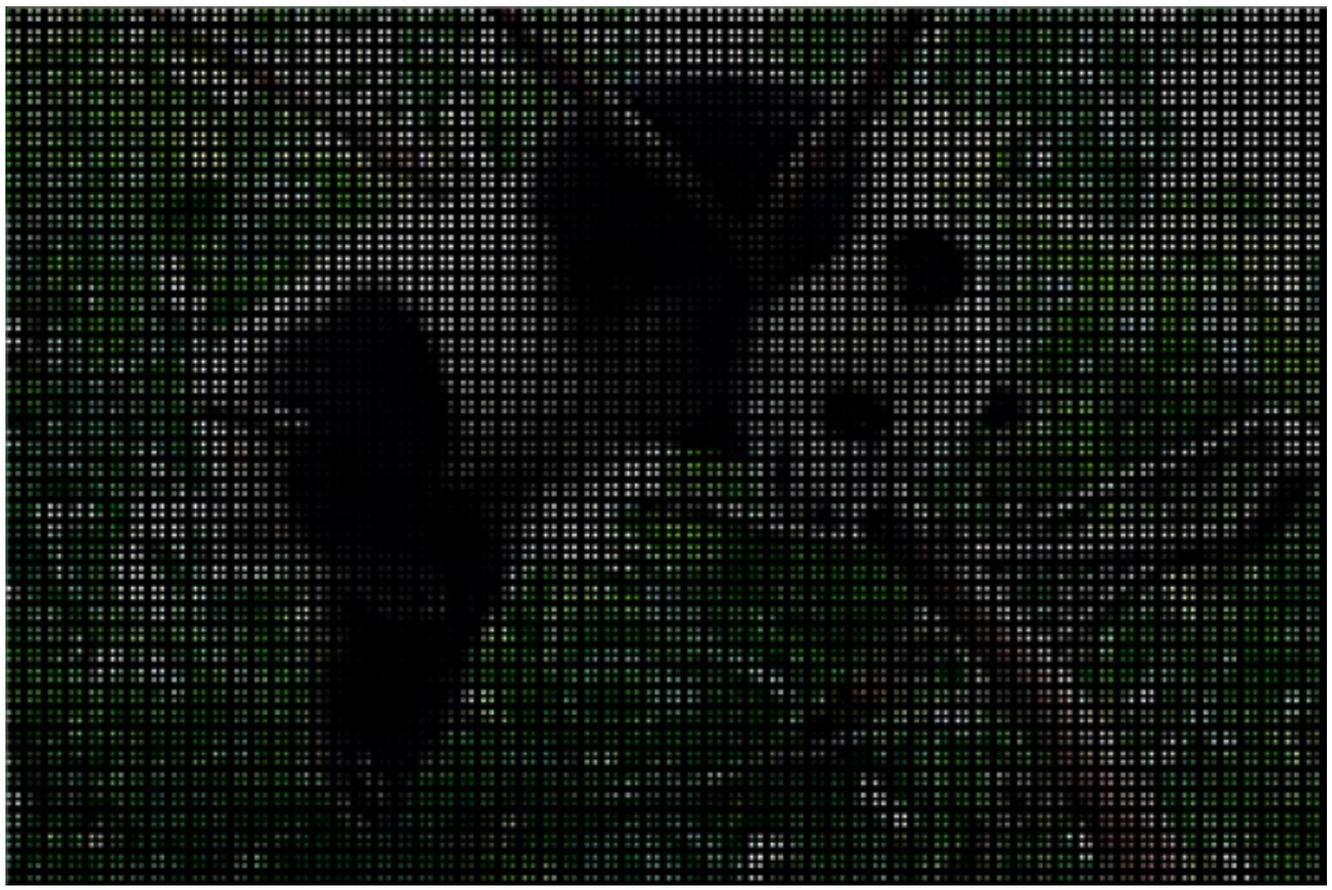
Interpolation Techniques

- Common interpolation techniques:
  - Zeroth order – nearest neighbor
  - First order – bilinear
  - Second order – bicubic

Naive Image Scaling

- Consider resizing an image to a large resolution
- Simple approach: Take all the pixels in input and place them in an output location.





Naive Image Scaling Code

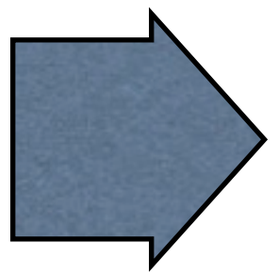
```
//scale factor
int k = 4;

//create an output image that is k times bigger
pixel **output = new pixel*[k*H];
output[0] = new pixel[k*H*k*W];
for (int i=0; i<k*H; i++) {
   output[i] = output[i-1] + k*W;
}

//copy the pixels over
for (int row = 0, row < H; row++) {
   for (int col = 0; col < W; col++) {
      output[k*row][k*col] = input[row][col];
   }
}
```

Using the “Inverse” Transform

- The problem with the previous approach is that we skip many pixels in the output
- We can adjust the loop so that we do one copy for each output pixel instead of each input pixel



Inverse Image Scaling Code

```
//scale factor
int k = 4;

//create an output image that is k times bigger
pixel **output = new pixel*[k*H];
output[0] = new pixel[k*H*k*W];
for (int i=0; i<k*H; i++) {
   output[i] = output[i-1] + k*W;
}

//Loop for each output pixel instead.
for (int row = 0, row < k*H; row++) {
   for (int col = 0; col < k*W; col++) {
      output[row][col] = input[row/k][col/k];
   }
}
```

Image Zooming Results

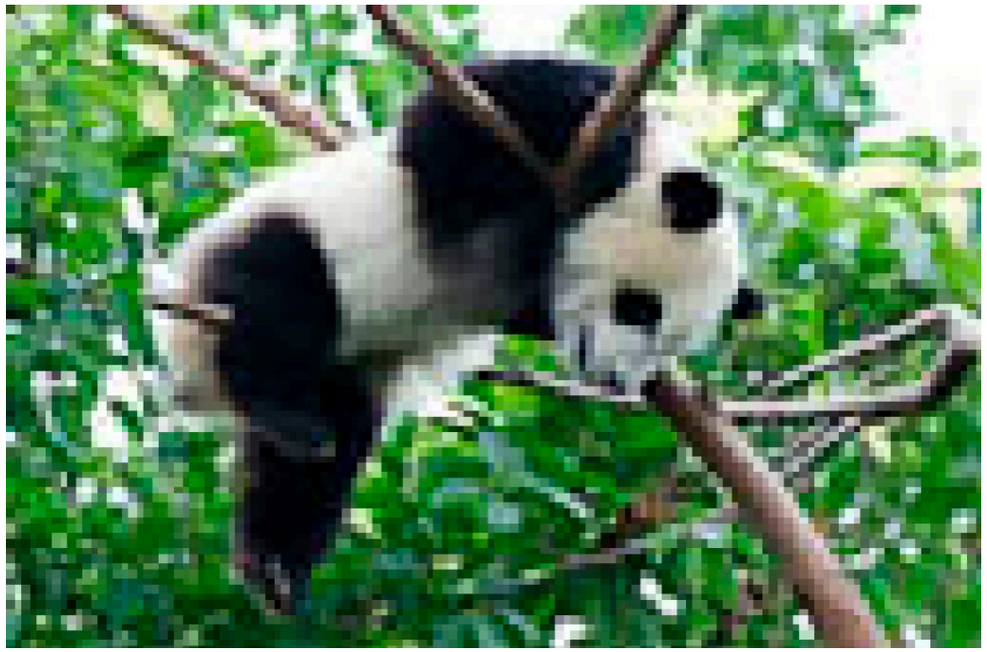
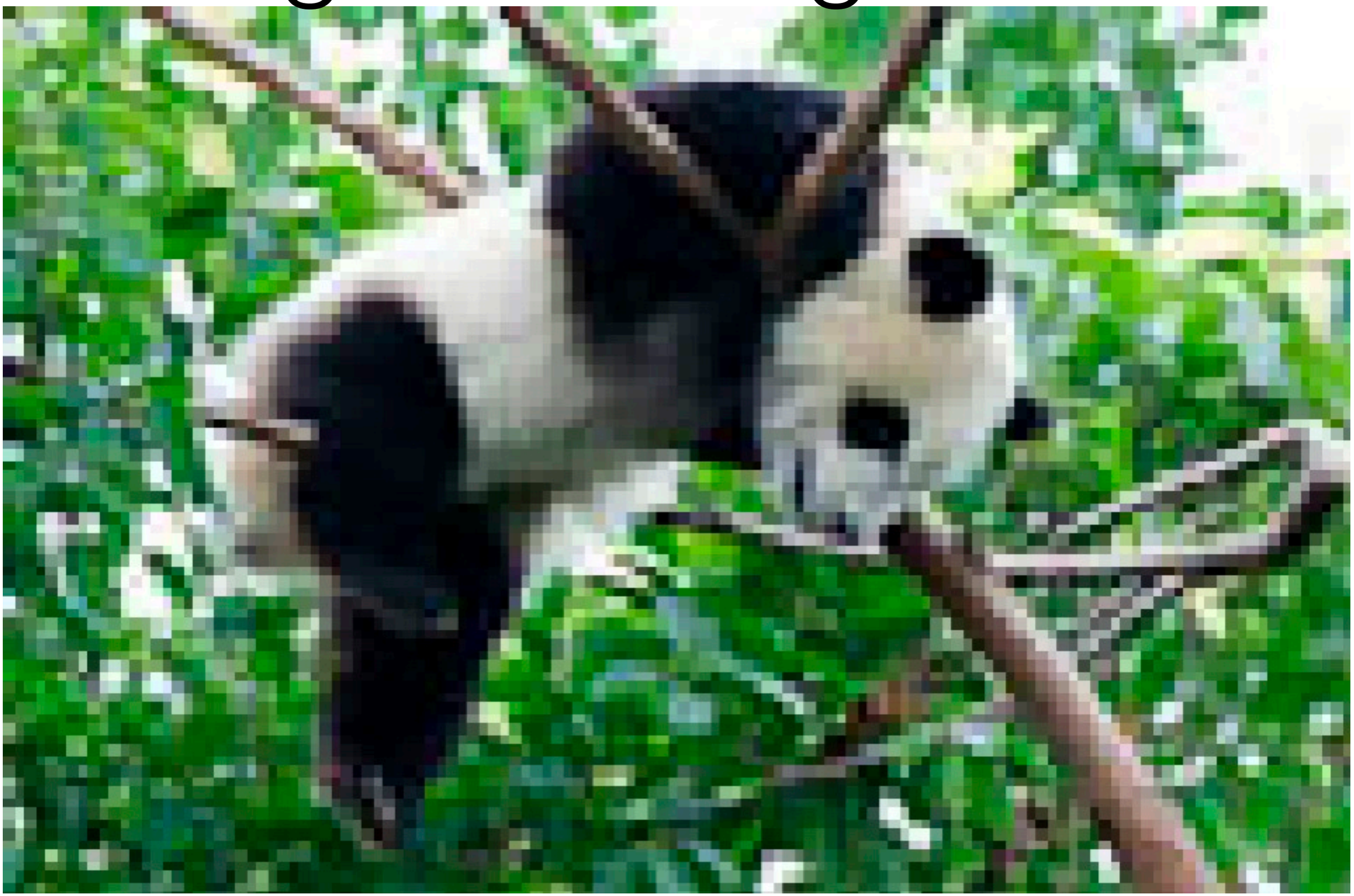


Image Zooming Results



What is the Problem?

- The output image is too “blocky”
- Because our image reconstruction rounds to the nearest integer pixel coordinates
    - Rounding to the “nearest” is why this type of interpolation is called **nearest neighbor interpolation**

Nearest Neighbor Interpolation

- Assume that a destination location (x, y) maps backward to source location (u, v).

- The source pixel nearest location (u, v) is located at (round(u), round(v)) and the source pixel at that image is then carried over as the value of the destination.

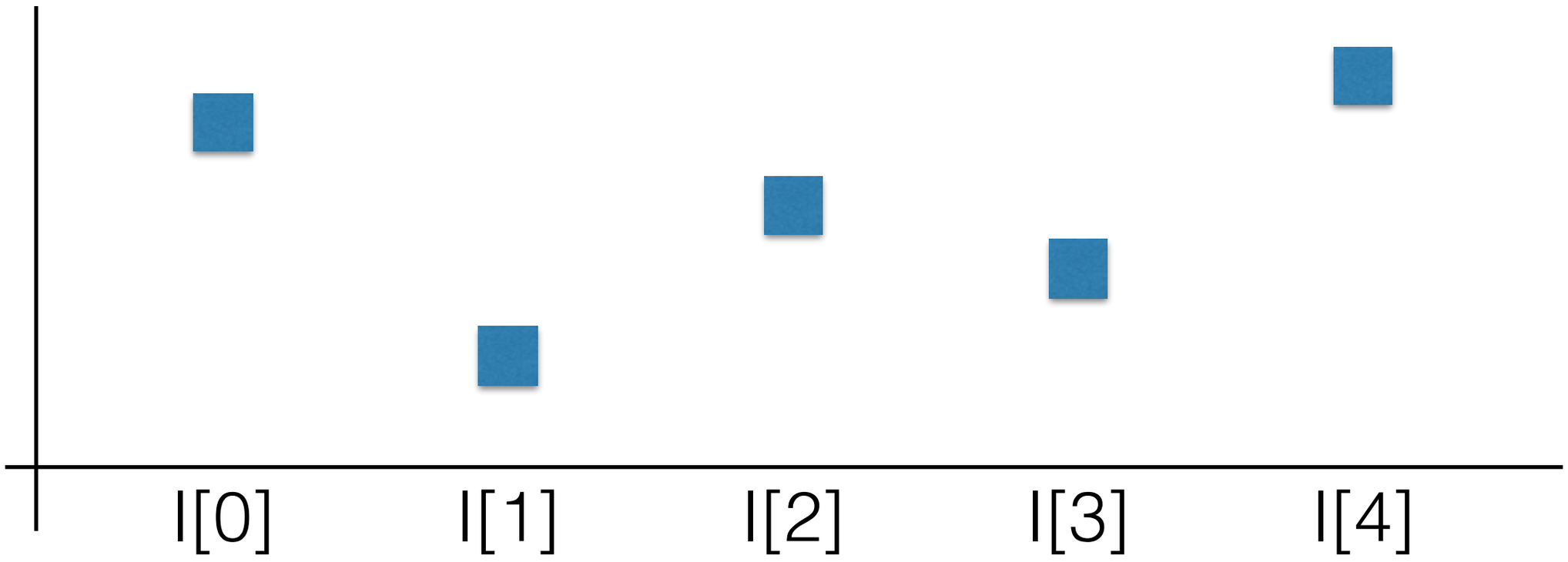
- In other words, Output[y][x] = Input[round(v)][round(u)]

- Nearest neighbor interpolation is computationally efficient but of generally poor quality, producing images with jagged edges and high graininess.

Point Spread Function (PSF) Demo

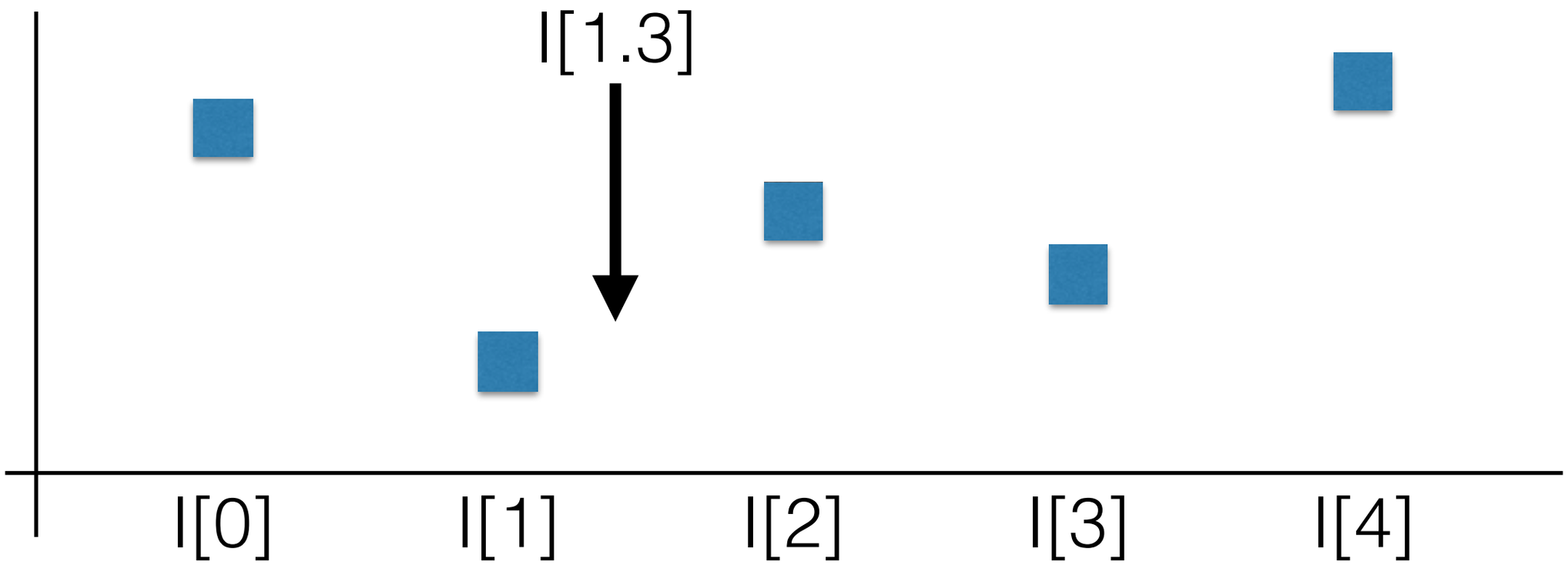
Nearest Neighbor Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?



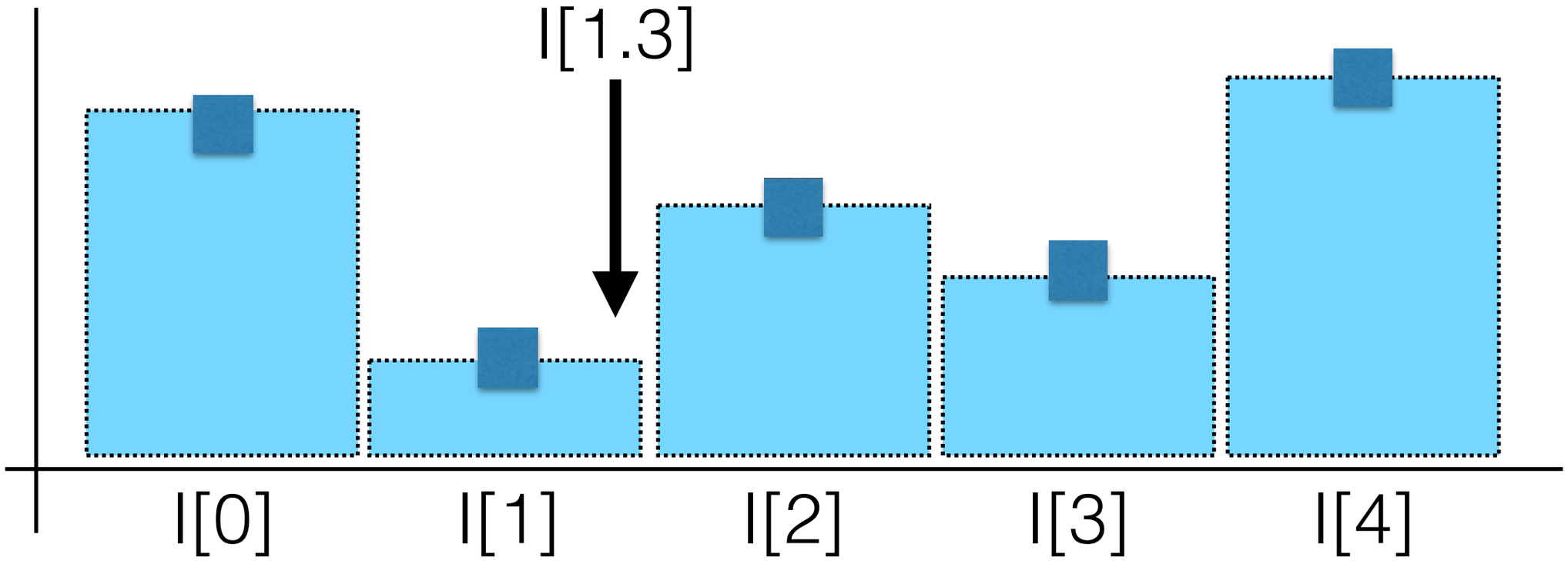
Nearest Neighbor Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?



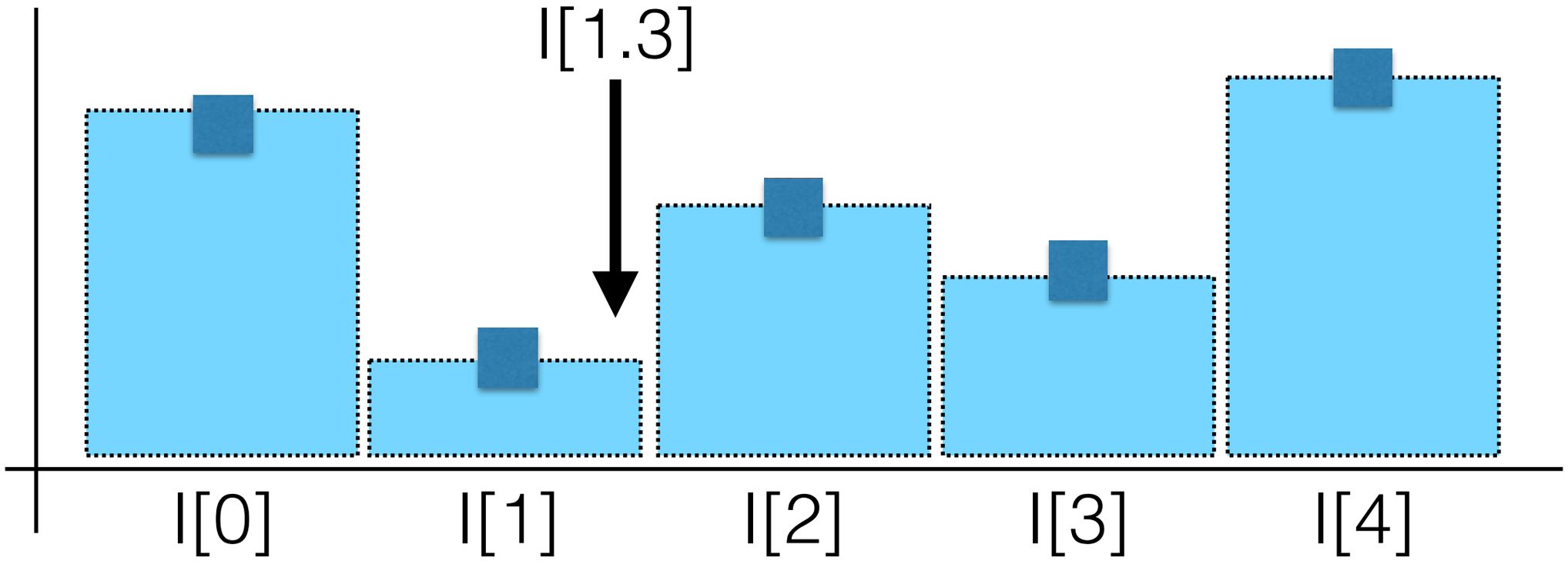
Nearest Neighbor Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?



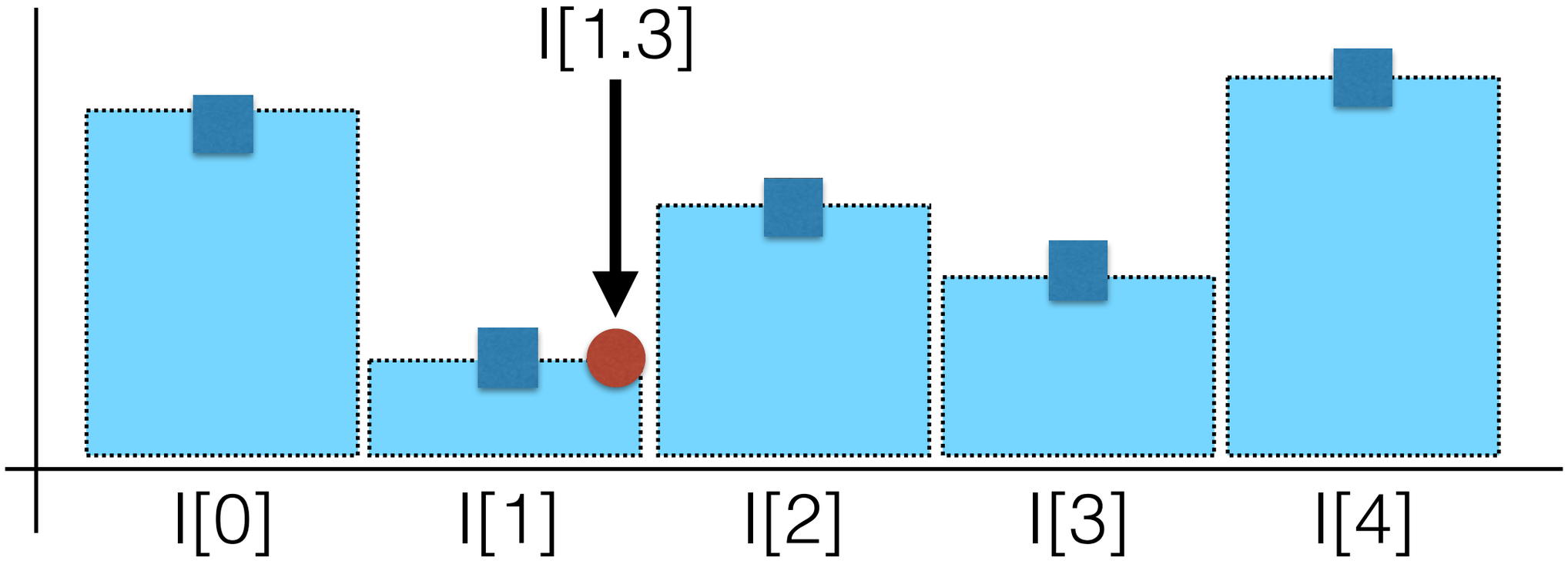
Nearest Neighbor Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?
  - I[1.3] = I[round(1.3)] = I[1]



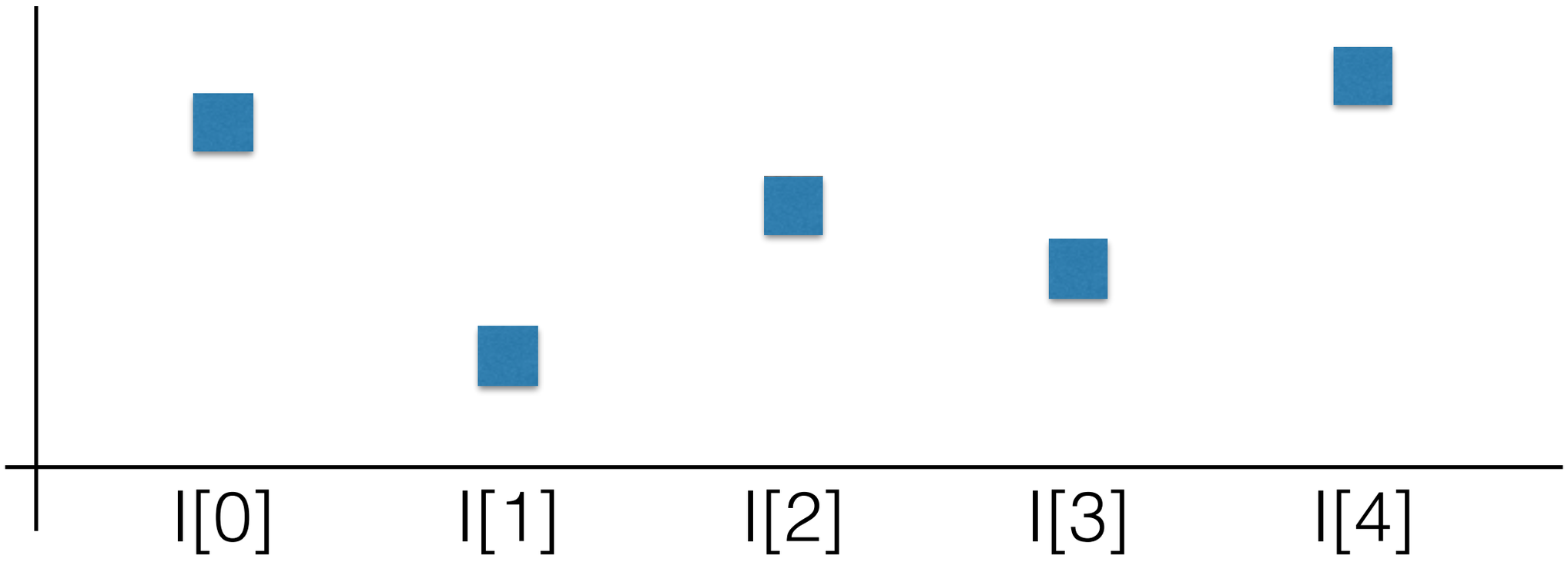
Nearest Neighbor Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?
  - I[1.3] = I[round(1.3)] = I[1]



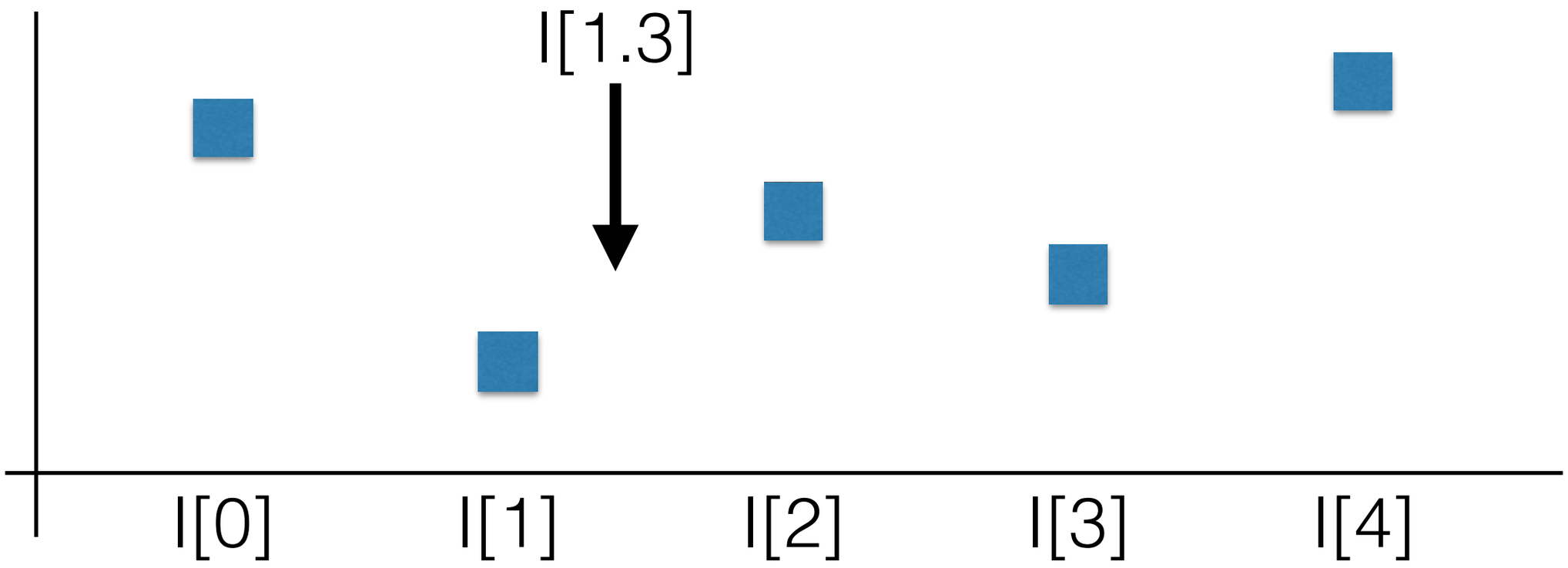
Linear Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?



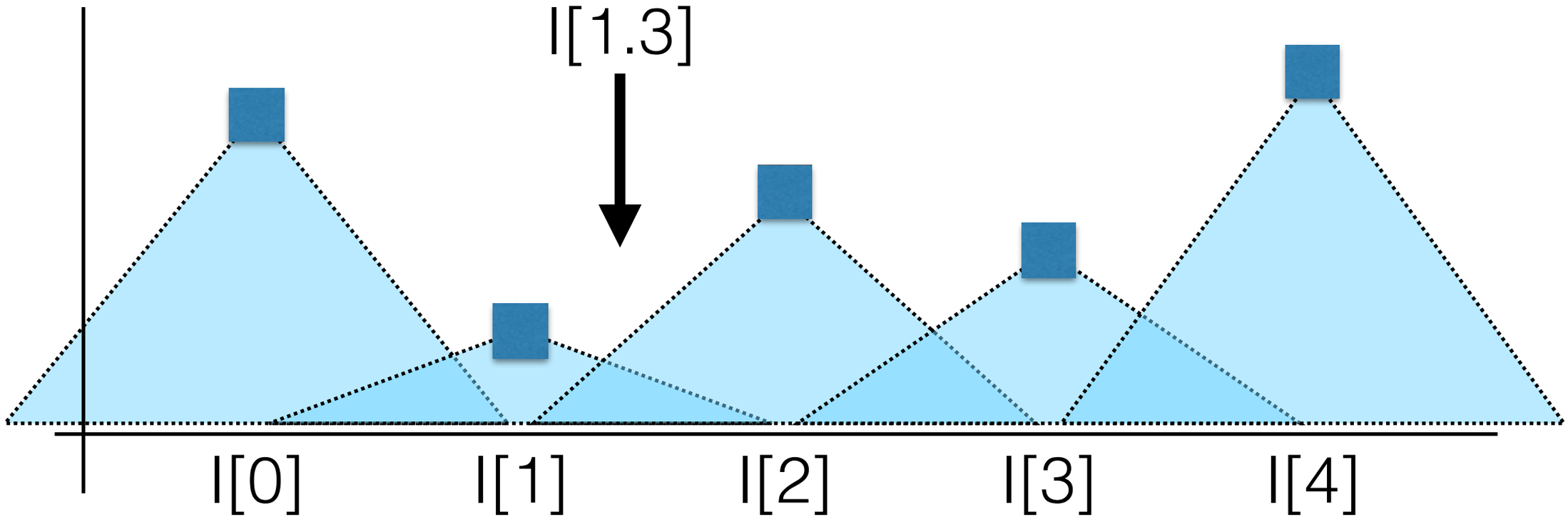
Linear Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?



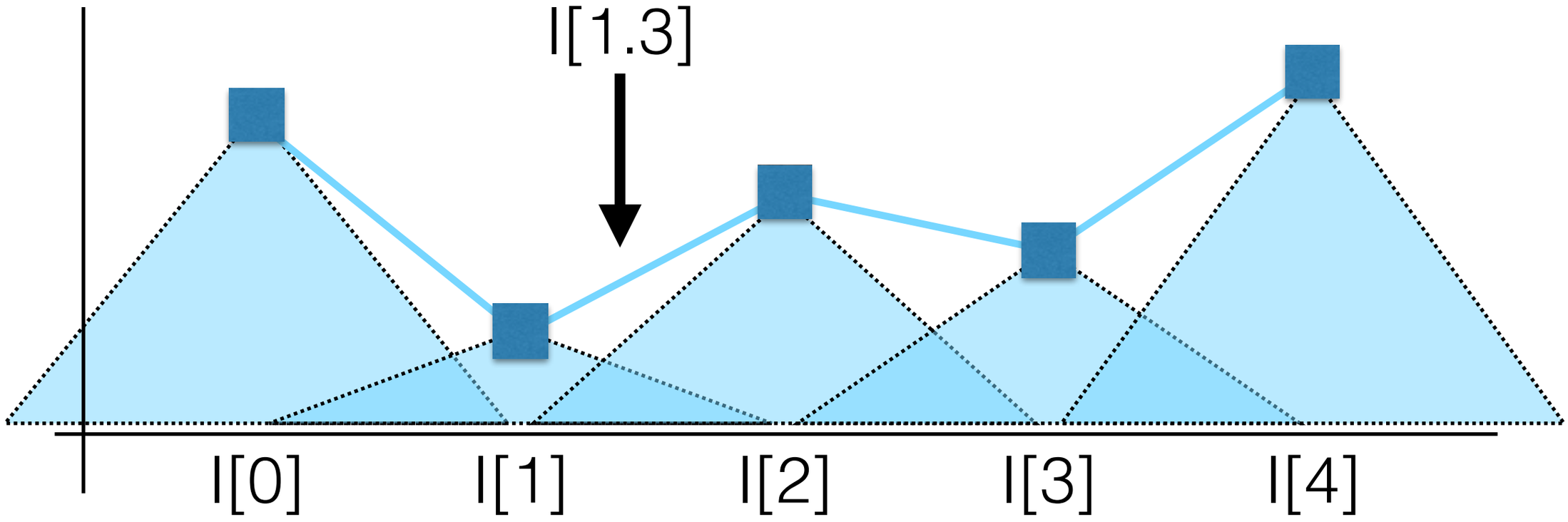
Linear Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?



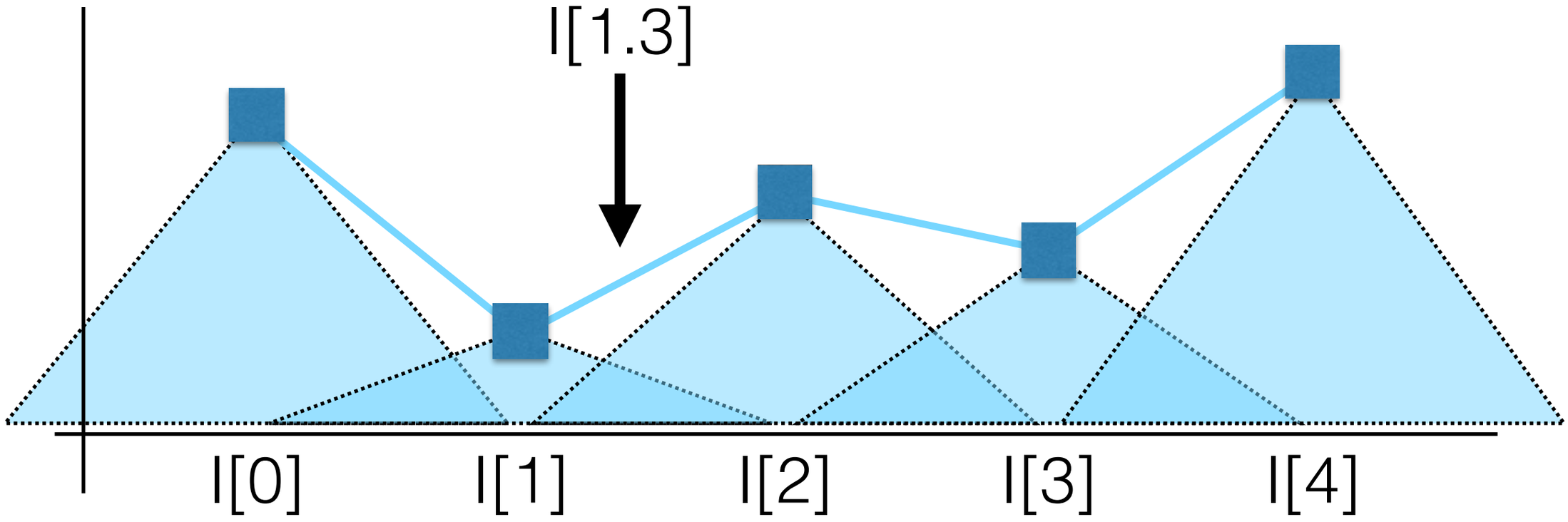
Linear Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?



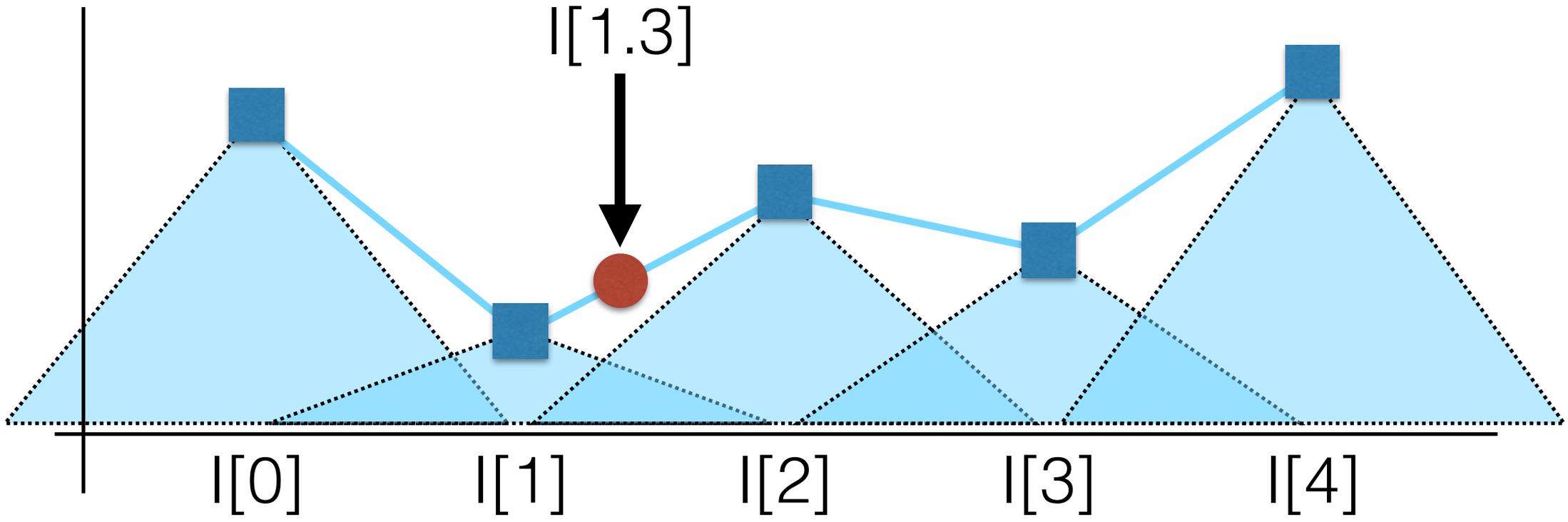
Linear Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?
  - I[1.3] = 0.7*I[1] + 0.3*I[2]



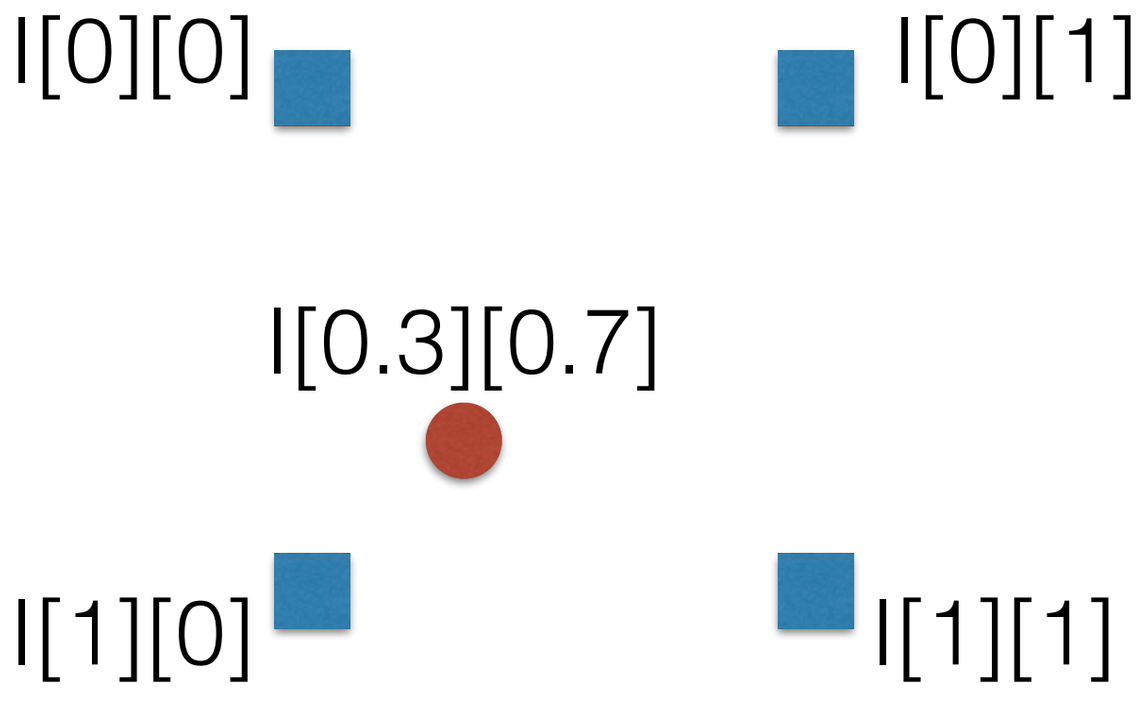
Linear Interpolation w/ PSFs

- Consider a 1-dimensional, grayscale image I spread horizontally
- What value is I[1.3] ?
  - I[1.3] = 0.7*I[1] + 0.3*I[2]



Bilinear Reconstruction

- Examine the four nearest neighbors, corners of the box which contain a sample
- Weight according to horizontal and vertical offsets relative to the box
- We can express this as linear interpolations in both the x and y direction



Bilinear Interpolation

- **Bilinear interpolation** assumes that the continuous image is a linear function of spatial location.

- Linear, or first order, interpolation combines the four points surrounding location (x,y), where (x, y) is the backward mapped coordinate that is surrounded by the four samples at (j,k), (j, k+1), (j +1, k), and (j+1, k+1)

$$
\begin{aligned}
j &= \lfloor x \rfloor, \\
k &= \lfloor y \rfloor, \\
a &= x - j, \\
b &= y - k, \\
I'(x', y') &= [1-b, b] \cdot \begin{bmatrix} I(j,k) & I(j+1,k) \\ I(j,k+1) & I(j+1,k+1) \end{bmatrix} \cdot \begin{bmatrix} 1-a \\ a \end{bmatrix}.
\end{aligned}
\tag{7.8}
$$

Bilinear Interpolation

- Bilinear interpolation is a weighted average where pixels closer to the backward mapped coordinate are weighted proportionally heavier than those pixels further away.

- Bilinear interpolation acts like something of a rigid mechanical system

1. Two rods vertically connect the four samples surrounding the backward mapped coordinate.

2. A third rod is connected horizontally which is allowed to slide vertically up and down the fixture.

3. A ball is attached to this horizontal rod and is allowed to slide freely back and forth across the central rod, determining the interpolated sample value wherever the ball is located.

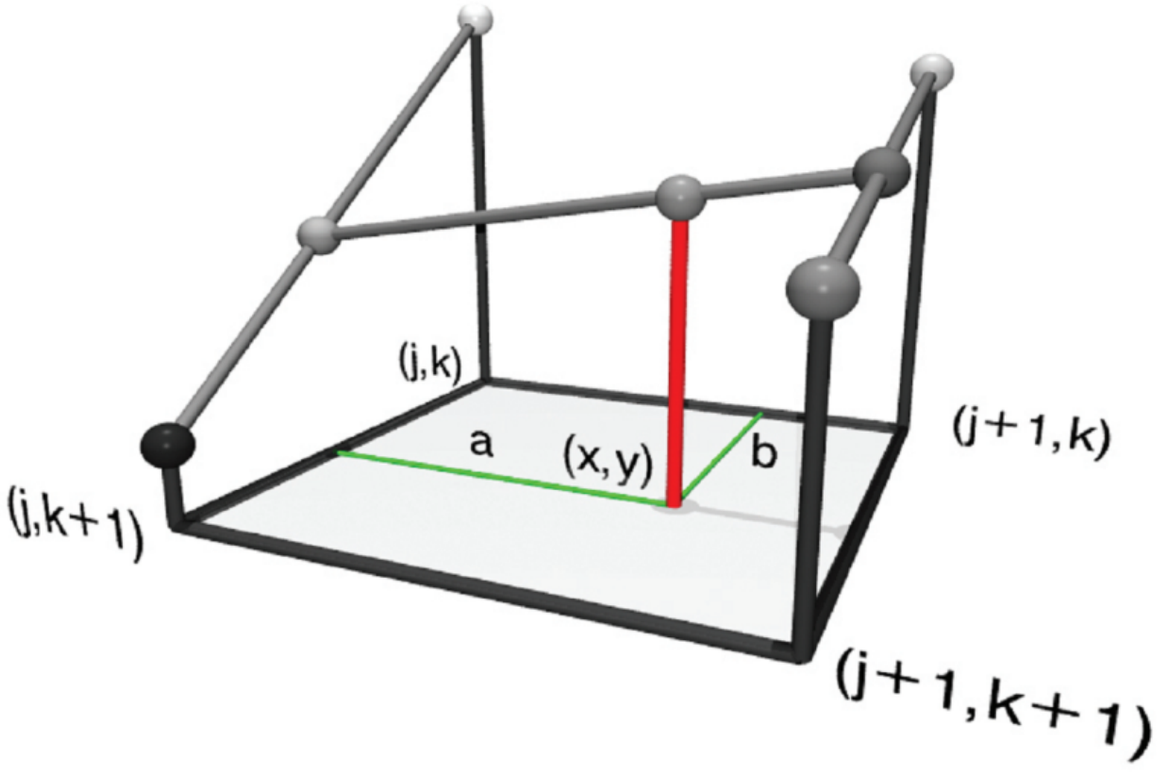
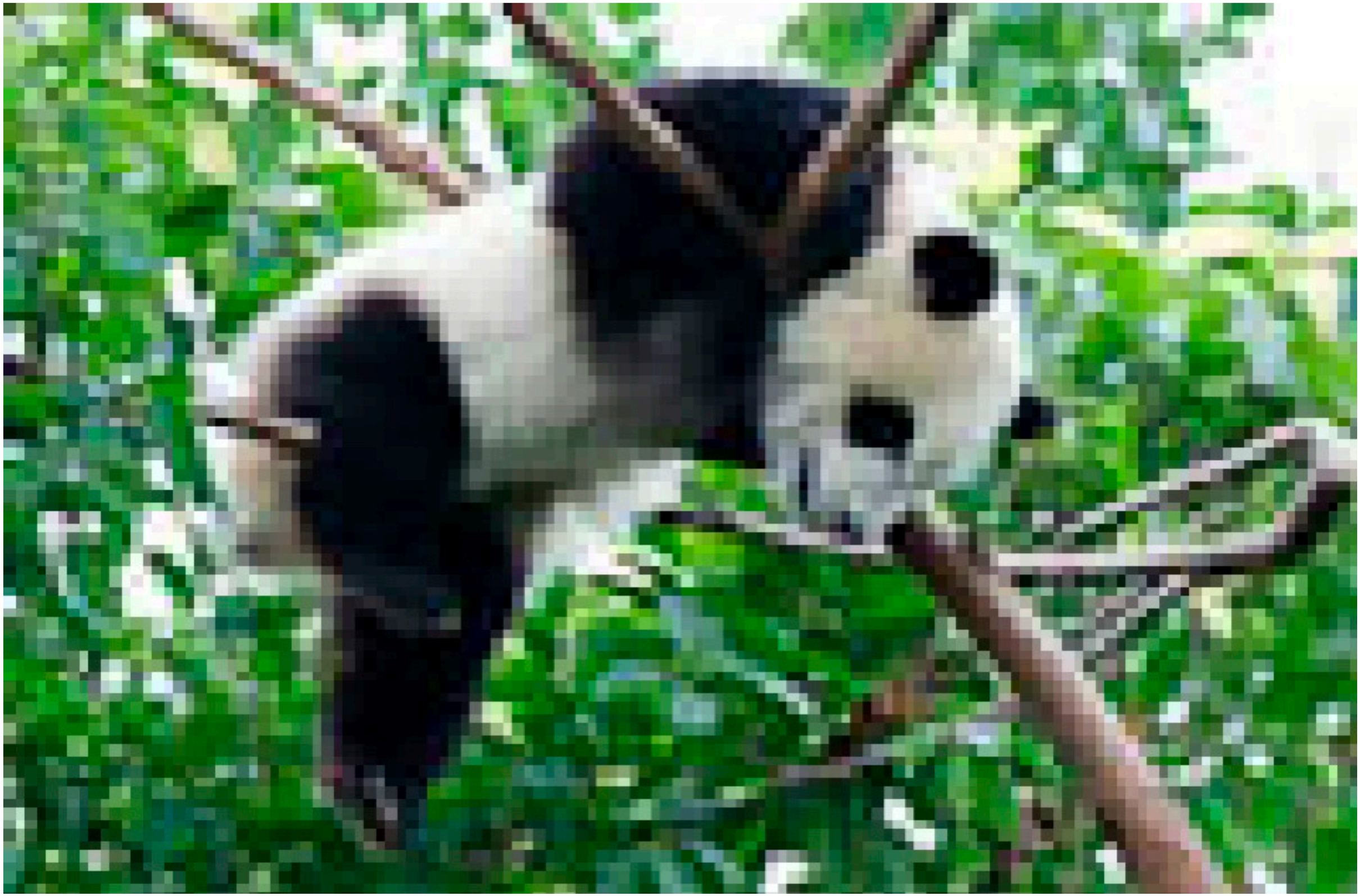
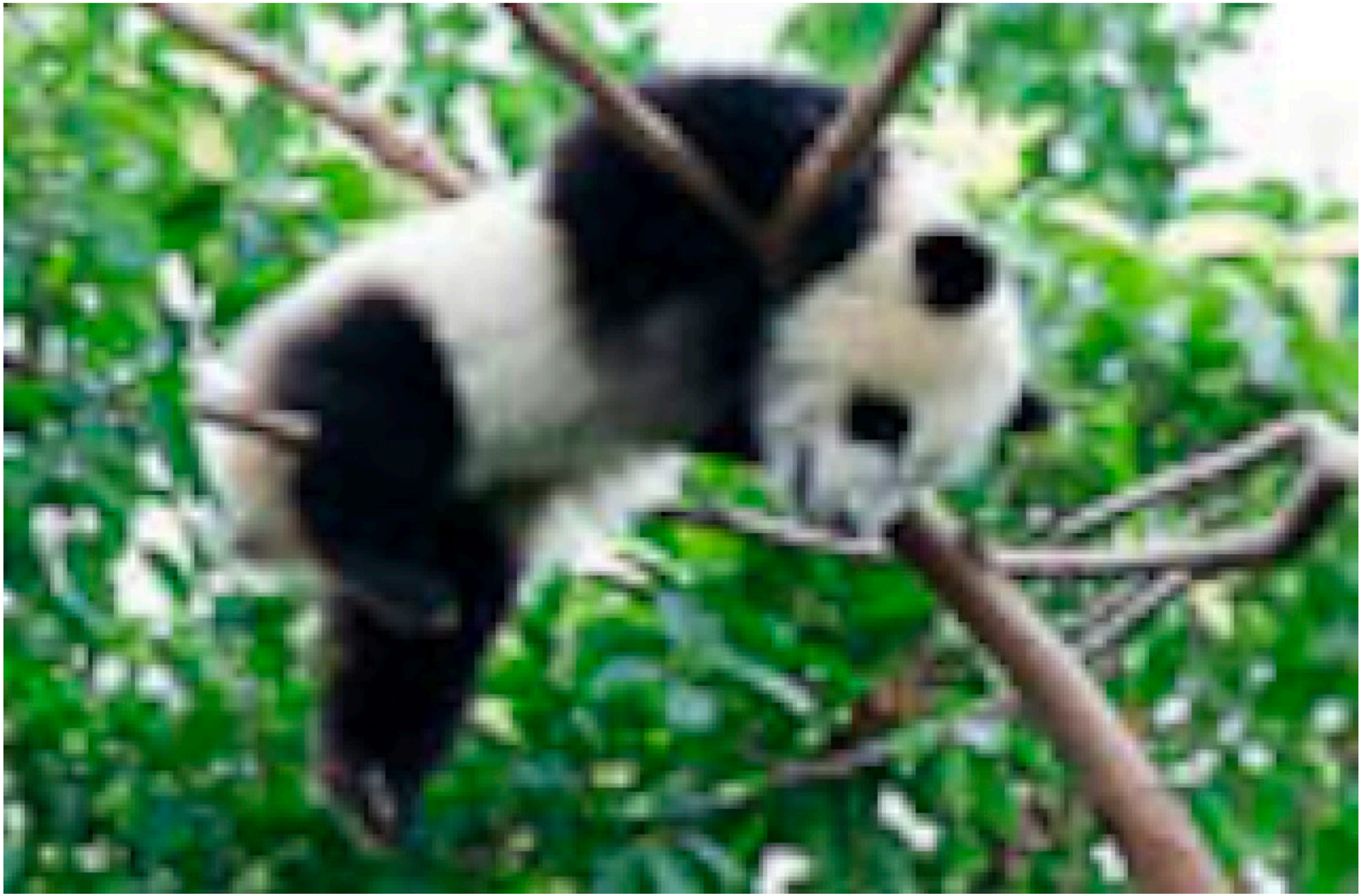


Figure 7.5. Bilinear interpolation.

Nearest Neighbor Example



Bilinear Example



Bicubic Interpolation

- In bicubic interpolation, the destination sample is a non-linear weighted sum of the 16 samples nearest to the reverse mapped location.

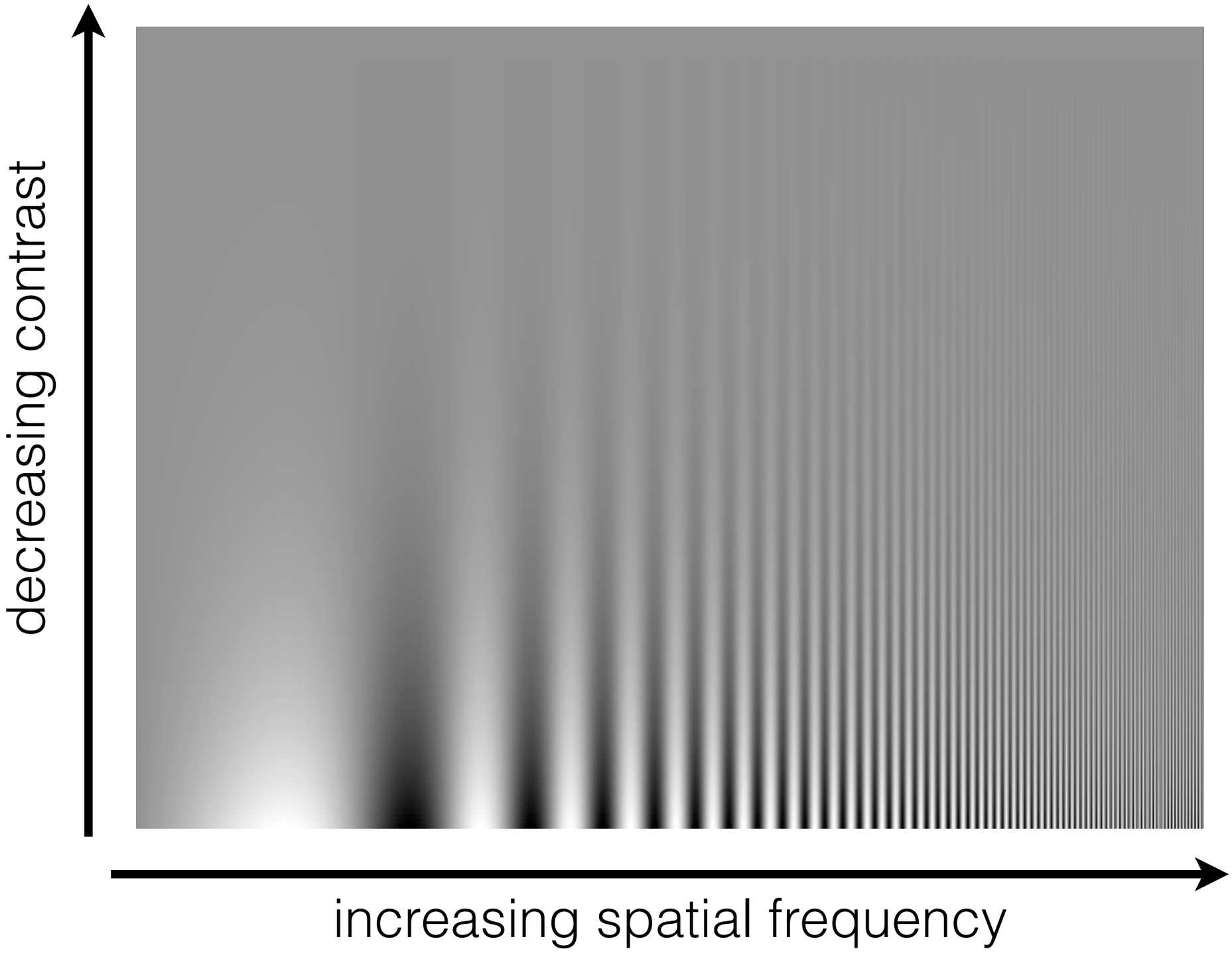
- This gives some nice properties, of second order interpolation:
    - Everywhere continuous
    - More computational effort required
- Can also do even better
    - Lanczos filtering
    - Edge sharpening?

Bicubic (from Photoshop)

Dynamic Range

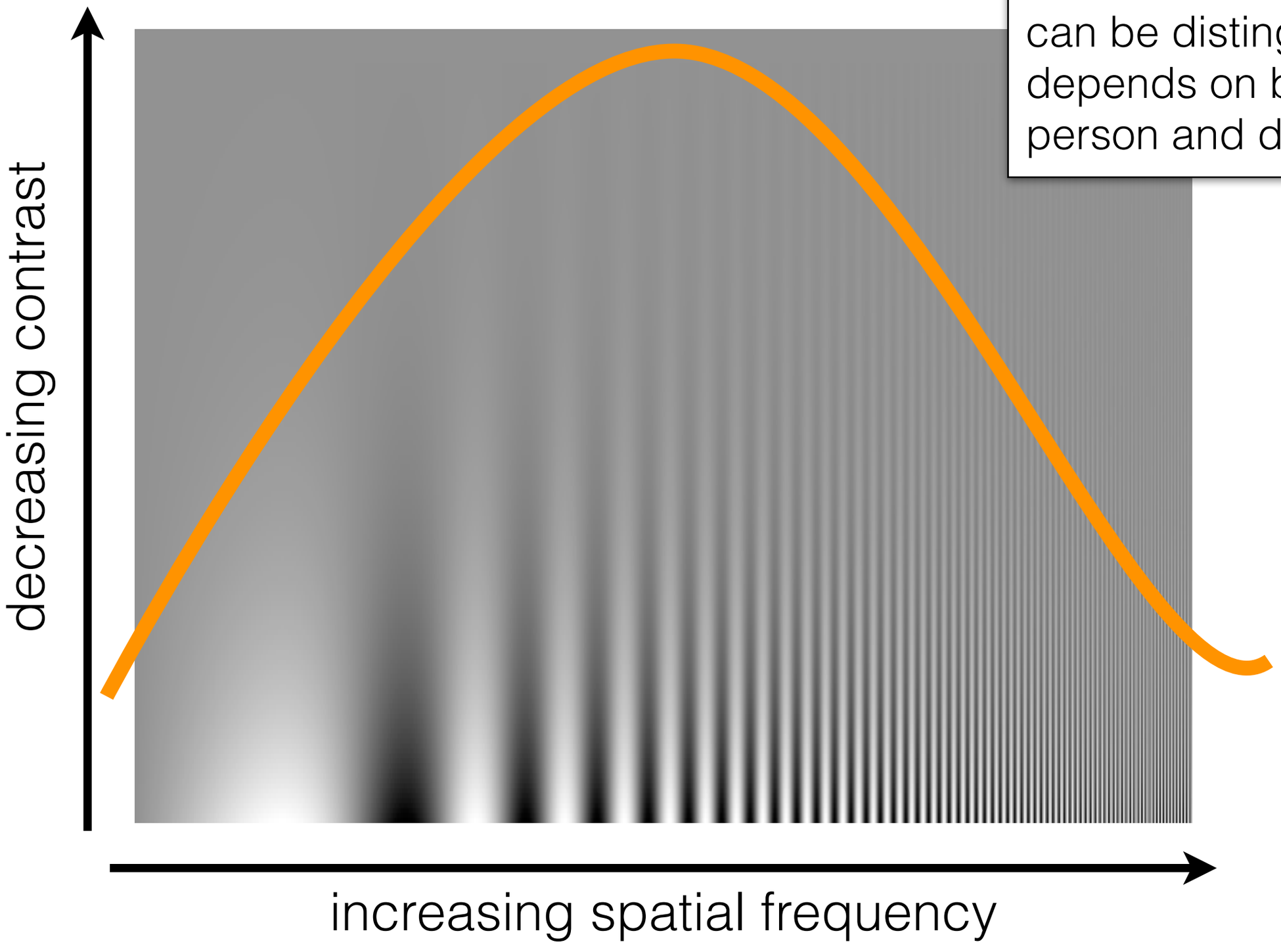
Contrast Sensitivity Function

Campbell-Robson Chart



Contrast Sensitivity Function

Campbell-Robson Chart



Where the bands can be distinguished depends on both the person and distance

Contrast Sensitivities Vary by Channel

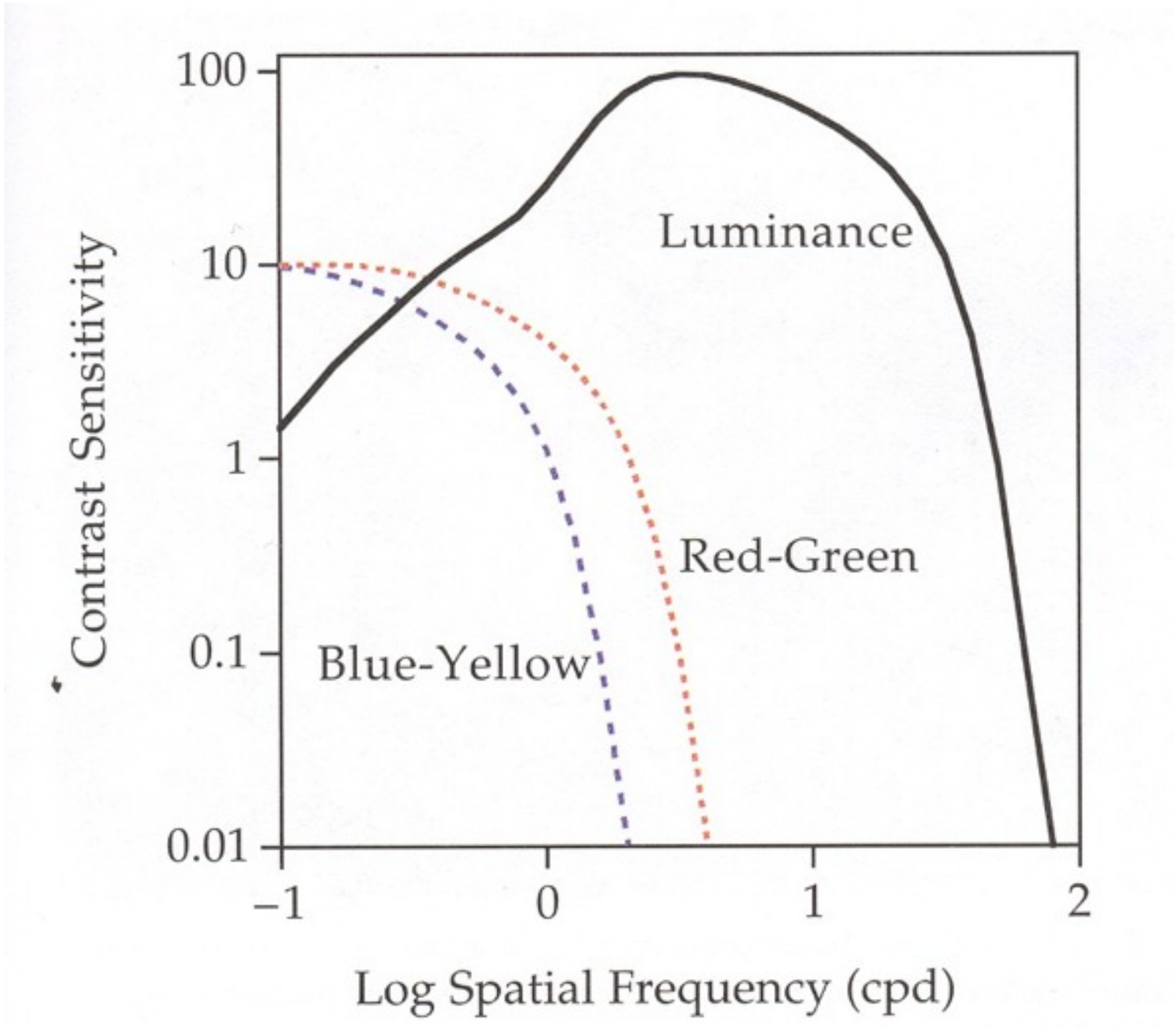


Figure 1-18. Spatial contrast sensitivity functions for luminance and chromatic contrast.

Photoshop demo

- Image > Mode > Lab color
- Go to channel panel, select Lightness
- Filter > Blur > Gaussian Blur , e.g. 4 pixel radius
  - very noticeable

- Undo, then select a & b channels
- Filter > Blur > Gaussian Blur , same radius
  - hardly visible effect

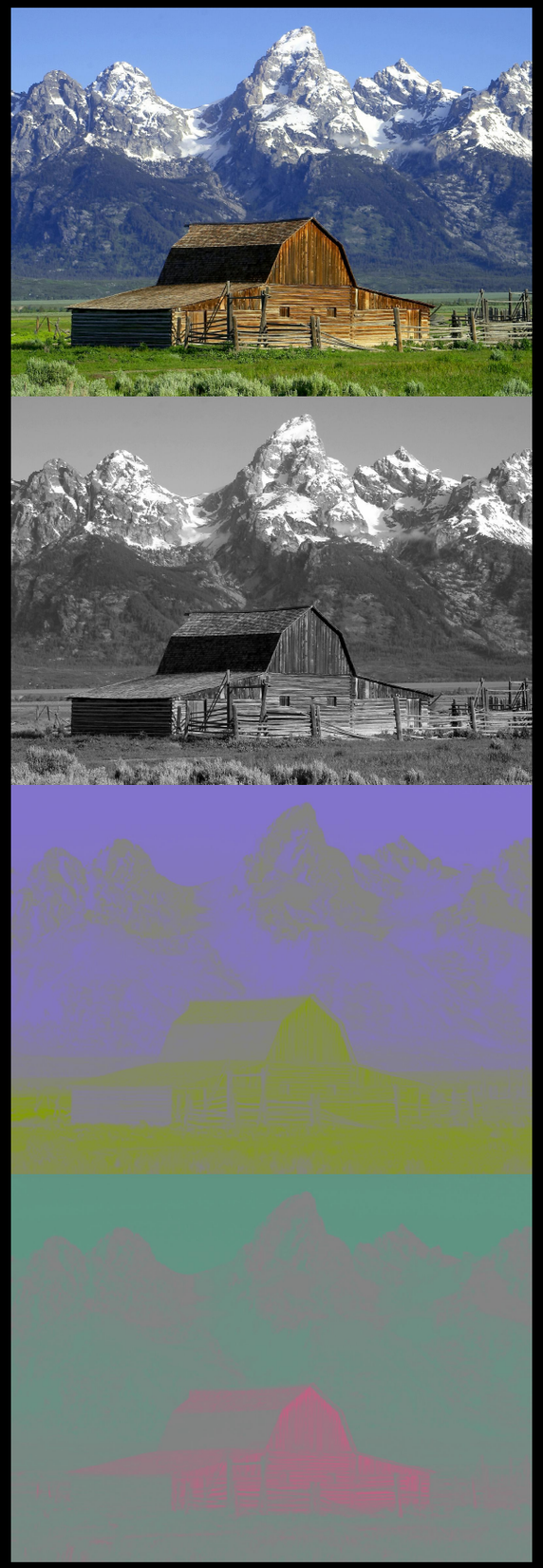
Original

Blur Lightness

Blur a & b

Opponents and image compression

- JPG, MPG, television
- Color opponents instead of RGB
  - YCrCB, similar to YUV
- Compress color more than luminance
  - downsample by factor of two for jpeg
  - less bandwidth for TV
- Exploit contrast sensitivity function
  - Compress high frequencies more

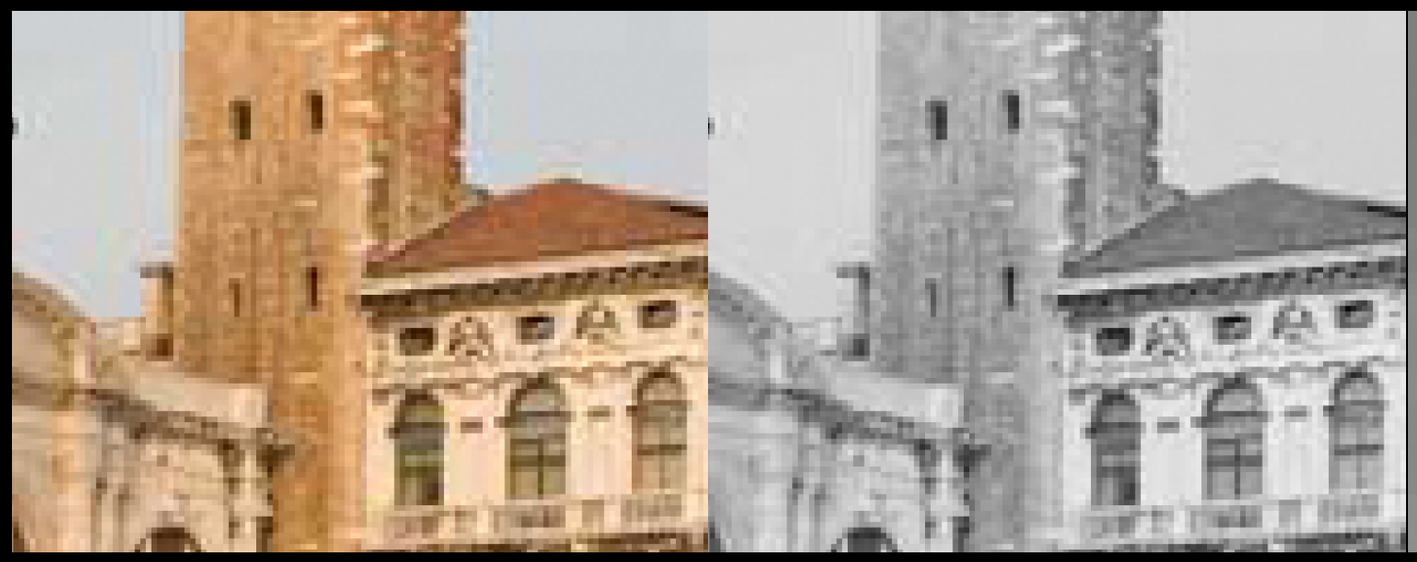


Example

- 800 x 533 image

RGB Lightness a

Low quality JPEG (0 in Photoshop, 172 KB)



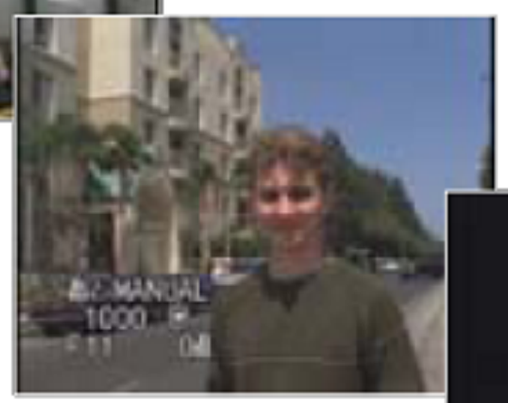
High quality JPEG (12 in Photoshop, 460 KB)

The World is a High Dynamic Range (HDR)

1:1

1:1,500

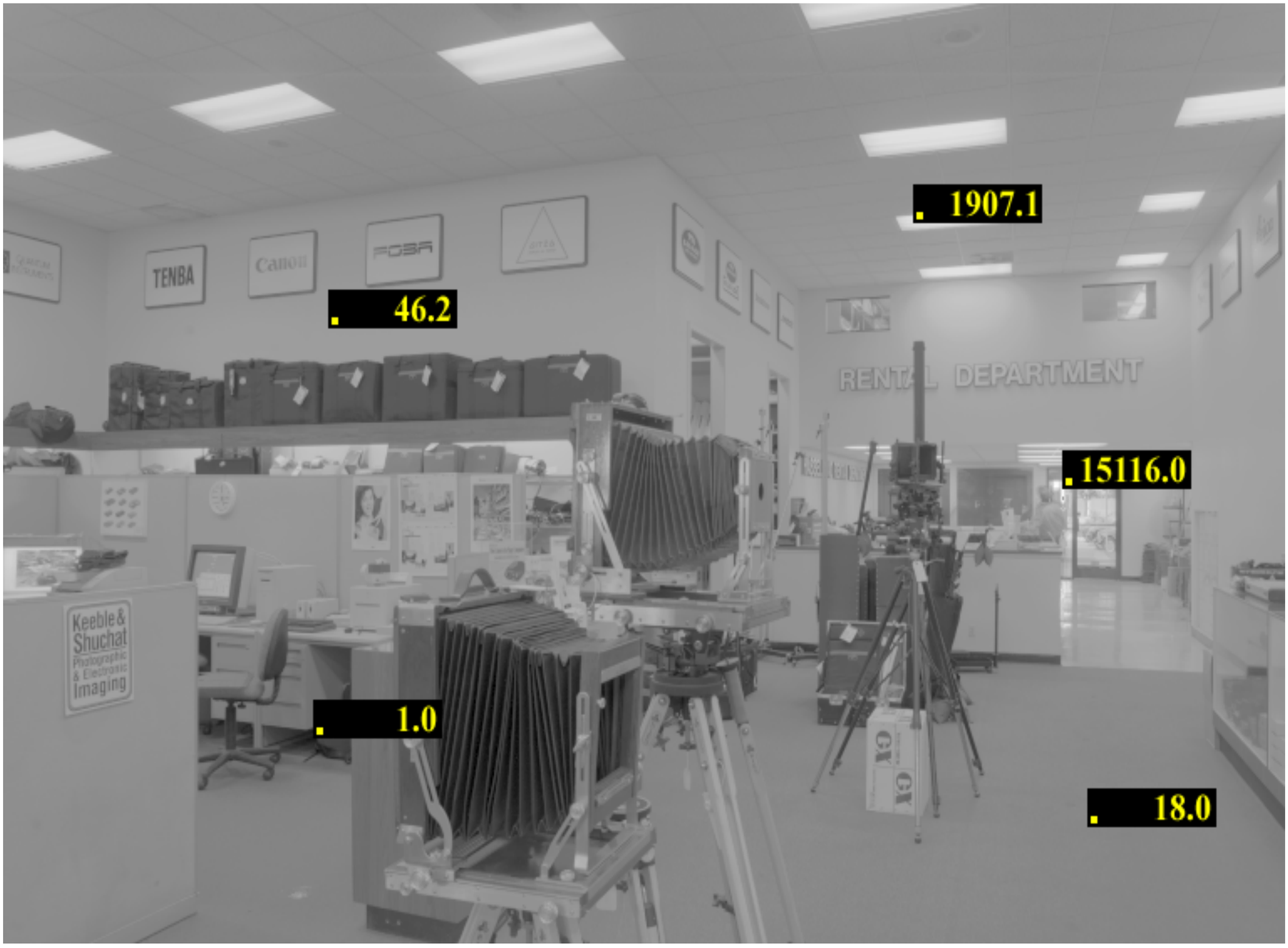
1:25,000



1:400,000

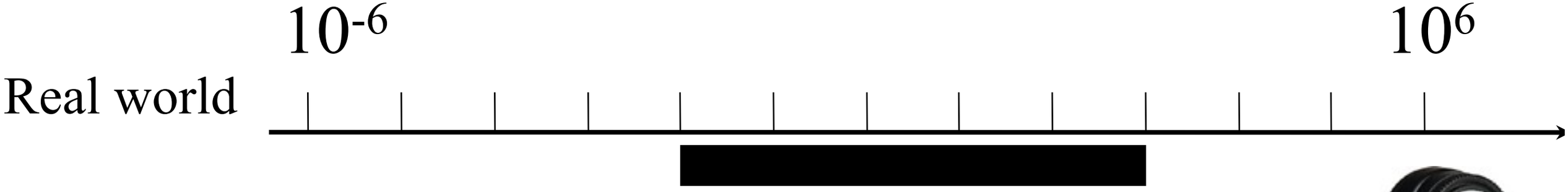
1:2,000,000,000

The World is a High Dynamic Range (HDR)



Real World Dynamic Range

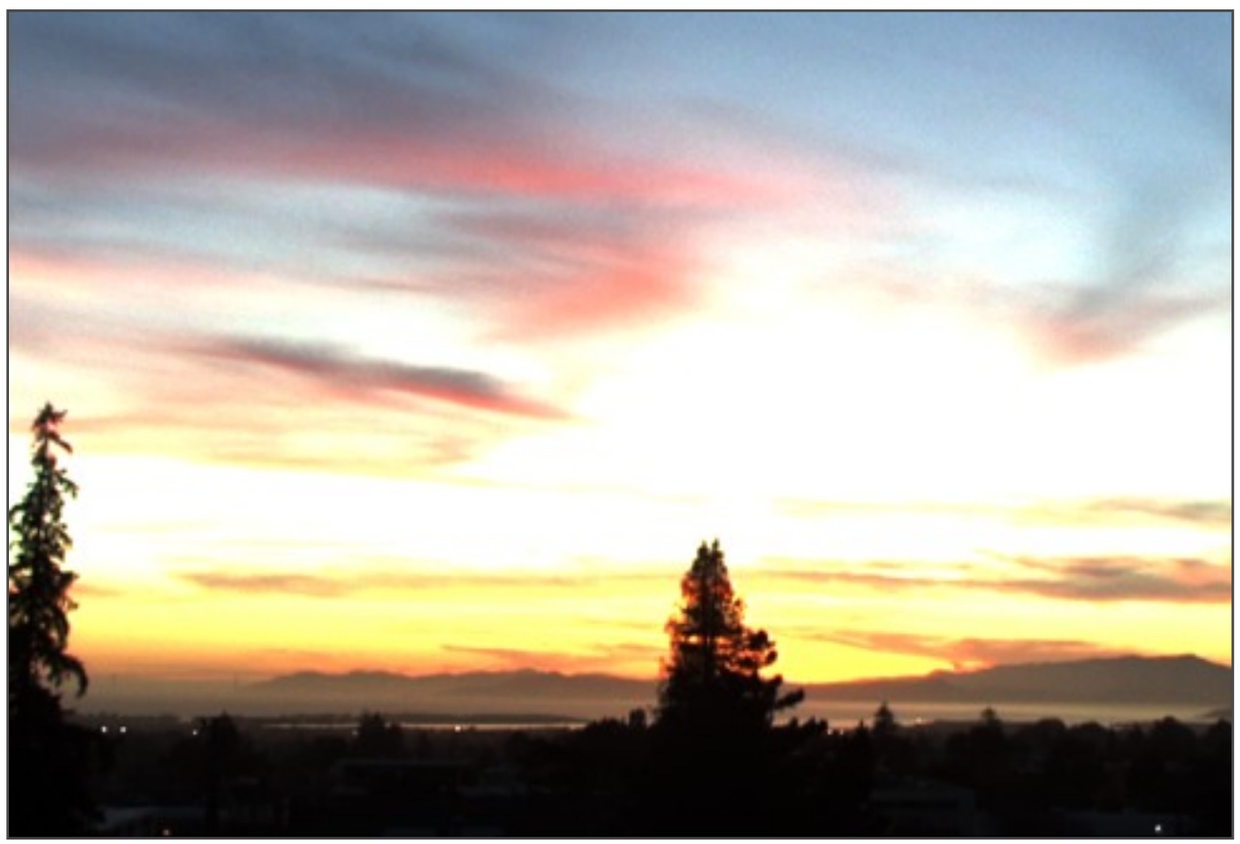
- Eye can adapt from ~10-6 to 106 cd/m2 (candela/area)
- Often 1:100,000 in a scene



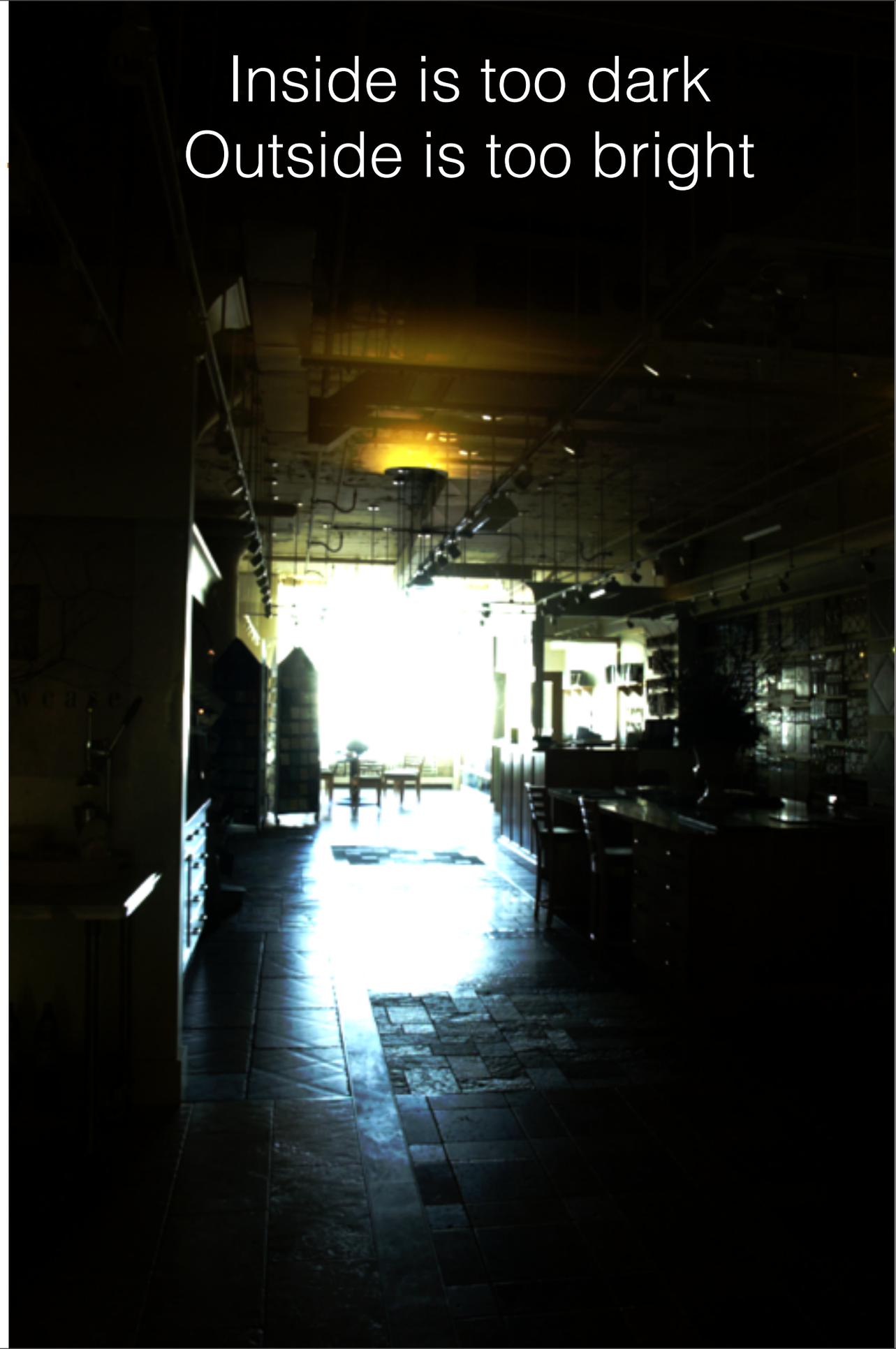
spotmeter

Examples

Sun overexposed
Foreground too dark



Inside is too dark
Outside is too bright

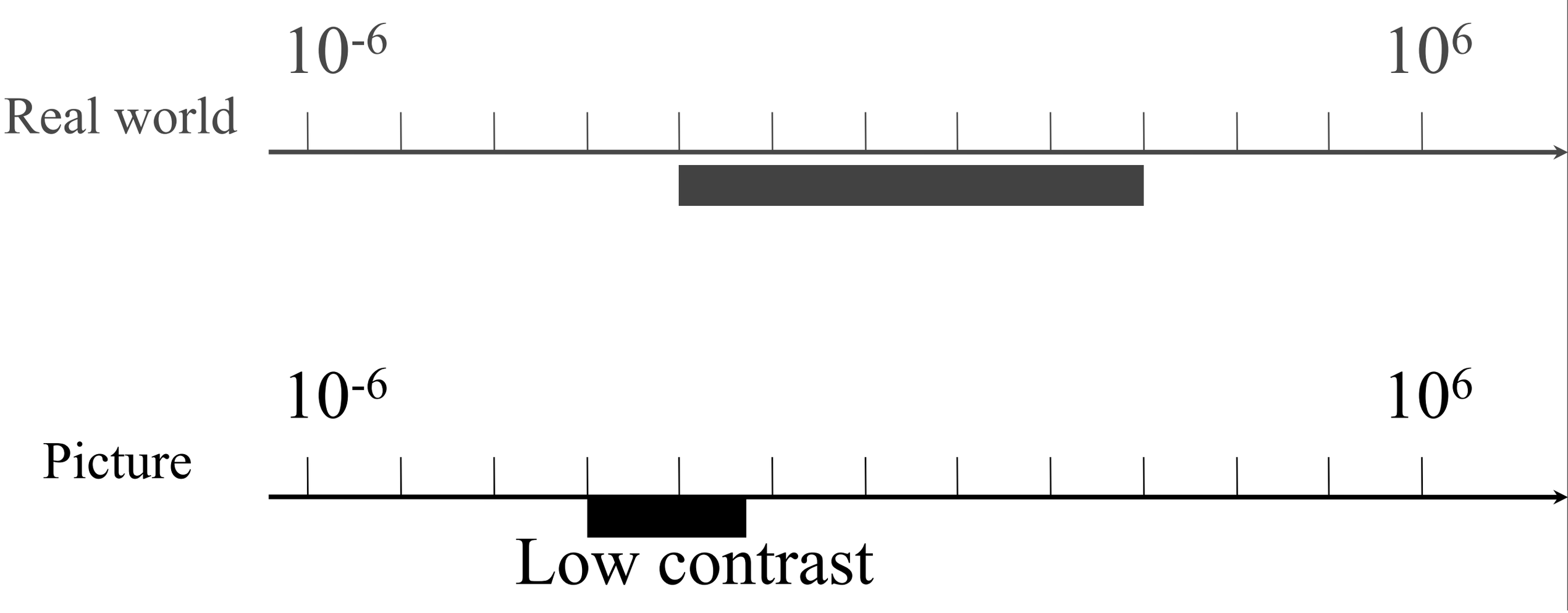


Dynamic Range in Displays

- Range of pure black vs. pure white?

Picture Dynamic Range

- **Typically 1: 20 or 1:50**
  - Black ■ is ~ 50x darker than white □
- **Max 1:500**



The Dynamic Range Problem

- For media (approximate and debatable)

  10:1 - photographic print (higher for glossy paper)
  20:1 - artist's paints
  200:1 - slide film
  500:1 - negative film
  1000:1 - LCD display
  2000:1 - digital SLR (~11 bits)

- Two Challenges

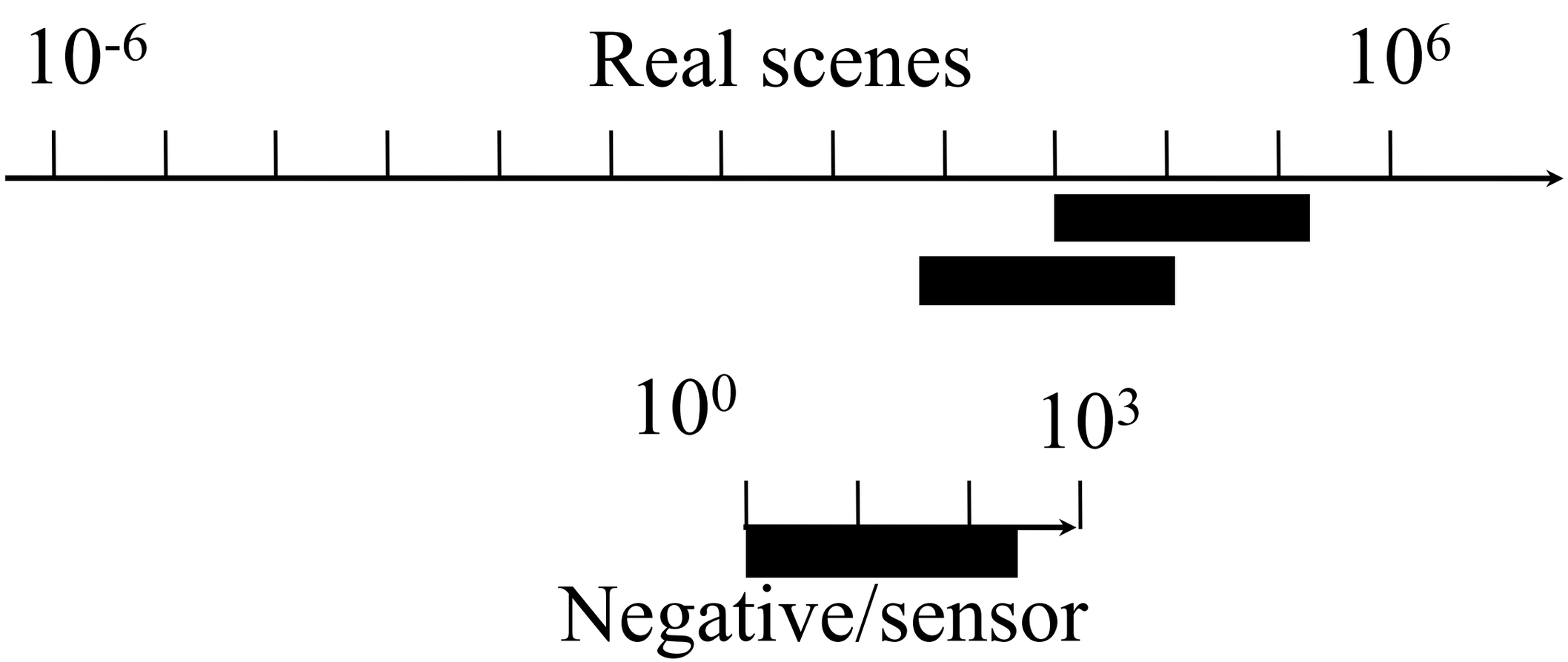
  1. Choosing which 6-12 bits of the world to include in your photograph (cell phone to professional SLR, respectively)

     - Metering the world to help you make this decision, since the world has more dynamic range than any light meter

  2. Compressing 12 bits into 4 bits for print, or 10 for LCD

Challenge 1: Recording the Information

- The range of illumination levels that we encounter is 10 to 12 orders of magnitudes
- Negatives/sensors can record 2 to 3 orders of magnitude
- Solution: Use HDR Formats



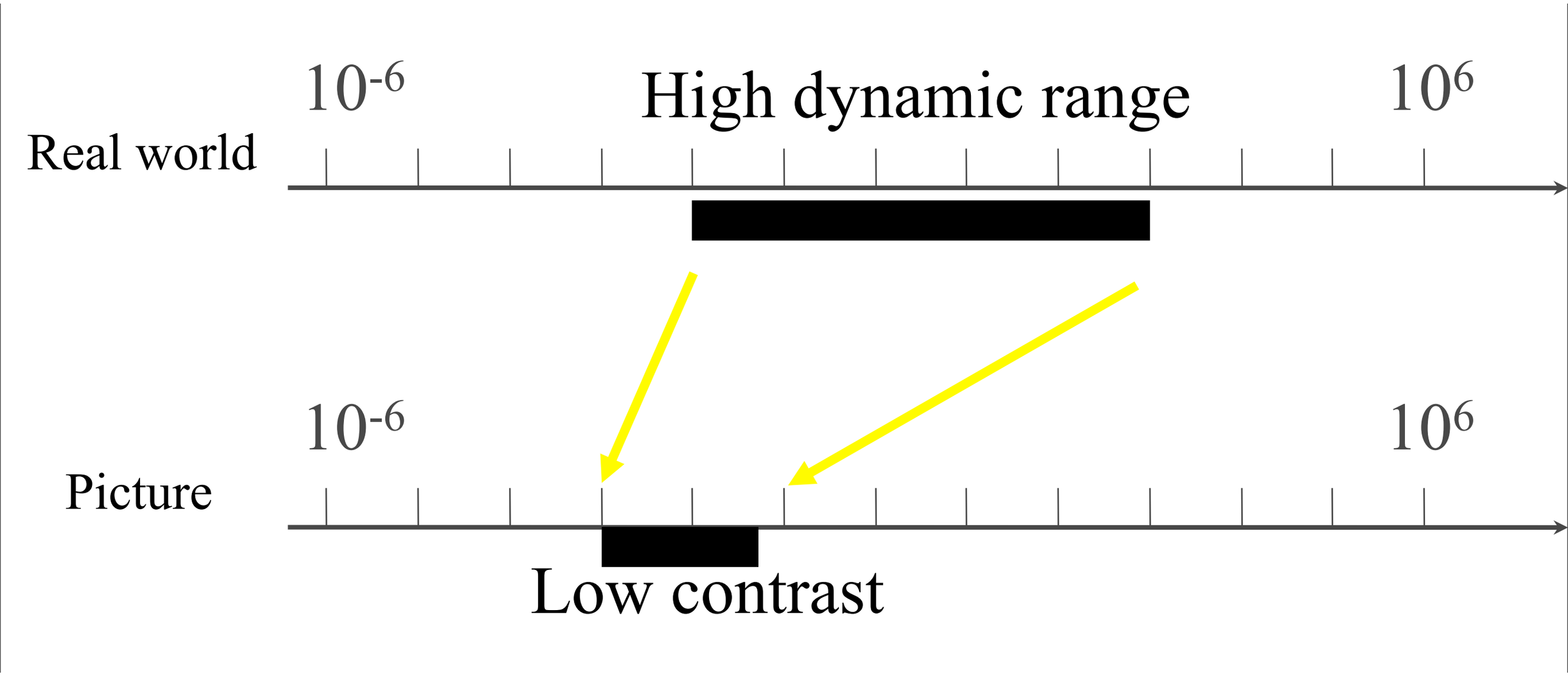
Best Technique for Shooting Interiors: HDR or Flash?

Have you ever tried to shoot an interior photograph and have it look like the shots in magazines or high end property brochures? If so then you probably know there are two routes to go: HDR or Flash. Photographer Dom Bower recently made a video showing the differences in both techniques and how you can combine them both to create a sort of hybrid image. Keep in mind that Dom is only using one single speedlight directly above the camera. Many of the amazing images you see for high end hotels and expensive properties often have dozens of light sources accenting very specific elements in the image. What techniques have you guys used in your interior photos? If you have examples, feel free to post your images in the comments below and check out Dom's final photos in the full post.

http://fstoppers.com/best-technique-for-shooting-interiors-hdr-or-flash
https://www.youtube.com/watch?v=i5ea1Kw0_pg

Problem 2: Displaying the Information

- Match limited contrast of the medium
- Preserve details
- Solution: **Tone Mapping** (we'll learn how next time!)



Without HDR + Tone Mapping

With HDR + Tone Mapping

http://www.stuckincustoms.com/2011/04/30/hdr-before-after/

http://abduzeedo.com/20-beautiful-hdr-pictures

Before

http://abduzeedo.com/20-beautiful-hdr-pictures

After

http://luminous-landscape.com/essays/hdr-plea.shtml

Not All Tone Mapping Produces Extreme Results,
Sometimes Just Beautiful Ones

HDR Image Acquisition

Important: Cameras are NOT Photometers

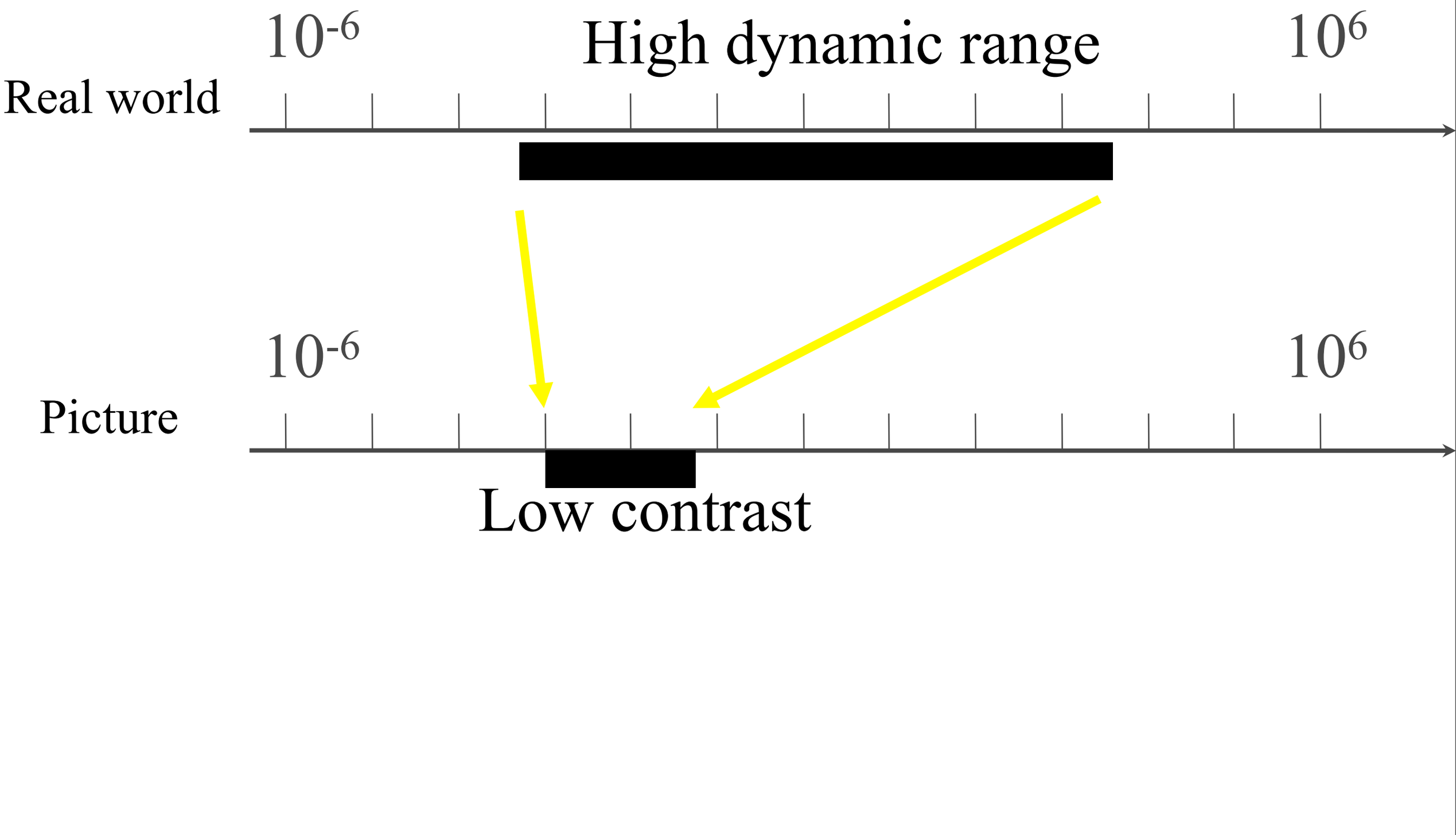
- Cameras are an imperfect device for measuring the radiance distribution of a scene because they cannot capture the full spectral content and dynamic range.

- Limitations in sensor design prevent cameras from capturing all information passed by lens.

- Idea: Recover response curve from multiple exposures, then reconstruct the radiance map

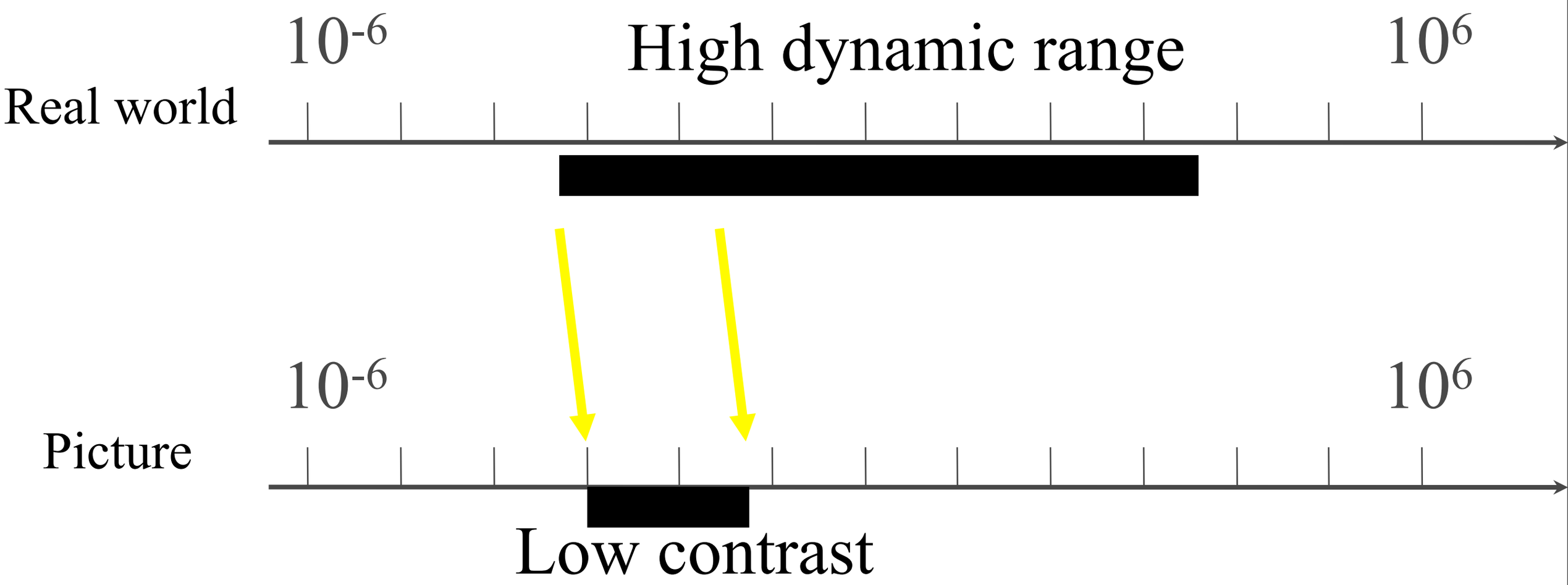
Multiple exposure photography

- **Sequentially measure all segments of the range**



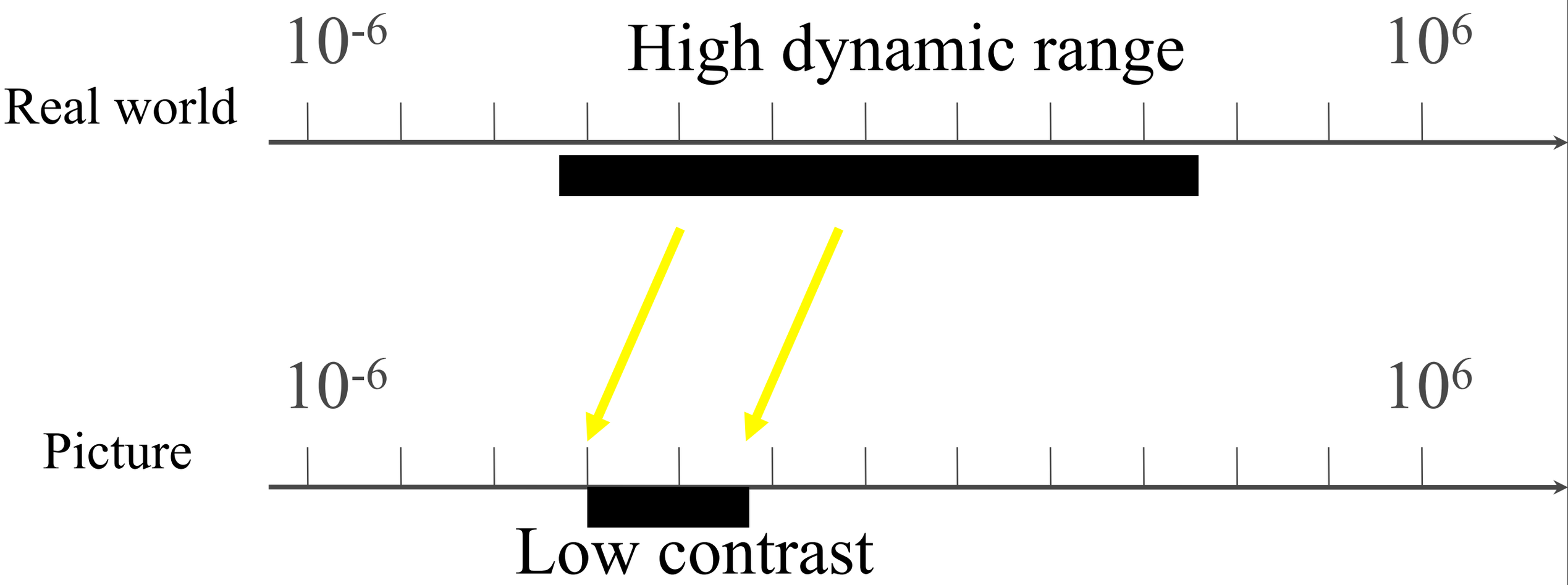
Multiple exposure photography

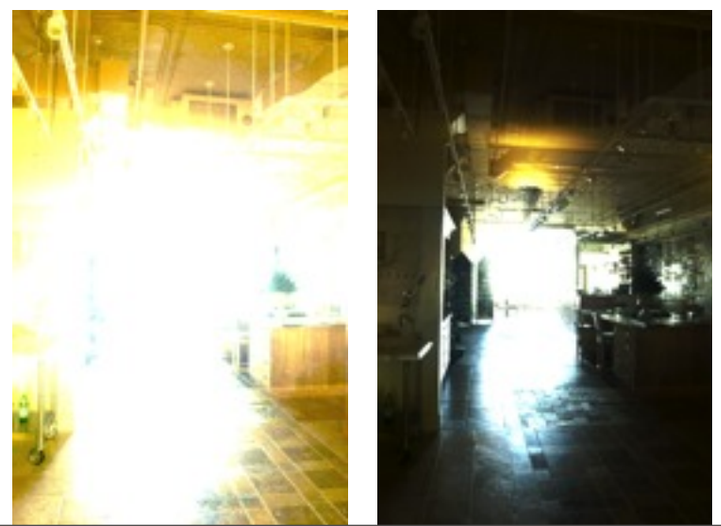
- **Sequentially measure all segments of the range**



Multiple exposure photography

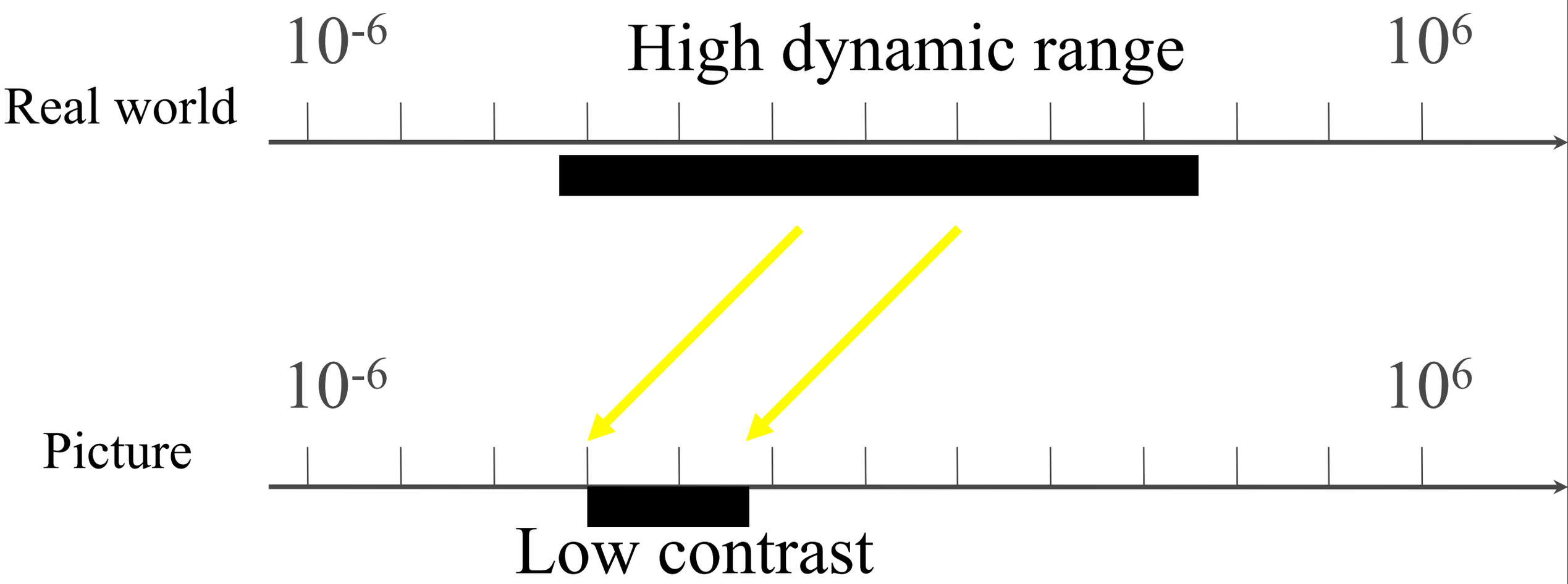
- **Sequentially measure all segments of the range**

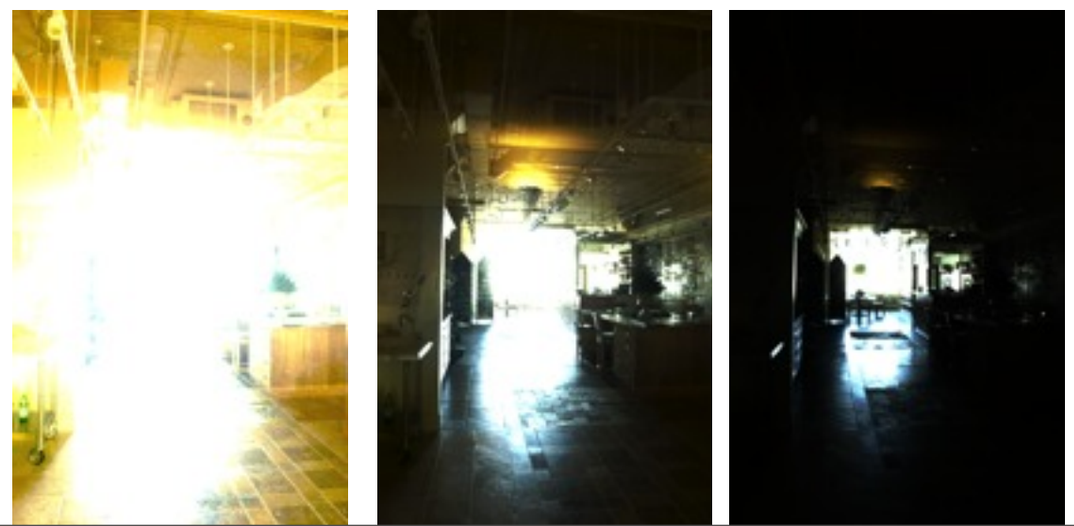




Multiple exposure photography

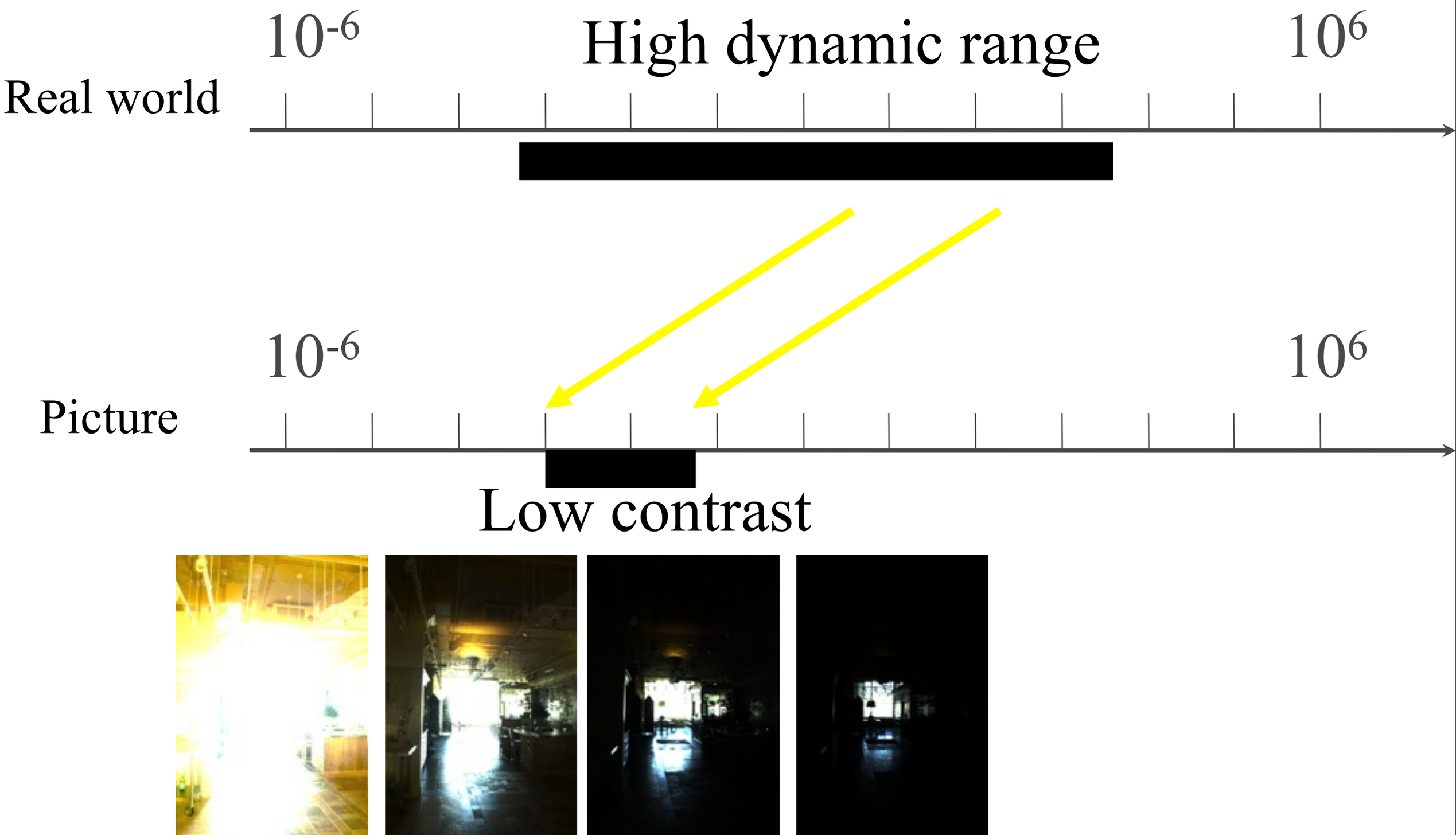
- **Sequentially measure all segments of the range**





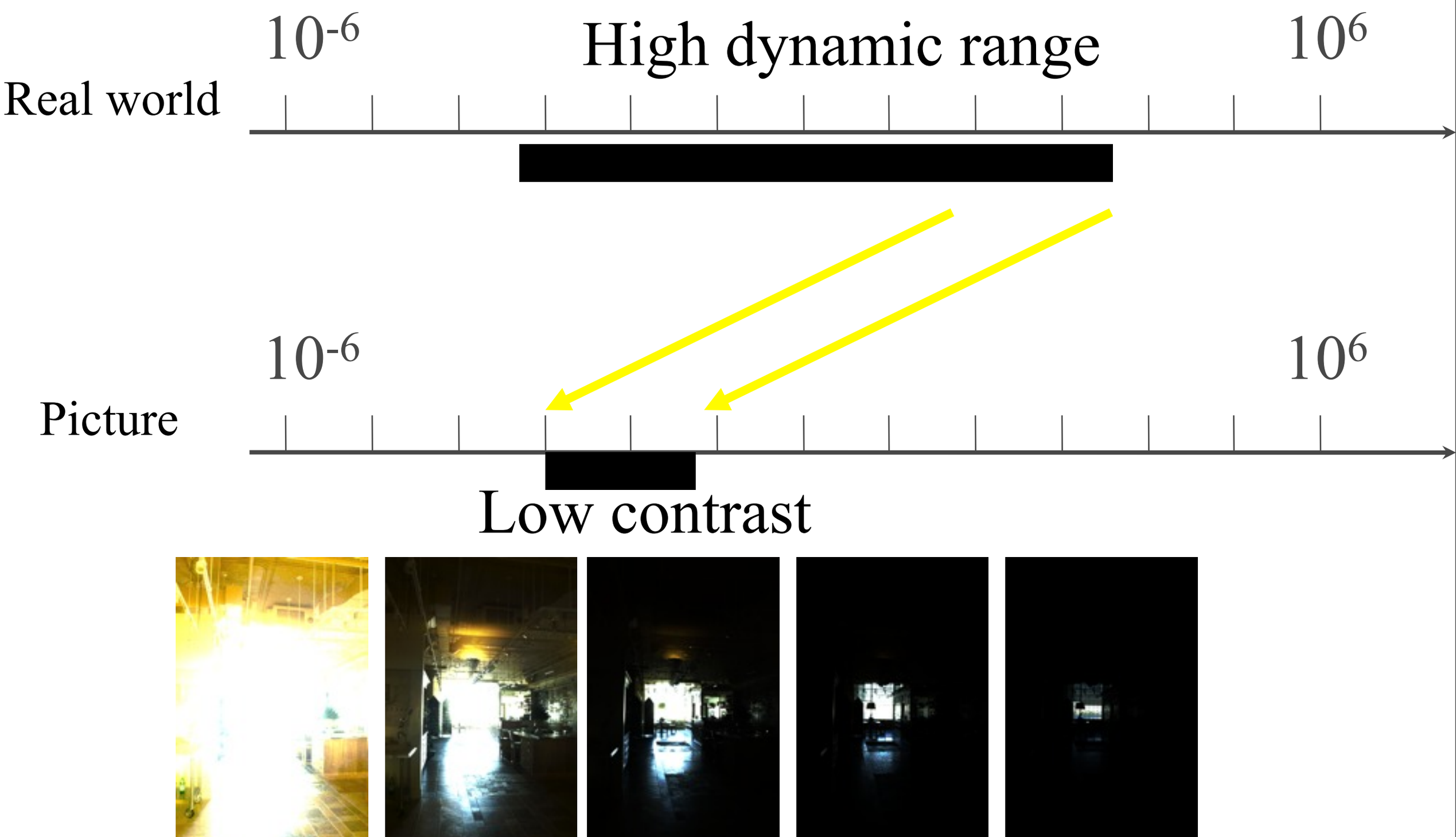
Multiple exposure photography

- **Sequentially measure all segments of the range**



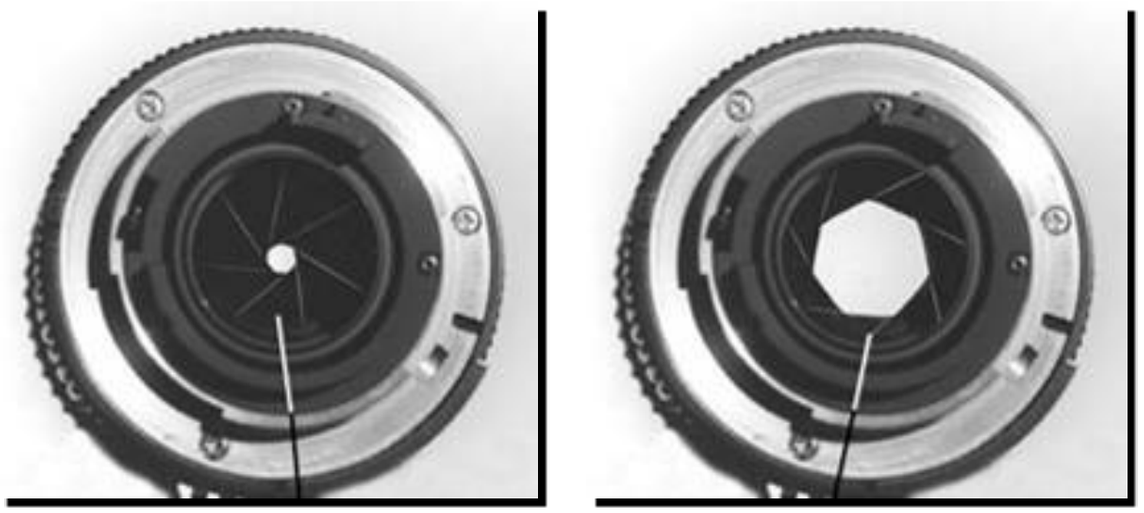
Multiple exposure photography

- **Sequentially measure all segments of the range**



Methods to Vary Exposure

- Shutter Speed
- Lens Aperture
- Film ISO (Sensitivity)
- Neutral density filters



How to Combine?

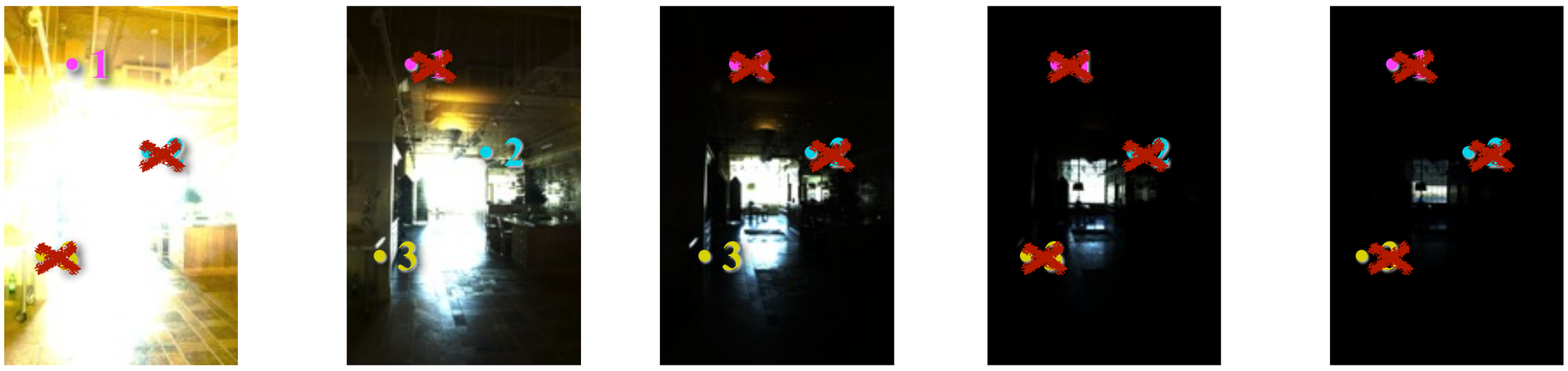
- Problem Statement: Given N images such that
  - Images are encoded linearly (the value at every pixel is directly related to the number of photons received at that location on the sensor)
  - Only the exposure changes: no motion
- Goal: Compute a single image which recovers the radiance of the scene

Two Problems with Dynamic Range

- Brightest regions (highlights) are clipped to some maximum value
- Shadow regions produce noise instead of true black

Simple Merging

- For each pixel:
  - Figure out which images are useful
  - Scale values appropriately according, ideally, to a factor relative to exposure
  - Eliminate clipped pixels, e.g. >0.99
  - Eliminate pixels that are too dark / too noisy, e.g. <0.002



A More Sophisticated Solution

Recovering High Dynamic Range Radiance Maps from Photographs

Paul E. Debevec Jitendra Malik

University of California at Berkeley[1]

ABSTRACT

We present a method of recovering high dynamic range radiance maps from photographs taken with conventional imaging equipment. In our method, multiple photographs of the scene are taken with different amounts of exposure. Our algorithm uses these differently exposed photographs to recover the response function of the imaging process, up to factor of scale, using the assumption of reciprocity. With the known response function, the algorithm can fuse the multiple photographs into a single, high dynamic range radiance map whose pixel values are proportional to the true radiance values in the scene. We demonstrate our method on images acquired with both photochemical and digital imaging processes. We discuss how this work is applicable in many areas of computer graphics involving digitized photographs, including image-based modeling, image compositing, and image processing. Lastly, we demonstrate a few applications of having high dynamic range radiance maps, such as synthesizing realistic motion blur and simulating the response of the human visual system.

CR Descriptors: I.2.10 [**Artificial Intelligence**]: Vision and Scene Understanding - *Intensity, color, photometry and threshold-*

true measurements of relative radiance in the scene. For example, if one pixel has twice the value of another, it is unlikely that it observed twice the radiance. Instead, there is usually an unknown, nonlinear mapping that determines how radiance in the scene becomes pixel values in the image.

This nonlinear mapping is hard to know beforehand because it is actually the composition of several nonlinear mappings that occur in the photographic process. In a conventional camera (see Fig. 1), the film is first exposed to light to form a latent image. The film is then developed to change this latent image into variations in transparency, or *density*, on the film. The film can then be digitized using a film scanner, which projects light through the film onto an electronic light-sensitive array, converting the image to electrical voltages. These voltages are digitized, and then manipulated before finally being written to the storage medium. If prints of the film are scanned rather than the film itself, then the printing process can also introduce nonlinear mappings.

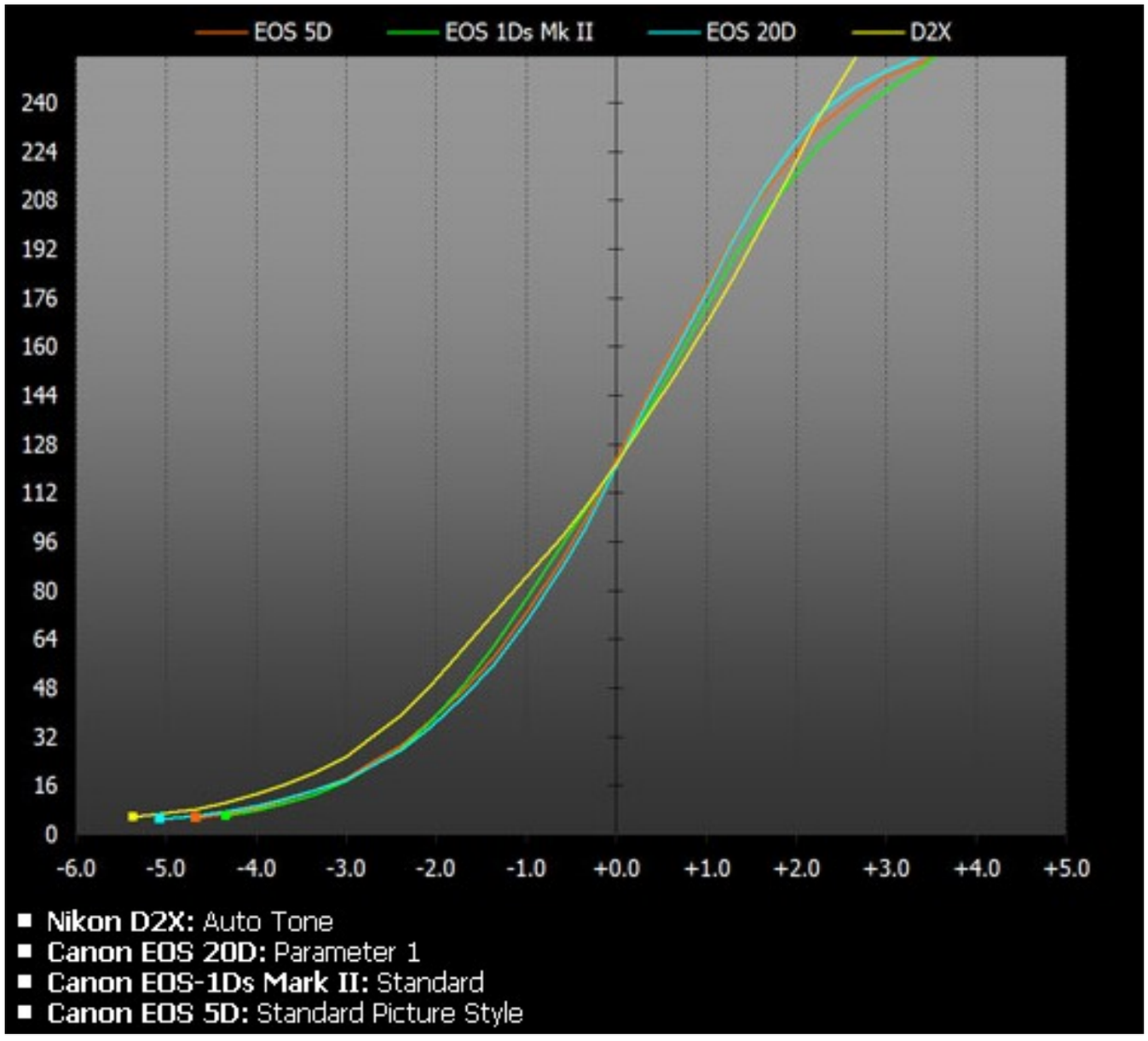
In the first stage of the process, the film response to variations in exposure X (which is $E\Delta t$, the product of the irradiance E the film receives and the exposure time Δt) is a non-linear function, called the "characteristic curve" of the film. Noteworthy in the typ-

Recovering Response Curves

- So far, we have assumed a mostly linear image
- Makes it easy to translate pixels values into a common reference range: scene irradiance L(x,y)
  - At least up to a global scale
- But often, images have received a non-linear response curve
- To make better use of dynamic range
  - Compress dark & bright tones, preserve more contrast in the middle

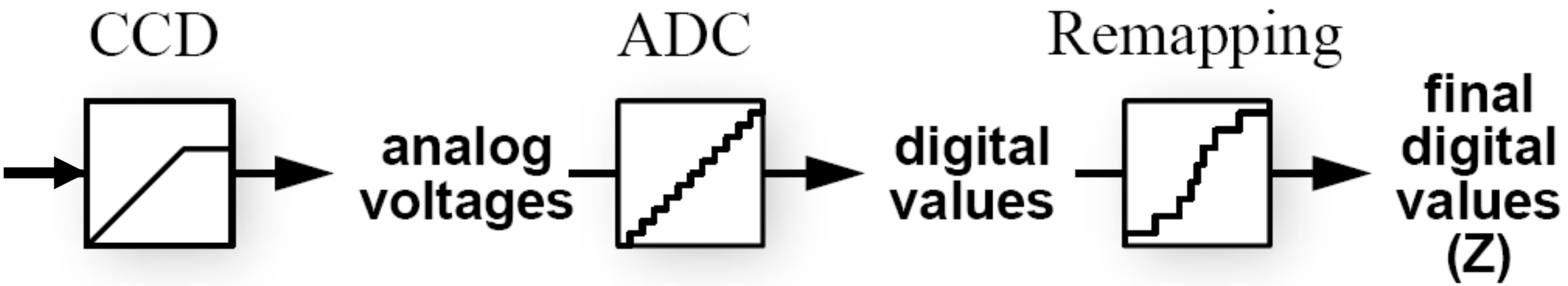
Example Response Curves

- In general, the response function is not provided by camera makers who consider it part of their proprietary product differentiation.



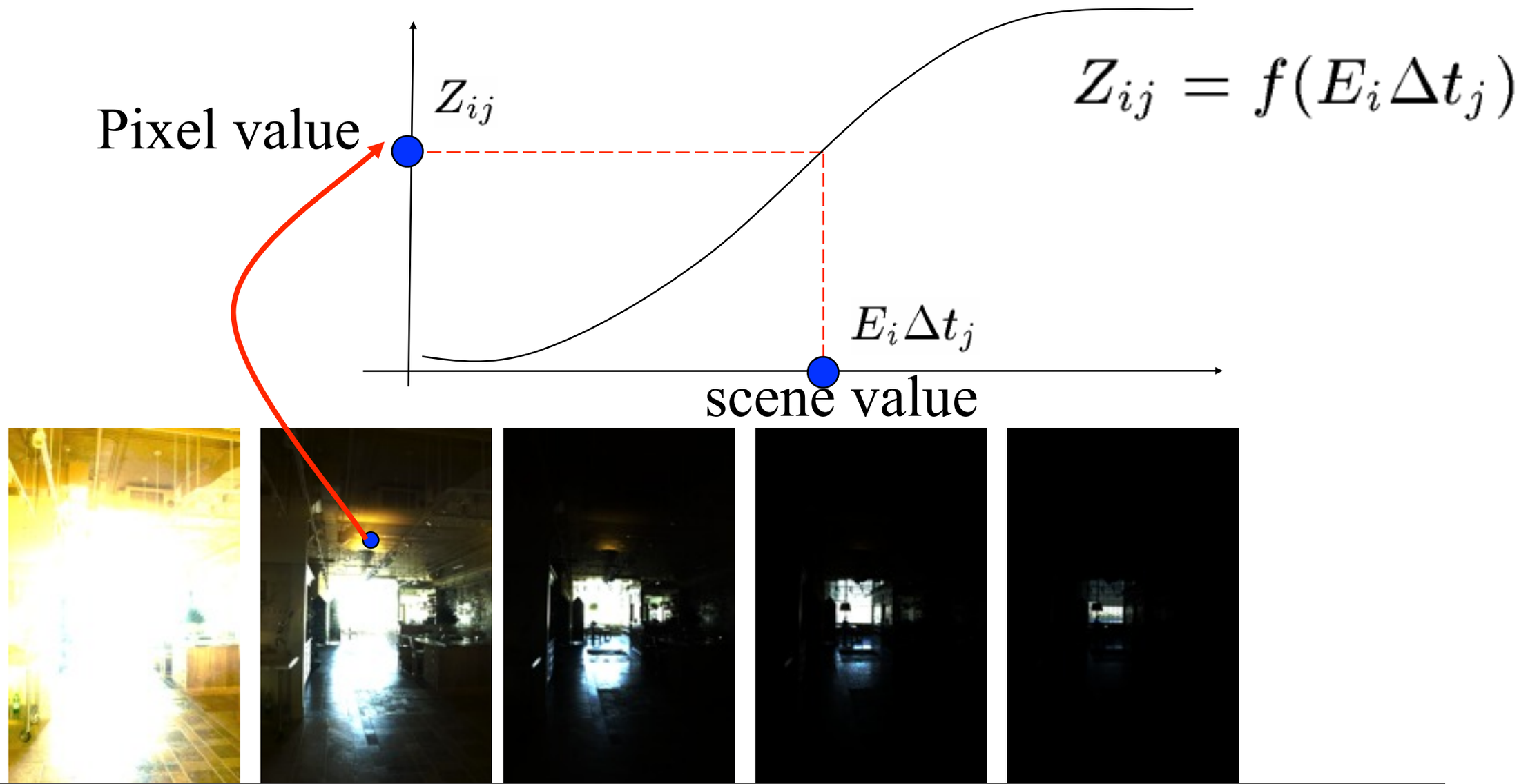
Recovering Response Curves

$$Z_{ij} = f(E_i \Delta t_j)$$



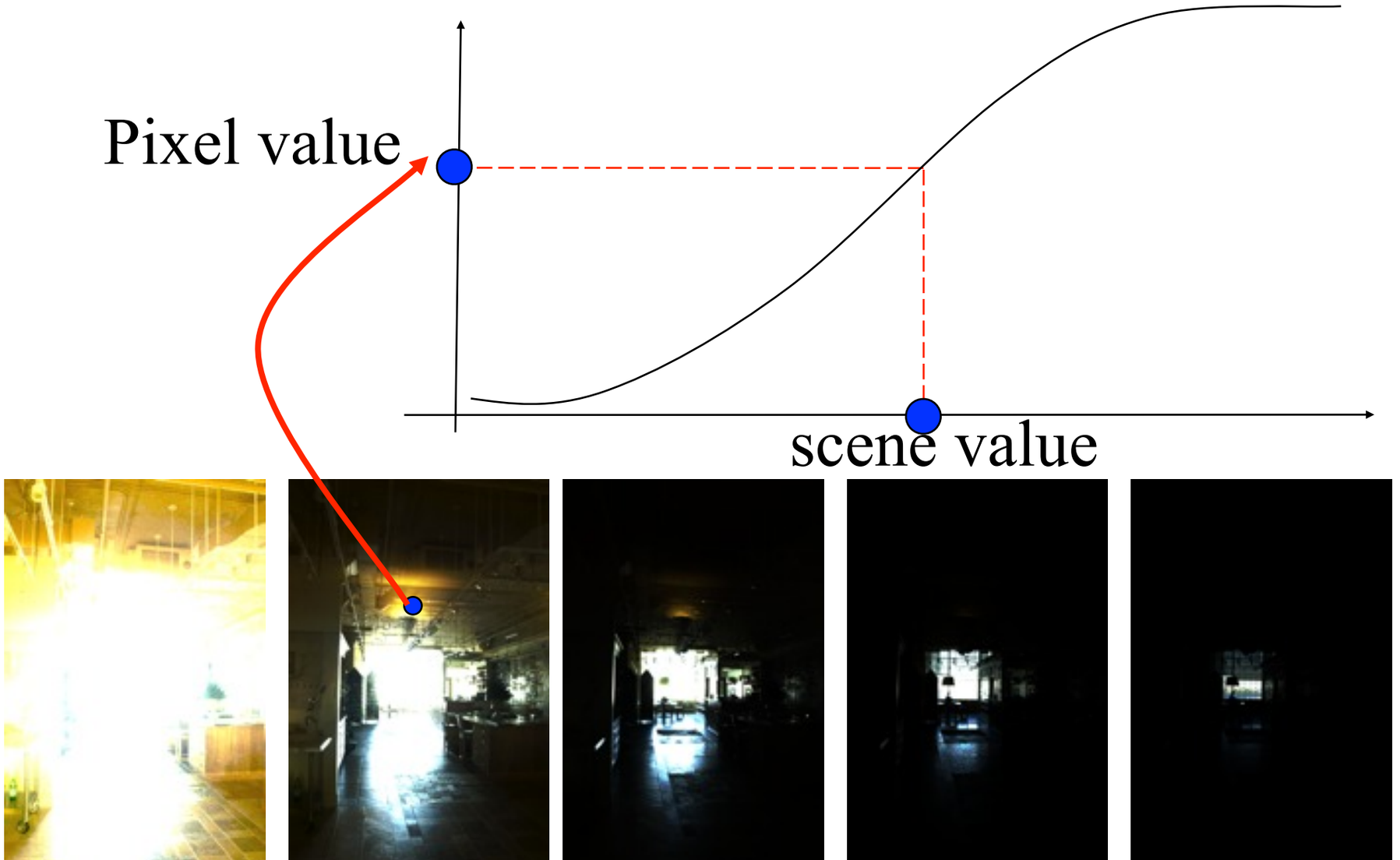
Response Curve

- Maps scene light intensity (radiance) into pixel color
- We are interested in the inverse response curve



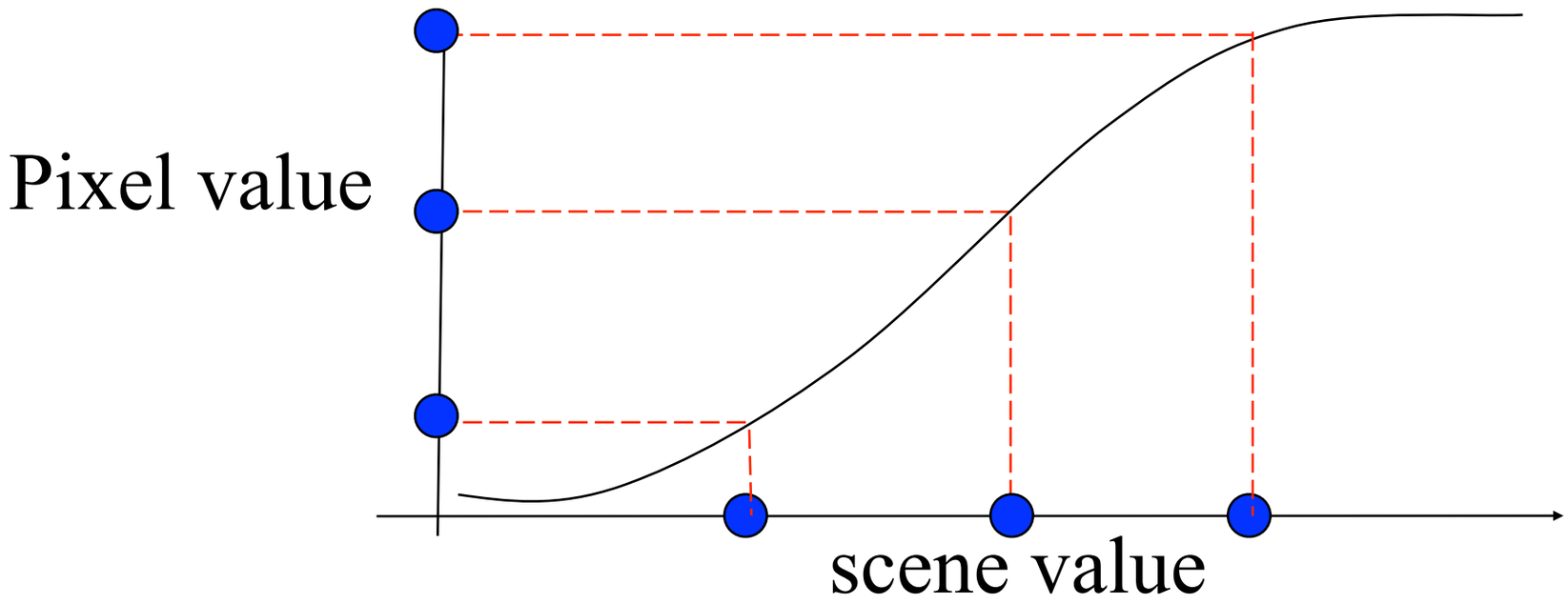
If We Knew the Response Curve...

- For each pixel:
  - For each frame:
    - If not black & not saturated, convert to absolute scene value
  - Take average if well-exposed in multiple frame



Calibrating the Response Curve

- Basic idea:
  - Measure different scene luminance levels and observe pixel value
- How do we vary the luminance reaching the sensor?



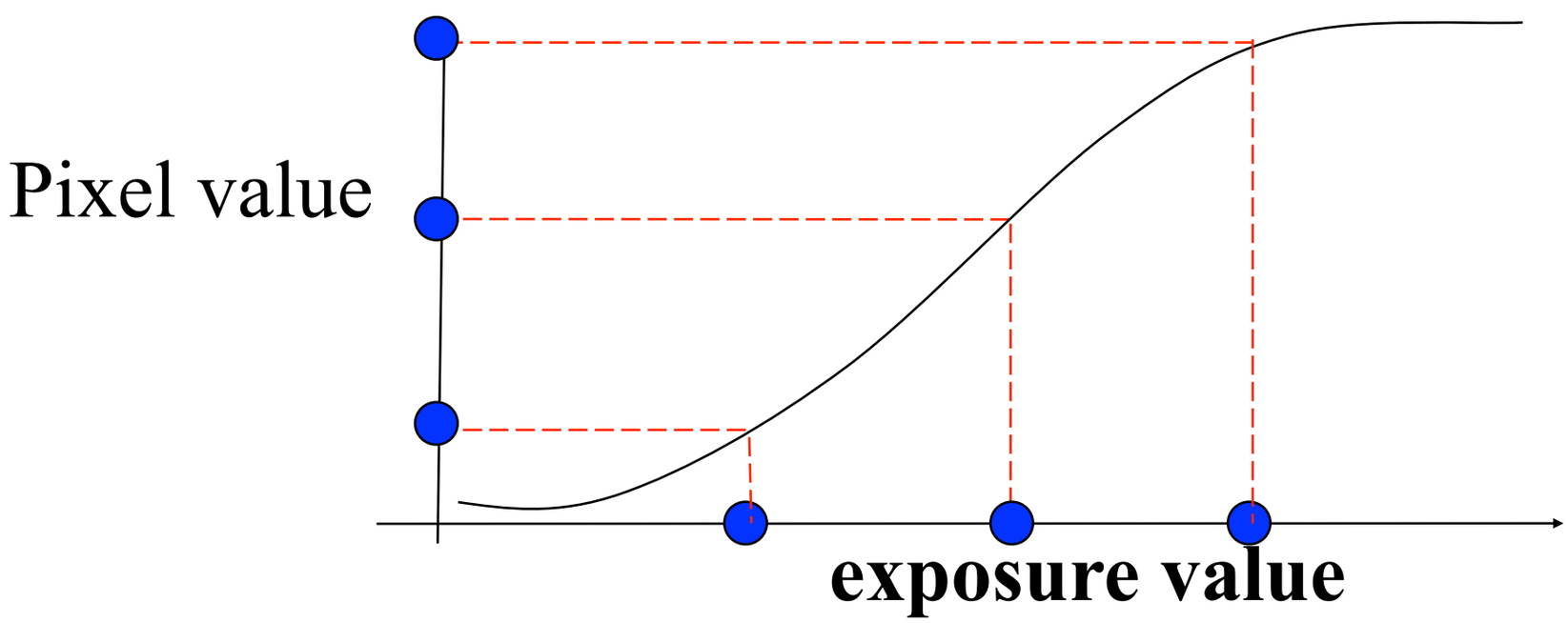
Calibrating the Response Curve

1. Vary scene luminance

    - Have a calibrated light source with an intensity knob

2. Vary exposure (shutter speed) and observe pixel value for one fixed scene luminance

    - Exposure (e.g. shutter speed) linearly scales luminance
    - But cannot vary exposure more finely than by 1/3 stop



Calibration Algorithm

Image series

Δt = 10 sec | **Δt = 1 sec** | **Δt = 1/10 sec** | **Δt = 1/100 sec** | **Δt = 1/1000 sec**

$$\text{Pixel Value } Z = f(\text{Exposure})$$

exposure: essentially # photons

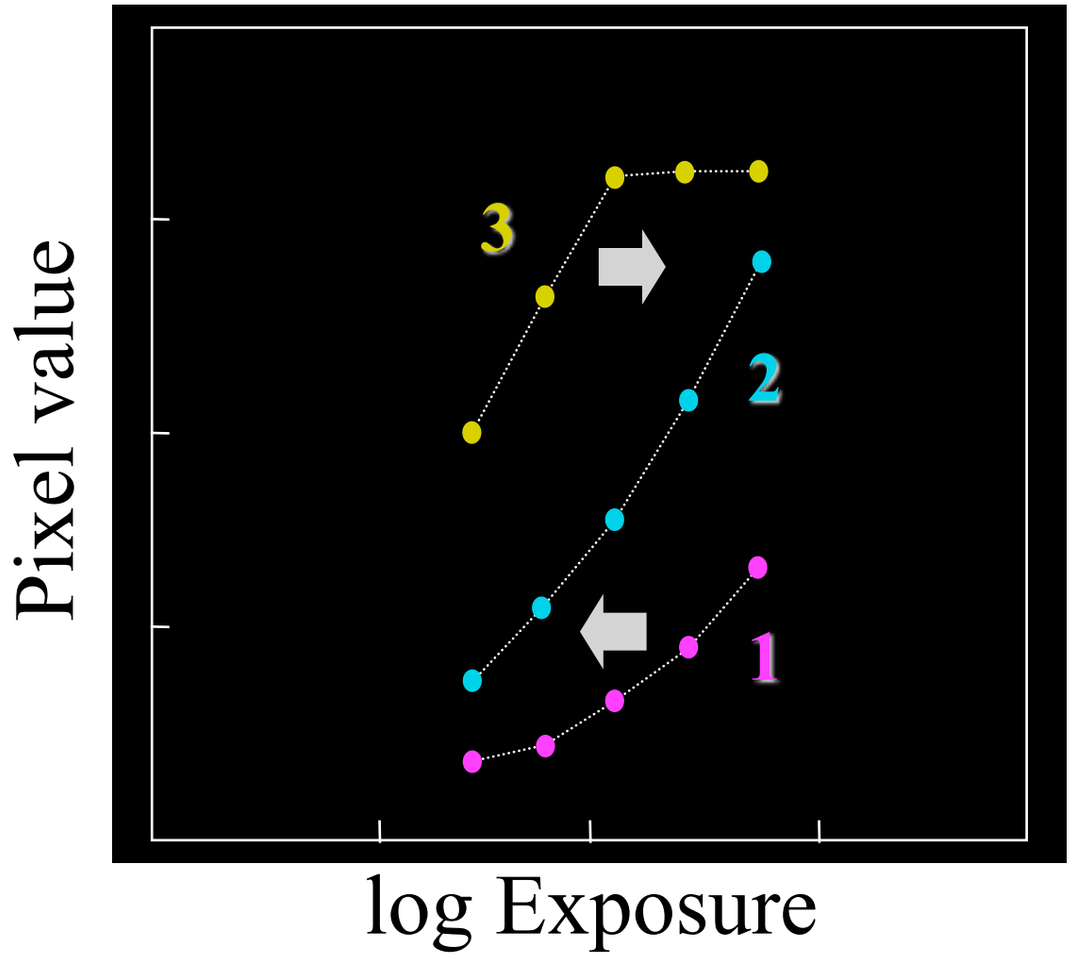
$$\text{Exposure} = \text{Radiance} \times \Delta t$$

$$\log \text{Exposure} = \log \text{Radiance} + \log \Delta t$$

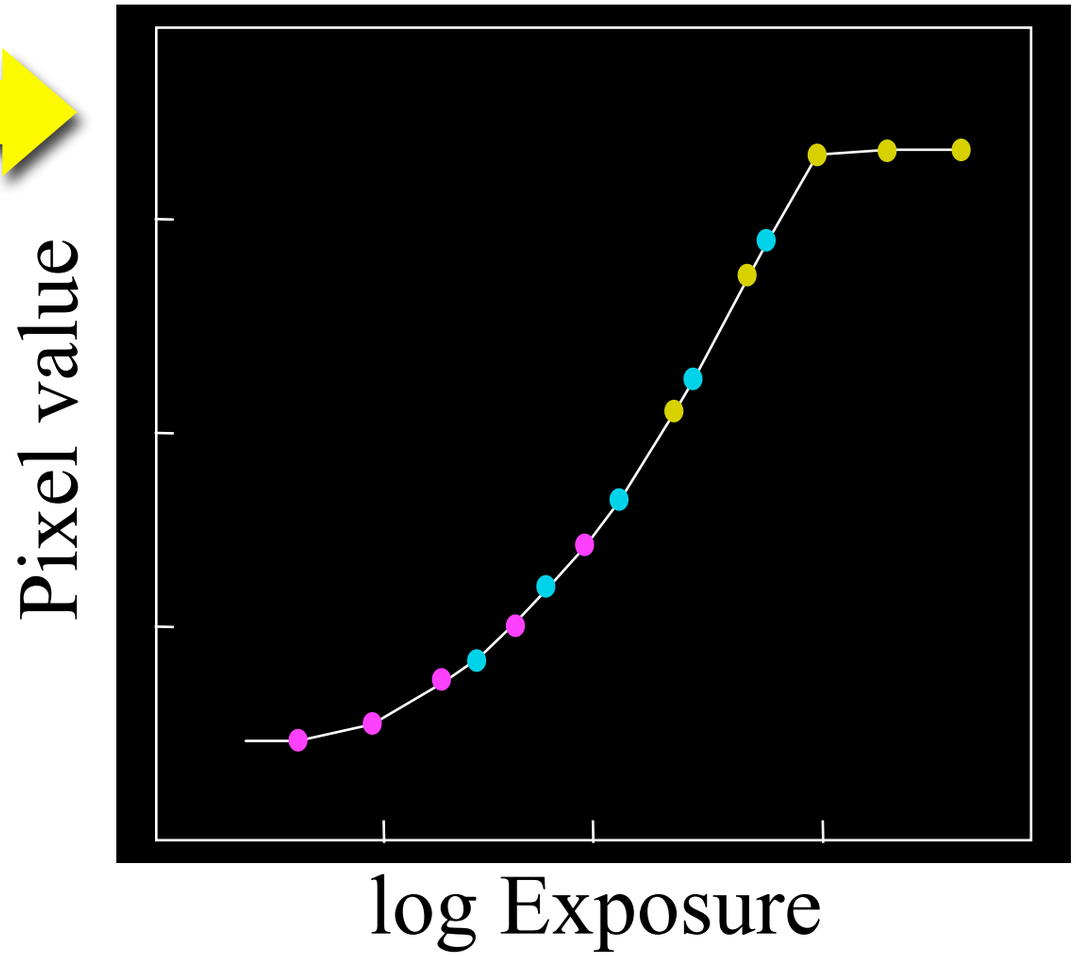
Response Curve

- Radiance is unknown, fit to find a smooth curve

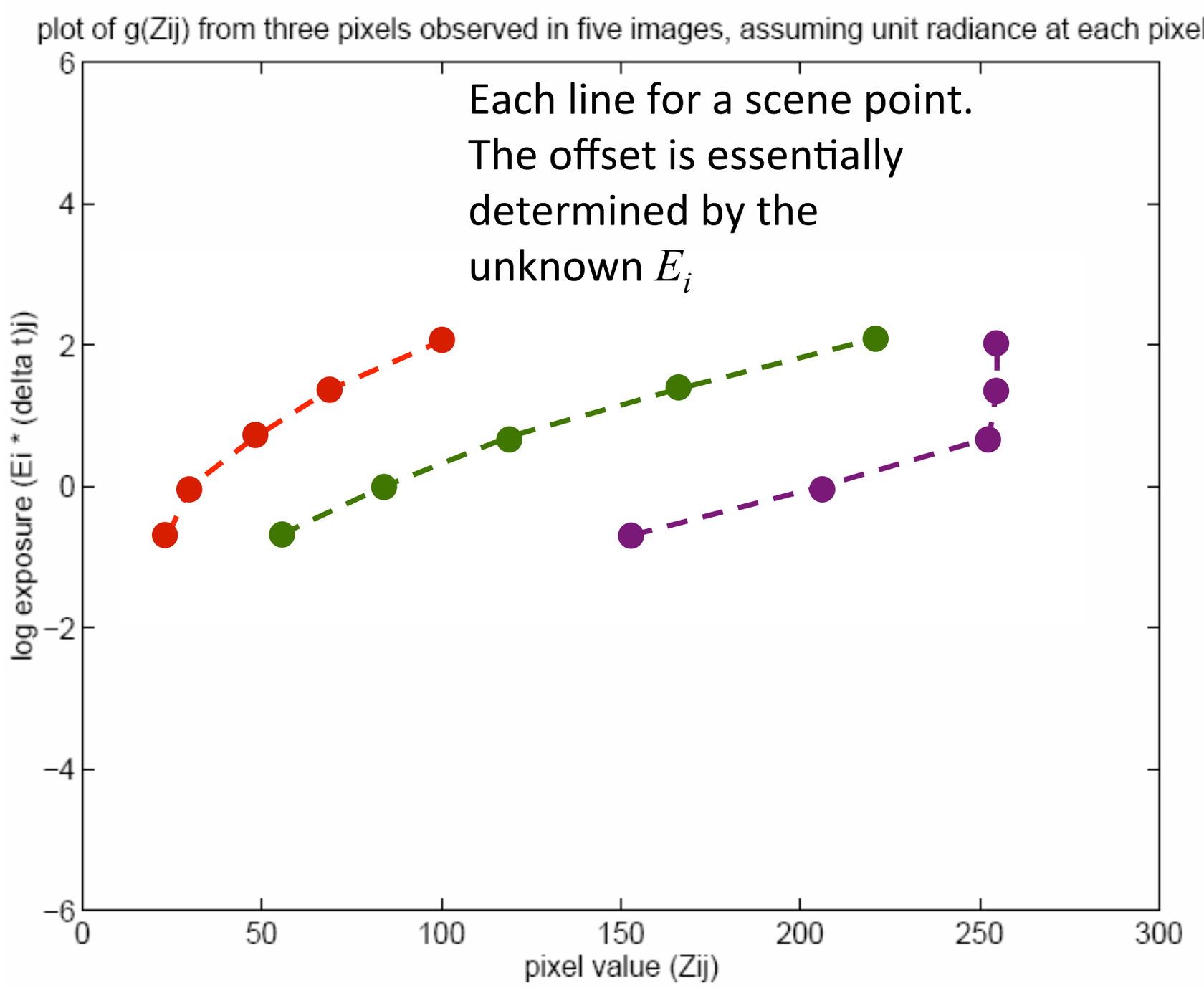
Assuming unit radiance for each pixel



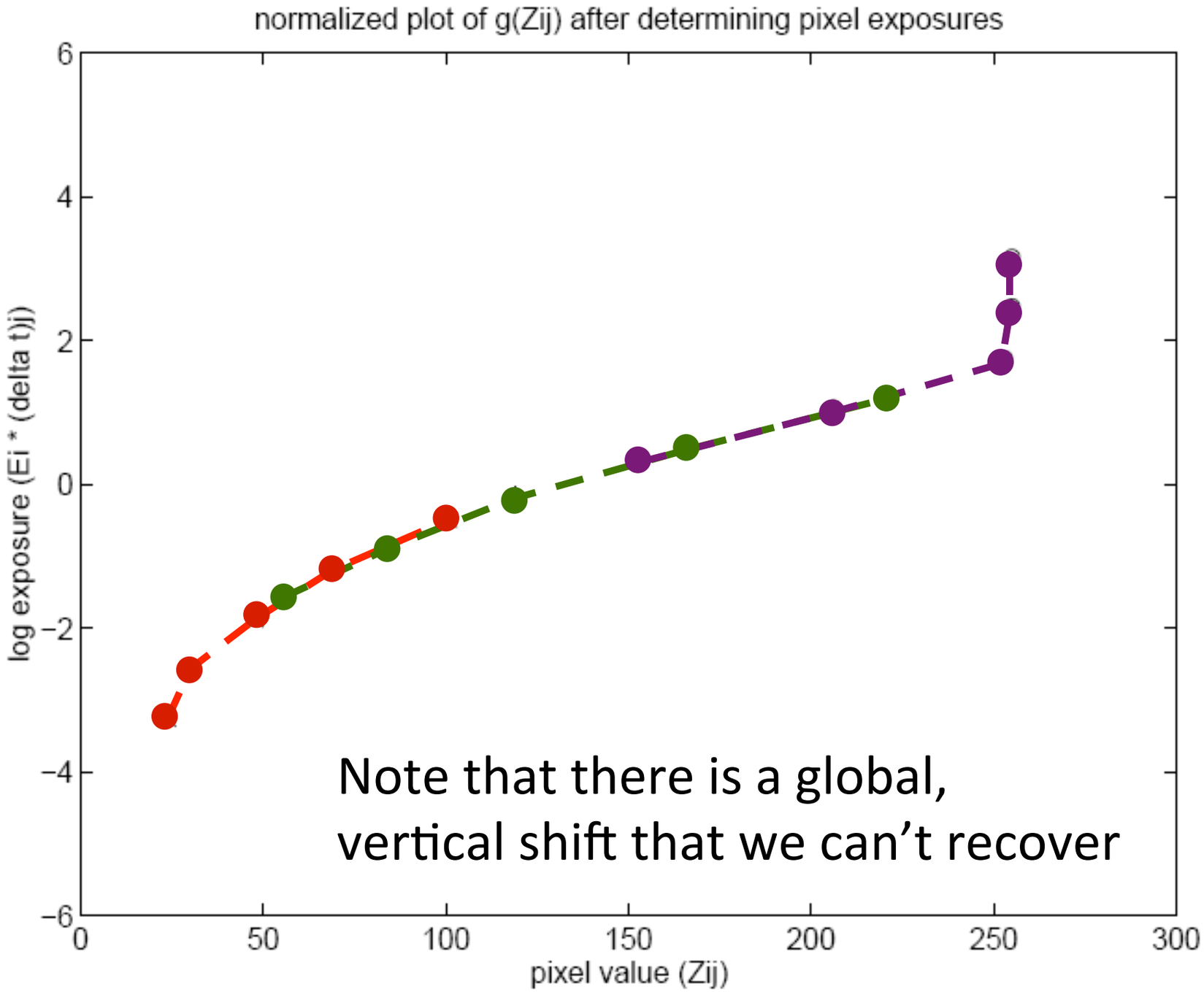
After adjusting radiances to obtain a smooth response curve



Main Idea Behind the Math

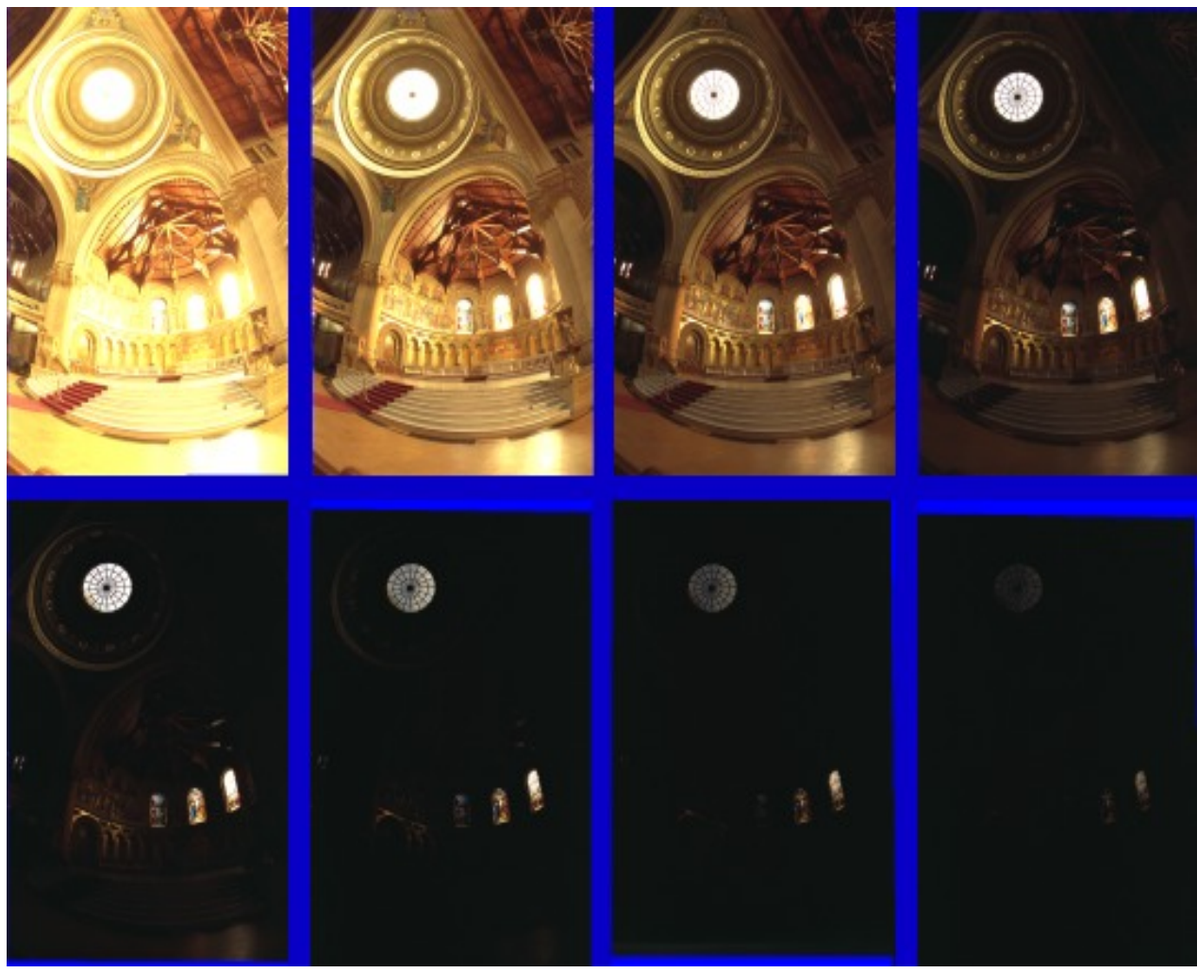


Main Idea Behind the Math

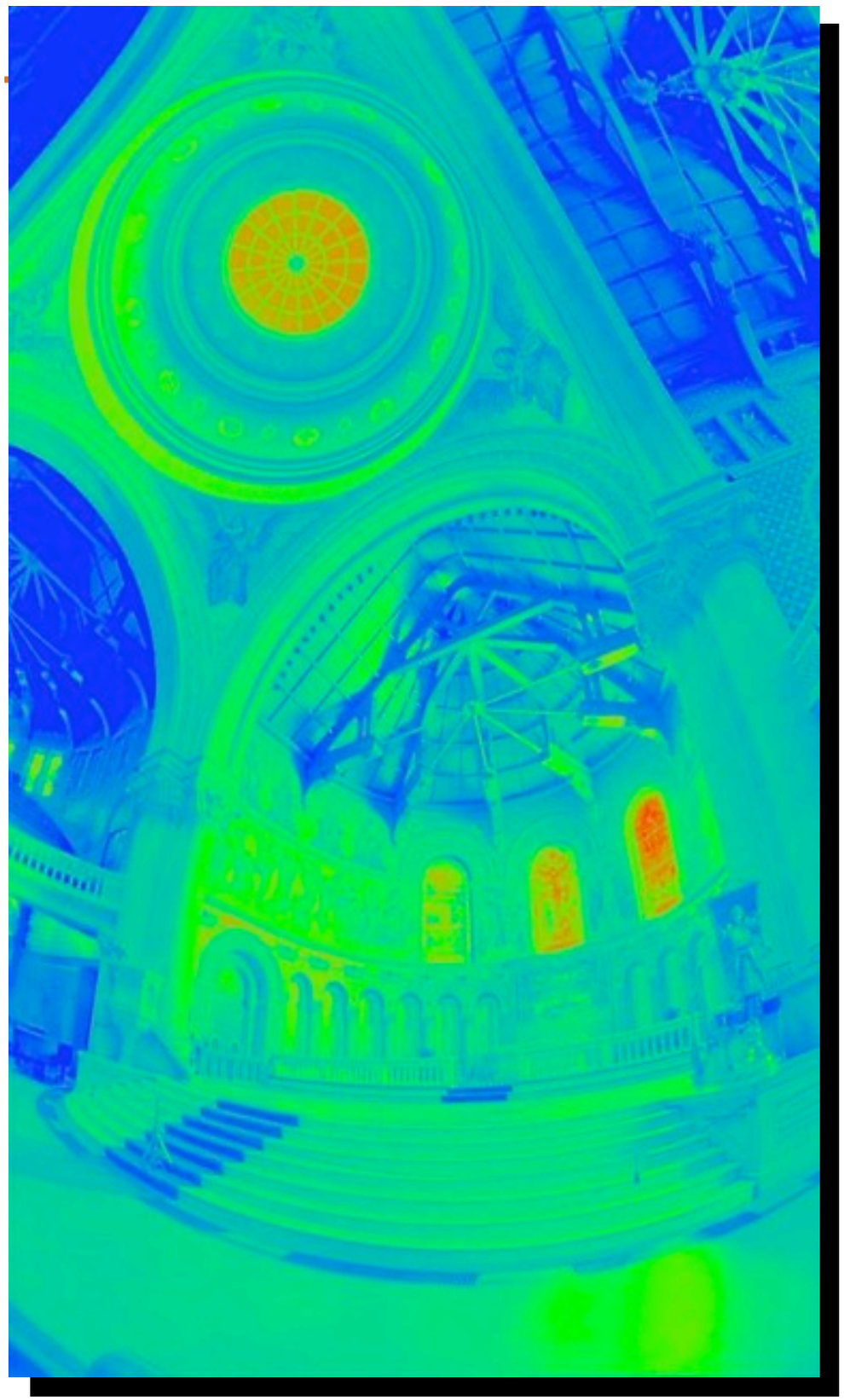


Recap

- Curve calibration
  - Take many images of static scene (1/3 stop)
  - Solve optimization problem
- HDR multiple-exposure merging
  - Take multiple exposures (e.g. every 2 stops)
  - (optional) align images
  - For each pixel, use picture(s) where properly exposed
    - Use inverse response curve and exposure time
  - Output: one image where each pixel has full dynamic range, stored e.g. in float aka radiance map



The Radiance Map



Lec15 Required Reading

- Szeliski, 10.2

Photographic Tone Reproduction for Digital Images

Erik Reinhard
University of Utah

Michael Stark
University of Utah

Peter Shirley
University of Utah

James Ferwerda
Cornell University

Abstract

A classic photographic task is the mapping of the potentially high dynamic range of real world luminances to the low dynamic range of the photographic print. This tone reproduction problem is also faced by computer graphics practitioners who map digital images to a low dynamic range print or screen. The work presented in this paper leverages the time-tested techniques of photographic practice to develop a new tone reproduction operator. In particular, we use and extend the techniques developed by Ansel Adams to deal with digital images. The resulting algorithm is simple and produces good results for a wide variety of images.

CR Categories: I.4.10 [Computing Methodologies]: Image Processing and Computer Vision—Image Representation

Keywords: Tone reproduction, dynamic range, Zone System.

Image courtesy of Cornell Program of Computer Graphics

1 Introduction

The range of light we experience in the real world is vast, spanning approximately ten orders of absolute range from star-lit scenes to sun-lit snow, and over four orders of dynamic range from shadows to highlights in a single scene. However, the range of light we can reproduce on our print and screen display devices spans at best about two orders of absolute dynamic range. This discrepancy leads to the *tone reproduction* problem: how should we map measured/simulated scene luminances to display luminances and produce a satisfactory image?

Fast Bilateral Filtering for the Display of High-Dynamic-Range Images

Frédo Durand and Julie Dorsey
Laboratory for Computer Science, Massachusetts Institute of Technology

Abstract

We present a new technique for the display of high-dynamic-range images, which reduces the contrast while preserving detail. It is based on a two-scale decomposition of the image into a base layer, encoding large-scale variations, and a detail layer. Only the base layer has its contrast reduced, thereby preserving detail. The base layer is obtained using an edge-preserving filter called the *bilateral filter*. This is a non-linear filter, where the weight of each pixel is computed using a Gaussian in the spatial domain multiplied by an influence function in the intensity domain that decreases the weight of pixels with large intensity differences. We express bilateral filtering in the framework of robust statistics and show how it relates to anisotropic diffusion. We then accelerate bilateral filtering by using a piecewise-linear approximation in the intensity domain and appropriate subsampling. This results in a speed-up of two orders of magnitude. The method is fast and requires no parameter setting.

CR Categories: I.3.3 [Computer Graphics]: Picture/image generation—Display algorithms; I.4.1 [Image Processing and Computer Vision]: Enhancement—Digitization and image capture

Keywords: image processing, tone mapping, contrast reduction, edge-preserving filtering,weird maths

Figure 1: High-dynamic-range photography. No single global exposure can preserve both the colors of the sky and the details of the landscape, as shown on the rightmost images. In contrast, our spatially-varying display operator (large image) can bring out all details of the scene. Total clock time for this 700x480 image is 1.4 seconds on a 700Mhz PentiumIII. Radiance map courtesy of Paul Debevec, USC. [Debevec and Malik 1997]

1 Introduction