

ACRONYMS — THIS CHAPTER

ACRONYMS

ADC	analog-to-digital converter
AE	autoexposure
AWB	automatic white balance
BSSRDF	bidirectional scattering-surface reflectance distribution function
CC	CameraCalibration matrix (in the DNG pipeline); Creative Commons in a licensing context
CCD	charge-coupled device
CIE	Commission Internationale de l'Éclairage (International Commission on Illumination)
CIELAB	CIE Lab color space (lightness L with a, b color-opponent axes)
CMOS	complementary metal-oxide-semiconductor
CT	computed tomography
D50	CIE standard illuminant D50 (~5000 K daylight)
D65	CIE standard illuminant D65 (~6500 K daylight)
DFT	discrete Fourier transform
DNG	Digital Negative (raw file format)
ETTR	expose to the right
EV	exposure value
EXIF	Exchangeable Image File Format
FFCC	Fast Fourier Color Constancy
HDR	high dynamic range
HSL	hue, saturation, lightness
ICC	International Color Consortium
IDE	integrated development environment
ISP	image signal processor
LMU	Ludwig Maximilian University of Munich
MATLAB	Matrix Laboratory (software)
ND	neutral density (filter)
NeRF	neural radiance field
PSNR	peak signal-to-noise ratio
RANSAC	RANdom SAmple Consensus
SDR	standard dynamic range
SIMD	single instruction, multiple data
SNR	signal-to-noise ratio
SSIM	structural similarity (index)
UCS	uniform color space
YUV	luma (Y) with blue- and red-difference chroma (U, V)

LECTURE NOTES 1 — IMAGES, C++

Course notes cut from *Computational Photography, an AI-powered Slopendum* for this lecture. Each section links back to its home in the book; sections marked **optional** go beyond what the lecture needs.

► Contents • 61 sections on this page, from 38 chapters of the book

What programming language for computational photography?

Light and optics

The color of objects: illumination times reflectance

in the book [↗](#)

Most ordinary photographed objects emit negligible visible light of their own; they only *reflect* whatever light falls on them. So the color that reaches your eye is the result of two spectra multiplied together, wavelength by wavelength:

$$C(\lambda) = S(\lambda) \rho(\lambda). \tag{1}$$

Here $S(\lambda)$ is the illuminant (the spectrum of the light source actually present) and $\rho(\lambda)$ is the surface's spectral reflectance, the fraction of light it sends back at each wavelength. Reflectance is the object's intrinsic, fixed signature: the apple's $\rho(\lambda)$ is high in the reds and low in the greens and blues, whatever the lighting. Illumination is whatever happens to be shining. Their per-wavelength product is what leaves the surface and reaches the eye ([Figure 2.1.28](#)). A red apple under white light reflects mostly red; the same apple under pure green light has almost nothing to reflect and looks nearly black, because there is no red illumination for its reflectance to act on.

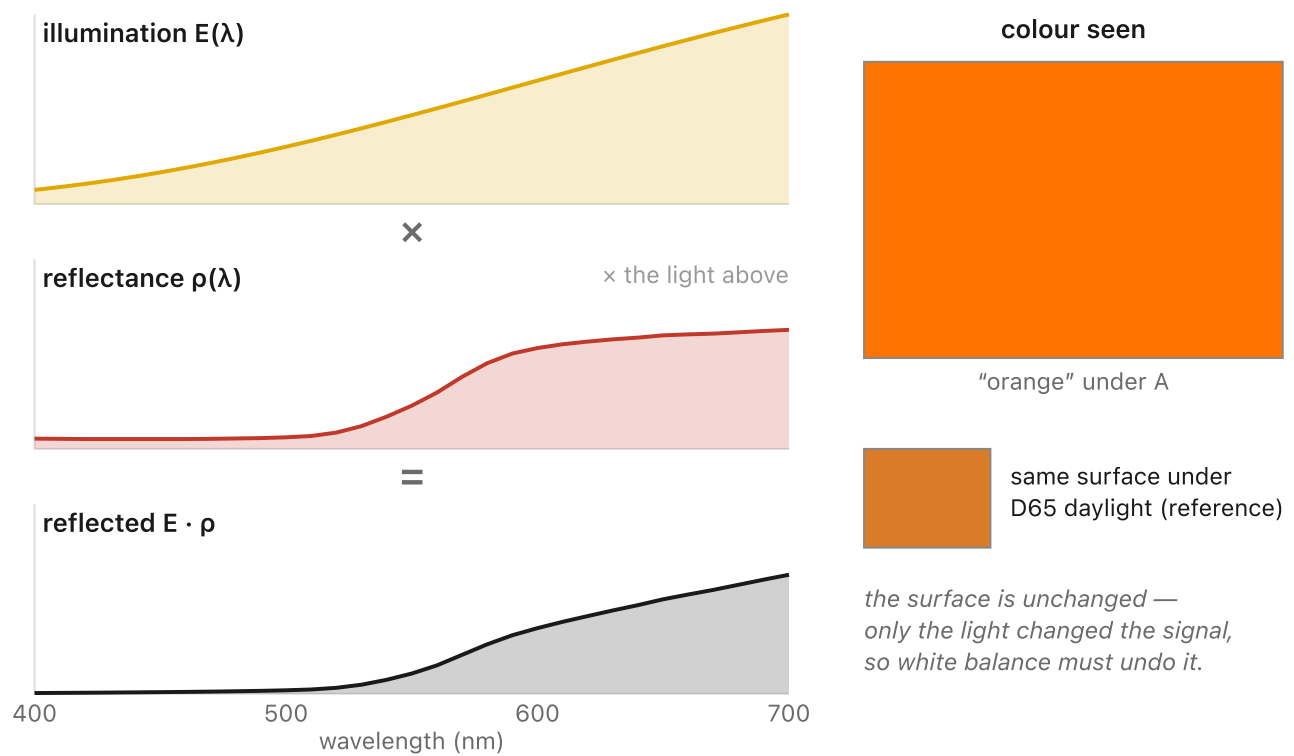


Figure 2.1.28 (reproduced). Color is illumination $\times$ reflectance. A surface's fixed reflectance spectrum $\rho(\lambda)$ (here an apple's, high in the reds) is multiplied per wavelength by the illuminant spectrum $S(\lambda)$ actually present. The product $C(\lambda) = S(\lambda) \cdot \rho(\lambda)$ is the light that reaches the eye. Change the illuminant and the product changes even though the surface has not: the same reflectance under white versus reddish versus greenish light yields three different stimuli. One swatch shows the surface's unadapted appearance under the chosen illuminant; a smaller reference swatch shows it under D65 for comparison.

This multiplicative entanglement is consequential, and it is the first instance of a principle that runs through the whole book.

💡 BIG LESSON (L2.6) — THE MULTIPLICATIVE WORLD

Much of imaging is *multiplicative*, not additive. The light reaching the eye is illumination **times** reflectance; contrast is a **ratio** of intensities; a shadow darkens a region by a **factor**, not by subtracting a constant. Because the structure is multiplicative, the natural tools are **multiply, divide, and log**, not just add. This is why, again and again, we will work with ratios and with logarithmic encodings rather than raw linear values. It is also why **white balance** is hard: to recover an object's true color we must *divide out* the unknown illuminant $S(\lambda)$ from the product $C(\lambda)$, undoing a multiplication we never observed directly. (The additive counterpoint, light from different sources *adding*, and what it means for choosing an encoding, comes in Color technology.)

Why this multiplicative structure matters so much is a question the rest of the book keeps answering. Our visual system does not care about the illumination for its own sake. It cares about *objects*, about the reflectance $\rho(\lambda)$ that tells it what a thing is and whether it is food, a face, or a threat. So the brain works hard to discount the illuminant, factoring the product $C(\lambda) = S(\lambda) \rho(\lambda)$ back apart to recover the reflectance and hold an object's apparent color and lightness steady as the light changes. That discounting is the perceptual machinery of lightness and color constancy, studied in *Human vision and color*, and a camera has to imitate it deliberately, which is exactly what white balance does. Much of imaging is this running effort to see past the illumination to the reflectance underneath.

Image measurements as integrals over the plenoptic function

The pixel integral

OPTIONAL
in the book ↗

A pixel does not sample a single ray; it integrates over four groups of variables, each weighted by how the pixel responds there ([Figure 2.4.2](#)):

1. the aperture: all the ray directions (θ, ϕ) the lens admits toward this pixel, a solid angle set by the f-number;
2. the pixel's area (x, y) on the sensor, a finite patch, not a mathematical point;
3. **wavelength** λ , weighted by the sensor's and color filter's spectral response; and
4. the exposure time t , the interval the shutter is open.

The spectral axis, λ , is the one this book develops furthest elsewhere: collapsing a whole spectrum onto a few numbers weighted by $r(\lambda)$ is exactly where *color* is born, and how a sensor is built to make that measurement in three or more channels, together with what the resulting numbers mean perceptually, is the subject of [Part 3](#),

[Visual perception and color](#) and [Part 4, Color spaces and color technology](#), and in particular the chapter on [Chapter 4.6, Sensing color: multiplexing strategies](#). Here we treat λ simply as one more axis the pixel integrates over.

Schematically, the value recorded at a pixel is

$$\text{pixel value} = \iiint\!\!\!\int \underbrace{P(x, y, \theta, \phi, \lambda, t)}_{\text{plenoptic radiance}} \underbrace{r(\lambda) k(x, y, \theta, \phi)}_{\text{pixel's response}} d(\text{aperture}) d(\text{area}) d\lambda dt, \quad (2)$$

the scene radiance integrated over aperture $\times$ pixel area $\times$ wavelength $\times$ time, weighted by the pixel's spectral response $r(\lambda)$ and the geometric weight k that the aperture and footprint impose. Measurement is integrating the plenoptic function against the pixel's sampling kernel, and the *support* of that kernel along each axis is exactly a photographic knob. Because the underlying capture counts photons, this integral is linear in scene radiance, a fact we will need both for high-dynamic-range merging and for the noise analysis in [Chapter 2.8, Noise, signal-to-noise ratio and dynamic range](#).

💡 **BIG LESSON (L2.26) — A PIXEL VALUE IS AN INTEGRAL**

No pixel measures "the" value at a point in space, time, or color: it reports an **integral** over everything the photosite cannot tell apart. Four axes appear explicitly above: the **spatial footprint** (the pixel's area, plus the optical blur smeared across it), the **exposure time** (a window, not an instant), the **spectrum** (wavelength, weighted by the channel's spectral sensitivity), and the **solid angle of rays** the aperture admits; a fifth, **polarization**, is usually lumped in too. Naming all the axes at once is what makes **motion blur**, **defocus**, **color**, **vignetting**, and **aliasing the same phenomenon** (a finite integral standing in for an idealized point sample), and the *width* of the integral along each axis is a knob the rest of the book spends its time controlling.

A pixel integrates over the aperture and over its own area

$$\text{pixel} = \iiint \text{radiance over aperture} \times \text{pixel area} \times \text{time} \times \text{wavelength}$$

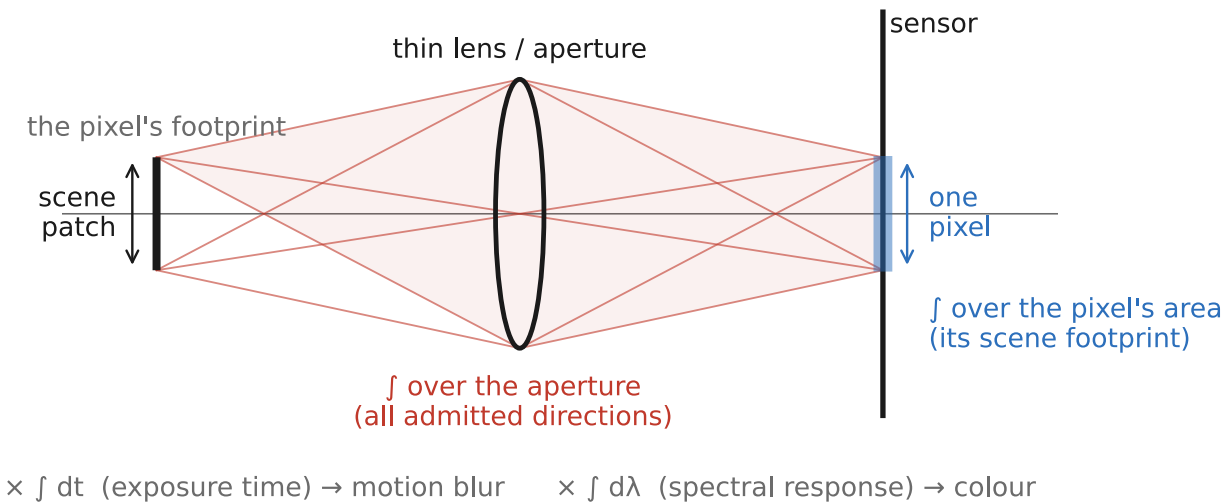


Figure 2.4.2 (reproduced). A pixel value as a four-dimensional integral, a measured slice of the plenoptic function. The thin lens (drawn as an ellipse) admits a cone of rays; the value recorded at one finite-area pixel is the scene radiance integrated over (1) the aperture (which rays/directions the lens admits), (2) the pixel's area on the sensor, (3) wavelength (the spectral response), and (4) the exposure time. Aperture $\rightarrow$ depth of field; pixel area $\rightarrow$ sampling/antialiasing; time $\rightarrow$ motion blur; wavelength $\rightarrow$ color.

The four domains are the four knobs. This matters because each integration domain is governed by a control the photographer actually sets: the integral is the camera's exposure settings, written as mathematics:

- the aperture integral is set by the f-number $N = f/D$: stopping down shrinks the cone of directions each pixel sums over (less light, more depth of field); opening up widens it;
- the exposure-time integral is set by the shutter speed: a longer interval integrates over more time (more light, but more motion blur);
- the pixel-area integral is fixed by the sensor's pixel pitch (the physical size of a photosite) so larger photosites collect more photons at equal sensor irradiance and exposure, while larger sensor area does not by itself imply larger pixels;
- the wavelength integral is shaped by the spectral response (the sensor's sensitivity curve times the color filter over each photosite) which collapses a whole spectrum into a single number per channel.

The one familiar exposure control *missing* from this list is ISO, and the omission is instructive: ISO is not an integration limit. It scales the *result* of the integral after the light is collected (analog or digital gain), rather than changing how much light is gathered, which is exactly why pushing ISO amplifies noise along with signal, as the noise section will show.

The deeper reason to write capture this way is that each of the four integration domains earns its own topic later, and much of photography is the craft of choosing how to shape these four integrals. Every later sensing topic is just *what happens when you change the support of one axis*:

- integrating over the aperture is what produces depth of field (the lens chapter): a wide aperture integrates over a fat cone of directions, so an out-of-focus point spreads its energy across many pixels: the circle of confusion. The light-versus-blur trade lives entirely on this one axis;
- integrating over the pixel's area is sampling and antialiasing (Basic Image Processing): a finite footprint makes each sample a *box-filtered average* rather than a point sample, pre-blurring the image before it is laid on the grid: a wider footprint suppresses aliasing at the cost of resolution;
- integrating over time is motion blur (Motion/Video): a long exposure sums a moving scene over the interval, the temporal sibling of the aperture, trading light against sharpness;
- integrating over wavelength is color sensing: the spectral response collapses a whole spectrum into one number per channel, exactly as a cone does ([Chapter 3.2, Perceptual color and trichromatic vision](#)). Because a color-filter array gives each photosite a *different* $r(\lambda)$, color is spatially multiplexed across the Bayer mosaic and must be reconstructed by demosaicking.

The rest of the sensing story reduces to one question, *which kernel, how wide, and what does it lose*: antialiasing widens the spatial kernel, depth of field is the aperture kernel, motion blur is the time kernel, and demosaicking inverts the wavelength multiplexing. Each is also a source of noise: a narrower kernel along any axis (a smaller footprint, a shorter exposure, a stopped-down aperture) integrates *fewer* photons, and fewer photons mean a noisier estimate, which is exactly where [Chapter 2.8, Noise, signal-to-noise ratio and dynamic range](#) picks up.

All four axes are then summed into one number: the conventional sensor couples them, discarding how the light was distributed across directions, area, color and time alike. The *ideal* sensor would do the opposite: decouple the axes, measuring each one separately so we could decide *after* the shot how to combine them. Chasing that decoupling is a whole branch of computational photography, which the end of the chapter frames.

Framed this way, the measurement integral is a general language for imaging. Every imaging system, from a pinhole to a light-field camera, is described by *which* weighting it applies to the plenoptic function before integrating: the choice of weights over direction, area, wavelength, and time is the design of the system. And because integration is linear, every such system is a linear operator on the plenoptic function, the viewpoint that [Chapter 6.7, Image formation as a linear system](#) develops in full.

For the sake of simplicity, we routinely speak of imaging in terms of *infinitesimal* quantities: a pinhole that admits a single ray, an exposure that freezes an instant, a pixel that samples a point, a single wavelength of light. These idealizations keep the geometry clean and the diagrams legible. But no real measurement is infinitesimal. A sensor reports a number only by *integrating* over a finite aperture, a finite area, a band of wavelengths, and a stretch of time, because every axis of the plenoptic function must be summed over to gather enough photons to register at all. The idealized ray and the frozen instant are the limit the mathematics reaches for; the finite integral is what the camera actually delivers, and the gap between the two is where depth of field, motion blur, color mixing, and noise all live.

Off-axis ($\cos^4$) falloff. One prediction falls out of this irradiance integral. Rays reaching the corners of the frame arrive obliquely, so the same scene radiance delivers *less* irradiance there: the aperture-and-area geometry of the pixel integral works out to a $\cos^4 \theta$ dimming toward the edges, called natural vignetting ([Figure 2.4.3](#)). It is distinct from *optical* vignetting (lens elements physically blocking edge rays, covered in Optics) in that natural vignetting is a clean consequence of the pixel-integral geometry alone.

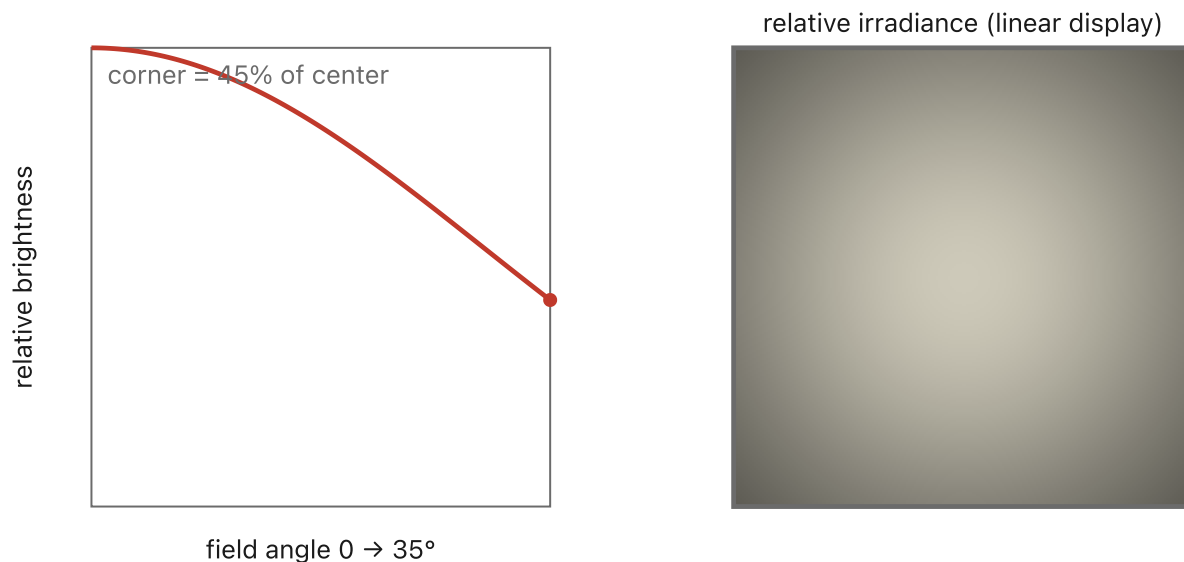


Figure 2.4.3 (reproduced). Natural vignetting: the $\cos^4 \theta$ falloff. Relative pixel irradiance is plotted against the off-axis angle θ , beside a preview of a frame darkening toward its corners. Even with uniform scene radiance, corner pixels receive less light, falling as $\cos^4 \theta$, the combined effect of oblique incidence, the foreshortened aperture, and the longer path to the corner. This is the geometry of the sensor irradiance integral, separate from optical vignetting. Widen the field of view to exaggerate the effect.

To interpret the integral as expected photoelectrons, let $r(\lambda)$ include quantum efficiency and the conversion from radiant energy to photons, $\lambda/(hc)$. If r denotes quantum efficiency alone, that factor must appear explicitly. Optical transmission and the projected-area factor belong in k . Fixing these weights makes the expected measurement linear; the observed count is random, and saturation or later encoding breaks this simple linear relation.

Sensors: photosites, CCD vs CMOS

Photon to number: the photosite

OPTIONAL
in the book [↗](#)

A digital sensor is a two-dimensional array of photosites, one per pixel. Each photosite is a small potential well that counts photons: arriving light frees electrons, the well accumulates that charge, and at the end of the exposure the charge is converted to a voltage and then, by an analog-to-digital converter (ADC), to a number. Because the well literally counts arrivals, the number it reports is, to good approximation, linear in the amount of light that fell on it. That linearity is not a minor technical detail: it is what lets us treat a raw image as a calibrated radiometric measurement, and it is the foundation that makes HDR merging, deblurring, and depth estimation meaningful later in the book.

💡 **BIG LESSON (L2.35) — SENSORS CONVERT PHOTONS TO ELECTRONS LINEARLY**

A photosite is at heart a **photon counter**: each absorbed photon frees a charge carrier, an **electron** or equivalently a *hole* (a missing electron), and the well accumulates them in direct **proportion** to the light that arrives. So the raw signal is **linear in scene radiance**, before any gamma or tone curve. This linearity is not a detail; it is what makes raw arithmetic *trustworthy*: HDR merging **adds** exposures, white balance is a per-channel **multiply**, flat-field correction is a **divide**, and all three are valid only because the numbers are proportional to light. Display encoding (gamma) deliberately hides this linearity for the eye, so computation should undo it and work in **linear light**.

Two revolutions are often conflated. The first was electronic imaging: turning light into an analog electrical signal (the era of the CCD and analog video). The second, the one computational photography lives on, was digital: putting an ADC in the path so the signal becomes *numbers we can compute on*. A sensor that captures light electronically but never digitizes it is a television camera, not a computer's eye.

Above each photosite sits a microlens that funnels light onto the light-sensitive area, necessary because wiring occupies part of each pixel, so the fill factor is less than 100%. Under the microlens sits a color filter: a bare photosite is colorblind (it only counts photons), so a mosaic of color filters, most commonly the Bayer pattern, is laid over the grid and full color is reconstructed later by demosaicking. That is only the briefest mention here; how a colorblind sensor is made to see color, and the several ways to multiplex the measurement, is the subject of its own chapter in the next part, [Chapter 4.6, Sensing color: multiplexing strategies](#) (§4.2). Finally, the map from scene radiance to the *rendered* pixel value is the camera response curve: raw is roughly linear, while a JPEG has a non-linear curve (gamma plus a tone curve) baked in: recovering that curve is exactly the problem HDR has to solve.

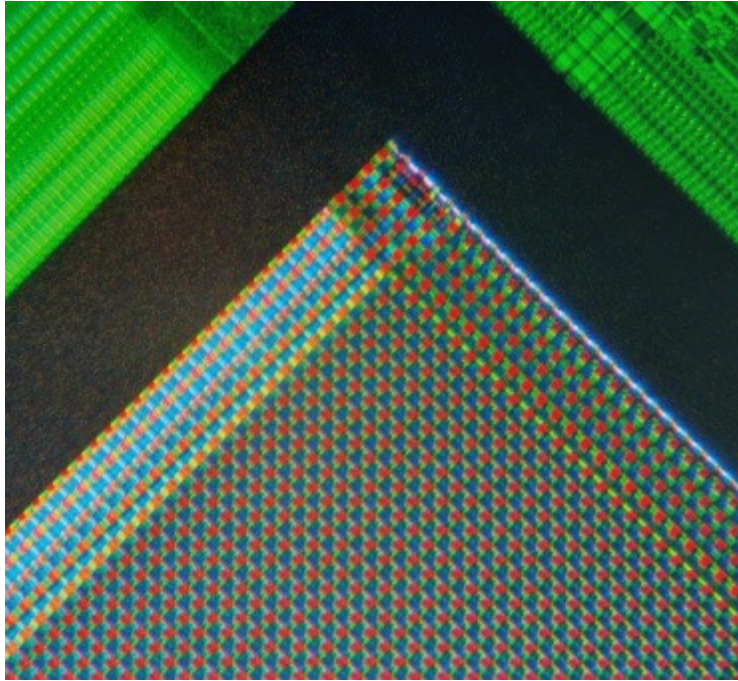


Figure 2.7.2 [\(reproduced\)](#). A sensor surface under the microscope. The regular grid of photosites is overlaid with the Bayer color-filter mosaic: twice as many green filters as red or blue, because the eye resolves luminance (carried mostly by green) more finely than color. Each site is one pixel's light-collecting well; full color at every pixel is reconstructed later by demosaicking.

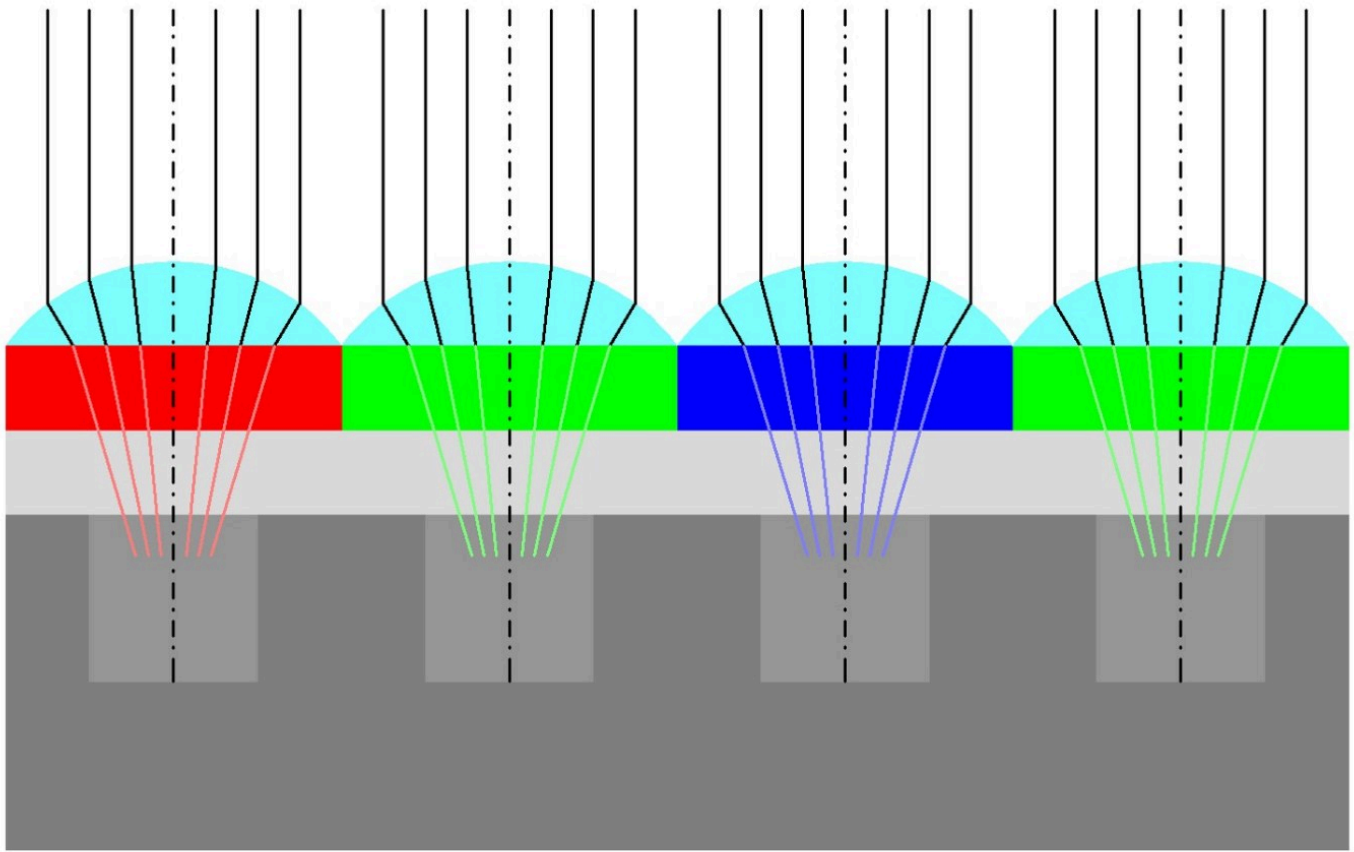


Figure 2.7.3 (reproduced). A photosite in cross-section. Light enters through a microlens that gathers it onto the photosite's active area (wiring leaves the fill factor below 100%), passes a color filter (one of the Bayer mosaic's red, green, or blue), and reaches the photodiode, where it frees electrons that accumulate as charge. The charge becomes a voltage and then a digital number. Counting photons is what makes capture linear in light.

How big is a photosite, really? Pixel sizes span more than a factor of ten across cameras, and the range maps directly onto the sensor-size story below. A flagship phone crams its pixels down to about $0.7\text{--}1.0\text{ }\mu\text{m}$; a 1-inch compact sits near $2.4\text{ }\mu\text{m}$; APS-C runs $3.5\text{--}4\text{ }\mu\text{m}$; a full-frame sensor is $4\text{--}6\text{ }\mu\text{m}$; and the largest-pixel cameras (low-resolution full-frame bodies, astronomy and scientific sensors) reach $8\text{ }\mu\text{m}$ and beyond. The telling comparison is biological: a foveal cone is about $2\text{ }\mu\text{m}$ across ([Chapter 3.2, Perceptual color and trichromatic vision](#)), so a big camera pixel is several cones wide, while a modern phone pixel has shrunk *below* the size of a single cone ([Figure 2.7.4](#)). Size matters here because a larger collecting area receives more photons at equal irradiance and exposure, increasing shot-noise standard deviation but reducing noise relative to signal, which is the whole reason larger sensors look cleaner ($\rightarrow$ *The impact of sensor size*, below). The phone's tiny wells force it to make up the deficit computationally, with burst capture and denoising rather than glass and silicon.

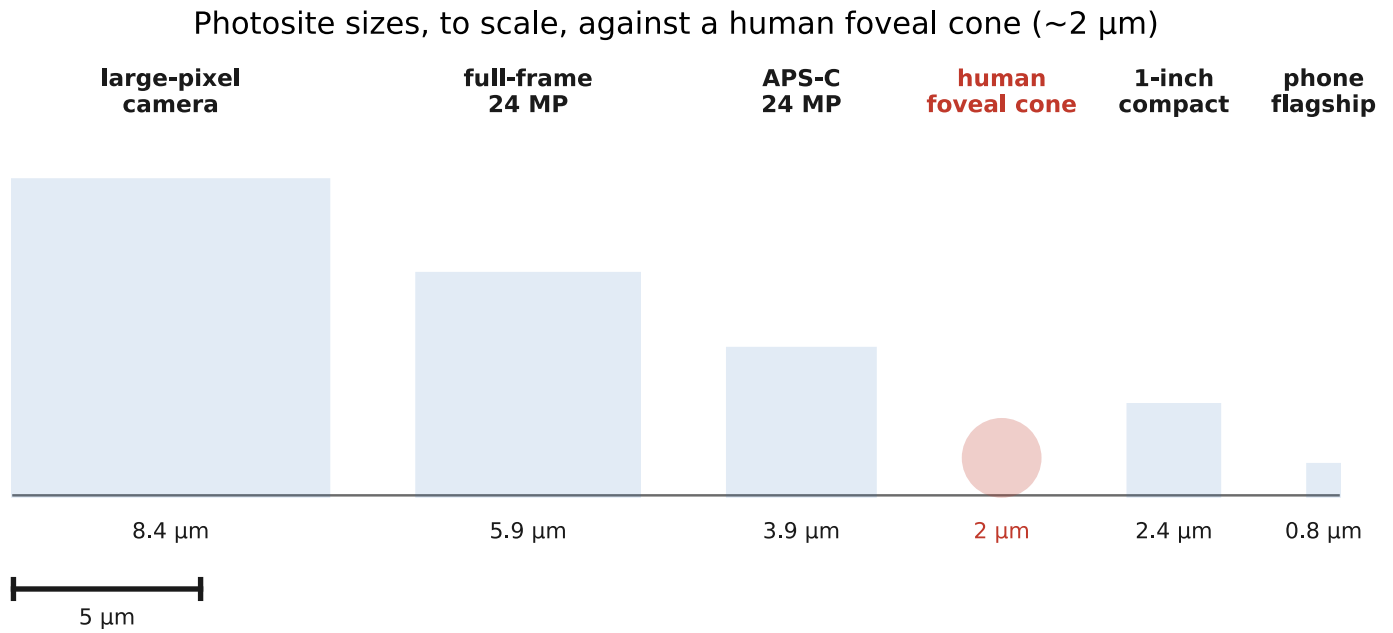


Figure 2.7.4 (reproduced). Photosite sizes, to scale, against a human cone. Drawn at a common micrometer scale: a large-pixel camera ($\sim 8.4\ \mu\text{m}$), a full-frame pixel ($\sim 5.9\ \mu\text{m}$), an APS-C pixel ($\sim 3.9\ \mu\text{m}$), a human foveal cone ($\sim 2\ \mu\text{m}$, the biological yardstick), a 1-inch compact pixel ($\sim 2.4\ \mu\text{m}$), and a flagship-phone pixel ($\sim 0.8\ \mu\text{m}$). A big camera pixel spans several cones; a modern phone pixel is smaller than one cone, which is why phones must recover image quality computationally rather than optically.

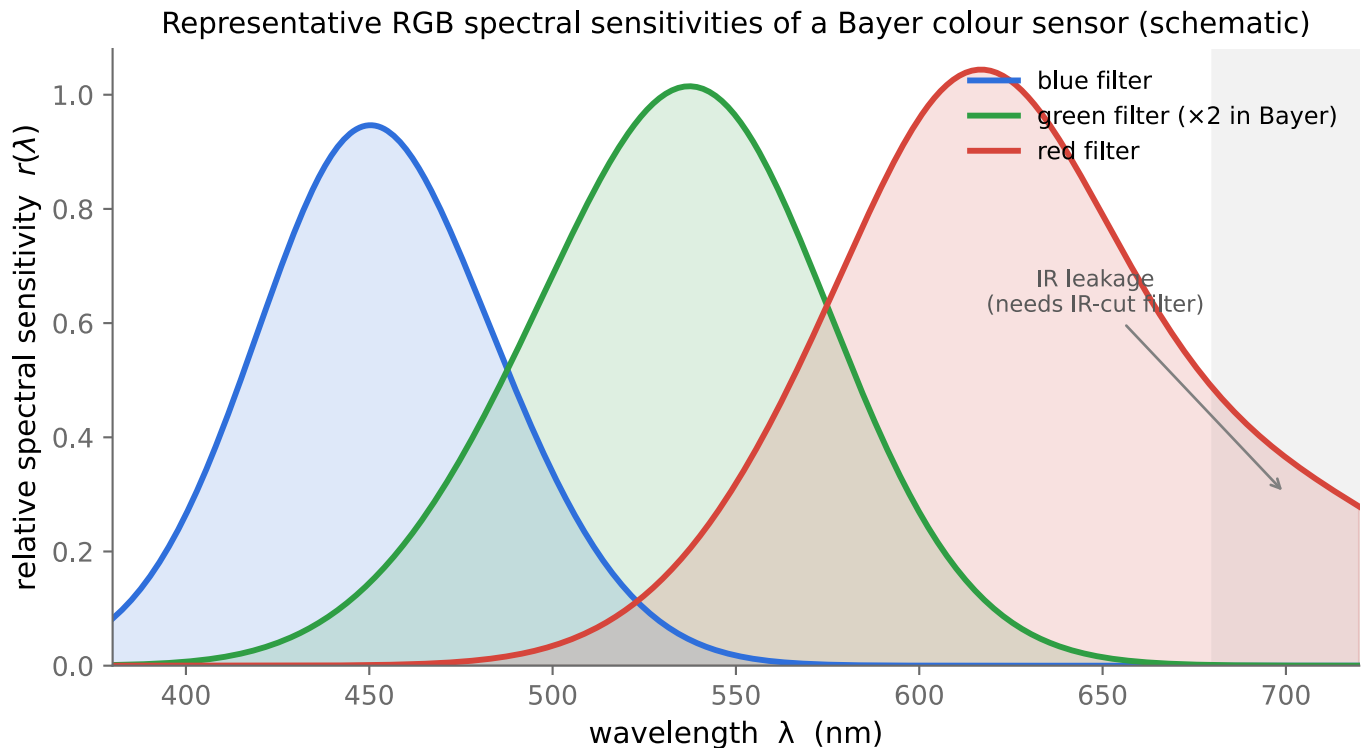


Figure 2.7.5 [\(reproduced\)](#). Representative RGB spectral sensitivities of a digital color sensor. Each curve is the probability that a photon of a given wavelength, passing the red, green, or blue Bayer filter, is counted by the photosite beneath, the $r(\lambda)$ of the pixel integral. Note the broad, heavily overlapping passbands (nothing like three clean spikes), the doubled-up green sitting where the eye is most sensitive, and the long-wavelength leakage that forces an infrared-cut filter in front of the sensor. Because each channel is one number, an integral of the spectrum against its curve, very different spectra can map to the same RGB triple (metamerism, taken up with color in part 3, [Chapter 3.2, Perceptual color and trichromatic vision](#)). Curves are schematic, drawn to typical silicon-plus-Bayer shapes rather than one measured device.

Adaptation and constancy

[in the book](#) ➤

Discounting the illuminant

A white shirt can remain recognizably white under daylight or a warm lamp despite a change in the spectrum reaching the eye. Color constancy describes this approximate stability of surface-color judgments. It does not imply recovery of the surface's full reflectance spectrum. Illuminant estimation, adaptation, and spatial context all contribute.

Von Kries adaptation models an independent gain on each cone class [Von Kries](#) ([Figure 3.3.16](#)). Gains chosen to map an adapting white to a reference white give a useful approximation, not exact illuminant removal for every surface. [Chapter 4.2, White balance and white point](#) develops the corresponding transform and its limits.

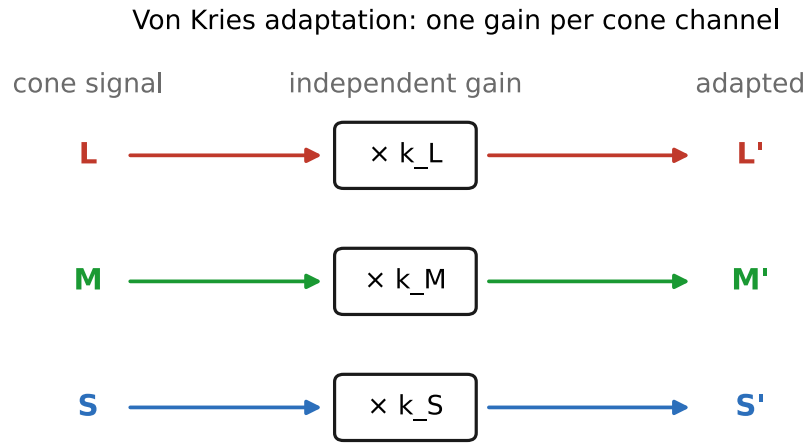


Figure 3.3.16 (reproduced). Von Kries cone-gain adaptation. The same surface viewed under two different illuminants produces different cone responses; an independent multiplicative gain on each of the *L*, *M*, and *S* channels rescales them so the surface reads as the same color. This per-channel gain is the perceptual blueprint for camera white balance.

Camera white balance estimates gains or a more general transform to render the scene for the intended output. Gray-world, bright-pixel, and learned methods use different evidence. These engineering estimators need not duplicate the visual system's computation [Freeman & Brainard \(Bayesian color constancy\)](#).

In low light, weak cone signals and increased rod contributions reduce color discrimination. A camera that accumulates a burst may record color that was difficult to see during capture. Night Sight uses learned white balance as part of a pipeline demonstrated down to 0.3 lux [Liba et al. 2019](#). That illuminance is a capture condition, not a universal threshold below which no person has useful color information. The desired rendering may also preserve some warmth or darkness rather than fully neutralizing the scene.

BIG LESSON (L3.12) — THE EYE DISCOUNTS THE COLOR OF THE ILLUMINANT

Chromatic adaptation and scene interpretation help stabilize surface color across illuminants. Per-channel cone gains are a useful approximation. Camera white balance makes a related rendering choice for the output conditions, with accuracy limited by the measurements and estimated illuminant.

Linear color reproduction and CIE XYZ

White balance and white point

Encoding: linear, gamma or log

in the book [↗](#)

Linear image values are proportional to the measured or represented light. An encoding maps them to stored numbers. The choice matters both for precision and for processing: adding encoded values generally does not add the corresponding light.

Uniform integer steps in linear light devote relatively few levels to dark regions. Human contrast sensitivity often depends more closely on relative than absolute differences, so a nonlinear encoding can use a limited bit budget more effectively [Poynton 2012](#). This is an engineering approximation, not a universal law of brightness. Visibility also depends on the background, spatial structure, noise, and viewing conditions.

For example, a range of 100:1 divided into successive one-percent ratios needs about $\ln(100)/\ln(1.01) \approx 463$ intervals. This motivates allocating codes by relative precision, but it does not prove that eight bits always suffice. Smooth gradients and subsequent editing can expose eight-bit quantization.

The name “gamma correction” encourages the wrong mental model. An encoding changes the numbers used to represent light; decoding reverses that change. There may also be a display nonlinearity to compensate and a deliberate contrast change to apply, but these are separate jobs even when all three involve power curves. Poynton's discussion emphasizes the continuing value of perceptual encoding after the original CRT motivation [Poynton 2012](#). Removing an old display technology does not remove the need to allocate a finite code range according to visibility.

RGB color spaces

OPTIONAL

in the book [↗](#)

An RGB space chooses primary chromaticities and a reference white; its stored representation also specifies a transfer function. Different choices serve different devices and workflows. sRGB is a common SDR delivery space. Display P3 extends its red and green coverage. Adobe RGB extends especially into greens and cyans useful in some print workflows. Rec. 2020 has still wider primary chromaticities. ProPhoto is a large working space with imaginary primaries.

With fixed integer precision, a larger gamut generally spreads codes more widely in color. The effect varies across the gamut and depends on the transfer function. Use sufficient precision for editing, then choose the delivery space from the output requirements. A screen cannot display every working-space value merely because software can store it.

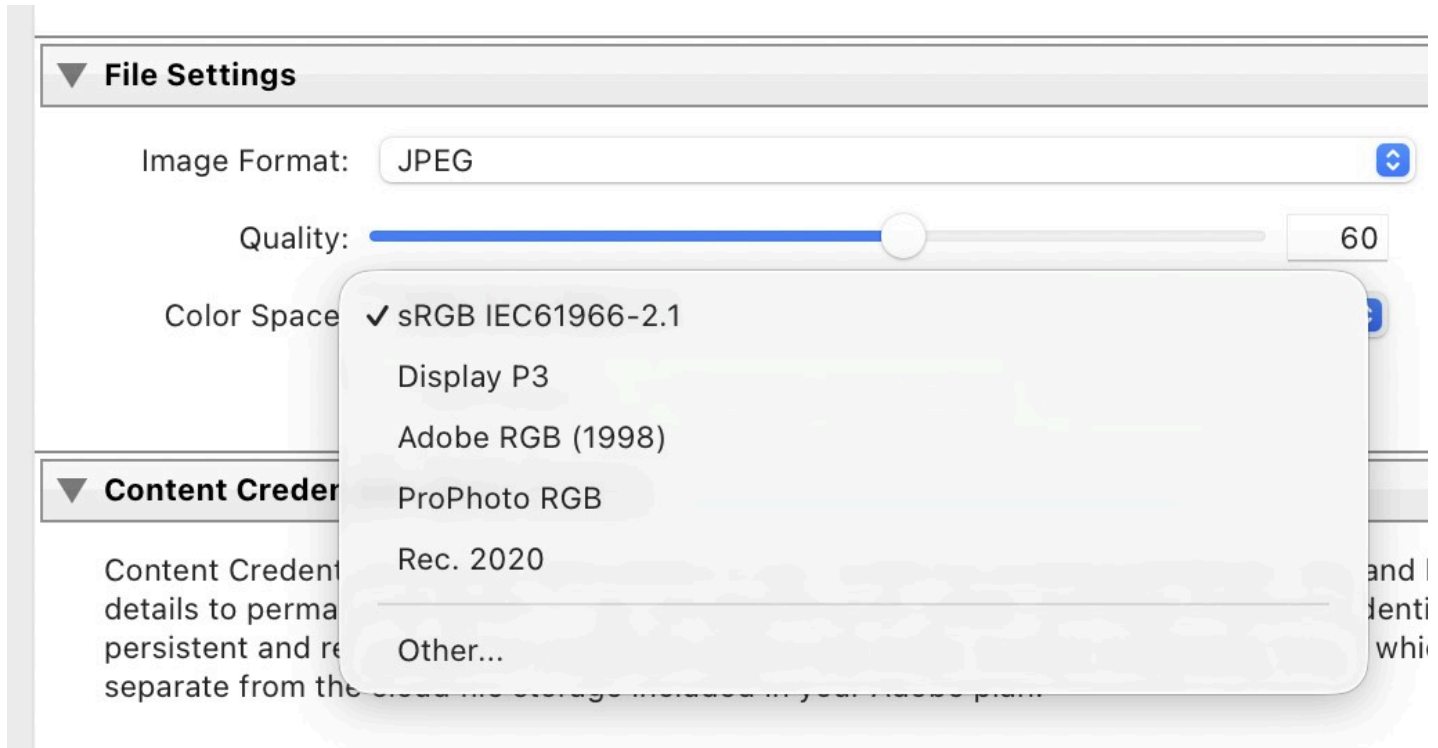


Figure 4.4.1 (reproduced). *The choice, as a photographer meets it. Lightroom's export panel asks for a color space alongside the file format and the JPEG quality, and the menu is the zoo of this chapter: sRGB for the web and for anyone whose screen you cannot know, Display P3 and Adobe RGB when the destination is wider, ProPhoto RGB and Rec. 2020 for editing and for delivery to displays that do not exist yet. Nothing in the pixel values says which one was meant, which is why the choice has to be recorded in the file and why getting it wrong shows up as color that is merely plausible rather than right. Screenshot of Adobe Lightroom Classic.*

💡 **BIG LESSON (L4.8) — THERE IS NO SINGLE RGB: GAMUT VERSUS BIT BUDGET**

Choose a working gamut large enough for the content and operations, with sufficient numeric precision. Choose a delivery space the receiving system can interpret. Wider primaries alone do not improve an image, and converting an sRGB image to a wider space does not recover colors previously clipped.

Display-oriented spaces describe a reference set of primaries. Working spaces can instead choose imaginary primaries to enclose more physical colors. ProPhoto's green and blue are outside the spectral locus; its red is not the imaginary pair's other member. Such coordinates are useful for calculation, but their nonnegative cube contains values no physical light can produce.

Camera raw is different. Its channels are spectral measurements, often one color sample per photosite in a mosaic, not weights of three emission primaries. A camera would admit an exact universal XYZ matrix if its sensitivities spanned the standard observer's matching functions. Real cameras generally only approximate this condition, so color correction is fitted and depends on spectra and illumination. It is not meaningful to invert an arbitrary three-by-many sensitivity matrix and call the result three physical display primaries.

Camera profiles can provide multiple illuminant calibrations. Do not assume every DNG contains exactly two matrices at fixed temperatures; use the tags and conventions of the actual profile. Camera color-space menu settings usually govern rendered output and previews, while raw channel interpretation remains camera-specific.

The correction matrix also propagates noise. With equal independent input variances σ^2 , output channel i has variance $\sigma^2 \sum_j M_{ij}^2$. More generally use $M \Sigma M^T$. Negative coefficients do not alone prove amplification; their magnitudes and the input covariance matter. The filter design and color fit balance sensitivity, color accuracy, and noise, as developed in [Chapter 4.6, Sensing color: multiplexing strategies](#).

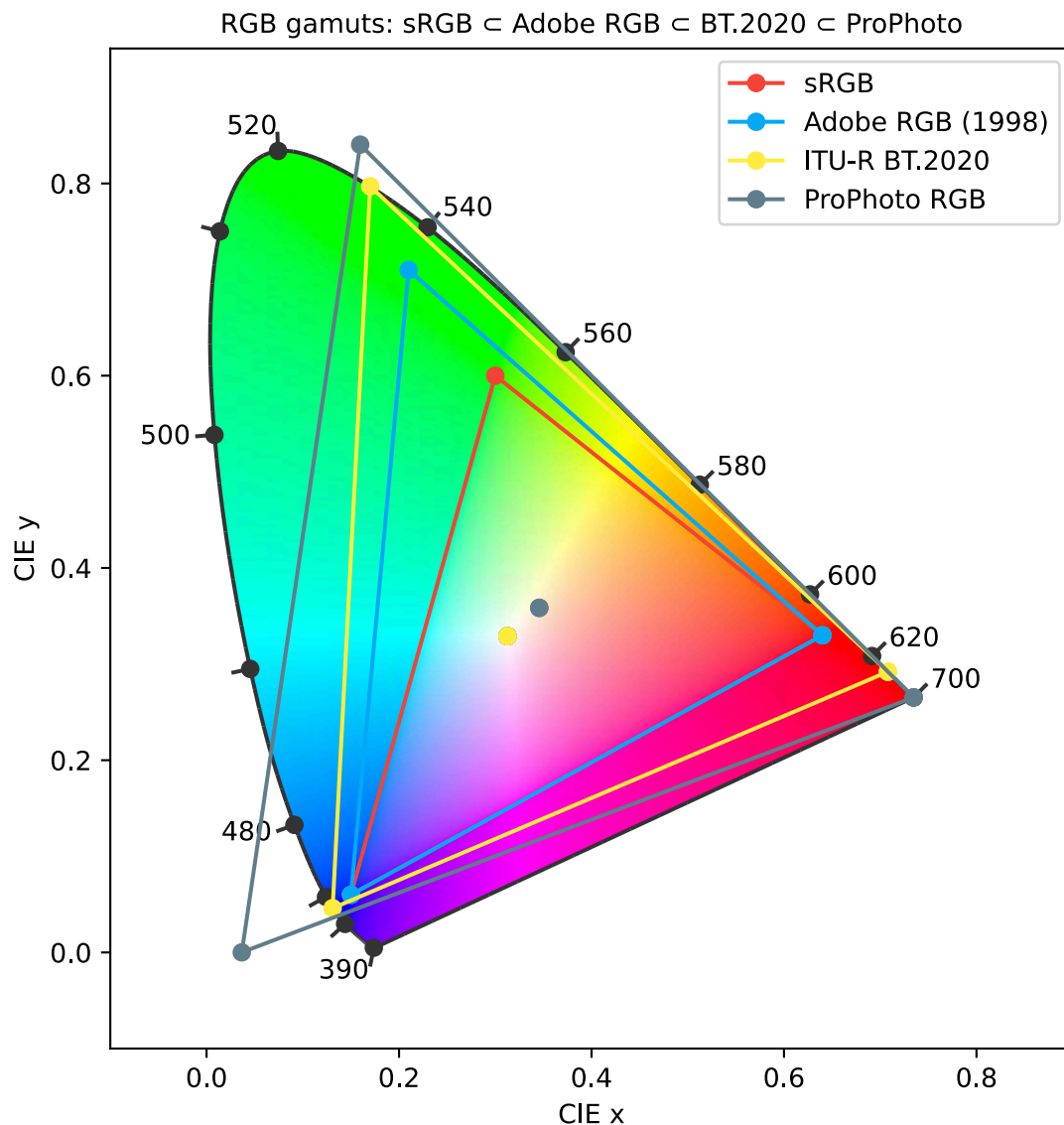


Figure 4.4.2 (reproduced). RGB primaries on the chromaticity diagram. The triangle vertices are the red, green, and blue primaries of each space, and the triangle they enclose is that space's gamut. sRGB / Rec. 709 is the small triangle, and Adobe RGB, Rec. 2020 and ProPhoto enclose progressively larger areas: each really does contain the one before it, primary by primary. ProPhoto is the one to look at twice. Two of its three primaries lie outside the spectral locus, so they are not colors at all, and a corner of its triangle names nothing that can be seen. A change of space is a 3×3 matrix between these triangles.

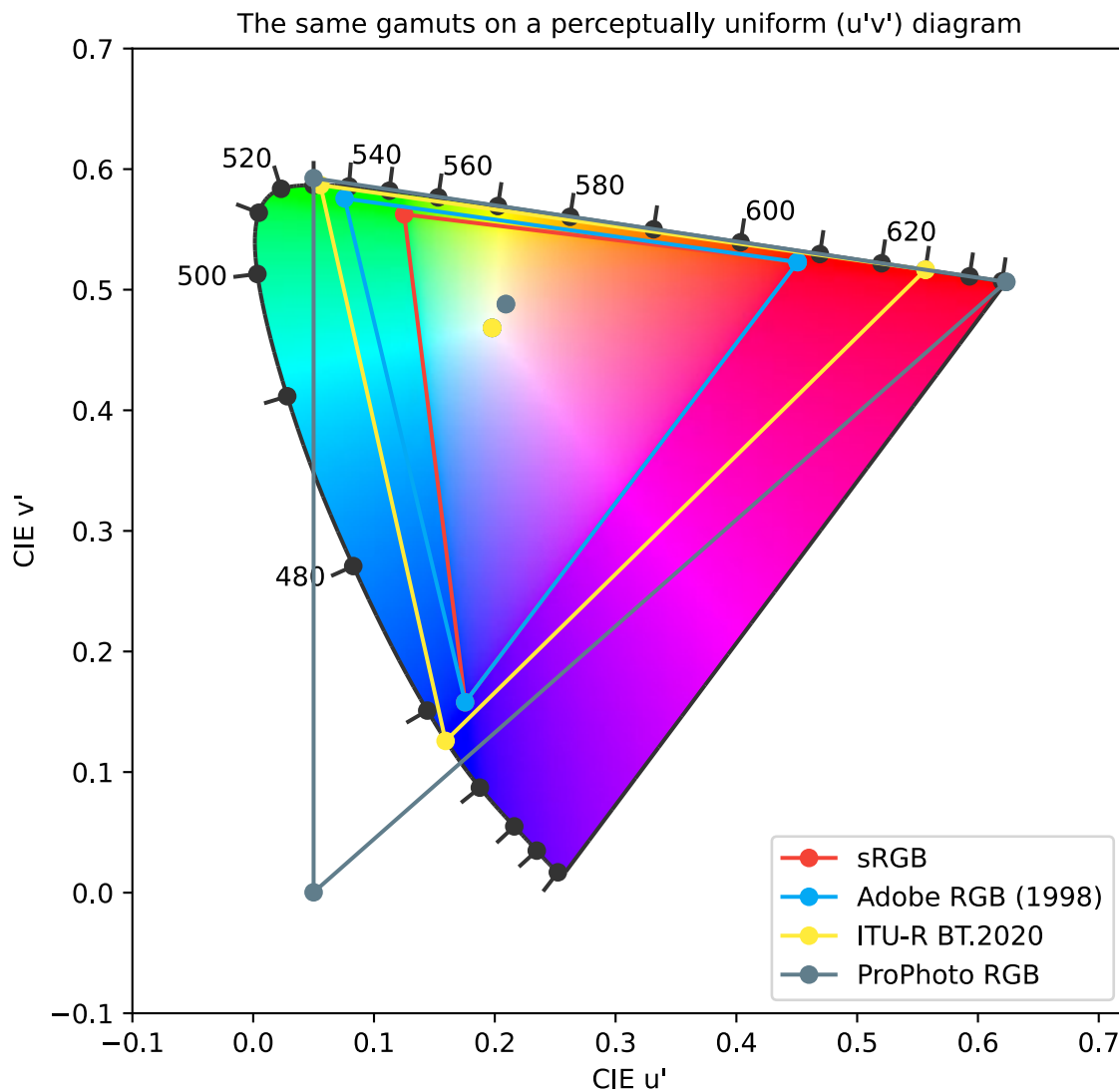


Figure 4.4.3 ([reproduced](#)). The same RGB gamuts on a perceptually uniform diagram. The identical sRGB, Adobe RGB, BT.2020 and ProPhoto primaries of the previous figure, replotted on the CIE 1976 UCS ($u'v'$) chromaticity diagram, which spaces colors far more evenly by perceived difference. Compared with the xy diagram, the oversized green lobe collapses and the blues open up, so the triangles take on very different shapes and relative areas, from the very same numbers. The lesson: xy (and XYZ) measures color, it does not tell you how different two colors look. Rendered by colour-science.

SPACE

☒ sRGB / Rec.709

☐ Adobe RGB (1998)

☐ Display P3

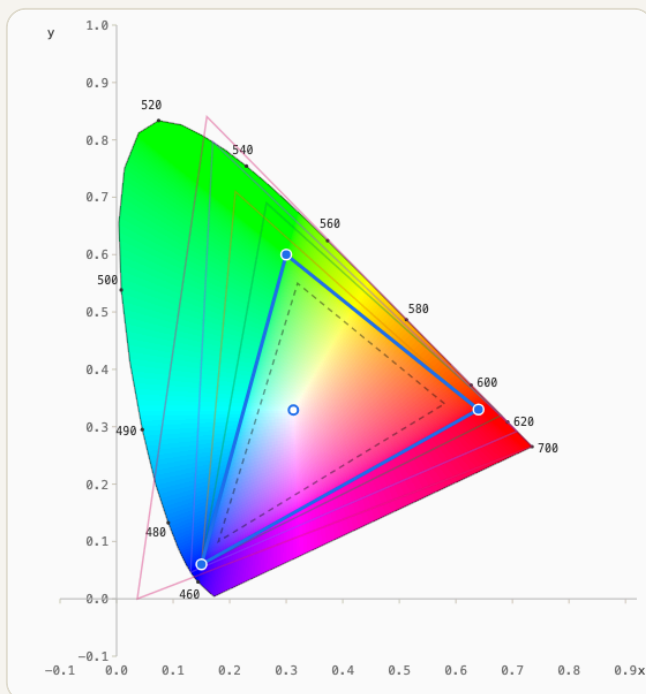
☐ Rec. 2020

☐ ProPhoto RGB

☒ editable

reset editable

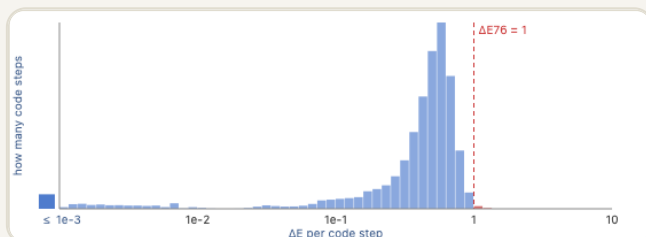
DIAGRAM ☐ more uniform (CIE 1976 $u'v'$)



sRGB / Rec.709: 33.3% of the horseshoe,
1.00x sRGB

QUANTIZATION power γ

bits



sRGB / Rec.709 at 8 bits, power γ 2.2 –
median step 0.492 ΔE , 90th percentile
0.709, 0.3% of sampled steps over $\Delta E76 =$
1

pure-power code range, dimmest non-black
code to white: 1:2.0e+5, 17.6 stops

Quantization model: a pure power curve for every

the colors themselves, with every pixel the selected
space cannot hold flagged



2.4% of pixels are outside sRGB / Rec.709
– a measured reflectance cube rendered
under D65 (CAVE, Columbia)

gamut, not each format's specified transfer function.
Actual 8-bit sRGB spans about 11.7 stops from code
1 to white. ΔE_{76} is a rough visibility proxy; 1 is not a
universal detection threshold.

Figure 4.4.4 ([reproduced](#)). *Chromaticity and color spaces, interactive*. Every gamut is drawn at once on the CIE horseshoe, faintly, and one is **selected** from the row of buttons: that one is drawn in full, carries the draggable handles, and drives everything else. Selecting it also sets the encoding that space is really used with, sRGB, Adobe RGB and Display P3 at $\gamma \approx 2.2$ in 8 bits, Rec. 2020 at 2.4 in 10, ProPhoto at 1.8 in 16; both sliders stay free afterwards. The axes run a little below zero so a primary can be dragged *outside* the locus, which is where two of ProPhoto's live. A checkbox swaps xy for the perceptually uniform u'/v' diagram, and the coverage readout is measured in u'/v' either way: sRGB holds **33.3%** of the visible chromaticities, Adobe RGB 38.8% (1.17× sRGB), Rec. 2020 57.4% (1.72×), ProPhoto 76.7% (2.30×), and the editable triangle live as you drag it. **Below the diagram, what quantization costs in that space.** At every code in the space's own cube, step one level along R , G and B , decode both ends through that space's transfer and that space's primaries, and measure the step as a full CIELAB ΔE , lightness and chroma together and not brightness alone; the histogram is the distribution of those steps, and 1 ΔE serves as a rough proxy for a just-noticeable difference, which in practice varies with the color and the viewing conditions. Its axis is **fixed**, 0.001 to 10 ΔE , so two spaces can be compared by eye rather than by reading the ticks; steps past either end are counted into a bar outside the axis instead of being squeezed in. It answers to all three knobs. Widen the primaries and the same codes have more color to cover: at 8 bits and $\gamma = 2.2$, **0.3%** of sRGB's code steps are visible, against 8.0% for Rec. 2020 and **23.3%** for ProPhoto, which is exactly why ProPhoto is specified at 16, where it is back to 0.0%. Change γ instead, in sRGB at 8 bits, and you get the other half of the story: 6.5% of steps visible at $\gamma = 1$, 0.3% at 2.2, none at 3.0, while the *median* step grows. Gamma does not remove the error, it levels it. Underneath, the **dynamic range** those same codes buy, from the dimmest non-black code up to white: 8.0 stops at $\gamma = 1$, which is one stop per bit and no more, against 17.6 at $\gamma = 2.2$ in the same 8 bits, 24.0 for Rec. 2020's 10 bits at 2.4, and 28.8 for ProPhoto's 16 at 1.8. Unlike the histogram, that number is set by the encoding alone and not by the primaries, since every space here puts white at $Y = 1$. Read it as an upper bound: the slider is a *pure* power, whose slope at zero is infinite, so it reaches further into the dark than a real transfer curve with a linear toe. sRGB's actual piecewise curve gives 11.7 stops at 8 bits, not 17.6. **On the right**, one sample throughout, a measured reflectance cube rendered under D65, with every pixel the selected space cannot hold flagged. Only 2.4% of these paints leave sRGB, 0.6% leave Display P3 and none leaves Rec. 2020, which is this section's argument in three numbers: real pigments barely need more than the oldest gamut.

This distinction explains two different kinds of color-conversion experiment. Between linear RGB working spaces built from the same standard observer, convert a color to XYZ and then to the destination coordinates; the two matrices compose into one exact coordinate transform. For a camera, photograph a set of known reflectances under a specified light and fit a matrix from the measured camera responses to their target XYZ values. It can fit those samples well and still make errors on other spectra. Two spectra that the camera records identically cannot be separated by any later matrix, even if the standard observer distinguishes them. More training patches improve the fit over a chosen population; they do not restore spectral information absent from the three measurements.

The choice of a working gamut also affects quantization. With a fixed integer bit depth, a wider set of primaries spreads the available code combinations over a larger range of colors. It can retain a saturated color that a smaller space clips, yet give coarser steps in a region shared by both spaces. Increasing bit depth or using floating point reduces that pressure. This is why a space suited to intermediate editing need not be the best choice for compact

delivery. The working space must retain the colors the pipeline needs; the output space must match the interpretation supported by the destination.

A caution about that diagram before we read too much into the triangle sizes: the xy chromaticity plane is a space for *measuring* color, not for judging how different two colors look. It stretches the greens enormously and cramps the blues and purples, so neither the relative areas of these gamuts nor the distances between their primaries track perceived difference. Re-plot the *identical* primaries on a perceptually more uniform diagram, the CIE 1976 UCS ($u'v'$) space, and every triangle changes shape ([Figure 4.4.3](#)): the relative areas and distances change. The newer diagram is more uniform, but it is still not a universal perceptual ruler. That XYZ (and its xy projection) is *not* perceptually uniform is precisely what will force the deliberately non-linear spaces of [Chapter 4.5, Advanced color: opponents, perceptually-uniform, and preferred](#).

Two design processes can lead to an RGB space. In a display-oriented design, begin with the primaries a reference display can emit, then describe colors as amounts of those lights. In a working-space design, begin with the colors that editing should retain, then choose a coordinate system that encloses them. The latter need not correspond to a manufacturable display. An imaginary primary is acceptable as a coordinate axis because no lamp is being asked to emit it while the image is stored or edited. Physical realizability becomes an output constraint when those values are converted for a screen or printer.

This distinction explains why choosing a larger working space does not upgrade the monitor. It changes which intermediate colors can be represented without clipping to the working cube. Displaying them still requires conversion and, where needed, gamut mapping. Conversely, a delivery space can be chosen for compatibility with the intended viewing path even when a larger space was useful during editing. The export dialog asks for an engineering choice about that destination, not a ranking in which the largest triangle is always best.

The noise formula can be read numerically. A correction row with coefficients 1.8, -0.9, and 0.1 has squared length 4.06. With equal independent input noise, its output standard deviation is therefore about 2.01 times the standard deviation of one input channel. The row may improve the color fit while increasing that channel's noise. This is why a camera characterization should be evaluated for both reconstruction error and noise propagation, using the actual covariance rather than assuming that every color matrix has the same cost.

Advanced color: opponents, perceptually-uniform, and preferred

Sensing color: multiplexing strategies

Recap: the full color pipeline, color management and ICC

OPTIONAL

Exposure and capture timing

Basic photography: exposure settings (shutter, aperture, and ISO)

OPTIONAL
in the book [↗](#)

For an unchanged scene and lens transmission, optical exposure scales as

$$H \propto \frac{t}{N^2}, \quad N = \frac{f}{D}. \quad (10)$$

Twice the integration time collects twice the light. Multiplying the f-number by $\sqrt{2}$ halves it. Thus f/2.8 at 1/250 s and f/4 at 1/125 s give approximately the same optical exposure, but the longer integration blurs motion more and the smaller aperture increases depth of field.

A stop is the base-two logarithm of a light ratio. The setting-based exposure value uses time in seconds:

$$\text{EV} = \log_2 \left(\frac{N^2}{t/(1\text{ s})} \right). \quad (11)$$

Increasing this EV by one halves optical exposure. Positive exposure compensation is a different convention: it asks an automatic mode for a brighter result relative to its metered choice. In aperture priority at fixed ISO, +1 stop will ordinarily double time. In manual mode with Auto ISO it may instead double gain while leaving the light collected unchanged.

ISO is not a third optical control. At fixed aperture and time, increasing gain scales the signal and its shot noise together. It can reduce the importance of noise added downstream of that gain, or select a sensor conversion-gain mode with lower input-referred read noise. It can also reduce highlight headroom. On an ISO-invariant sensor over the relevant range, low-ISO raw data brightened later can be close to a higher-ISO capture. The comparison in [Chapter 2.8, Noise, signal-to-noise ratio and dynamic range](#) applies only after matching output brightness and checking clipping.

The common association between high ISO and noise usually includes a reduction in collected light: we raise ISO because the scene is dim or the shutter must be fast. Add light, open the aperture or lengthen time when the photograph permits it. Raising ISO alone does not improve photon statistics.

Reflected-light metering measures light coming from the scene; it cannot separate a dark object in strong light from a pale object in weak light without additional information. Spot metering makes the chosen patch explicit. Evaluative metering uses more of the frame and may use recognized subjects. A gray card is a useful reference under the subject's illumination, not proof that a meter assumes exactly 18% reflectance. Snow and dark clothing are good tests of when compensation is needed.

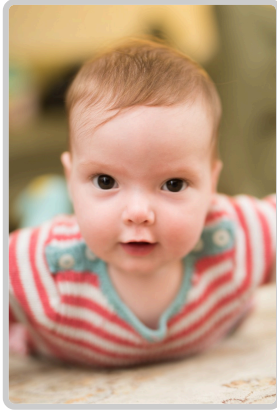
Expose to the right means collecting more light while retaining the highlights that matter. It is useful when time, aperture and saturation allow it; it is not a requirement to force every histogram against the edge. A longer exposure that blurs the subject is a poor trade for cleaner shadows. Inspect individual channels when possible, because a colored light can saturate one channel before the luminance preview looks white. Keep a margin when the camera's warnings refer to a processed preview rather than raw data.



Figure 5.2.1 (reproduced). Interactive: the faucet-and-bucket exposure analogy. Two buckets fill from faucets; the water caught is the exposure and the dashed line marks a correct one. The faucet diameter is the aperture (flow rate follows the opening's area, $\propto \text{Ø}^2$) and the fill duration is the shutter time. The left bucket is a fixed reference (wide faucet, short time); the right one you control, defaulting to the opposite (a narrow faucet open a long time), yet it catches the same water, because three stops of smaller aperture cancel three stops of longer time. Press to fill the

buckets and watch the narrow one keep pouring long after the wide one has finished; move either slider and the right bucket over- or under-fills (over- or under-exposes), with the swatch above it showing the resulting brightness: mid-gray when the exposure is correct, white when the bucket overflows into clipped highlights, dark when it is starved. A running stop tally makes the reciprocity explicit. The dashed line is one choice of target, and ISO, which changes how the reading is scaled rather than how much water is caught, has no place in the analogy.

Aperture and depth of field: a wide aperture isolates the subject; a small one keeps everything sharp. **Click a photo to enlarge.**



f/1.4 — wide aperture, shallow focus



f/22 — narrow aperture, deep focus

Photos credited in the book's image credits.

Figure 5.2.2 [\(reproduced\)](#). Aperture versus depth of field. Left: an infant portrait at f/1.4, the wide aperture throwing the background into a smooth blur that isolates the subject, admitting a lot of light but holding only a thin slab in focus. Right: a mountain scene at f/22, the narrow aperture keeping the near wildflowers and the distant fjord and peaks all sharp at once, at the cost of much less light (a longer shutter or higher ISO). The same *f*-number is both a brightness setting and a depth-of-field setting. Both photographs © Frédo Durand.

Shutter speed: freeze the motion, or let it blur. **Click a photo to enlarge.**



fast shutter, $1/3200$ s — motion frozen



slow shutter, 1 s — motion blurs to silk

Photos credited in the book's image credits.

Figure 5.2.3 (reproduced). Shutter speed versus motion blur. Left: a child bursting up through the water at $1/3200$ s, fast enough to freeze every flung droplet in mid-air. Right: a waterfall at 1 s, long enough for the moving water to smear into a silky veil while the rock stays sharp. A fast shutter records a single instant; a slow one integrates the subject's motion over its whole open time, and admits far more light doing so (here held back by a small aperture). The same shutter time is both a brightness setting and a motion-rendering choice. Both photographs © Frédo Durand.

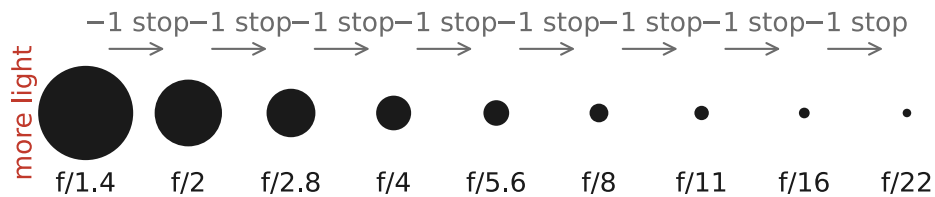


Figure 5.2.4 (reproduced). The f-stop progression as powers of two. The standard series $f/1.4, f/2, f/2.8, f/4, f/5.6, f/8, f/11, f/16$ drawn as aperture disks: each f-number is $\sqrt{2}$ times the previous, so the disk area, and the light admitted, halves at every step. The irregular-looking numbers are exactly the $\sqrt{2}$ spacing that makes each step one stop (a factor of two in light).

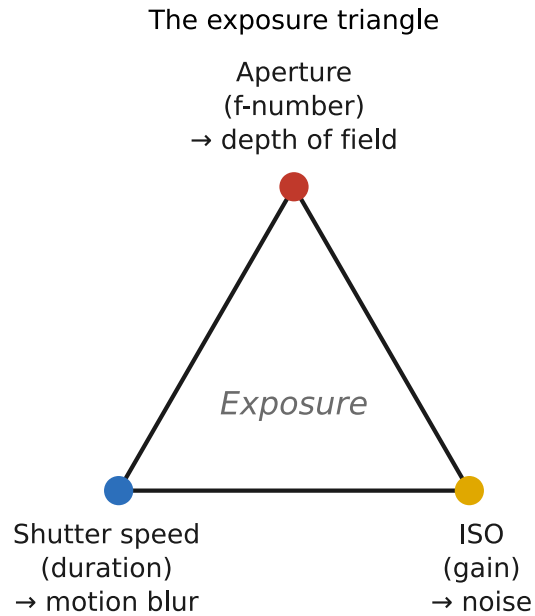


Figure 5.2.5 [\(reproduced\)](#). *The exposure triangle. Three controls set how bright a photo is, each in power-of-two stops (one stop = $\times 2$ light = 1 EV): shutter time (longer = brighter, but motion blur), aperture / f-number (wider = brighter, but shallower depth of field), and ISO (higher = a brighter rendering of the same light; the extra noise comes from the smaller exposure it lets you use, not from the setting itself). A stop of one can be traded for a stop of another to keep brightness constant while changing the side effects.*

The exposure triangle



Scene light



starlight → full moon → night → dark office → home → overcast → hazy sun → bright sun

Shutter



slower → more light + motion blur

Aperture



wider (small f) → more light + shallow depth of field

ISO



higher → brighter, not cleaner: gain scales noise with signal

Subject Sensor Auto-set (keep exposure constant):

2nd person depth



Subject motion

 [copy link](#)

Figure 5.2.6 [\(reproduced\)](#). *Try it yourself (web edition)*. A live exposure-triangle simulator: two 3D subjects stand at different depths and walk in opposite directions while you set shutter, aperture, ISO, scene brightness, and sensor size. It renders the scene physically by supersampling over time for real motion blur, over the aperture for real depth of field, and through a photon-shot-plus-read-noise model for real ISO grain. The noise model is illustrative, not a particular camera's. An "auto-set triangle" holds the rendered brightness constant so you feel each control's trade-off (freeze vs blur, deep vs shallow focus, clean vs noisy) rather than just its brightness. In the printed edition this stands in as a screenshot.

Metering to a middle tone. A reflected-light meter cannot tell what lightness you intend, so it renders what it measures as a middle tone. That suits an average scene; snow wants positive compensation and a dark subject negative. The 18% gray card is a practical reference, not the exact reflectance every meter is calibrated to. **Click any photo to enlarge.**



snow field — high-key (meter under-exposes)



old-town rooftops — average (meter is right)



dark cat — low-key (meter over-exposes)

Photos credited in the book's image credits.

Figure 5.2.7 ([reproduced](#)). *The 18% gray assumption and where it fails. The meter renders the average scene to mid-gray (~18%). For an ordinary scene this is right. For a snowfield (high-key), forcing the bright average to gray underexposes; the snow turns dull; the fix is to add +EV. For a black subject (low-key), forcing the dark average to gray overexposes; the fix is -EV. Matrix metering uses extra cues to guess intent.*

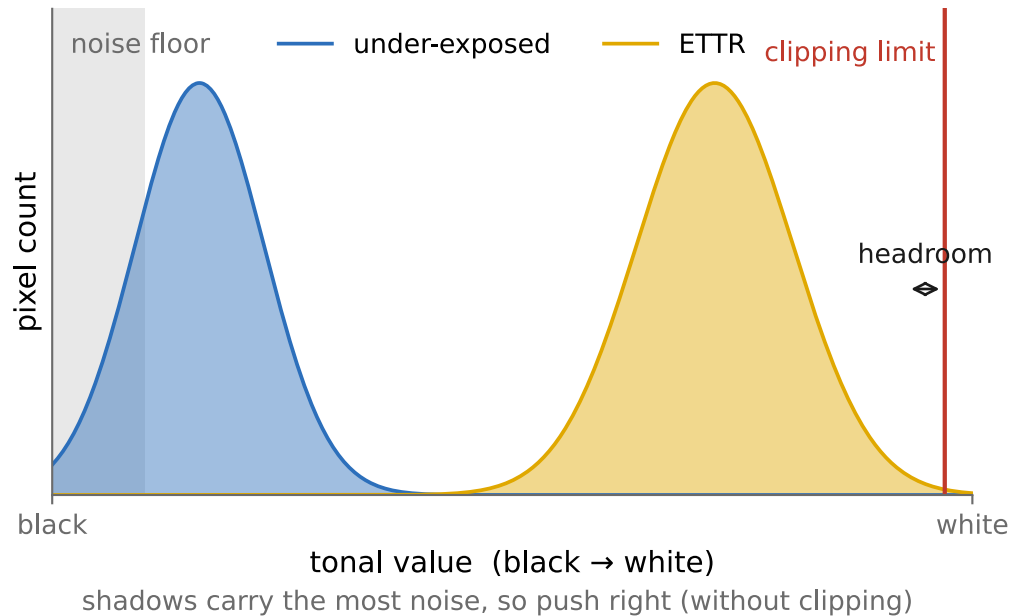


Figure 5.2.8 (reproduced). *Expose to the right (ETTR). Two captures of the same scene: (left) a timid exposure with the histogram bunched at the noise floor, later brightened in software, which amplifies shadow noise; (right) an exposure pushed as far right as possible without clipping the highlights, then pulled down in software, capturing more photons everywhere, especially in the noisy shadows. Brighter-but-not-clipped is the lower-noise capture, when motion and depth of field allow the longer exposure or wider aperture. Raising ISO alone adds no photons, and the comparison is made at the same final brightness.*

💡 BIG LESSON (L5.1) — PHOTOGRAPHERS MEASURE LIGHT IN STOPS

A stop is a factor of two. Time steps by two and f-number by $\sqrt{2}$. ISO and a software exposure slider can change rendered brightness by a stop without collecting another photon.

The f-number normalizes the physical opening by focal length. At the same f-number, a longer lens has a proportionally larger entrance pupil. For an extended scene of equal radiance, that compensates for the change in image scale so the on-axis irradiance is approximately unchanged, apart from transmission and close-focus effects. Aperture diameter alone would not give that useful comparison across focal lengths.

The familiar series $f/1.4$, $f/2$, $f/2.8$, $f/4$, $f/5.6$, $f/8$, $f/11$, $f/16$ follows from area. Each full-stop step multiplies the f-number by about the square root of two and therefore halves optical exposure at fixed time. Doubling time compensates for that step. For example, $f/4$ at $1/250$ s and $f/5.6$ at $1/125$ s collect approximately the same light per sensor area, with different motion blur and depth of field. Equal collected light does not make them interchangeable photographs.

Working with light

A face beside a window can have plenty of light and still photograph badly if the eyes fall into shadow. Move the subject or the source before changing exposure. Exposure scales the measurement; lighting changes the relationships within the scene.

Describe a source by its direction, apparent size, intensity, spectrum and timing. Direction places the shadows. Apparent size controls their penumbra. Intensity sets the photon budget and the balance with other sources. Spectrum affects color, and timing determines whether a flash freezes motion or a lamp produces bands. These properties are related in practice, but none substitutes for the others.

A large source can soften a shadow without filling it. Shadow contrast also depends on light arriving from other directions and on the surroundings. A small source can be useful for revealing texture or making a crisp graphic shadow. Soft light is a useful choice for many portraits, not a requirement for a good photograph.

Move a light sideways and the shadows move; enlarge it without changing its central direction and their edges soften. These are different changes. A small frontal source can make texture difficult to see because its shadows lie behind the relief from the camera's viewpoint. Raking side light reveals the same relief with visible shadows. A large source spans a range of directions, so a point near a shadow boundary can see part of the source while another part is occluded. That partial visibility produces the penumbra. A reflector, diffuser, or softbox should therefore be judged from the subject's position: what luminous area does the subject actually see?

Highlights provide another way to inspect the source. On a glossy object, their shape depends on both the source and the material's reflection function. A small bright source can produce a concentrated highlight; a broad source can produce an extended reflection. On glassware or a polished object, moving or enlarging the source can change the visible form more than changing exposure. Lighting controls the reflected image of the environment as well as the shadows cast onto the subject.

Developing and editing a photograph

Photographic representation and its limits

The contrast is limited: dynamic range and gamut

OPTIONAL
in the book [↗](#)

Two output constraints are easy to confuse. A print cannot emit the luminance of a sunlit scene, and its ratio between darkest and lightest regions may also be smaller. Scaling all values addresses the first mismatch but not the second. For a sense of scale, a sunlit scene with deep shade can span ten thousand to one or more, while a print under room light spans a few hundred to one at best. Tone mapping, local adjustments and lighting choices affect which distinctions survive.

A bright source can be suggested by a glow or surrounding dark tones even when the display cannot reproduce its luminance. That is a perceptual representation, not an absolute photometric measurement. Likewise, gamut mapping chooses a reproducible color when the target cannot display the original one. [Chapter 18.10, HDR displays](#)

and display mapping and [Chapter 4.7, Synthesizing color](#) describe the output limits; [Chapter 5.5, Developing and editing a photograph](#) shows the photographer's choices within them.

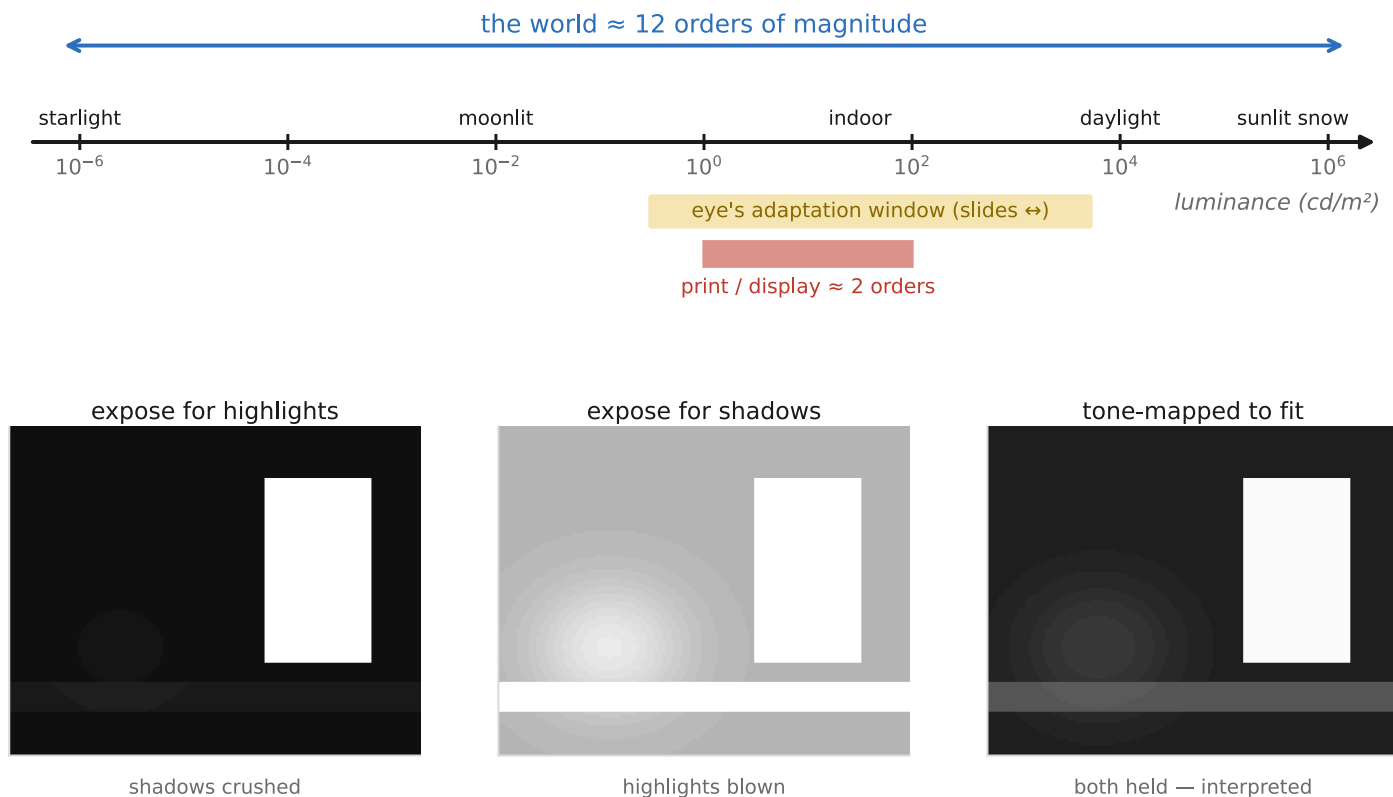


Figure 5.8.16 (reproduced). *The dynamic-range gap. A logarithmic luminance number line from 10^{-6} to 10^6 cd/m^2 shows the real world's ~ 12 -order span (starlight to sunlit snow); the eye's adaptation window slides along it (huge over time, far smaller at any instant); and the medium's ~ 2 -order print/display band sits as a narrow stripe. Beside it, one high-dynamic-range scene rendered three ways, exposed for the highlights (shadows crushed), exposed for the shadows (highlights blown), and tone-mapped to fit, annotated with the historical compensations: Brunelleschi's silver sky and an added flare/glow standing in for a brightness no print can emit.*

The history of depiction makes the intensity mismatch tangible. The lecture contrasts a painted sky with Brunelleschi's use of a reflecting silver surface for the sky in his perspective demonstration. A pigment and a reflection have different access to the surrounding illumination. Neither comparison can be reduced to an image's normalized pixel range.

The Zone System coordinates exposure, negative development and printing around planned tonal placements [Adams, *The Negative*](#). Dodging and burning modify those placements locally. Characteristic curves compress the ends of the range, while perceptual tone-reproduction methods can also model visibility, glare or adaptation [Ferwerda et al. 1996](#). These are distinct responses to limited output, not interchangeable names for a gamma curve. Retinex provides a related account of appearance through spatial relationships [Land & McCann 1971](#), but it should not be read as proof that the eye simultaneously measures the entire luminance span across all environments.

Convolution

A blur zoo

OPTIONAL
in the book [↗](#)

With the mechanics settled, the only choice left is the kernel's shape. Two are worth knowing.

The box filter gives every sample in a $(2r + 1) \times (2r + 1)$ window weight $1/(2r + 1)^2$. Its square footprint is not rotationally symmetric, and its frequency response has lobes and zeros. However, its nonnegative weights cannot create overshoot beyond the neighborhood's range; a blurred step becomes a ramp, not a ringing edge. Running sums can make large box averages inexpensive.

The Gaussian is the workhorse. Its weights fall off smoothly with distance from the center, following the bell curve

$$g(r) \propto \exp\left(-\frac{r^2}{2\sigma^2}\right), \tag{12}$$

where r is the distance to the center and σ (the standard deviation) sets the width, the blur's "radius" in spirit. Read it back: a neighbor's weight decays exponentially with the *square* of its distance, so nearby pixels count for a lot and far ones fade smoothly to nothing. The Gaussian is smooth, rotationally symmetric (no directional bias), and, as the next chapter shows, uniquely well-behaved in frequency, which is why it is the default blur almost everywhere.

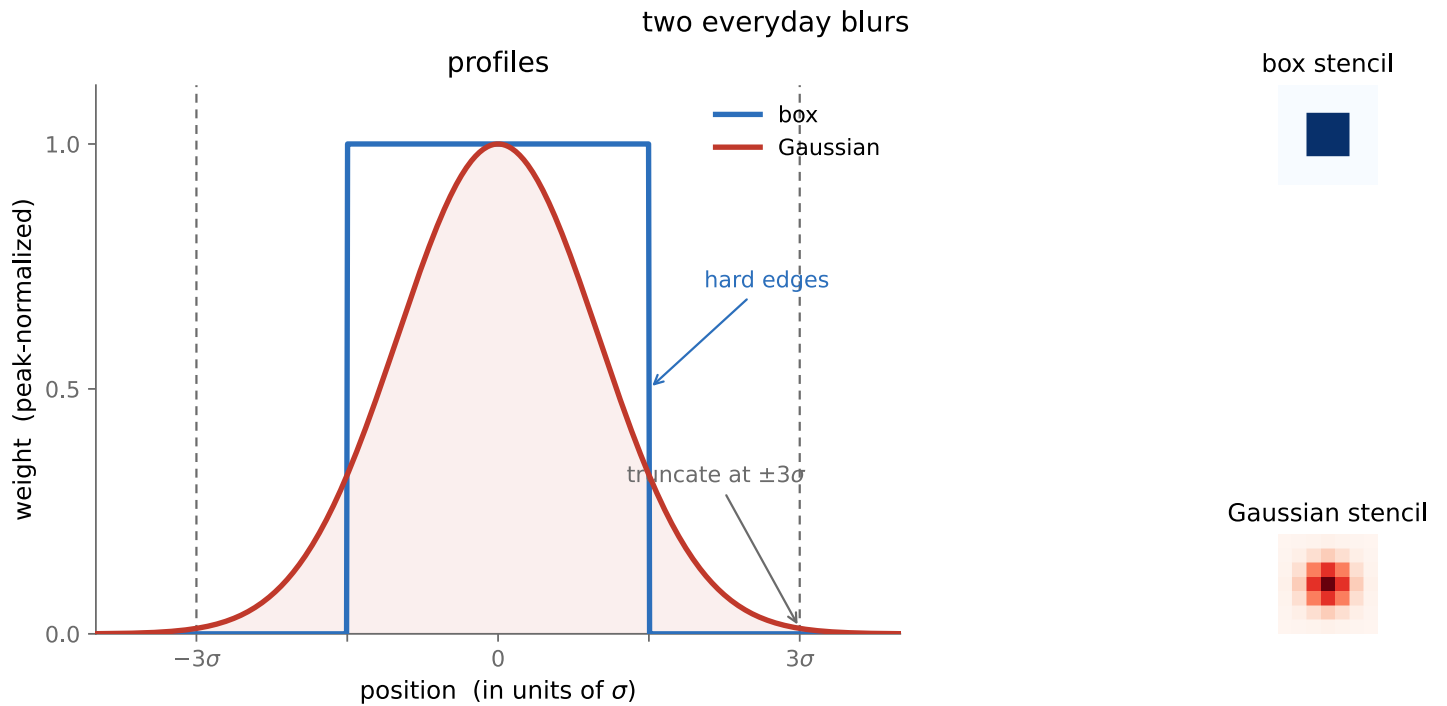
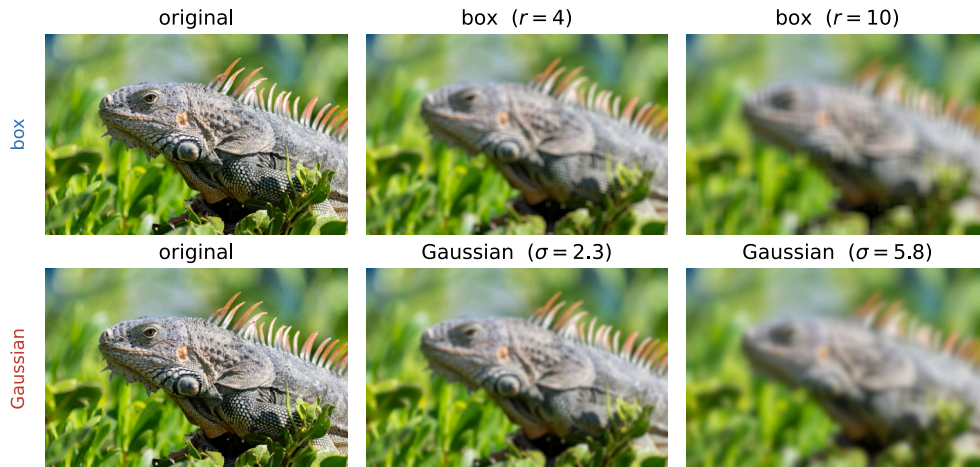


Figure 6.2.12 (reproduced). *The blur zoo: box vs. Gaussian. Top, the 1-D profiles: the box is flat then drops to zero abruptly; the Gaussian is a smooth bell that decays with distance, truncated near $\pm 3\sigma$. Bottom, the matching 2-D stencils (the actual weight grids). The box's hard edge causes directional blurring and faint ringing; the Gaussian's smooth falloff is isotropic and artifact-free, which is why it is the default. Both are normalized to sum to one so brightness is preserved.*

Run both on an actual photograph and the difference in *feel* is plain: the box smears fine texture into faintly blocky, slightly directional mush, while the Gaussian dissolves it smoothly and isotropically. Use a Gaussian when the blur should look natural (Figure 6.2.13).

the blur zoo on a real photo — box vs Gaussian, increasing radius



Matched smoothing per column ($\sigma \approx r/\sqrt{3}$); blurred in linear light. The box leaves a blockier, faintly directional smear; the Gaussian degrades detail smoothly and isotropically.

Figure 6.2.13 (reproduced). *The blur zoo on a real photo. The two everyday blurs, a box average (top) and a Gaussian (bottom), run on the same detailed photograph at increasing radius, matched per column to the same amount of smoothing ($\sigma \approx r/\sqrt{3}$, equal second moment) and computed in linear light. The box's hard square window leaves a blockier, faintly directional smear; the Gaussian's smooth falloff degrades fine-scale texture gracefully and without a preferred direction, the reason it is the default blur.*

Better than reading about them, you can try the blurs directly ([Figure 6.2.14](#)): switch kernels, change the width, and watch what each does to real texture.

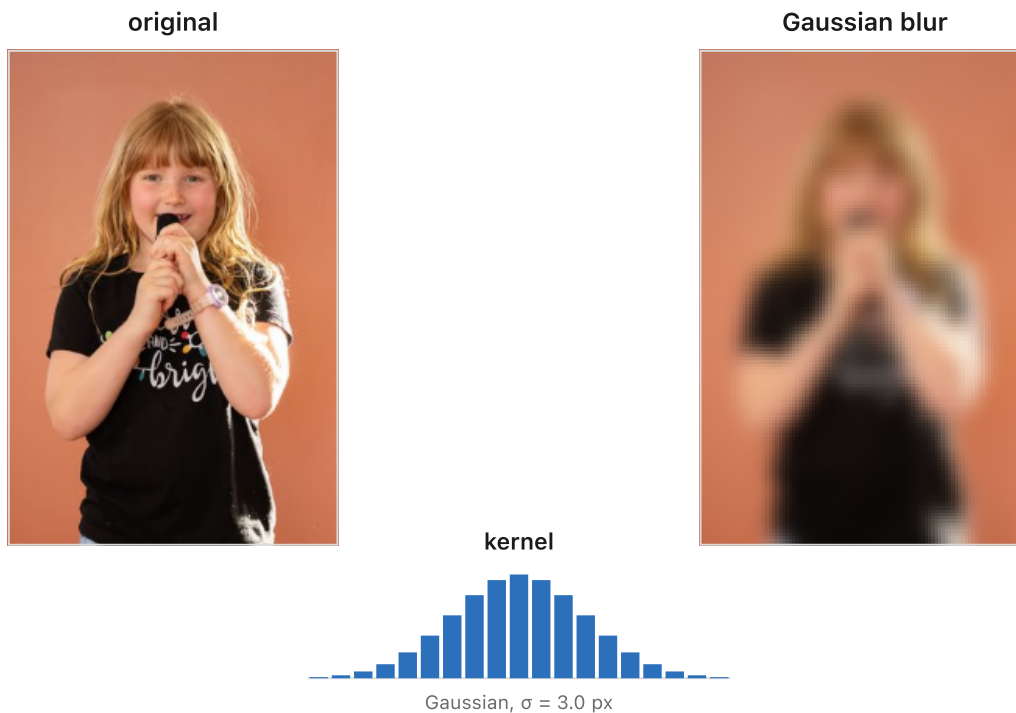


Figure 6.2.14 [\(reproduced\)](#). A blur playground. Pick a photo and a blur kernel (box, disk, triangle, the "bilinear" tent, or Gaussian) and set its size; the original and the blurred result sit side by side, with the kernel drawn below. Box, triangle, and Gaussian factor into two cheap 1-D passes; the disk is a true 2-D circular average, the shape a defocused lens paints ($\rightarrow$ [Chapter 2.5, Depth of field](#)).

An ideal Gaussian has infinite support. At a distance 3σ , its weight is $e^{-9/2} \approx 0.011$ of the peak; a one-dimensional cut at $\pm 3\sigma$ discards about 0.27% of the mass. Choose support for the required accuracy and renormalize the sampled, truncated weights. Square truncation preserves separability; circular truncation generally does not. Sampling and truncation make rotational symmetry approximate.

To simulate physical mixing, decode to linear light, filter, then encode for display. Aesthetic smoothing can deliberately use another space, but it should be named. Linear-light processing is essential when the blur is part of a calibrated forward model. For images with transparency, filter premultiplied color and alpha consistently to avoid color bleeding from transparent pixels.

The visible encoding error is easiest to see where black meets white. Averaging their encoded values gives a middle code value; decoding that value generally yields less light than averaging the original linear intensities. A blurred highlight can therefore acquire a dark transition when filtering is done in the display encoding. The arithmetic is still linear in the stored numbers, but it no longer represents linear mixing of light.

Resampling and upsampling

Domain operations: moving pixels around

OPTIONAL
in the book [↗](#)

Recall the three families of image operation from [Chapter 8.3, Tone point operations](#), sorted by what each output pixel is allowed to read. Resampling is the domain family: the underlying continuous image is remapped in position; discrete resampling generally blends its recorded colors. Formally a domain transform applies a function f from the plane to itself (from $\mathbb{R}^2$ to $\mathbb{R}^2$) to the image's coordinates, with the rule that if the point (x, y) carried color c in the input, then $f(x, y)$ carries that same color c in the output. The range (the colors) is untouched; only the domain (the coordinates) is remapped.

That one definition spans a huge amount of ground. When f is a uniform scale it is resizing; a rotation straightens a horizon; an affine or perspective f fixes keystoneing or stitches a panorama; and a freely chosen f is a warp: picture the image printed on rubber and stretched. The free-form case, including how a user even *specifies* such an f (and the morphing it unlocks), gets its own [Chapter 11.2, Antialiasing for complex transforms](#). Here we take f as given and concentrate on the hard part shared by *every* domain transform: the new positions almost never land on the input grid, so we must invent values between the samples.

Image formation as a linear system

Imaging as Bayesian inference with probabilistic priors

Image representation

in the book [↗](#)

Problem set

PS1 builds the image class, point operations, and color (YUV, gamma, white balance). → [Chapter 1.5, Problem sets](#) (appendix).

The basics of representing images digitally are simple enough to make computational photography easy and fun for beginners: just two-dimensional arrays of red, green, and blue values. However, details matter, and you will save yourself a lot of trouble by paying attention to how exactly images encode information.

An image is an array

in the book [↗](#)

A color image is stored as a grid of channel values, commonly red, green, and blue ([Figure 8.1.1](#)). These are samples or measurements; drawing enlarged pixels as squares is a display convention, not a complete model of sensor integration.

More formally, an image is a *function* defined over a 2-D domain, the grid of integer positions (x, y) , whose values live in a range, here the space of colors. We store that function as a three-dimensional array of size $\text{height} \times \text{width} \times \text{channels}$ and write $I(x, y, c)$ for the value at column x , row y , channel c . A grayscale image has a single channel, and we drop the index and write $I(x, y)$.

An image is a grid of pixels — an array of numbers

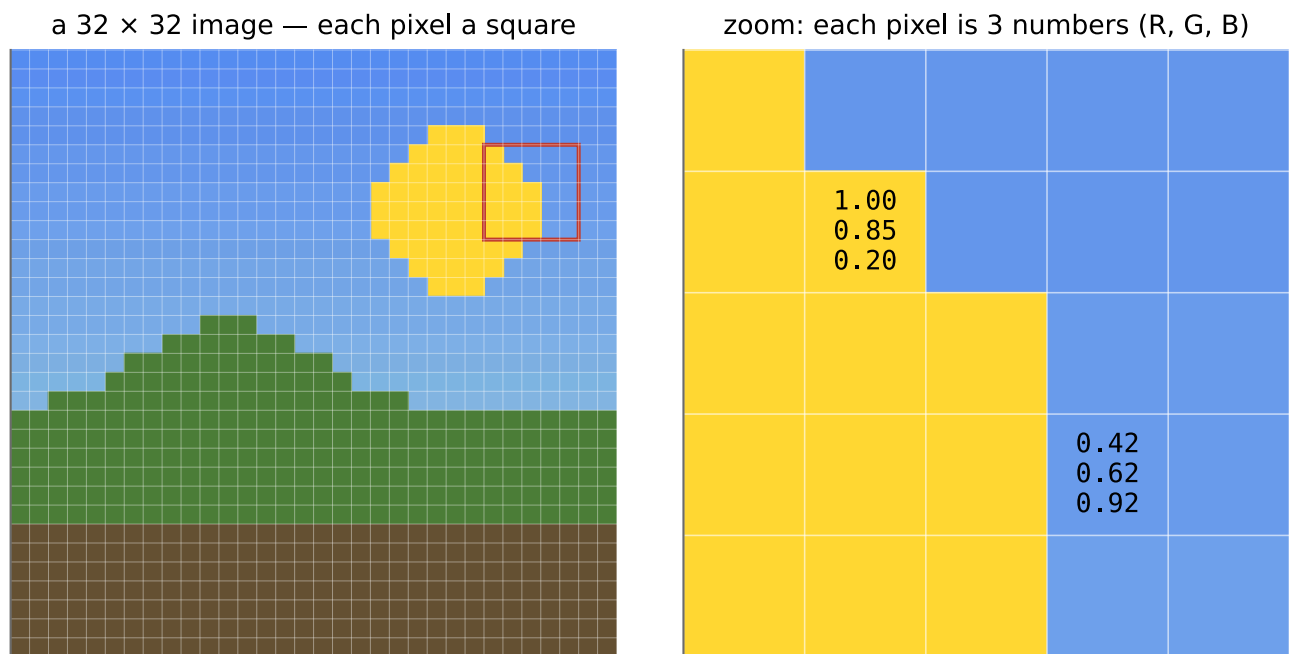


Figure 8.1.1 ([reproduced](#)). A small image and a magnified detail. Seen whole (left), you read a scene; zoomed in (right), each pixel resolves into a little square holding three numbers, its red, green, and blue values. There is nothing beneath the picture but this array, and every operation in this part of the book acts on it.

Zoom into a photograph and you reach the array of numbers underneath

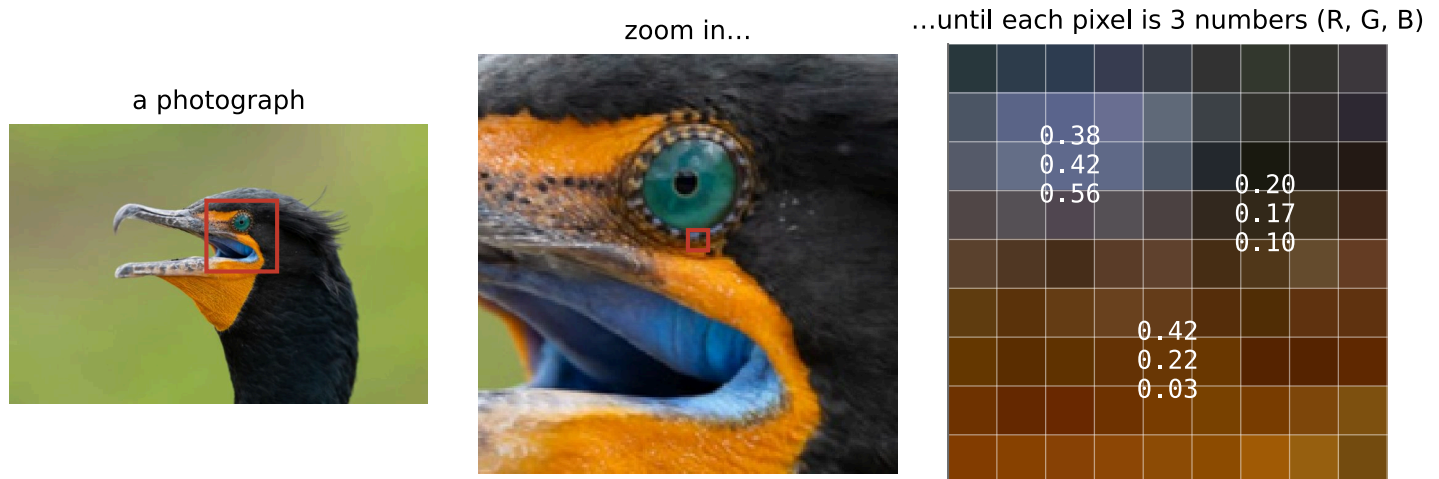


Figure 8.1.2 (reproduced). Zooming into a real photograph in three steps until the array shows through. Left: the whole image. Middle: a zoom onto the eye. Right: a zoom so deep that individual pixels become visible squares, each just an (R, G, B) triple of numbers. The schematic in Figure 1 is the literal truth of every photograph.

An RGB image is three grayscale planes — what each channel “sees”

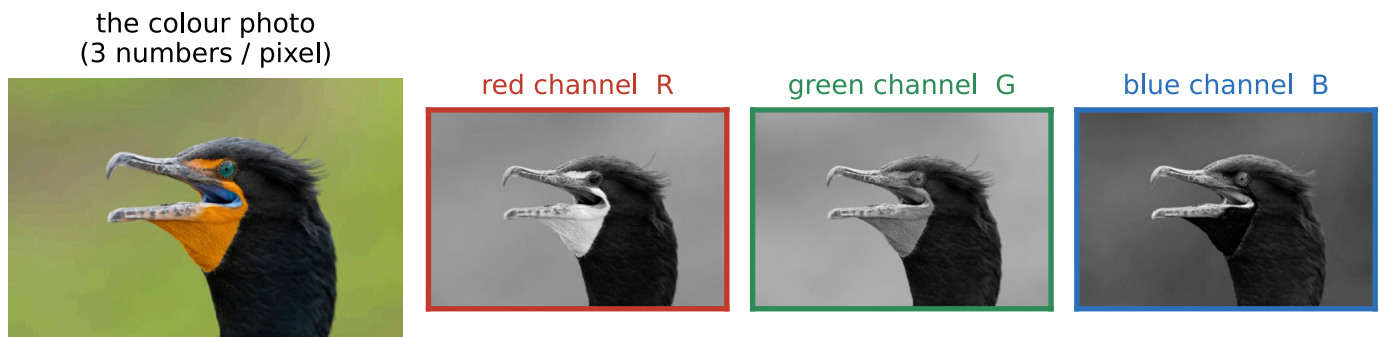


Figure 8.1.3 (reproduced). A color photograph is three grayscale planes. The color image (left) holds one number per pixel for red, one for green, one for blue; shown separately (right, as the grayscale each really is), each plane is bright where that channel is strong. The orange throat lights up the red plane, the teal eye and beak the green and blue, the foliage the green. Stack the three planes back up and the color photo returns.

Usually a pixel carries exactly three channels, but nothing forces that. Some images add a fourth (for transparency, which we return to under alpha below) and scientific hyperspectral images keep dozens or hundreds of narrow wavelength bands instead of three broad ones. One very practical trap follows from this freedom, and it bites beginners constantly: a Portable Network Graphics (PNG) file may carry an alpha channel, so a file you *assume* is three-channel RGB hands you four channels instead. If the rest of your code expects three, every pixel downstream is misread. The fix is trivial once you know to look (check the channel count on load), but the symptom (colors scrambled, code not crashing) is hard to diagnose if you do not.

Sidebar 2: why floats in $[0, 1]$?

We use float32 for working images. Zero and one serve as black and reference white in normalized examples, but intermediate values may lie outside that interval. Changing an integer array to float changes its numeric representation; it does not decode its transfer function or identify its color space.

Float vs 8-bit vs more bits

[in the book](#) ➤

Use floating point for intermediate arithmetic. It provides more precision and headroom than 8-bit storage, but still rounds and can overflow. In normalized examples, 0 denotes black and 1 a chosen white level. Values above 1 can retain highlights; negative values can arise from noise after black subtraction, color transforms, or filters. Clip only where the next operation or output representation requires it.

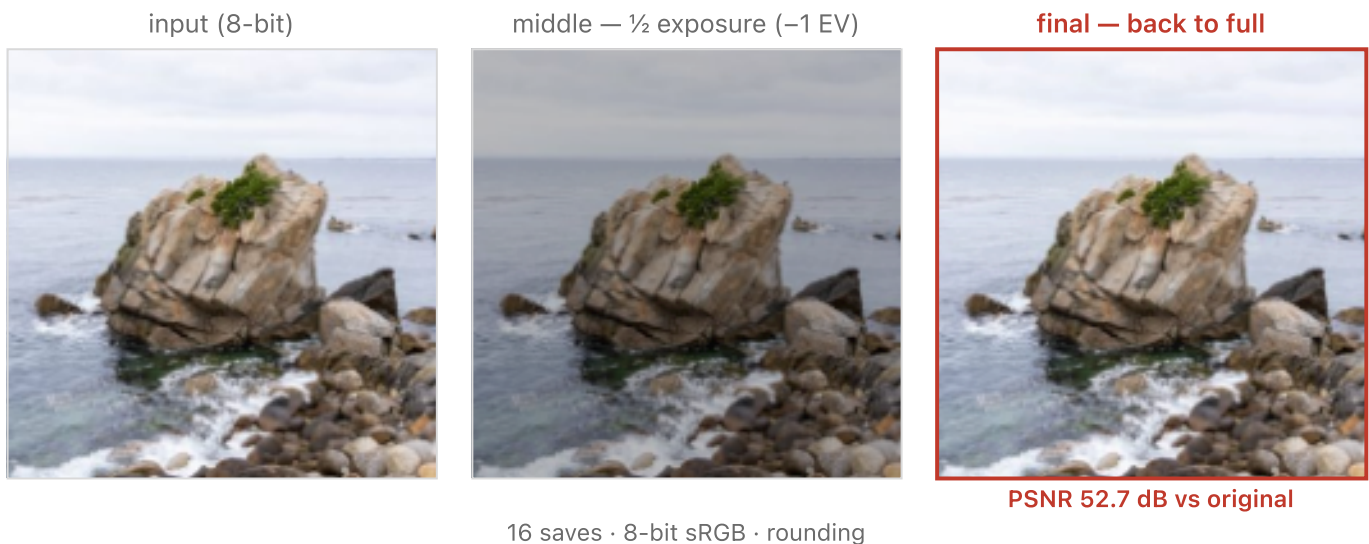


Figure 8.1.4 (reproduced). *Why 8-bit arithmetic accumulates error. Starting from an 8-bit encoded image, darken it by the chosen number of stops in N equal linear-light steps, then brighten it back in N more. Each step is exact in float (decode, scale, re-encode) but is then quantized to the chosen bit depth and clipped, so every step rounds; an encoding switch quantizes linear values instead. Back at the original exposure the image can show banding or a tonal bias, and the PSNR against the starting image measures it. Larger N does not necessarily mean more damage, because a small enough step can round straight back to the same code. Random dither and error diffusion trade structured bands for noisier, less visible error, but neither recovers what rounding discarded. The dependable fix is the chapter's: keep precision in float and quantize once, at the end. A smooth gradient such as a sky shows the bands most clearly.*

Repeated exposure changes followed by quantization are a stress test, not a model of every edit. The result depends on the step size, encoding, rounding, and clipping. More iterations need not monotonically worsen the image:

sufficiently small changes can round back to the same codes. Gamma encoding allocates finer linear-light steps to shadows, but there is no universal equivalence between 8 encoded bits and a fixed number of linear bits.

Reminder



L2.47 [show](#): when the quantization step is small relative to sensor noise, additional capture bits add little information. Coarse quantization, strong edits, and smooth synthetic gradients can still reveal banding. See [Chapter 2.8, Noise, signal-to-noise ratio and dynamic range](#).

Three kinds of operation

in the book [↗](#)

The domain/range split we just drew also sorts everything we will *do* to an image into three families, each the basis of one of the next several chapters ([Figure 8.1.13](#)):

- **point operations** touch the range only: a pixel's new value depends solely on its old value, the same rule applied everywhere (brightness, contrast, tone curves, gamma);
- **domain operations** change spatial coordinates; interpolation may also change values: each output pixel pulls its value from a *different* input location (translation, rotation, resizing, lens correction);
- **neighborhood operations** compute each pixel from a *window* of its neighbors (blur, sharpen, edge detection).

This is a preview; each family gets its own development later (point operations come first). But the taxonomy already earns its keep, because it predicts cost: a point operation reads each pixel once, while a neighborhood operation reads many input pixels per output pixel. We will flag which family every new tool belongs to as we go.

Three kinds of image operation: which input pixels set an output pixel

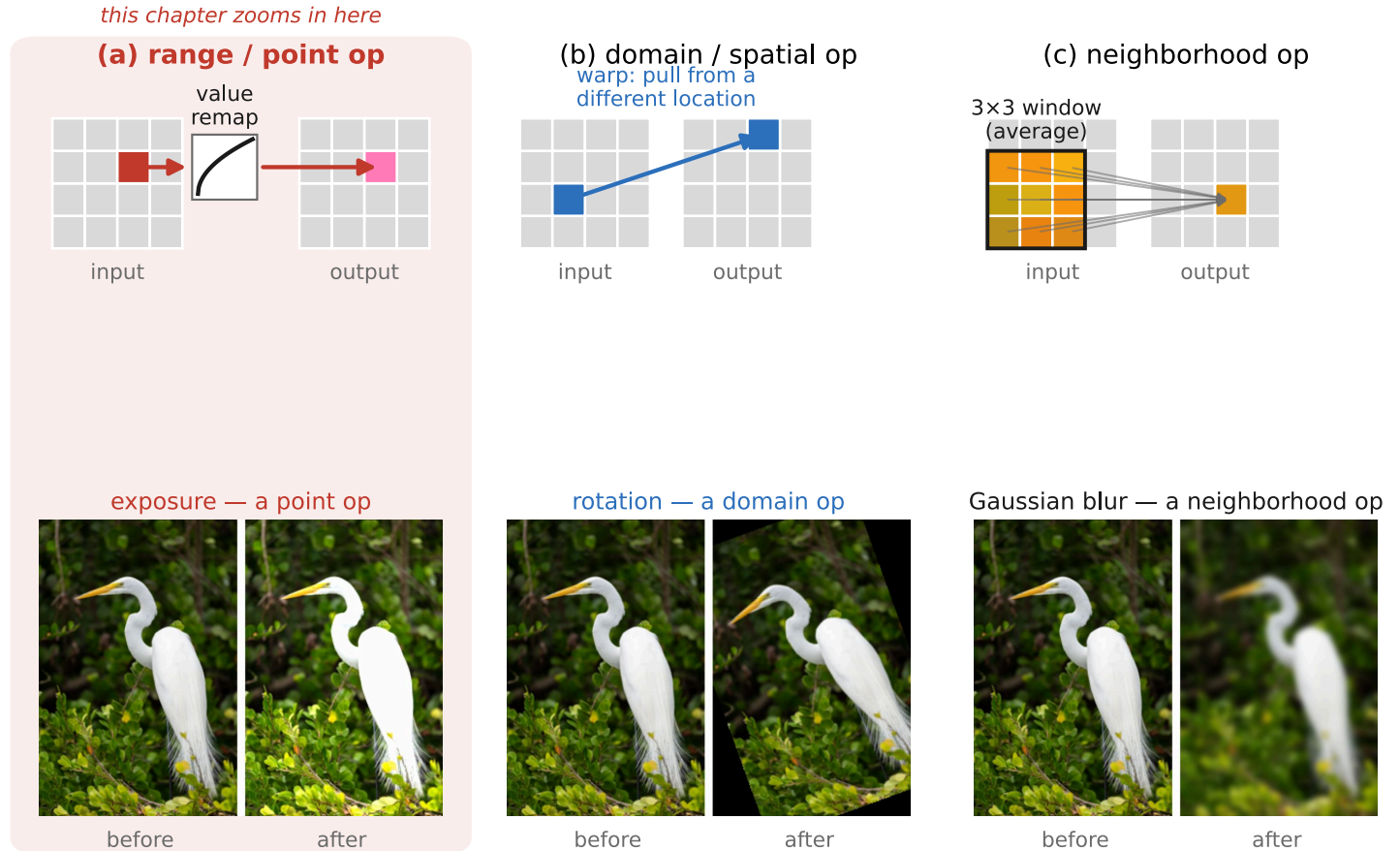


Figure 8.1.13 (reproduced). The three kinds of operation, classified by which input pixels determine an output pixel. (a) Range / point: the output at a location depends only on the input at the same location, through a value-remapping curve (brightness, gamma, levels). (b) Domain / spatial: the output pixel is fed from a different input location, a geometric warp or resample (translation, rotation, lens correction). (c) Neighborhood: the output pixel is a function of a window of input pixels, e.g. a 3×3 convolution (blur, sharpen, edge detection).

What the numbers mean

[in the book](#) ➤

Images usually encode a pixel as a triplet of numbers, but the exact meaning of those numbers can vary significantly. The numbers usually stand for photometric quantities (think of the number of photons received at a photosite), but they may be encoded non-linearly. And different images may use different color spaces, and even RGB comes in many variants. A large share of image-processing bugs ("why are my colors wrong, my blend dark, my HDR muddy?") trace straight back to encoding and color-space mismatches.

💡 BIG LESSON (L4.1) — PIXEL NUMBERS ARE MEANINGLESS ON THEIR OWN

The numbers in an image array carry no meaning by themselves. The same triple (0.5, 0.2, 0.1) is a different color, and a different amount of light, depending on two things the array does not contain:

- its **color space**: which primaries and white point those R, G, B refer to (sRGB? Adobe RGB? Display P3? a camera's native raw space?); and
- its **value encoding**: how a stored number maps to light (**linear**, **gamma**, **log**, or a camera "look" curve).

Strip those two away and you are left with a grid of anonymous numbers. Every operation that matters later (white balance, HDR merging, blur and deblur, any color conversion) silently assumes a particular answer to both questions, and gets the wrong result if the assumption is wrong. Hence the discipline of the whole book: never touch a pixel without knowing its color space and its value encoding. These are two common causes of otherwise puzzling image differences.

What the numbers actually mean: encoding

[in the book](#) ↗

💡 BIG LESSON (RECURRENCE OF (L4.6) AND (L4.4)) — MIND THE ENCODING

Use the domain expected by the operation. Decode the transfer function before linear-light arithmetic, and encode appropriately for display. Decoding a rendered file does not undo tone mapping or restore scene-linear measurements.

The same stored value can represent different light levels under different transfer functions. This matters immediately when averaging two images or adjusting exposure. Use the space required by the operation, and convert deliberately:

- Linear values can be proportional to scene light or to display light. Physical light mixing uses linear values in the relevant domain. A model-aware denoiser or appearance filter may use another representation.
- Gamma-like encodings allocate more code values to dark linear-light intervals. They are useful for storage and display interfaces, but are not exactly perceptually uniform.
- Log encoding represents multiplicative exposure changes as additive offsets. Scaling log contrast can compress a range; subtracting a constant alone shifts exposure without reducing the ratio between positive endpoints.

See [Chapter 4.3, Encoding: linear, gamma or log](#).

A deblurring model must account for the domain in which the physical blur occurred ([Figure 6.8.1](#)). An encoded-domain solver can be valid if it includes the nonlinear encoding in its forward model. Poisson blending instead chooses a gradient-domain appearance objective, which can legitimately differ across spaces ([Figure 8.2.1](#)).



Figure 6.8.1 (reproduced). Deblurring belongs in linear light. A sharp image is blurred in linear light (a true average of light), then the blur is inverted two ways: on the linear values (the right model) and on the gamma-encoded values (the wrong model, with halos and lost contrast, because gamma and blur do not commute). Both use the same finite number of iterations, so neither recovers the source exactly; at the default settings the linear result scores 28.9 dB and the gamma one 26.0 dB, and the ordering can depend on the source and the blur. Same operation, right and wrong encoding. Pick a real photo (the default) or a synthetic grating (set its contrast and period) that dips into the shadows, where the gamma curve bends hardest and the error is largest; on a photo the gap is not always visually obvious, since most local neighborhoods span a narrow tonal range where gamma is nearly linear. Even where the PSNR is clearly worse, it is the grating that makes the error plain. Developed fully as an inverse problem in [Chapter 7.1, Imaging as Bayesian inference with probabilistic priors](#).



Figure 8.2.1 ([reproduced](#)). The same seamless clone, solved in three color encodings. One composite (a dolphin cloned into a pool) run through the identical Poisson solve in linear, gamma, and Chong's illumination-invariant log space ([Chong et al. 2008](#)), then converted back to sRGB for display, so the only difference between the panels is the space the arithmetic ran in. The re-leveling that hides the seam behaves differently in each: linear leaves the pasted dolphin a garish purple, gamma is milder, and log lands closest to a natural gray. A forward reference; the interactive version and the full story are in [Chapter 10.8, Poisson image editing](#).

There is a further wrinkle: the values in a finished JPEG, or a scanned film frame, are usually *not even a clean gamma encoding of scene light*. The camera's processor (or the film's chemistry) has already applied tone curves, saturation boosts, and contrast S-curves to make the picture *look good* straight out of the camera: beautification, not measurement. So a consumer JPEG is roughly: scene light $\rightarrow$ camera "look" $\rightarrow$ gamma $\rightarrow$ 8-bit, with the physical meaning of the numbers scrambled along the way.

State the encoding and color space at every stage. Decoding sRGB recovers display-linear RGB; it does not undo a camera's tone mapping to recover scene radiance. Raw development also needs black-level, white-level, and camera-color calibration.

Store encoding, color space, range, and alpha convention with the array. A conversion must update both the numbers and their description. Relabeling an sRGB array as linear does not decode it, and an integer-to-float cast does not perform a color conversion. Validate operation inputs and make conversions explicit when intent or information loss is involved.

[Figure 8.2.2](#) distinguishes a deliberate look from an encoding mistake. A neutral or punchy rendition can be intentional; double encoding is a different error. [Chapter 45.15, Rendering a raw DNG — from sensor numbers to a faithful image](#) develops the raw-data interpretation in detail.

fig-numbers-mean-versions: one set of pixel values, three defensible-looking images

Nothing in the array says which of these it is. Only your bookkeeping does.

1 scene-linear, sRGB-encoded
radiometrically correct, and dull



2 wrong encoding
the same numbers, gamma applied twice



3 + the beauty pass
what a viewer expects to see (a macaw against foliage)



1 scene-linear, sRGB-encoded
radiometrically correct, and dull



2 wrong encoding
the same numbers, gamma applied twice



3 + the beauty pass
what a viewer expects to see (a hot spring)



Figure 8.2.2 (reproduced). One set of pixel values, three defensible-looking images. From a single raw DNG: (1) the scene-linear data correctly encoded to sRGB, radiometrically right but dull, because it carries no camera tone curve; (2) the same numbers with the encoding mishandled (gamma applied twice), milky and washed; (3) the correct render plus the beauty pass, what a viewer expects. None is "the" picture; each is a different, defensible reading of the same array, which is the whole point.

A black-level error can lift shadows and introduce a color cast before any tone adjustment (Figure 8.2.3). Subtract the calibrated offset in the same units as the decoded raw samples. An image histogram with no values near zero is a diagnostic clue, not a measurement of black level: the scene may contain no dark surface, and flare can add real light. In this book's own case (the figure), the file declared a black level of zero while nothing in the data sat below about 0.18 of full scale, with the highest floor in the green channel, which is where the cast came from. Prefer trusted calibration, masked sensor pixels, or matched dark frames to subtracting the image minimum.

Black level: a sensor calibration, not a number you can read off the picture

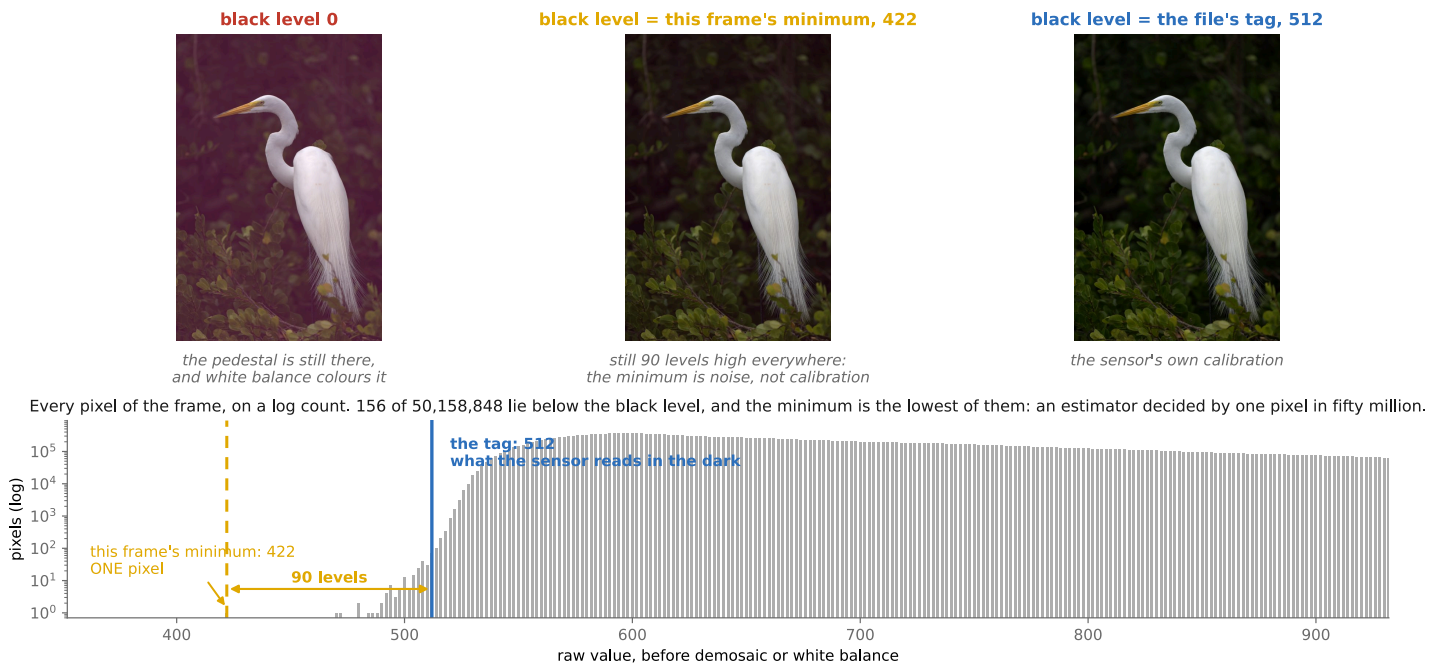


Figure 8.2.3 (reproduced). A file that lies about its numbers. The same DNG rendered twice by the same pipeline; the only difference is the black level. Left: the level taken at the file's word (zero), so the sensor's pedestal stays in the data, every value is lifted by roughly 0.18, and the shadows cannot reach black; the green channel's larger pedestal tints the whole frame. Right: the per-channel pedestal measured from the data and subtracted. A real bug from building this book's raw developer, caught because the histogram had no left half.

A useful programming analogy is to make the image representation part of its type. A buffer tagged as linear sRGB and a buffer tagged as encoded Display P3 should not be interchangeable merely because both contain three floats per pixel. A physical-light blend could require linear inputs; a color conversion could declare its source primaries and reference white. The conversion then becomes an explicit operation rather than an assumption hidden in a function name.

Such a type system would not choose the intended rendering or infer missing metadata, but it could catch a large class of mistakes: adding images with different representations, applying a transfer function twice, or treating a log value as a linear intensity. In ordinary array code, carry those assumptions as metadata and check them at boundaries. That is manual type checking for the image pipeline.

The black-level example came from a practical failure in this book's raw developer. A dusk photograph initially appeared milky with a green cast despite otherwise plausible processing. The supplied zero black-level value was accepted without checking whether it described the decoded samples. An offset mismatch early in the pipeline can survive every later stage and look like a poor tone or white-balance choice. The example motivates checking the decoder's units and offset handling, together with calibration; the minimum value in that photograph alone cannot prove the correct subtraction.

Tone point operations

Problem set

PS1 implements brightness/contrast, gamma, and white balance as point operations. → [Chapter 1.5, Problem sets](#) (appendix).

The simplest image-editing operations are independent at each pixel: each output pixel depends only on the *value* of the input pixel at the very same spot, with no reference to its neighbors or to where it sits. These are point operations, and they cover a remarkable fraction of the edits photographers actually make: exposure, brightness, contrast, levels, freehand tone curves, gamma, and the lookup tables that bake them in. This chapter is about the *tonal* half of that family: what they can and cannot do, and the single question that decides whether each is done right or wrong, namely the numerical space it acts in. The *color* half, saturation and vibrance, black-and-white conversion, and video color grading, is the companion chapter [Chapter 8.5, Color point operations](#).

Two framing cautions before the toolkit. First, these sliders are not independent: brightness, exposure, contrast, levels, and the tone curve all bend the *same* transfer line, so they overlap and partly undo one another; the general tone curve at the end of the chapter literally subsumes the rest, and any given look can be reached several ways. Second, they are not all equally meaningful: a few are objective, radiometric operations faithful to how light behaves (exposure is a multiply in linear light), while most are stylistic (brightness, contrast, the S-curve) with no single physically correct form, only a choice of space and taste. Knowing which kind you are using tells you whether "which space?" has a right answer (radiometric: yes) or merely a pleasant one (stylistic: pick the look).

Three kinds of operation

[in the book](#) ➤

Recall the distinction between changing values, remapping coordinates, and combining neighboring samples ([Figure 8.1.13](#); [Chapter 8.1, Image representation](#)). These categories describe useful building blocks, not every possible image algorithm.

A point operation touches the range only. The output pixel depends solely on the input pixel's value at the *same* location, and one and the same rule is used everywhere:

$$I_{\text{out}}(x, y) = f(I_{\text{in}}(x, y)). \quad (13)$$

There is a single input-to-output curve f , applied identically at every pixel. In words: read a pixel's value, find where that value lands on the curve, write the answer back: no neighbors consulted, nothing moved. Brightness, contrast, exposure, gamma, and lookup tables are all point operations, and we begin here because they are the simplest to reason about and the cheapest to run.

A domain operation maps output coordinates to input coordinates, $I_{\text{out}}(x, y) = I_{\text{in}}(x', y')$. For noninteger positions, interpolation combines neighboring samples and can change their values; see [Chapter 6.5, Resampling and upsampling](#).

A neighborhood operation sits in between: the output pixel is computed from a whole *window* of input pixels around the same spot. Blur averages a pixel with its neighbors; sharpening, edge detection, and the bilateral filter

all read a neighborhood. Those get their own chapter on [Chapter 6.2, Convolution](#).

Three kinds of image operation: which input pixels set an output pixel

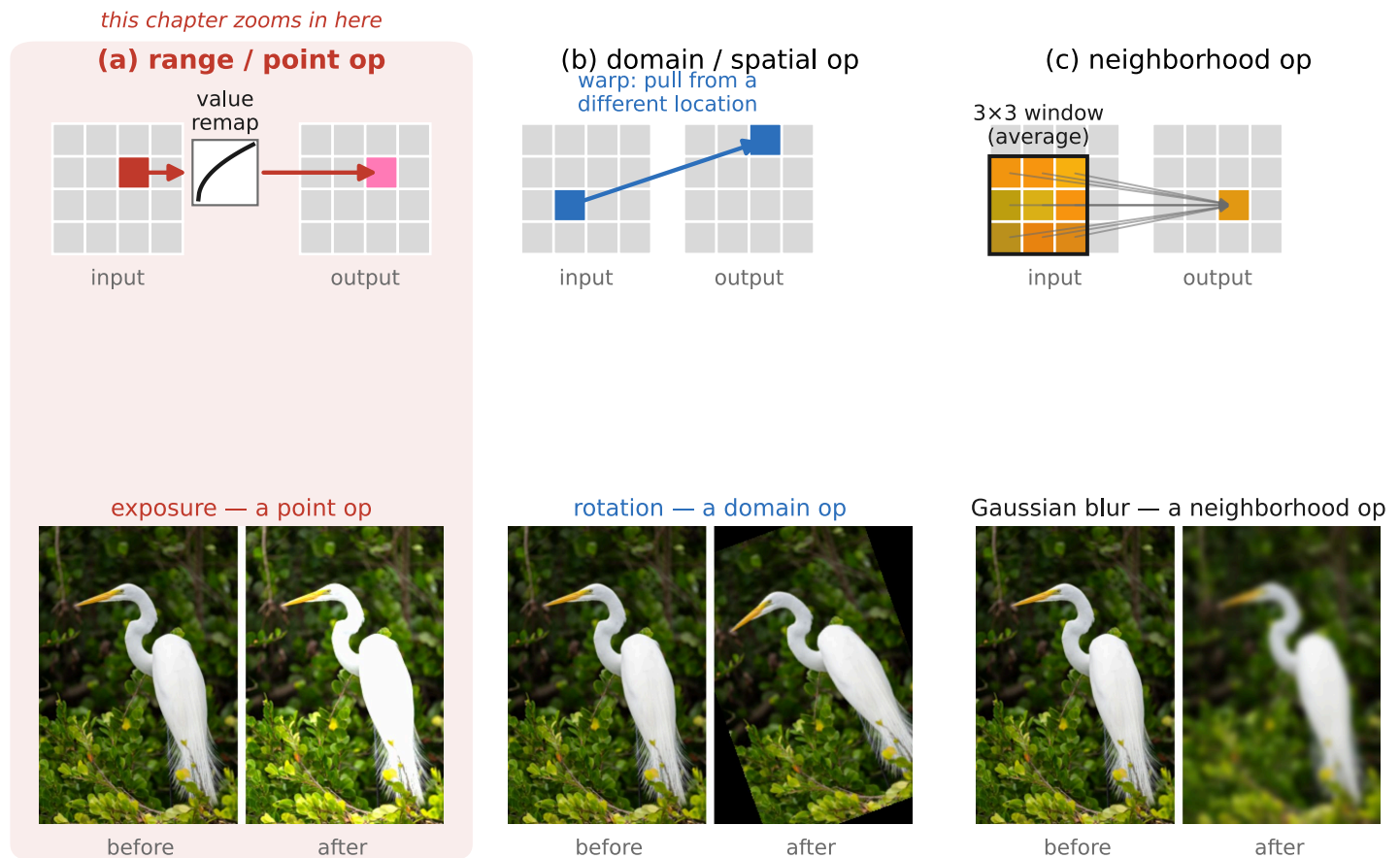


Figure 8.1.13 (reproduced). The three families of image operation, sorted by what each output pixel may read. Point / range (left): the output depends only on the input pixel's value at the same location: one curve applied everywhere (exposure, contrast, lookup tables). Domain (center): pixels are relocated, their colors unchanged (resize, rotate, warp). Neighborhood (right): the output is computed from a window of nearby input pixels (blur, sharpen, bilateral). This chapter is the left column.

We develop tonal curves on a scalar value L . Applying a nonlinear curve independently to RGB channels can change hue and saturation. Scaling the channels using a mapped luminance behaves differently; [Chapter 8.5, Color point operations](#) treats this choice.

The point-operation curve

[in the book](#) ➤

Since a point operation is just one curve f applied to every pixel, *the curve is the operation*, and photo editors hand it to you directly: the curves tool plots input value across the horizontal axis, output value up the vertical, and lets you bend the line by hand. The "do nothing" case (leave every value untouched) is the 45° diagonal. Everything

in this chapter is some way of bending that diagonal: lift it to brighten, steepen it to add contrast, pin its ends to set black and white points, or just drag it freehand.

Controlling that mapping is not a digital invention. Photography has *always* been the craft of deciding how scene radiance becomes printed tone (film stock, exposure, development, paper grade, and the enlarger all shape it) and the digital tone curve is the most direct handle on the same problem. There is, though, one question we must answer before we bend anything, and it runs through the whole chapter: in which numerical space does the curve act?

Sidebar 4: the Zone System

The Zone System connects scene exposure placement to negative development and print tones. Exposure placements are one stop apart; the printed zones do not form a sequence of doubling luminances. [Chapter 18.2, Global tone mapping](#) distinguishes these two uses of the zone scale.

The same image can be stored as linear-light values (proportional to the physical radiance the sensor measured), as gamma/sRGB-encoded values (the perceptually spaced numbers a PNG actually holds), or as log values (handy when the range is enormous). We met these encodings in [Chapter 8.1, Image representation](#); here they stop being a storage detail and start changing the *result*. Apply the very same arithmetic in linear light versus in sRGB and you get visibly different pixels, so every operation below carries a note about the space it belongs in.

💡 BIG LESSON (L4.6) — ADDITIVE VS. MULTIPLICATIVE → CHOICE OF ENCODING

Exposure multiplies linear-light values; independent contributions of light add. Log coordinates turn exposure ratios into differences. An offset in encoded values is a distinct appearance control. Choose the domain from the operation and its model.

Exposure: a multiply in linear light

[in the book](#) ➤

An exposure adjustment scales linear-light values. During capture, a longer shutter or larger aperture can collect more light; raising ISO changes the camera's gain or rendering convention and does not itself add photons. A software exposure change scales an existing measurement without improving its capture SNR or recovering clipped data. For s stops, the multiplier is 2^s :

$$L_{\text{out}} = 2^s L_{\text{in}}. \quad (14)$$

If E denotes the output encoding, the displayed curve is $E(2^s E^{-1}(V))$. Its shape depends on E : a pure power law gives a scaled straight line before clipping, while sRGB also has a toe and offset. Values above display white can be retained in the working image even when the preview clips them ([Figure 8.3.2](#)).

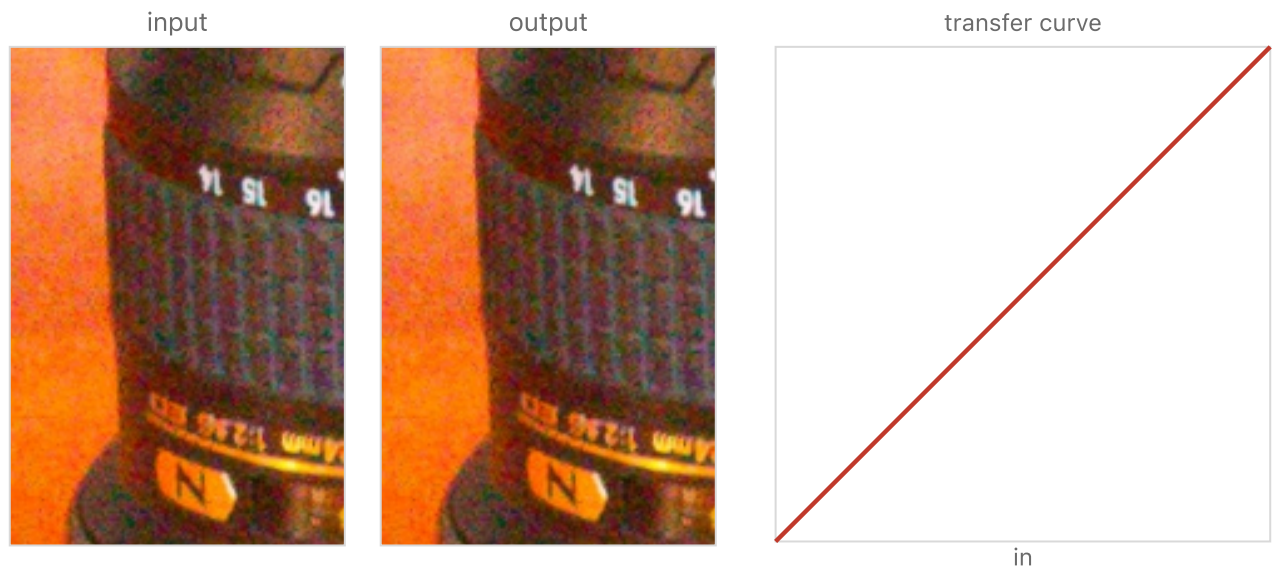


Figure 8.3.1 (reproduced). A point-operation playground. Every pixel is mapped through the same function, independent of its neighbors: exposure, brightness, contrast, gamma, saturation, threshold, and invert all compose into the single input→output transfer curve shown at right. Adjust the controls (or load your own image) to watch the curve and the result change together; the rest of this section covers these one at a time.

Exposure: multiply by 2^{stops} in linear light, then re-encode



Figure 8.3.2 (reproduced). Exposure as a multiply in linear light. Left, the input. Center, the same photo at +1 stop ($\times 2$ in linear light), brighter throughout, with the brightest highlights clipped to white. Right, the transfer curve $L_{\text{out}} = 2^s L_{\text{in}}$ drawn in display (sRGB) space: it bows above the 45° diagonal, lifting the shadows strongly and flattening into the ceiling at 1. The 45° diagonal (no change) is shown for reference.

The "in linear light" qualifier is not pedantry. It marks the difference between a correct exposure change and a wrong one. The library makes the distinction concrete: `imageops.exposure(img, stops)` decodes to linear light, multiplies, then re-encodes; ask it for the naive version instead and it multiplies the gamma-encoded numbers directly, which is exactly what happens when you forget to linearize. The two part ways sharply, and the next figure is devoted to showing how.

💡 **BIG LESSON (L8.3) — EXPOSURE SCALES LINEAR LIGHT (A MULTIPLY) OR SHIFTS LOG VALUES (AN ADD)**

Exposure scaling multiplies linear-light values by $k = 2^s$ or adds s in base-2 log units. For a pure gamma encoding $V = L^{1/\gamma}$ it multiplies V by $k^{1/\gamma}$. For an arbitrary transfer function, decode, scale, and encode. Output clipping is a separate operation.

Exposure +1 stop — in linear light (correct) vs naively on the gamma values

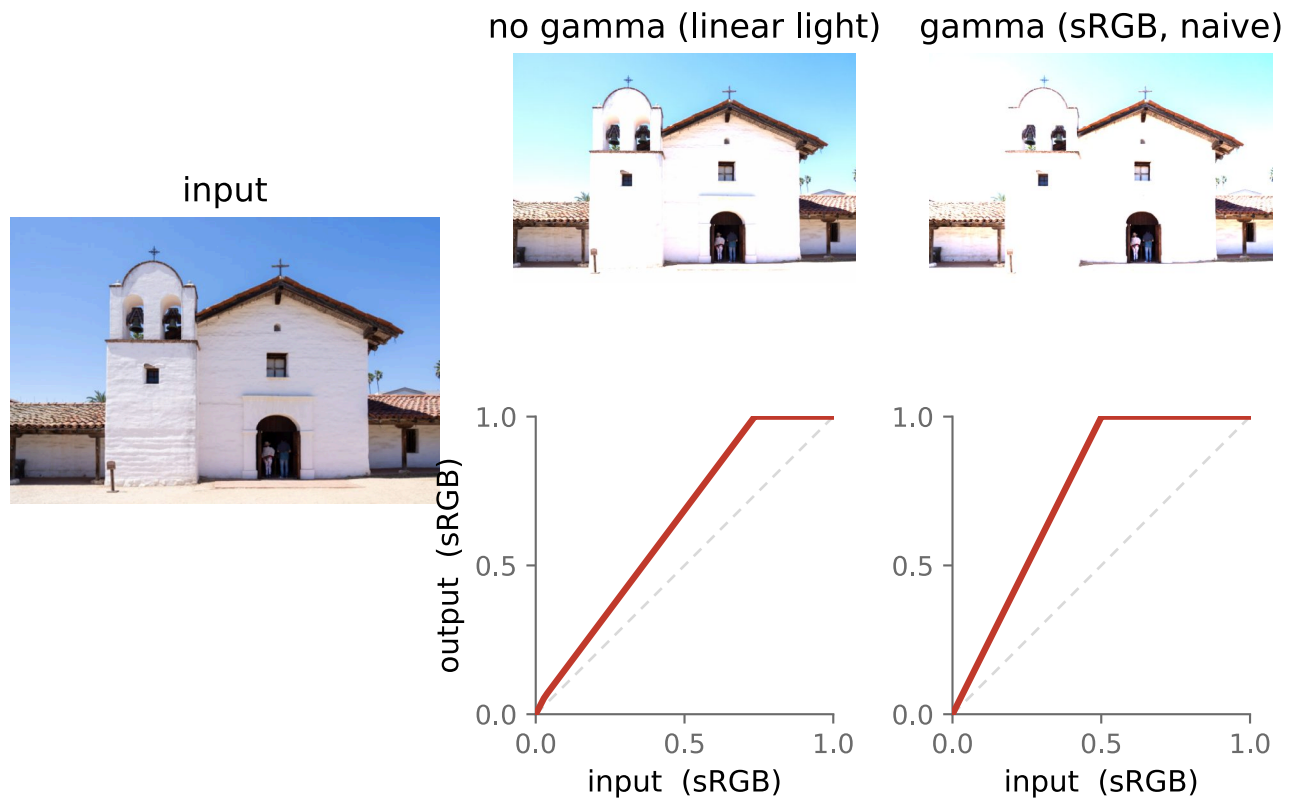


Figure 8.3.3 (reproduced). Why the space matters for exposure. The same "+1 stop" done two ways. Top, the correct way, multiply in linear light (decode $\rightarrow \times 2 \rightarrow$ re-encode): tones brighten smoothly and in proportion. Bottom, the naive way, multiply the gamma/sRGB numbers directly: the curve is a straight ramp of slope 2 that clips everything above input $1/2$ to white, blowing out the entire upper half of the tonal range. Same intended edit, very different pixels; exposure belongs in linear light.

Doing it in the encoded domain, correctly. Sometimes you cannot leave the encoded values (an 8-bit buffer, a shader handed display-referred pixels), so it pays to know exactly what a stop costs there. For a **pure gamma** curve $V = L^{1/\gamma}$ the decode-scale-encode round trip collapses to a single multiply,

$$V_{\text{out}} = (2^s V_{\text{in}}^\gamma)^{1/\gamma} = 2^{s/\gamma} V_{\text{in}}, \quad (15)$$

so an s -stop change on gamma-encoded values is just multiplication by $2^{s/\gamma}$ (at $\gamma = 2.2$ a stop is $\times 2^{1/2.2} \approx 1.37$, not $\times 2$). The instant the transfer function is not a pure power the shortcut breaks: real sRGB carries a linear toe and an offset power segment, so there is no closed-form multiply and you must decode, scale, and re-encode,

$$V_{\text{out}} = \text{OETF}_{\text{sRGB}}(2^s \cdot \text{EOTF}_{\text{sRGB}}(V_{\text{in}})). \quad (16)$$

The examples below accept input values in $[0, 1]$. Positive exposure adjustments can produce outputs above 1; the code preserves them instead of clipping them:

Listing 3 · exposure on gamma-encoded values

[link](#)[copy](#)[embed](#)

Python

C++

Halide

```
def expose_gamma(V, stops, gamma=2.2):      # pure-gamma shortcut: one multiply
    return (2.0 ** (stops / gamma)) * V      # == ((2**stops) * V**gamma) ** (1/gamma)

def expose_srgb(V, stops):                  # real sRGB: decode, scale, re-encode
    lin = V / 12.92 if V <= 0.04045 else ((V + 0.055) / 1.055) ** 2.4
    lin *= 2.0 ** stops
    return 12.92 * lin if lin <= 0.0031308 else 1.055 * lin ** (1 / 2.4) - 0.055
```

Use the power-law shortcut only when that is the actual transfer function. Use the piecewise sRGB formula for sRGB, and the specified inverse/forward functions for other encodings.

Contrast: steepen about a pivot

OPTIONAL

in the book [↗](#)

Where exposure moves all the tones the same way, contrast spreads them apart or pulls them together. The idea is to pick a pivot value, a mid-gray that stays put, and push every other value *away* from it (more contrast) or *toward* it (less). Values above the pivot get brighter, values below get darker, and the pivot itself never moves. As a curve, that is a straight line through the pivot whose slope is the contrast gain g :

$$L_{\text{out}} = g(L_{\text{in}} - p) + p, \quad (17)$$

where p is the pivot. Read it as: measure how far a value sits from the pivot, scale that distance by the gain, add it back. A gain above 1 steepens the line and raises contrast; a gain below 1 flattens it and lowers contrast. The steeper the line, the sooner it slams into the floor at 0 and the ceiling at 1, so strong contrast clips both shadows and highlights ([Figure 8.3.5](#)).

💡 **BIG LESSON (L8.4) — CONTRAST IS A PIVOT: CHOOSE WHAT STAYS PUT**

A contrast change rotates values about a chosen pivot: $L_{\text{out}} = p + g(L_{\text{in}} - p)$. The pivot p is exactly what is left unchanged, and it also decides whether the picture gets brighter or darker overall: a high pivot pulls most of the image down (darkening), a low pivot pushes it up (brightening). Unlike exposure, contrast is not radiometrically meaningful, it is a stylistic stretch, so the *encoding matters less* (no single space is "correct"); what matters is a sensible pivot, and the sensible pivot *does* depend on the space: roughly 18% mid-gray in linear light, the numeric midpoint in gamma or log. Pick the pivot deliberately: it is the whole character of the operation.

Sidebar 5: 18% gray, and the space it lives in

A useful example pivot is 0.18 in normalized linear light, which encodes to about 0.461 in sRGB. A surface's 18% reflectance is not automatically a sensor value of 0.18: illumination and exposure determine its reading. Meter calibration is not universally an 18% target. To compare contrast operations, map the same chosen pivot into each working space.

Contrast: steepen the tone curve about a mid-gray pivot (here in sRGB)

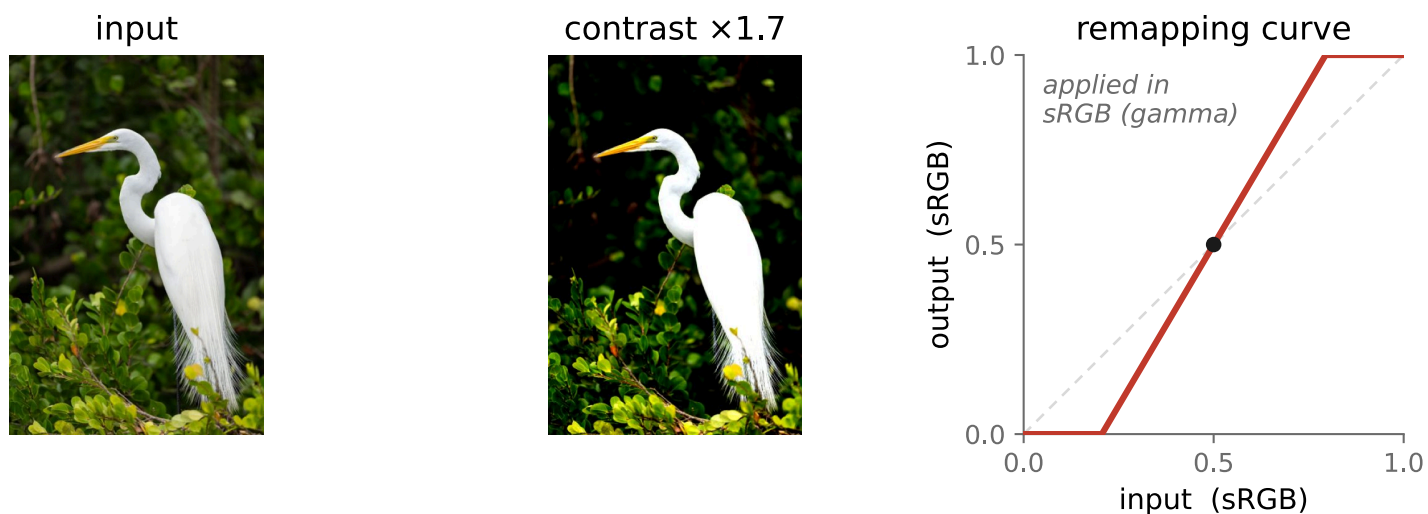


Figure 8.3.5 (reproduced). Contrast as a steepening about a pivot. Left, the input; center, increased contrast (gain > 1): shadows deepen, highlights brighten, midtones near the pivot barely move; right, the transfer curve, a straight line through the mid-gray pivot p whose slope is the gain. Steepening drives the line into the floor (0) and ceiling (1), clipping the darkest and lightest tones. The 45° diagonal (gain 1, no change) is shown for reference.

The pivot and the domain both affect contrast. A linear-domain adjustment preserves its chosen linear pivot; a log-domain scale changes luminance ratios by a power. An encoded-domain adjustment acts on displayed code

differences. Choose the appearance you want and inspect the clipping at both ends ([Figure 8.3.6](#)). On the photographs in the figure, the three give recognizably different looks. Steepening the encoded values, what most editors do, pivots about a perceptual mid-gray and roughly tracks how the eye judges contrast. Steepening linear values weights the highlights more and, on these images, reads harsher, with shadows crushed faster. Steepening log values stretches ratios uniformly and reads gentlest across a wide range. None is wrong; they are three looks, and an editor can offer the choice.

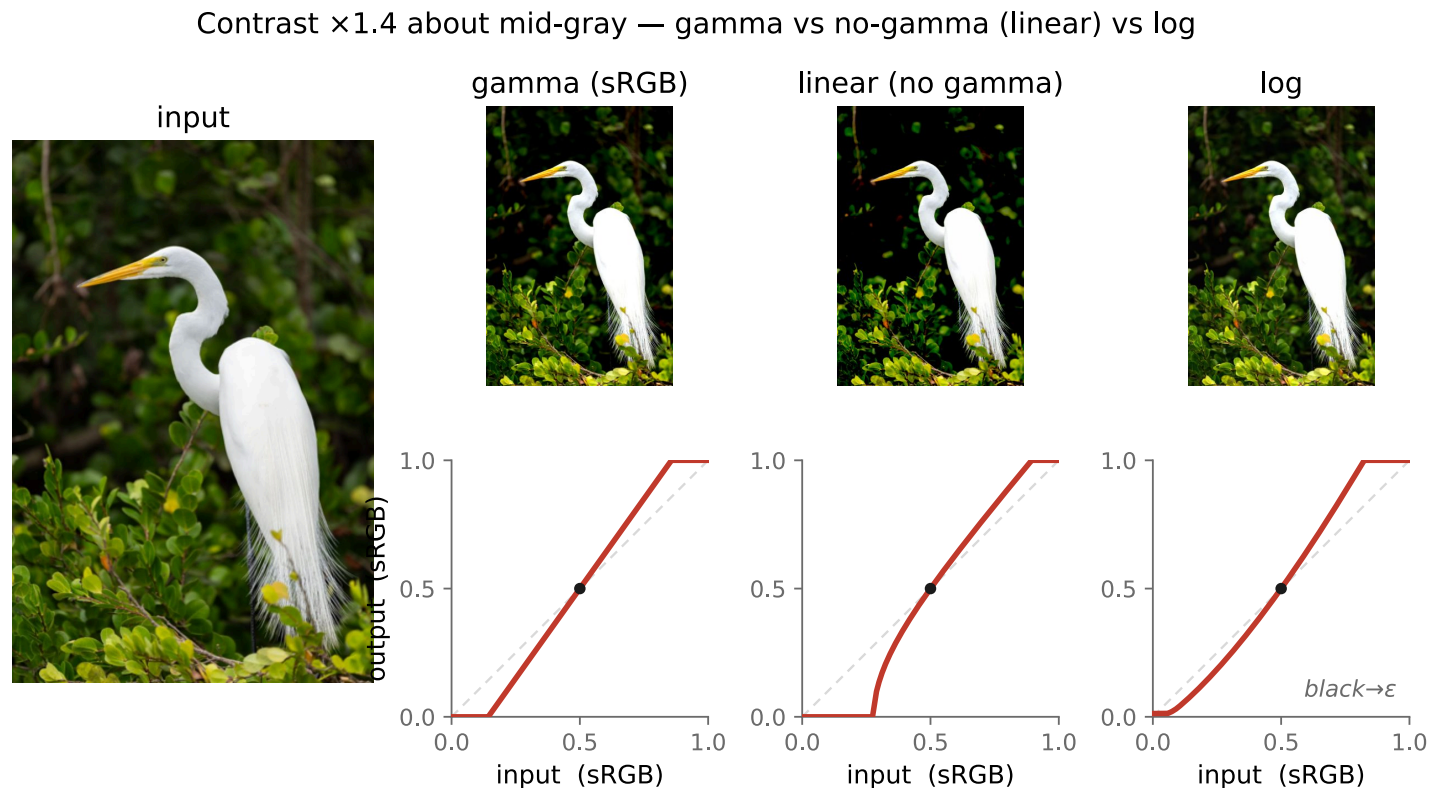


Figure 8.3.6 ([reproduced](#)). The same contrast boost in three spaces. The identical gain about a mid-gray pivot, applied in sRGB/gamma (perceptual mid-gray, the usual editor behavior), in linear light (acting on physical radiance, harsher in the highlights, crushes the shadows), and in log/density (a uniform stretch in ratios, gentlest across a wide range). For the log version the black is clamped to a small ϵ to avoid $\log(0)$. Each panel shows the output and its transfer curve; the look differs because the pivot and the spacing of tones differ between spaces.

Sidebar 6: the S-curve

A straight contrast line clips abruptly the moment it touches 0 or 1. Photographers usually prefer a sigmoid (S-shaped) tone curve instead: steep through the midtones (where the eye is most sensitive) but easing gently into the shadows and highlights so the extremes *roll off* rather than clip. This is exactly the shape of film's response and of a camera's built-in "look," and it is the bridge both to the general tone curve below and to the tone-mapping operators of the next chapter.

Auto-exposure and auto white balance

[in the book](#) ↗

Auto-exposure chooses capture settings and auto white balance estimates a color correction. Both depend on uncertain scene information: a dark object and an underexposed bright object may yield the same measurement, as may a colored surface and colored illumination.

Auto-exposure: metering

[in the book](#) ↗

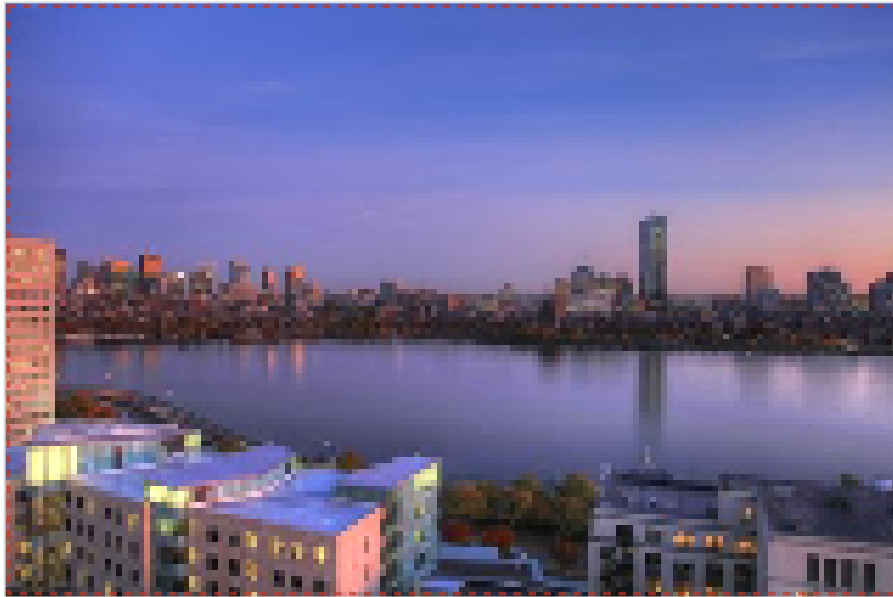
A meter estimates scene brightness, then the exposure program chooses shutter duration, aperture, and ISO within constraints. Shutter duration and aperture affect photons collected; ISO affects gain and often the selected exposure. A motion limit, depth-of-field requirement, or highlight constraint can take priority over average brightness.

Average and center-weighted metering. The simplest meter averages the whole frame to one number and exposes so that average is middle gray (see [Chapter 5.2, Exposure and capture timing](#)). It is fooled by anything that is not average: a bright sky pulls the average up, so the meter *under*-exposes and the foreground falls into shadow (the classic backlit-subject silhouette); a snow field reads too bright and is rendered a dingy gray. Center-weighted metering improves the guess by weighting the middle of the frame most, on the assumption that the subject is usually near the center.

Matrix / evaluative metering. Combine measurements from several regions, optionally using focus, face, color, or distance information. Different cameras use different decision rules. Nikon's 3D Color Matrix metering, for example, compares zone brightness and color and the subject distance reported by the lens against a database of reference scenes, which Nikon describes as tens of thousands of photographs; Canon's evaluative metering and other makers' equivalents are cousins. A subject-weighted result may deliberately sacrifice some highlights; an image-wide average does not establish the photographer's intent.

Spot metering is the opposite extreme: meter one small zone and place *that* tone at middle gray. It hands the photographer exact control over which tone is anchored (the basis of Ansel Adams's Zone System) at the cost of having to choose wisely.

exposed for the metered region



Average metering

metered tone	16.3% → 18%
exposure gain	×1.10
highlights clipped	0.4%

Bright areas pull the average up
→ small gain → subject dark.

Figure 8.10.1 [\(reproduced\)](#). Auto-exposure metering modes (interactive). Pick a scene and a metering mode and the demo scales the exposure (a multiply in linear light) so the metered region averages to $\approx 18\%$ gray, reporting the metered tone, the gain, and the fraction of highlights it clips. Average meters the whole frame: a bright sky or water drags the average up, the gain comes out small, and the subject falls dark. Center-weighted biases toward the middle but is still pulled by a big bright surround. Spot meters one small zone, moved and resized with the sliders: it places that zone at middle gray at the cost of clipping the bright background. That is correct only for a subject of average reflectance; spot-meter a white shirt and it comes out gray. Same scene, same target: only where you meter changes, and the exposure with it. Try the backlit sunset scene, then move the spot from the dark foreground to the bright sky.

Learned metering. An estimator can learn exposure preferences or use recognized subjects to weight the measurement. Burst pipelines can choose shorter frames to limit motion and saturation, then merge them to reduce noise. Exposure compensation and AE lock remain useful when the intended subject or mood differs from the automatic choice.

For video and preview, both exposure and white-balance estimates also need temporal stability. Damping small frame-to-frame changes reduces flicker, but too much damping delays adaptation after a lighting change.

A zone pattern retains information that an overall average discards. A bright upper region, a darker lower region, and a focused subject between them can suggest backlighting, whereas a uniformly bright frame may be snow or a white wall. Such evidence can guide an exposure decision, but does not settle whether the photographer wants a silhouette. Exposure compensation supplies that missing instruction. The progression from one average to spatial patterns and learned statistics adds evidence about intent without making intent observable directly.

White balance and color constancy

White balance is the harder color-estimation problem. For a simple diffuse surface, the spectrum reaching the sensor depends on the product of the illuminant spectrum and the surface reflectance, so a white shirt under tungsten light sends warm, orange-ish light to the sensor, and the very same shirt under open shade sends cool, bluish light. Yet to a human the shirt looks *white* under both, because the visual system discounts the illuminant, the color constancy of the perception chapter. A camera has no such automatic constancy; it must *estimate* the illuminant and divide it out, and that operation is white balance (Figure 8.10.2).

warm light, before: daylight balance



warm light, after: balanced on the floor tiles



open shade, before: daylight balance



open shade, after: balanced on the white fence



Figure 8.10.2 (reproduced). White balance, before and after, on two real captures. Top, an arcade under warm artificial light; bottom, a white fence in open shade. Each is shown first at the camera's daylight balance, which is what a daylight film would have recorded, and then balanced on a white surface in the scene. Before balancing, the warm scene's floor tiles read $1.53 : 1 : 0.51$ in red, green and blue, and the shaded fence $0.89 : 1 : 1.16$; after it, both read neutral. White balance is the camera's stand-in for the eye's color constancy.

A common approximation uses independent gains in a suitable linear color basis [Von Kries](#):

$$R' = k_R R, \quad G' = k_G G, \quad B' = k_B B, \quad (27)$$

These gains form a diagonal transform. A known neutral supplies their ratios, with a common scale left for exposure. Camera channels and cone-like adaptation coordinates are different bases; Bradford adaptation is not simply a diagonal scaling in arbitrary RGB. See [Chapter 4.2, White balance and white point](#) for that distinction and [Land & McCann 1971](#) for the Retinex connection.

Subtract the calibrated black level before applying sensor-channel gains. Apply them before nonlinear rendering, in the linear basis for which they were estimated. A common scalar exposure commutes with a linear color matrix, but unequal channel gains generally do not: $MD \neq DM$. A gain moved after encoding needs the conjugate operation $g(Dg^{-1}(v))$, not the same numerical multiply. White balance may precede demosaicking; its placement must match the reconstruction and color profile.

Automatic white balance

in the book [↗](#)

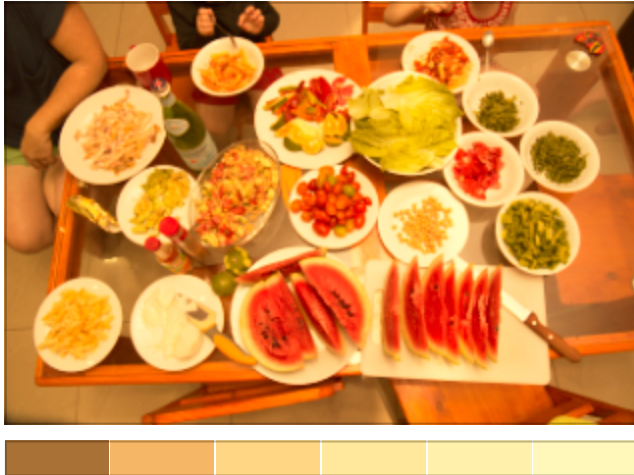
In practice the camera has no labeled neutral patch, so it must *guess* the illuminant from the image itself: automatic white balance (AWB). The classic assumptions are statistical. Gray-world [Buchsbaum 1980](#) assumes the scene averages to gray, so the channel gains are

$$k_c = \frac{\text{mean gray}}{\text{mean}_c}, \quad (29)$$

Here mean gray can be the average of the three channel means. Use only valid samples and guard near-zero channel means; normalizing the gains to a chosen reference channel separates their common exposure scale. This scales the channels so their means agree ([Figure 8.10.4](#)). The bright-pixel (white-patch) assumption ([Lee 1986](#)) instead takes the brightest pixels to be a white highlight reflecting the illuminant directly, and balances on those. Both fail in predictable ways: gray-world tips on a scene dominated by one color (a forest, a red wall), bright-pixel tips on a colored specular highlight.

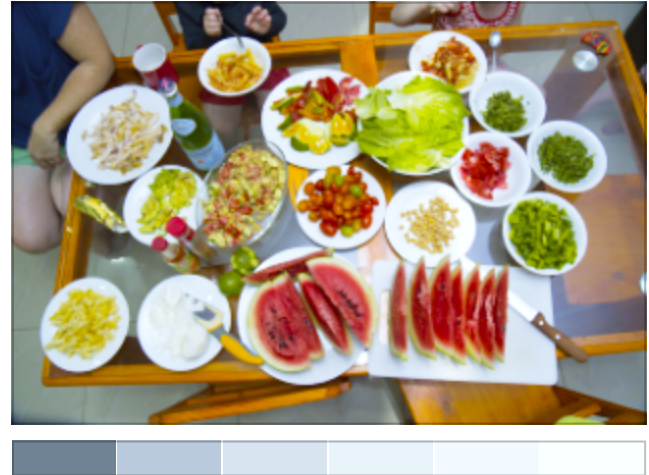
Reject saturated samples and unstable ratios near black when estimating gains. White-patch statistics can select different surfaces in different channels, and a scene need not contain a neutral highlight. Under mixed lighting, one global gain cannot neutralize every region. A neutral reference helps, but white balance cannot reconstruct wavelengths missing from the illumination.

as shot, under tungsten



a neutral wedge, as shot

balanced by gray world



the same wedge, balanced

angle to the true light

● gray world	4.9°
● brightest pixels	1.0°
● shades of gray	0.4°
● gray edge	3.2°
uncorrected:	34.5°

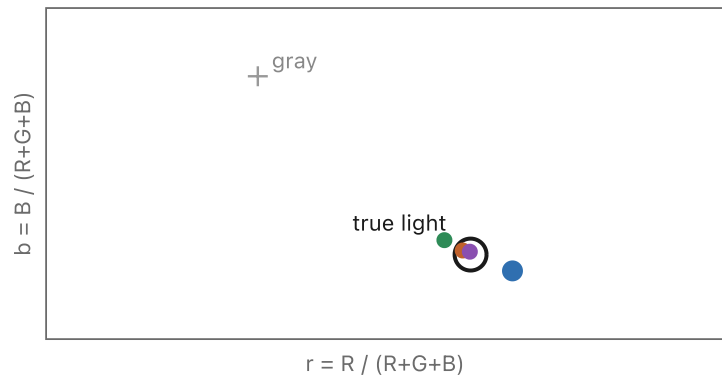
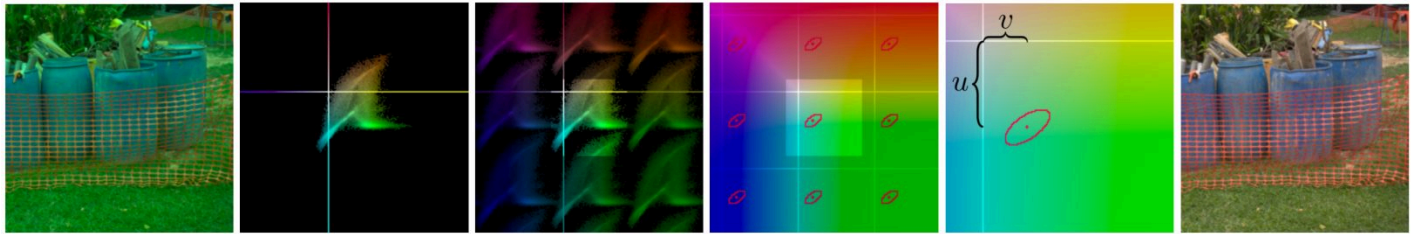


Figure 8.10.4 (reproduced). Four simple estimators on the photographs of [Figure 4.2.1](#) (interactive). Each photograph is relit under a known light, so every estimate can be scored as the angle between the estimated and the true illuminant in RGB, the usual measure in color constancy. Gray world takes the channel means; brightest pixels averages the top few percent by luminance, a white patch that tolerates noise; shades of gray (Finlayson and Trezzi, 2004) replaces the mean with a Minkowski p-norm, which is gray world at $p = 1$ and tends to the per-channel maximum as p grows; gray edge (van de Weijer, Gevers and Gijsenij, 2007) asks the same of the image's derivatives, on the assumption that edges, not surfaces, average to gray. Saturated pixels and pixels too dark to carry a ratio are left out. Over the fifteen frames at the defaults (brightest 2%, $p = 6$) the mean errors are about 8° for gray world, 4.8° for brightest pixels, 5.4° for shades of gray and 4.4° for gray edge, against 34.5° for leaving a tungsten frame uncorrected. Each fails where its assumption does: gray world is 21° off on a yellow café front and 14° on a warm still life, gray edge 11° on a portrait against smooth walls, and on the two frames shot under daylight every method makes the picture worse, since there was no cast to remove. Interactive: choose the photograph and the estimator, set the brightest-pixel percentage and the Minkowski p , and mark the pixels that vote; the plot shows every estimate against the true light in rg chromaticity.

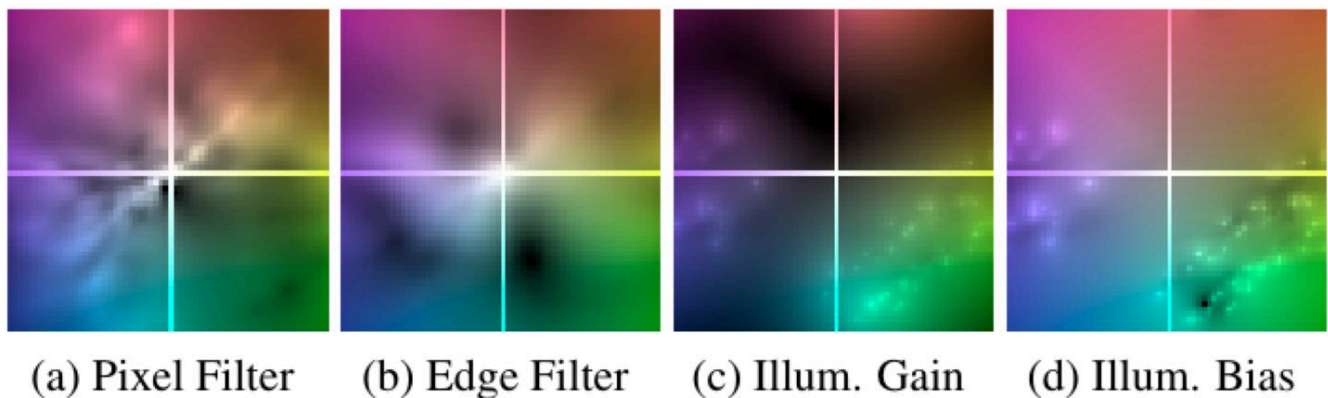
Learned estimators can use richer image statistics than gray-world. FFCC [Barron and Tsai, 2017](#) estimates illumination from a small periodic log-chroma histogram.

For positive linear values, let $u = \log(G/R)$ and $v = \log(G/B)$. Under a global diagonal gain, $u' = u + \log(k_G/k_R)$ and $v' = v + \log(k_G/k_B)$. Thus the model turns illumination into a histogram translation. FFCC uses Fourier-domain filtering, resolves periodic ambiguity, and estimates a posterior rather than only a peak. The periodicity belongs to its histogram representation, not to physical illumination.



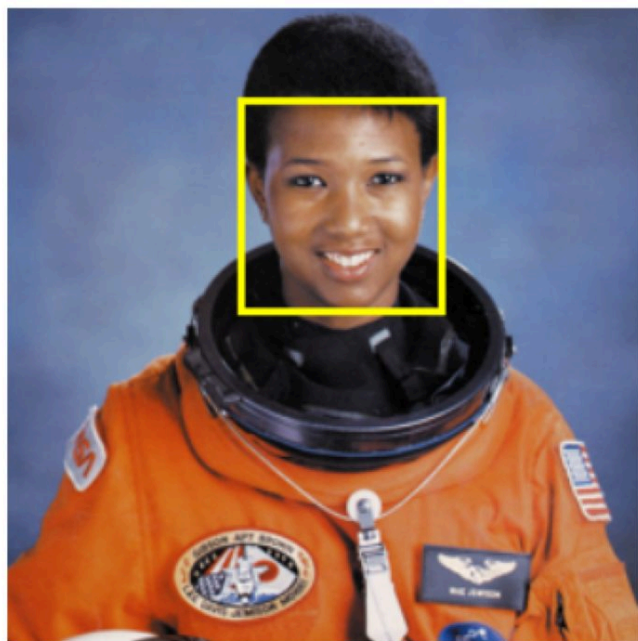
Reproduced from Barron & Tsai (2017), with permission of the authors.

Figure 8.10.5 (reproduced). *The Fast Fourier Color Constancy pipeline, from Barron. The input image is binned into a two-dimensional log-chroma histogram; a learned filter is convolved over that histogram (efficiently, in the Fourier domain) to produce a heat map whose peak localizes the illuminant; and that estimate is read back out as the per-channel gains that white-balance the image. The reframing to keep is that color constancy has become localization in a histogram: because changing the light merely translates the log-chroma histogram, finding the light is finding the shift. This is the method's overview (Barron 2017).*

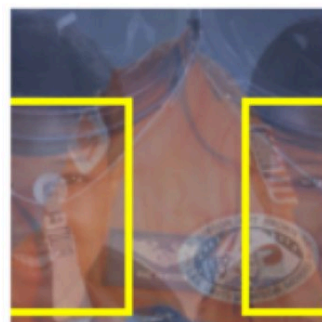


Reproduced from Barron & Tsai (2017), with permission of the authors.

Figure 8.10.6 (reproduced). *The FFCC model, from Barron. The estimator is a single learned filter applied to the log-chroma histogram, trained discriminatively so that convolving it with any scene's histogram puts its response peak at the true illuminant. Because illuminant change is a translation of the histogram, one shift-invariant filter (a convolution) suffices, and the whole estimate reduces to a few small Fourier-domain operations, which is what makes the method both accurate and exceptionally fast. This is the learned model at the heart of the method (Barron 2017).*



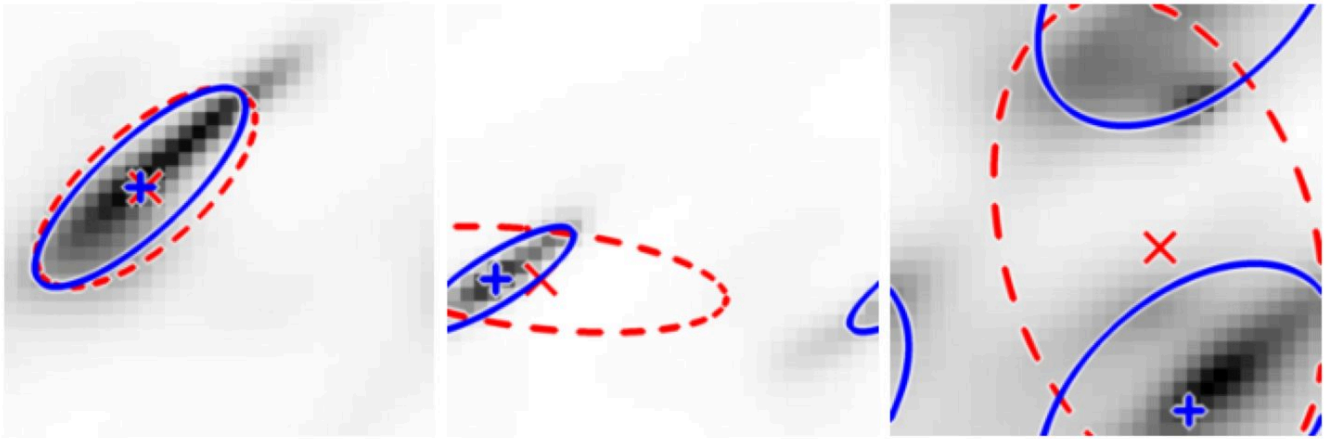
(a) Image A



(b) Aliased Image B

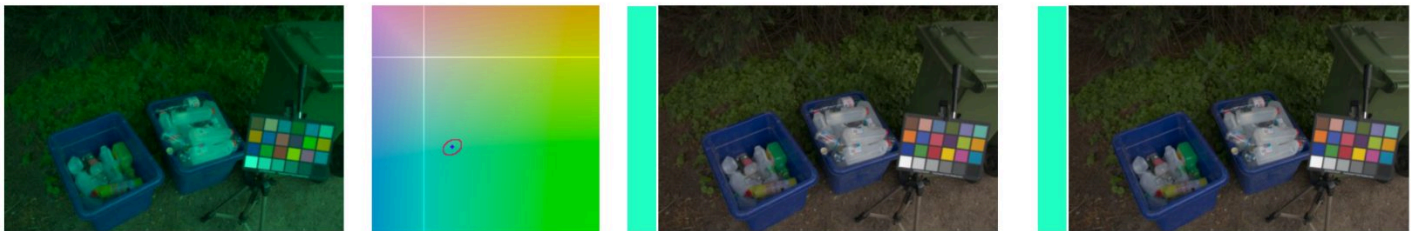
Reproduced from Barron & Tsai (2017), with permission of the authors.

Figure 8.10.7 (reproduced). *Illuminant estimation as localization, from Barron. Convolving the learned filter with a scene's log-chroma histogram yields a response surface over the chroma plane; the white balance is read off as the location of its peak. Seeing the estimate as the argmax of a heat map is what makes the rest possible: it turns "what color was the light?" into "where is the peak in this histogram response?", a question a convolution answers directly, and one whose confidence we can read from how sharp the bump is. From the FFCC method (Barron 2017).*



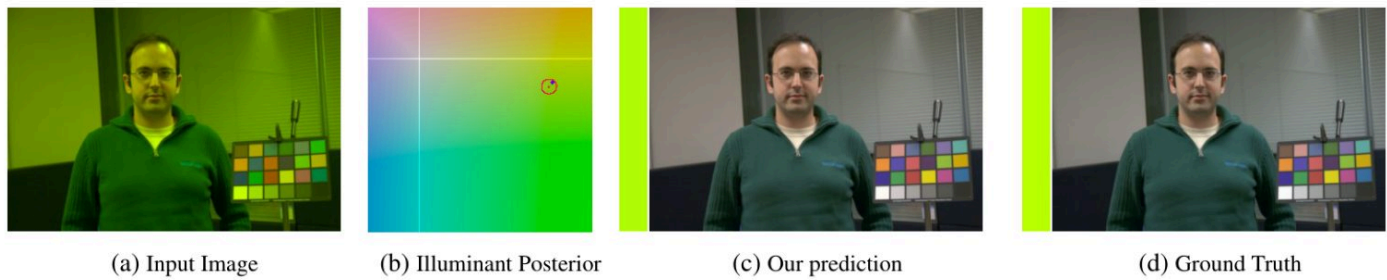
Reproduced from Barron & Tsai (2017), with permission of the authors.

Figure 8.10.8 (reproduced). Reading a confidence from the FFCC response, from Barron. Rather than just take the peak, the method fits a von Mises distribution (the natural bell curve for a quantity that wraps around, as hue does) to the response surface. The fitted mean gives the illuminant estimate and its spread gives a calibrated uncertainty, so the camera knows not only its best guess but how much to trust it, which matters when fusing the estimate over a video or with other cues. The lesson is that localizing in a histogram hands you a full distribution, not just a point. From the FFCC method (Barron 2017).



Reproduced from Barron & Tsai (2017), with permission of the authors.

Figure 8.10.9 (reproduced). FFCC white-balance results on the Gehler–Shi benchmark, from Barron. Scenes from the standard color-constancy dataset, each shown with its raw cast and after correction by the learned illuminant estimate, with the recovered light compared to the ColorChecker ground truth. Notice how a method whose entire inference is a small Fourier-domain convolution over a histogram nevertheless tracks the true illuminant across very different scenes: the priors gray-world only gestures at have been learned. The first set of published results (Barron 2017).



Reproduced from Barron & Tsai (2017), with permission of the authors.

Figure 8.10.10 (reproduced). Further FFCC results on Gehler–Shi, from Barron. A second set of benchmark scenes corrected by the method, including cases that defeat the classic statistical assumptions: frames dominated by a single strong color, where gray-world would drain the very hue that belongs in the scene. The takeaway is that the learned histogram filter has absorbed enough of natural-scene statistics to keep the color when gray-world would lose it, which is exactly the gap between a hand-coded statistic and a learned estimator that this chapter has been tracing. More published results (Barron 2017).

Sidebar 10: Dataset, Gehler-Shi + NUS (Cheng)

Color-constancy sets with ColorChecker ground-truth illuminant; lead with NUS (8 cameras, Cheng, Prasad & Brown 2014 (NUS color-constancy set)), cite Gehler-Shi as the classic baseline (Gehler et al. 2008 (ColorChecker / Gehler-Shi set)). <https://www2.cs.sfu.ca/~color/data/shi_gehler/> · <https://cvil.eecs.yorku.ca/projects/public_html/illuminant/illuminant.html>. See the [Chapter 45.16, Datasets](#) appendix.

Skin tones: a calibration target, a memory color, and a fairness caveat

Test exposure and white balance across a range of skin tones and lighting conditions. A face detector should not imply that every face belongs at one brightness or one color. Historical calibration practices, including Shirley cards, motivate the broader discussion in [Chapter 40.2, Ethics of computational photography](#).

In CompPhotoRoom

In [Chapter 8.13, A raw developer in the browser](#), White balance is a real Kelvin axis (Planck's law against the CIE observer), and the Auto button on the Tone panel is a learned auto-adjustment: a regression trained on the MIT-Adobe FiveK dataset (Bychkovsky et al. 2011) against expert C, in the app's own slider space. Its ? tells the full story.

This is a localization problem in a representation chosen to simplify the physics. The histogram retains the distribution of channel ratios while discarding their positions in the picture. A learned filter can score possible shifts using distributions seen in training. The estimated two-dimensional shift specifies relative channel gains,

leaving their common scale to exposure. The method replaces the single gray-world assumption with a learned statistical model, but surface color and illumination can still admit competing explanations.

Recap ISP, non-destructive editing OPTIONAL

Image metrics

SSIM

OPTIONAL
in the book [↗](#)

Structural similarity (SSIM) [Wang et al. 2004](#) compares local brightness, contrast, and correlation. For windows centered at the same location in the two images, a common form is

$$\text{SSIM}(I, J) = \frac{(2\mu_I\mu_J + C_1)(2\sigma_{IJ} + C_2)}{(\mu_I^2 + \mu_J^2 + C_1)(\sigma_I^2 + \sigma_J^2 + C_2)}. \quad (31)$$

The means μ_I, μ_J describe local brightness; the variances σ_I^2, σ_J^2 describe contrast; the covariance σ_{IJ} describes agreement after subtracting the means. The positive constants C_1, C_2 stabilize nearly flat or dark windows. Identical windows score one. Scores can be negative, particularly when local patterns are anticorrelated. Wang et al.'s formulation specifies the window and constants needed to reproduce a score.

Same PSNR, very different quality



MSE/PSNR can't tell these apart; SSIM can — pixel error $\neq$ perceived quality.

Figure 9.3.2 (reproduced). *Pixel error is not perceived quality.* Two degraded versions of the same image, contrived to have the same PSNR (identical mean squared error) yet very different appearance. One degradation is perceptually mild: a near-uniform change (a small contrast or brightness shift, a faint blur) that the eye largely forgives; the other is perceptually severe: structured damage concentrated where it shows (blocky artifacts, noise on a face, a smeared edge). MSE and PSNR score them equal; SSIM scores the structured-damage version far worse, matching what you see. This is the chapter's central demonstration that a single pixel-error number can hide a large perceptual gap.

SSIM is a map before it is a number. Because the window slides across the image, what SSIM produces first is a *per-pixel* structural-similarity image. Its displayed values indicate local agreement under the chosen visualization. The single "SSIM" value people quote is just that map's average. The map retains spatial information, because it localizes the damage in a way a scalar never can.

SSIM is a per-pixel map: dark where structure broke, bright where it survived



Averaging the map gives the scalar SSIM people report — but the map itself localizes the damage, which a single PSNR number cannot.

Figure 9.3.3 (reproduced). *SSIM is a per-pixel map, not just a scalar.* A clean photo, a copy with localized damage (one band heavily JPEG-compressed and noised, the rest untouched), and the resulting SSIM map: it is bright (near 1) wherever the two images match and drops dark exactly over the damaged band. Averaging the map gives the scalar mean SSIM people quote, but the map itself tells you where the structure failed, which a single PSNR number cannot.

SSIM adds useful spatial information but remains sensitive to window size, scale, and alignment. A luminance-only implementation misses purely chromatic errors. Multi-scale SSIM compares several resolutions. Record the implementation, input range, channel handling, window, and border convention; the name alone is not a complete evaluation protocol.

Morphing

Two interpolations: domain (shape) and range (color)

OPTIONAL
in the book [↗](#)

The cure has two knobs. Name them precisely, because the method is their product.

Range interpolation (color) is the cross-dissolve we already have: at a location x , blend the two colors,

$$C(x) = (1 - t) C_0(x) + t C_1(x), \quad (32)$$

a straight linear blend of pixel values.

Domain interpolation (location) treats a feature *point* as a vector and interpolates *where it is*. For a corresponding pair, a point P in I_0 and the matching point Q in I_1 , the in-between location is

$$V = (1 - t)P + tQ. \quad (33)$$

At $t = 0$ the feature sits where I_0 put it; at $t = 1$ where I_1 put it; in between it travels straight along the segment PQ (Figure 11.3.2). This is interpolation of the domain, the coordinate grid itself, as opposed to the color living on it.

To realize a domain interpolation as an image operation we warp: move pixels spatially while leaving their colors alone, $C'(x, y) = C(f(x, y))$. In practice we use the inverse / backward map (for each *output* pixel, look up where it came from, $\text{out}(x, y) = \text{im}(f^{-1}(x, y))$), so every output pixel is filled (no holes) and resampling is well-defined. This is exactly the engine of Chapter 11.1, [Warping](#); morphing is one of its biggest customers, calling it *twice per frame* (once for each source image).

The morph in one line: interpolate the domain like a vector, then interpolate the range (color). A morph is the product of a shape interpolation and a color interpolation: two independent linear blends, applied together (Figure 11.3.2).

Shape and colour are two orthogonal knobs of a morph: a real morph moves along the diagonal (both at once → a sharp face); a plain cross-dissolve moves along the colour axis alone (no warp → a ghost)



Figure 11.3.2 (reproduced). The two knobs of a morph, as orthogonal axes. One axis, range (color): the cross-dissolve $C(x) = (1-t)C_0(x) + tC_1(x)$, a linear blend of pixel values. Other axis, domain (shape): a feature point interpolated as a vector, $V = (1-t)P + tQ$, sliding the feature's location from where I_0 puts it to where I_1 puts it. A plain cross-dissolve moves along the color axis only (ghosts); a true morph is the product: interpolate position and color together. The corners of the square label the four extreme combinations.

Dynamic range and multiple-exposure capture

OPTIONAL
in the book [↗](#)

Problem set

PS4 merges an exposure bracket into a radiance map and tone-maps it. → [Chapter 1.5, Problem sets](#) (appendix).

Dynamic range is the ratio between the largest and smallest signal a sensor records. At the bright end, charge capacity or the readout chain limits the recorded signal. At the dark end, noise sets a gradual loss of precision rather than a hard black threshold. The usable dynamic range depends on the minimum signal-to-noise ratio (SNR) accepted for the task ([Chapter 2.8, Noise, signal-to-noise ratio and dynamic range](#)). That chapter develops

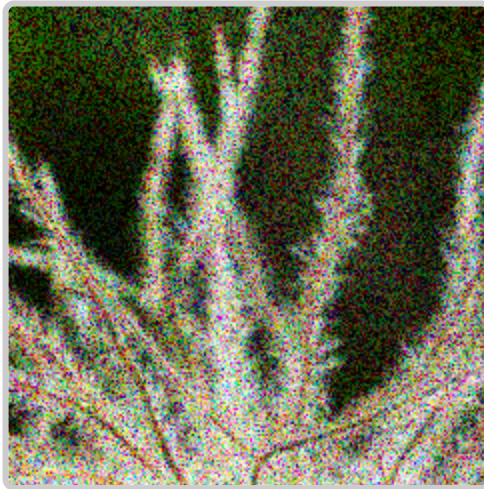
both ends: full-well capacity at the top (the maximum number of electrons a pixel can hold), read noise and photon noise at the bottom. The engineering definition places the floor where the signal equals the noise (0 dB), but that is too low for photographic purposes, where 30 dB is considered close to perfect and even 15 dB is severely noisy. [Figure 17.1.1](#) shows examples. Like depth of field, quantitative dynamic range is a question of tolerance, to noise in this case. Recall that bright scene values that saturate the photoreceptor may still be read below the maximum because of electronic noise. The ability to capture high dynamic range can further be limited by the veiling glare inside the lens, which lifts the darkest tones before the sensor sees them.

Click any picture to enlarge it. SNR in dB is $20 \log_{10}(\text{signal} / \text{noise})$; 20 dB is a noise of one tenth of the signal.

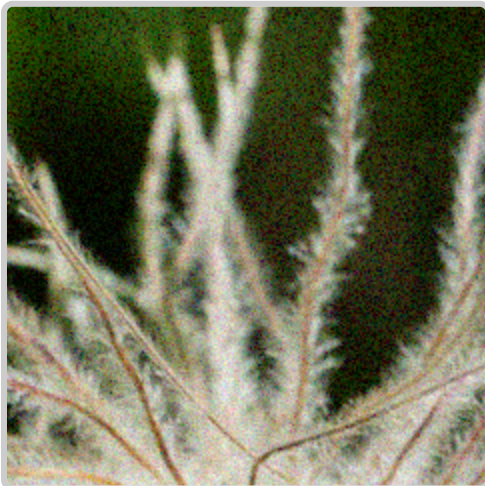
A 240 × 240 crop at 1:1, at six signal-to-noise ratios (stated at an 18% gray)



no added noise



0 dB: SNR 1, the engineering floor



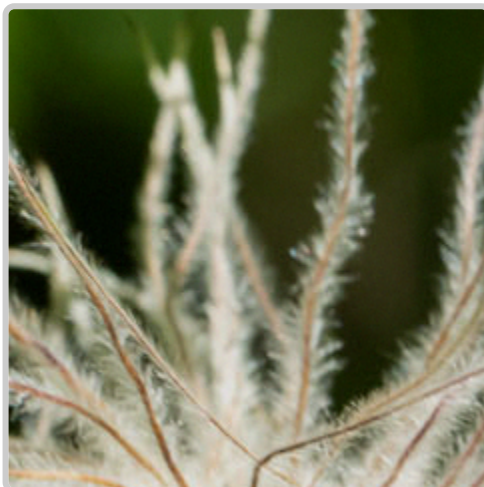
10 dB



15 dB: severely noisy



20 dB



30 dB: close to clean



40 dB

Noise added in linear light, growing with the square root of the signal. Photograph © Frédo Durand.

Figure 17.1.1 (reproduced). *What a signal-to-noise ratio looks like: a 1:1 crop of a photograph, one image pixel per screen pixel, with noise at 0, 10, 15, 20, 30 and 40 dB, where the SNR in decibels is $20 \log_{10}$ of the signal over the noise, so 20 dB is a noise of one tenth of the signal. The noise is added in linear light and grows with the square root of the signal, like photon noise, and the stated SNR is that of an 18% gray: the bright seed head is cleaner than the figure says and the dark background noisier. At 0 dB, the engineering floor of dynamic range, the picture is barely usable; at 15 dB it is severely noisy; at 30 dB the noise is hard to see.*

When the dynamic range of a scene exceeds that of the sensor, we capture multiple exposures to span the range. We may need to register these images in space (similar to denoising). [Chapter 13.2, Denoising by averaging](#) aligns and averages a burst of equal exposures, [Part 12, Matching pixels across space and time](#) covers the registration itself, and [Chapter 17.4, Ghosts and motion](#) deals with what moves between the frames of a bracket. We also need to calibrate the pixel values photometrically to bring them into the same absolute linear range proportional to scene-radiance values $L(x, y)$, which may require the calibration of the film or digital camera response curve (the beauty curve in particular, [Why scene-linear looks dull in Part 8, Basic image processing and ISP](#); [Chapter 17.2, Camera response and radiance calibration](#) recovers it). Motion, clipping in every frame, or inadequate shadow exposure can leave parts of that estimate uncertain.

💡 BIG LESSON (L2.46)

A common sensor dynamic-range measure divides the saturation signal by the read-noise standard deviation, in consistent units. A stricter quality threshold gives a smaller usable range. Bracketing extends coverage when its exposures provide reliable, registered measurements of the same scene.

Global tone mapping

Inpainting, texture synthesis, and object removal

Mixed-lighting white balance

OPTIONAL
in the book [↗](#)

Photograph a room at dusk with a lamp on and the curtains open, and the camera cannot solve the problem with a single global adjustment. The lamp throws a warm, orange light on one side of a face; the window lets in cool, bluish daylight on the other. White-balance for the lamp and the daylit cheek turns blue; white-balance for the window and the lamp-lit cheek turns orange. There is no global gain that neutralizes both, because the right correction is different in different parts of the frame, and an ordinary white balance has exactly one correction to give.

This is where the global white balance of [Chapter 8.10, Auto-exposure and auto white balance](#) fails. There, the camera's job was to *estimate one illuminant* and divide it out with a von Kries diagonal gain k_R, k_G, k_B : a single multiply in linear light. That model assumes one light color over the whole scene. Mixed lighting violates the assumption at the root: the illuminant is no longer one color but a field of colors that varies across the image. So we cannot estimate one gain; we have to estimate, at every pixel, *which light is doing the lighting there*, and correct accordingly.

[figure fig-mixed-illuminant-scene not built]

Figure 24.2.1 (reproduced). *Why one white balance is not enough. A scene lit by two differently-colored sources (warm interior light and cool daylight through a window) cannot be neutralized by a single global gain: balancing for one light leaves the other cast. A correct fix needs spatially-varying, multi-illuminant white balance.*

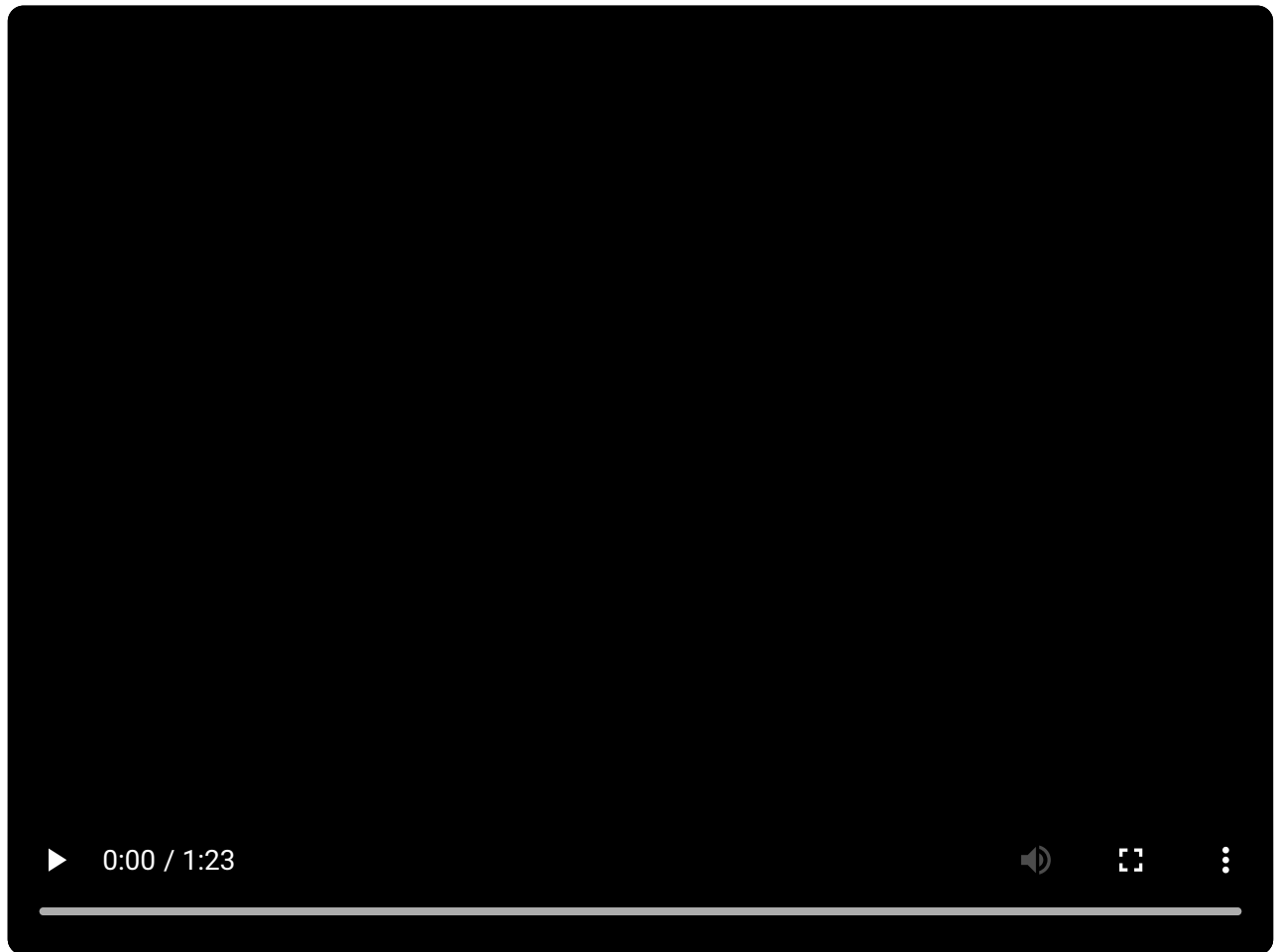
Correcting each region for its own light

OPTIONAL
in the book [↗](#)

With the mixing map $\hat{\alpha}(x)$ in hand, the correction is mechanical: at every pixel reconstruct the local light $L(x) = \hat{\alpha}(x) L^{(1)} + (1 - \hat{\alpha}(x)) L^{(2)}$ and divide the pixel by it, a per-pixel von Kries gain instead of one global gain. Where the lamp dominates we divide out the warm light; where the window dominates we divide out the cool light; in between we divide out the blend. With accurate estimates and a suitable diagonal model, neutral surfaces in both regions lose their casts *at once*, the thing a single gain could never do. The art and craft connection is direct: this is what a careful colorist does by hand with local color correction, masking the warm and cool regions and grading each separately. The method does it automatically and continuously, the per-pixel analog of dodging-and-burning the *color* of the light.

The estimate also supports lighting edits. Replacing $L(x)$ changes the local light color. To simulate switching off a source, however, retain its unnormalized contribution: write $C = \rho(i_1 L^{(1)} + i_2 L^{(2)})$ and set the appropriate

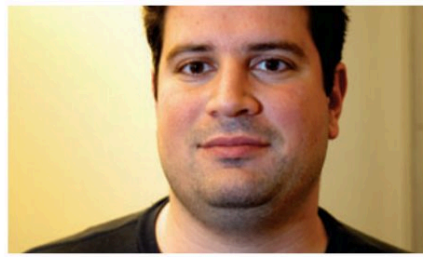
amplitude i_1 or i_2 to zero. Changing only the normalized mixture does not reproduce the brightness loss. Neither operation predicts new shadow geometry or view-dependent reflections.



Video 4. Video: *Light Mixture Estimation for Spatially Varying White Balance*: estimating, at every pixel, the mixture of two illuminants and white-balancing each region for the light that actually falls on it. Hsu, Mertens, Paris, Avidan & Durand 2008 ([Hsu et al. 2008](#)).



(a) Original image



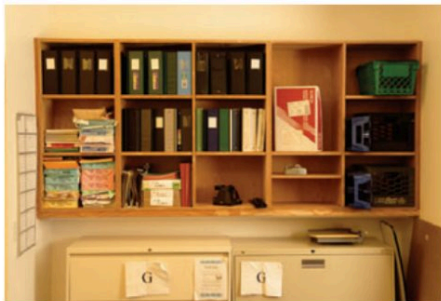
(b) Traditional white balance



(c) Our result

From Hsu, Mertens, Paris, Avidan & Durand (2008).

Figure 24.2.3 (reproduced). *Mixed lighting, where a single white balance must lose. The scene is lit by two illuminants of different color at once (a warm indoor source and cool daylight), so the cast a camera should remove changes from one part of the frame to the next. Balance for the warm light and the daylight regions go blue; balance for the daylight and the warm regions go orange; there is no one gain that neutralizes both, which is the whole problem a global white balance cannot solve [Hsu et al. 2008](#).*



(a) Input (tungsten and daylight)



(b) Voting (white material color)



(c) Voting (brown material color)



(d) Mixture constraints



(e) Interpolated mixture



(f) Our result

From Hsu, Mertens, Paris, Avidan & Durand (2008).

Figure 24.2.4 (reproduced). *The spatially-varying fix, in overview. Rather than guess one illuminant for the frame, the method estimates at every pixel how much of each of the two lights is contributing (a per-pixel mixture of the two illuminants) and then white-balances each region for the light that actually falls on it. The reframing to keep is that a mixed-light scene has no single answer, so the unknown becomes a smoothly-varying mixing map rather than a single gain [Hsu et al. 2008](#).*

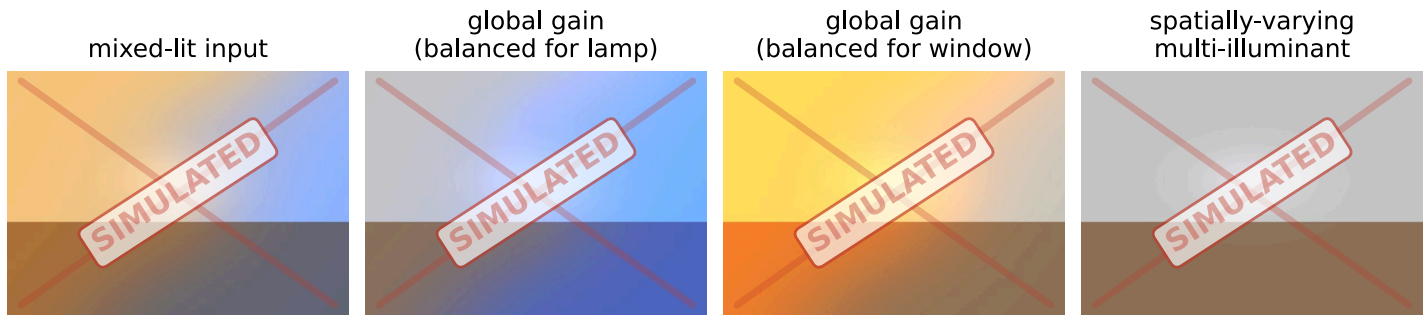
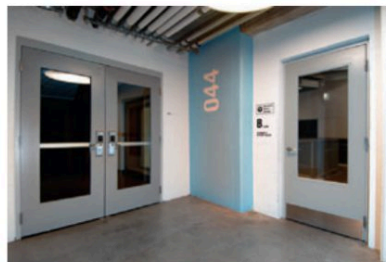
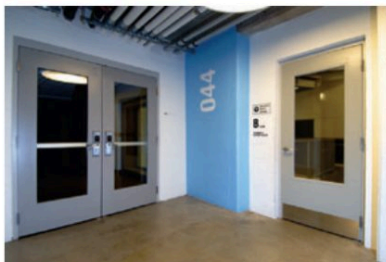
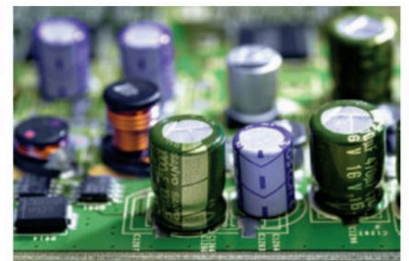
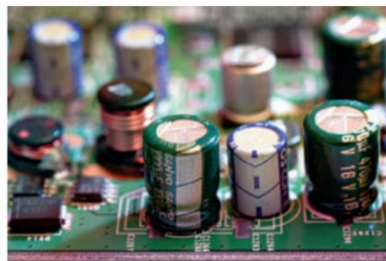
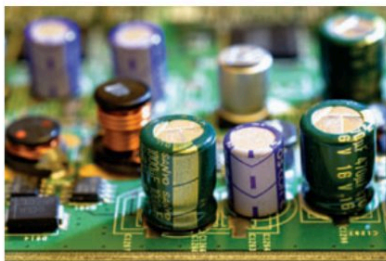


Figure 24.2.5 ([reproduced](#)). Global versus spatially-varying white balance. The same mixed-lit scene corrected four ways: the input; a single global gain balanced for the lamp (the window side goes blue); a single global gain balanced for the window (the lamp side goes orange); and the spatially-varying multi-illuminant result, which carries the right correction into each region so warm-lit and daylit areas read neutral at once. A global gain can only ever neutralize one of the two lights; treating the illuminant as a per-pixel mixture neutralizes both. (Illustration: a hand-built scene, not a faithful algorithm output.)



(a) Average white balance

(b) Local Color Shifts [Ebner 2004]

(c) Our result

From Hsu, Mertens, Paris, Avidan & Durand (2008).

Figure 24.2.6 (reproduced). *Global versus spatially-varying white balance, on a real result. The same mixed-light scene is corrected two ways: a single best global gain, which can only ever neutralize one of the two lights and leaves the other tinted, and the multi-illuminant result, which carries the right correction into each region so warm-lit and daylit areas both read neutral at once. This is the gain from treating the illuminant as a per-pixel mixture rather than one number for the frame* [Hsu et al. 2008](#).

Estimate a spatial illuminant map before applying local white balance. Check neutral surfaces under both lights and inspect transitions at shadows. Material assumptions, endpoint errors, and the diagonal color model limit the correction; smoothness by itself cannot identify the light.

Modern CPUs: memory hierarchy, parallelism, and what it takes to go fast

Light field networks

Recap: big lessons in this lecture

💡 BIG LESSON (L2.6) — THE MULTIPLICATIVE WORLD

Much of imaging is *multiplicative*, not *additive*. The light reaching the eye is illumination **times** reflectance; contrast is a **ratio** of intensities; a shadow darkens a region by a **factor**, not by subtracting a constant. Because the structure is multiplicative, the natural tools are **multiply, divide, and log**, not just add. This is why, again and again, we will work with ratios and with logarithmic encodings rather than raw linear values. It is also why **white balance** is hard: to recover an object's true color we must *divide out* the unknown illuminant $S(\lambda)$ from the product $C(\lambda)$, undoing a multiplication we never observed directly. (The additive counterpoint, light from different sources *adding*, and what it means for choosing an encoding, comes in Color technology.)

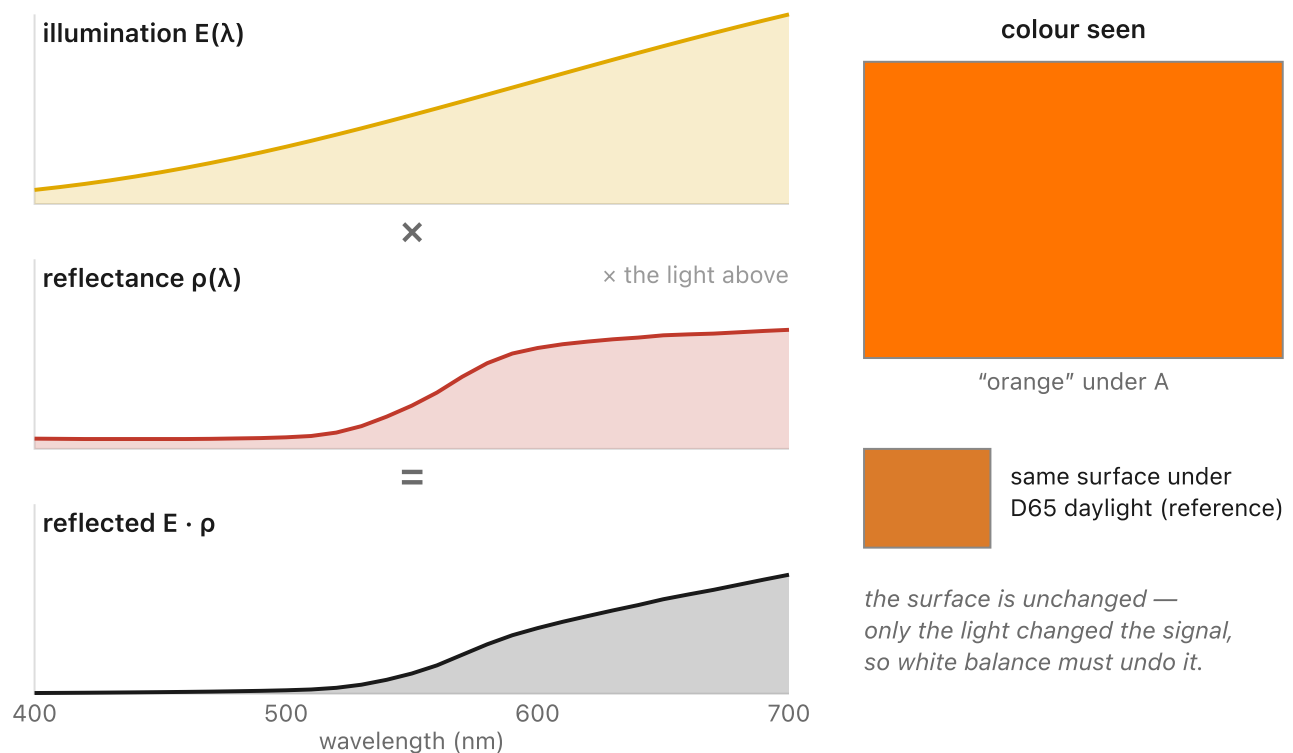


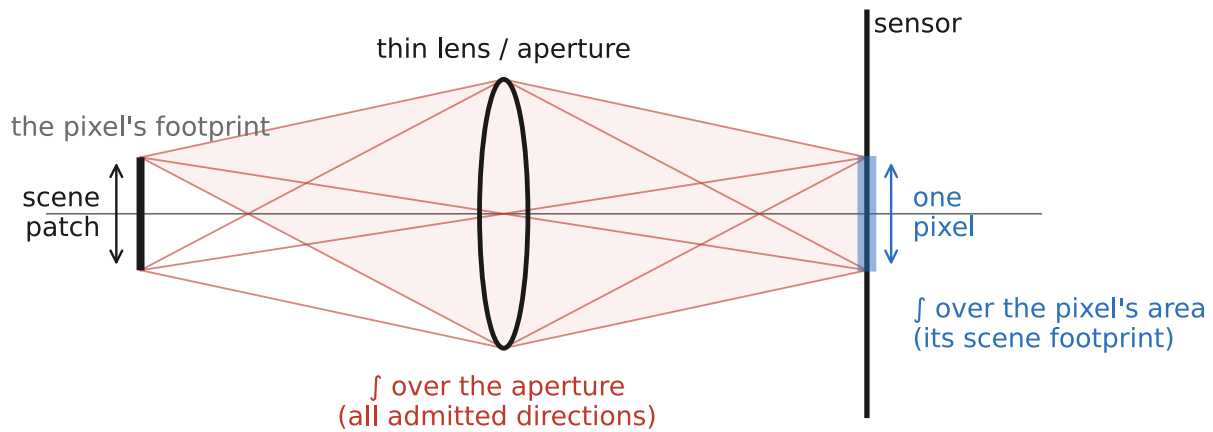
Figure 2.1.28 (reproduced). Color is illumination $\times$ reflectance. A surface's fixed reflectance spectrum $\rho(\lambda)$ (here an apple's, high in the reds) is multiplied per wavelength by the illuminant spectrum $S(\lambda)$ actually present. The product $C(\lambda) = S(\lambda) \cdot \rho(\lambda)$ is the light that reaches the eye. Change the illuminant and the product changes even though the surface has not: the same reflectance under white versus reddish versus greenish light yields three different stimuli. One swatch shows the surface's unadapted appearance under the chosen illuminant; a smaller reference swatch shows it under D65 for comparison.

💡 BIG LESSON (L2.26) — A PIXEL VALUE IS AN INTEGRAL

No pixel measures "the" value at a point in space, time, or color: it reports an **integral** over everything the photosite cannot tell apart. Four axes appear explicitly above: the **spatial footprint** (the pixel's area, plus the optical blur smeared across it), the **exposure time** (a window, not an instant), the **spectrum** (wavelength, weighted by the channel's spectral sensitivity), and the **solid angle of rays** the aperture admits; a fifth, **polarization**, is usually lumped in too. Naming all the axes at once is what makes **motion blur**, **defocus**, **color**, **vignetting**, and **aliasing the same phenomenon** (a finite integral standing in for an idealized point sample), and the *width* of the integral along each axis is a knob the rest of the book spends its time controlling.

A pixel integrates over the aperture and over its own area

$$\text{pixel} = \iiint \text{radiance over aperture} \times \text{pixel area} \times \text{time} \times \text{wavelength}$$



$\times \int dt$ (exposure time) $\rightarrow$ motion blur $\times \int d\lambda$ (spectral response) $\rightarrow$ colour

Figure 2.4.2 (reproduced). A pixel value as a four-dimensional integral, a measured slice of the plenoptic function. The thin lens (drawn as an ellipse) admits a cone of rays; the value recorded at one finite-area pixel is the scene radiance integrated over (1) the aperture (which rays/directions the lens admits), (2) the pixel's area on the sensor, (3) wavelength (the spectral response), and (4) the exposure time. Aperture $\rightarrow$ depth of field; pixel area $\rightarrow$ sampling/antialiasing; time $\rightarrow$ motion blur; wavelength $\rightarrow$ color.

💡 BIG LESSON (L2.35) — SENSORS CONVERT PHOTONS TO ELECTRONS LINEARLY

A photosite is at heart a **photon counter**: each absorbed photon frees a charge carrier, an **electron** or equivalently a *hole* (a missing electron), and the well accumulates them in direct **proportion** to the light that arrives. So the raw signal is **linear in scene radiance**, before any gamma or tone curve. This linearity is not a detail; it is what makes raw arithmetic *trustworthy*: HDR merging **adds** exposures, white balance is a per-channel **multiply**, flat-field correction is a **divide**, and all three are valid only because the numbers are proportional to light. Display encoding (gamma) deliberately hides this linearity for the eye, so computation should undo it and work in **linear light**.

💡 BIG LESSON (L3.12) — THE EYE DISCOUNTS THE COLOR OF THE ILLUMINANT

Chromatic adaptation and scene interpretation help stabilize surface color across illuminants. Per-channel cone gains are a useful approximation. Camera white balance makes a related rendering choice for the output conditions, with accuracy limited by the measurements and estimated illuminant.

💡 **BIG LESSON (L3.13) — CAPTURE AND VIEWING CONDITIONS BOTH AFFECT COLOR APPEARANCE**

A photograph is viewed under conditions different from those at capture. Its appearance depends on the display or print, the surround, adaptation, and cues within the picture. White balance and rendering should target those conditions and the photographer's intent; viewers do not automatically undo the captured illuminant.

💡 **BIG LESSON (L3.7) — THE EYE'S DYNAMIC RANGE IS ENORMOUS OVER TIME AND MODEST AT ANY INSTANT**

Distinguish sensitivity across adaptation states from the range visible in one scene under specified conditions. Local adaptation, glare, spatial structure, and the task all affect that range. A single stop count is insufficient to compare an eye, camera, and display.

The eye's three operating regimes across ~14 log units of luminance

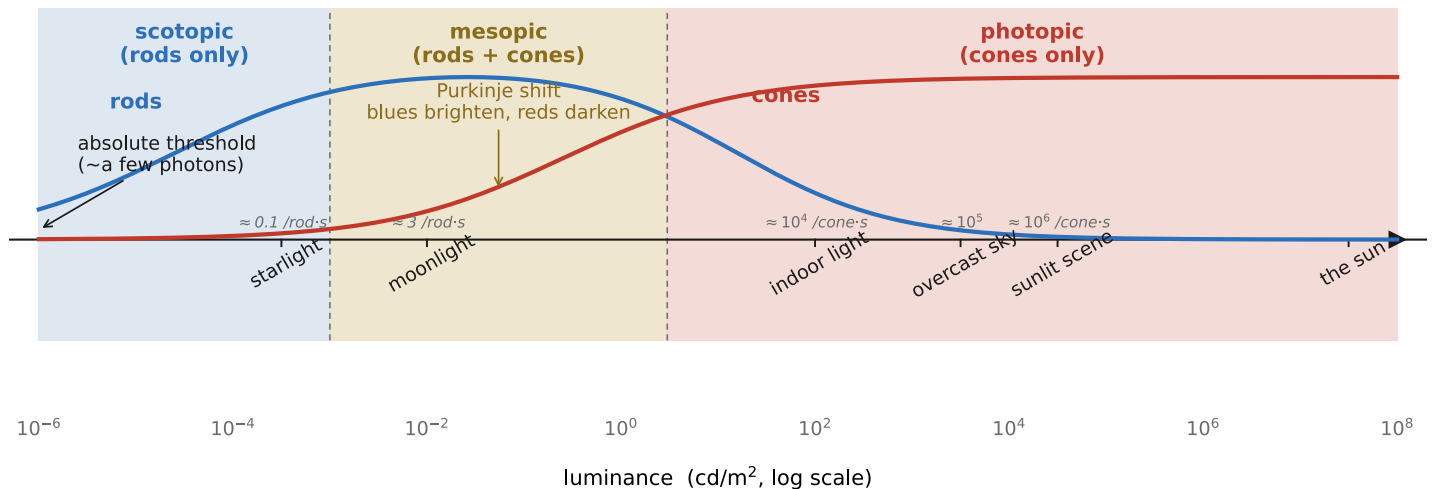


Figure 3.3.4 (reproduced). The three operating regimes on a log-luminance (cd/m^2) axis: scotopic (rods only), mesopic (rods and cones), and photopic (cones only), with the rod and cone activity curves, the absolute threshold of vision, the Purkinje shift in the mesopic band, and real-world landmarks placed along the axis: starlight → moonlight → indoor light → overcast sky → sunlight → the sun. Each landmark also carries the photons caught per receptor per second (photoisomerizations): from about 0.1 per rod per second under starlight (one photon every ten seconds or so) up to $\sim 10^6$ per cone per second in a sunlit scene. These counts fold in the pupil (iris), which narrows from about 7 mm in the dark to about 2 mm in bright light, and switch from per-rod (scotopic) to per-cone (photopic) as the receptors hand over.



adapted to **-0.7 stops** relative to the scene's mean

one exposure loses 0% of this patch to white and 12% to black · the patch spans 9.9 stops, the picture 11.4

Saturating: the same multiplier plus a roll-off, which is why the ends survive.

radiance map: office window — Byong Mok Oh

Figure 3.3.2 [\(reproduced\)](#). *One gain per fixation (interactive). A real radiance map, ten stops or more from its darkest corner to its brightest, far more than a display or any single instant of vision can show. **Move the pointer.** Inside the soft aperture the scene is rendered for the light at that spot; outside it, the same scene gets the one linear exposure a camera has to choose, and you can set it the two ways a photographer can: **for the average**, which places the mid-tones and burns the highlights out, or **for the highlights**, which keeps them and drops everything else into the dark. On the office the two are about six stops apart, and neither one shows you the room and the city outside the window at once, which is the whole trouble; on the design-center interior they are eleven stops apart and the gap is starker still. Nothing about the scene changes as you move, only the gain, and that is enough to make every part of it readable in turn: the readout says how much of what is under the aperture the single exposure loses at each end, to white and to black. **Saturating** is the previous figure's Naka-Rushton curve with its half-saturation slid to the local level, so the ends roll off; **gain only** is a pure multiplier, which rescales but compresses nothing, so whatever range the aperture itself contains still clips. Adaptation is lagged rather than instant, in the log domain because the mechanism is multiplicative; set the time constant to zero and the effect is uncanny, because instant*

adaptation is not what having eyes is like. The readout also shows where a single gain is not enough: park the aperture on a bright doorway edge and one fixation spans nearly the whole picture's range, which is what local tone mapping later in the book exists to handle. Eight radiance maps to choose from, each at its own proportions. **A cartoon in five specific ways:** the eye pools its adapting level over a far larger and softer region than this aperture and with a strong whole-field component, so the sharp boundary is an artifact; only the fast neural gain is modeled, not the photopigment bleaching that takes minutes nor the pupil; a real surround still contributes to the pool rather than being ignored; the eye never holds still, so you never see a scene at one gain the way a frozen aperture suggests; and the peripheral blur is a separate simplification standing for the fall-off of acuity away from the fovea, a different mechanism from adaptation and nothing like a uniform blur, present so you can see where you are looking and switchable off, and computed in linear light, which is why a blown highlight spreads into what is beside it. Radiance units in a Radiance file are arbitrary unless calibrated, so levels are quoted in stops relative to each scene's own geometric mean, not in cd/m^2 . Radiance maps by Paul Debevec and Byong Mok Oh.

💡 **BIG LESSON (L3.10) — CONTRAST DEPENDS ON A REFERENCE LEVEL**

Detectable increments often scale approximately with the background in the photopic range. State which contrast definition, stimulus, and viewing conditions a measurement uses. Absolute pixel error and perceived difference answer different questions; neither replaces the other.



L 0.05000 ΔL 0.00500

$$\Delta L / L = 0.1000 = 10.00\%$$

Lower ΔL until the patch just disappears — that ratio is your threshold. Then change L and find it again.

Linear luminance; the grey is encoded only when drawn.

Figure 3.3.6 [\(reproduced\)](#). *Measure your own Weber fraction (interactive). A patch of increment ΔL on a background L . Lower ΔL until the patch just disappears: the ratio $\Delta L/L$ you land on is **your** threshold, and it is the quantity Weber's law is about — the smallest detectable increment is a roughly constant fraction of the background. Now change L by a factor of ten and find the threshold again: ΔL will have moved by roughly the same factor while the ratio comes back to about where it was. In good conditions that ratio is a few percent; on a screen in a lit room expect worse, because light reflected off the glass adds a constant veil to patch and background alike, turning the quantity into $\Delta L/(L + L_0)$ and hurting dark backgrounds most. Two caveats: you know where the patch is, which lets you detect a smaller increment than a proper forced-choice experiment would, and Weber's law holds only over the middle of the range — near black the threshold stops falling and flattens out. Values are linear luminance; the gray is encoded only when drawn.*

💡 **BIG LESSON (L3.11) — SURFACE APPEARANCE CAN REMAIN STABLE AS ILLUMINATION CHANGES**

Surface-lightness and color judgments can remain relatively stable as illumination changes. This constancy is partial and uses several cues. It differs from recovering the complete reflectance spectrum, and it does not imply that brightness or the appearance of lighting stays fixed.

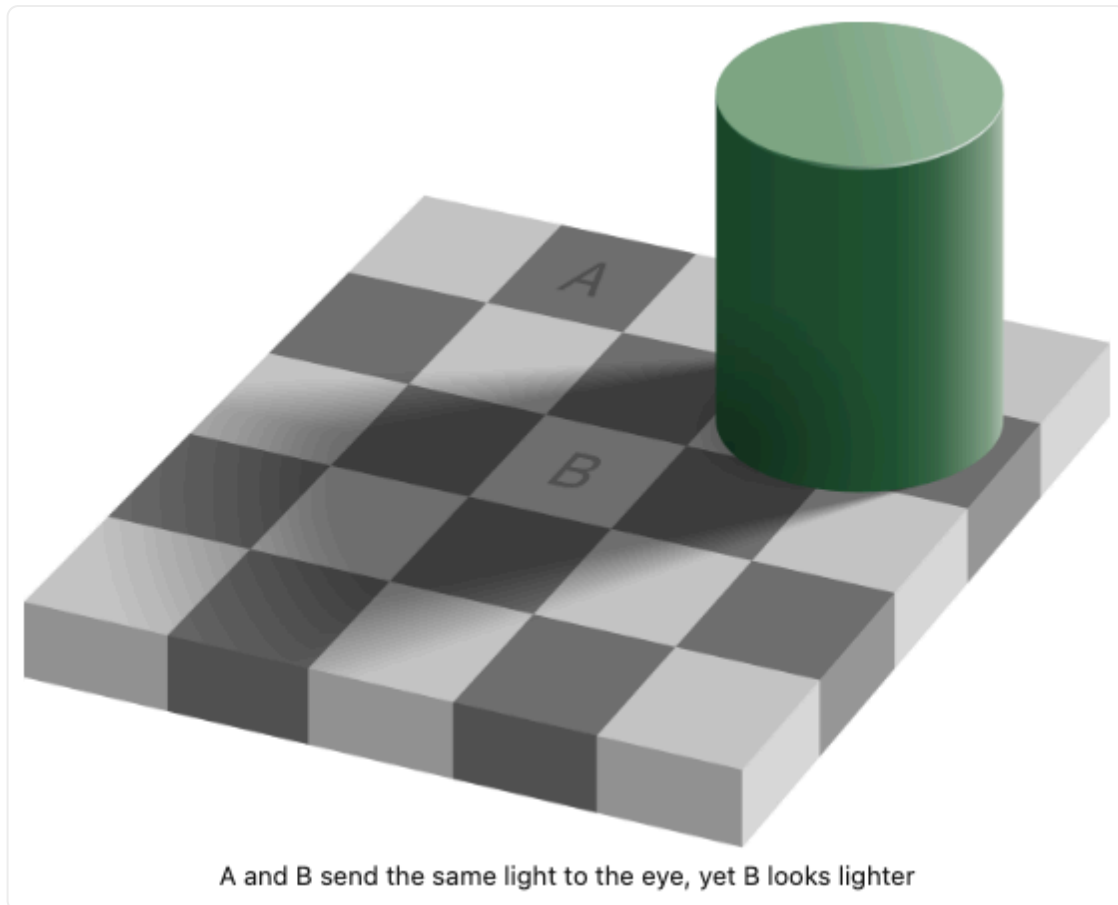


Figure 3.3.9 (reproduced). *Adelson's checker-shadow illusion*. The square labeled A (in light) and the square labeled B (in the cylinder's shadow) have identical luminance, yet B looks far lighter, shown side by side with a proof strip connecting the two equal-gray squares. The visual system discounts the inferred shadow and reports surface lightness, not luminance.

💡 **BIG LESSON (L4.8) — THERE IS NO SINGLE RGB: GAMUT VERSUS BIT BUDGET**

Choose a working gamut large enough for the content and operations, with sufficient numeric precision. Choose a delivery space the receiving system can interpret. Wider primaries alone do not improve an image, and converting an sRGB image to a wider space does not recover colors previously clipped.

💡 **BIG LESSON (L5.1) — PHOTOGRAPHERS MEASURE LIGHT IN STOPS**

A stop is a factor of two. Time steps by two and f-number by $\sqrt{2}$. ISO and a software exposure slider can change rendered brightness by a stop without collecting another photon.

💡 **BIG LESSON (L7.5) — FACTORIZATION IS INVERTING A PRODUCT, AND IT IS FUNDAMENTALLY ILL-POSED**

Factorization problems seek to decompose a number, or an image, into a *product* of two unknown pieces (scalars, or whole matrices) given only the product. This is fundamentally ill-posed: many factor pairs multiply to the same result, and unconstrained formulations often admit the degenerate "put everything in one factor, leave the other the identity" split. Additional constraints, calibration, or priors breaks the tie. In the fully general case it is our linear problem $\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{n}$, except we must now solve for the operator $\mathbf{A}$ *as well as* the unknown $\mathbf{x}$. Examples include white balance, intrinsic images, matting, and blind deblurring.

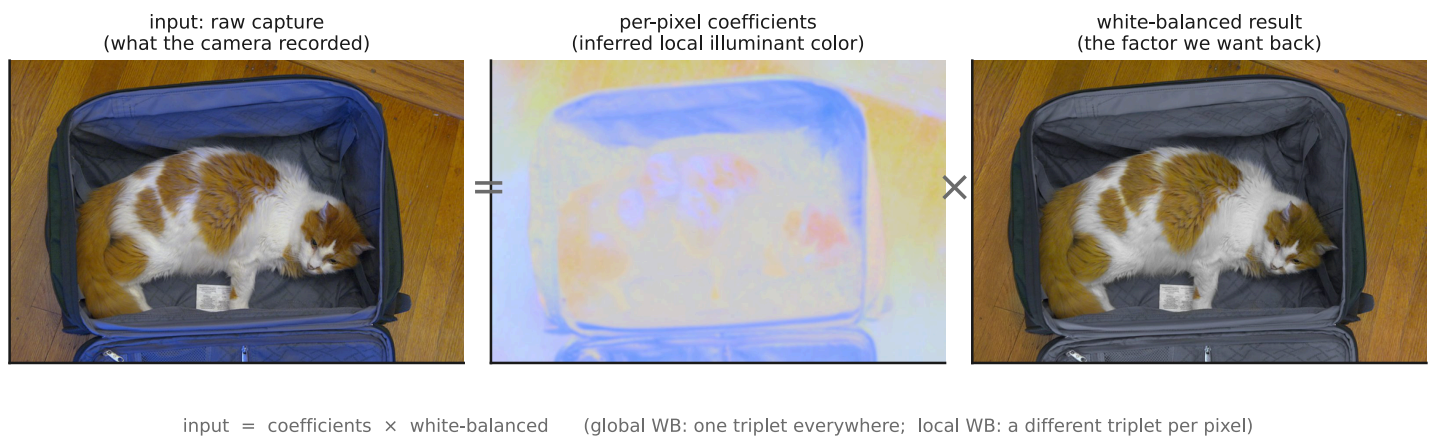


Figure 4.2.3 (reproduced). Local white balance as a per-pixel factorization, on real mixed-lighting data. Under a warm lamp on one side and cool daylight on the other, the raw capture is, pixel by pixel, a white-balanced image times a spatially varying color coefficient. Read left to right, the input factors into two color triplets at every pixel: input = coefficients $\times$ white-balanced, the raw capture on the left, then the per-pixel coefficient field (the local illuminant color) times the white-balanced result we want back. We are not given the coefficients; we infer them as the per-channel ratio of the input to the result, computed in linear light. A global white balance would collapse the whole coefficient panel to one constant triplet; the local problem is to recover a different triplet at every pixel, which is why it needs a strong prior on how illumination varies. Data from Boyadzhiev et al., "User-Guided White Balance for Mixed Lighting Conditions" [Boyadzhiev et al. 2012](#).

💡 **BIG LESSON ((L8.1) · DITHER: TRADE STRUCTURED ERROR FOR UNSTRUCTURED NOISE)**

Dither replaces visible quantization structure with noise. Independent stochastic rounding produces approximately white noise; error diffusion shapes the error spatially and can leave patterns. Dither preserves an average under stated assumptions, not every pixel or every local region.

💡 **BIG LESSON (L8.2) — LOOP IN MEMORY ORDER (INNERMOST INDEX VARIES FASTEST)**

Prefer traversal that reuses nearby stored values. For scalar loops over packed HWC, use y, x, c order; for packed CHW, use c, y, x. Actual strides, tiling, and vectorized kernels can change the best schedule. Measure before changing the representation.

💡 **BIG LESSON (L4.1) — PIXEL NUMBERS ARE MEANINGLESS ON THEIR OWN**

The numbers in an image array carry no meaning by themselves. The same triple (0.5, 0.2, 0.1) is a different color, and a different amount of light, depending on two things the array does not contain:

- its **color space**: which primaries and white point those R, G, B refer to (sRGB? Adobe RGB? Display P3? a camera's native raw space?); and
- its **value encoding**: how a stored number maps to light (**linear**, **gamma**, **log**, or a camera "look" curve).

Strip those two away and you are left with a grid of anonymous numbers. Every operation that matters later (white balance, HDR merging, blur and deblur, any color conversion) silently assumes a particular answer to both questions, and gets the wrong result if the assumption is wrong. Hence the discipline of the whole book: never touch a pixel without knowing its color space and its value encoding. These are two common causes of otherwise puzzling image differences.

💡 **BIG LESSON (RECURRENCE OF (L4.6) AND (L4.4)) — MIND THE ENCODING**

Use the domain expected by the operation. Decode the transfer function before linear-light arithmetic, and encode appropriately for display. Decoding a rendered file does not undo tone mapping or restore scene-linear measurements.

💡 **BIG LESSON (L4.6) — ADDITIVE VS. MULTIPLICATIVE → CHOICE OF ENCODING**

Exposure multiplies linear-light values; independent contributions of light add. Log coordinates turn exposure ratios into differences. An offset in encoded values is a distinct appearance control. Choose the domain from the operation and its model.

💡 **BIG LESSON (L8.3) — EXPOSURE SCALES LINEAR LIGHT (A MULTIPLY) OR SHIFTS LOG VALUES (AN ADD)**

Exposure scaling multiplies linear-light values by $k = 2^s$ or adds s in base-2 log units. For a pure gamma encoding $V = L^{1/\gamma}$ it multiplies V by $k^{1/\gamma}$. For an arbitrary transfer function, decode, scale, and encode. Output clipping is a separate operation.

💡 **BIG LESSON (L8.4) — CONTRAST IS A PIVOT: CHOOSE WHAT STAYS PUT**

A contrast change rotates values about a chosen pivot: $L_{\text{out}} = p + g(L_{\text{in}} - p)$. The pivot p is exactly what is left unchanged, and it also decides whether the picture gets brighter or darker overall: a high pivot pulls most of the image down (darkening), a low pivot pushes it up (brightening). Unlike exposure, contrast is not radiometrically meaningful, it is a stylistic stretch, so the *encoding matters less* (no single space is "correct"); what matters is a sensible pivot, and the sensible pivot *does* depend on the space: roughly 18% mid-gray in linear light, the numeric midpoint in gamma or log. Pick the pivot deliberately: it is the whole character of the operation.

Contrast: steepen the tone curve about a mid-gray pivot (here in sRGB)

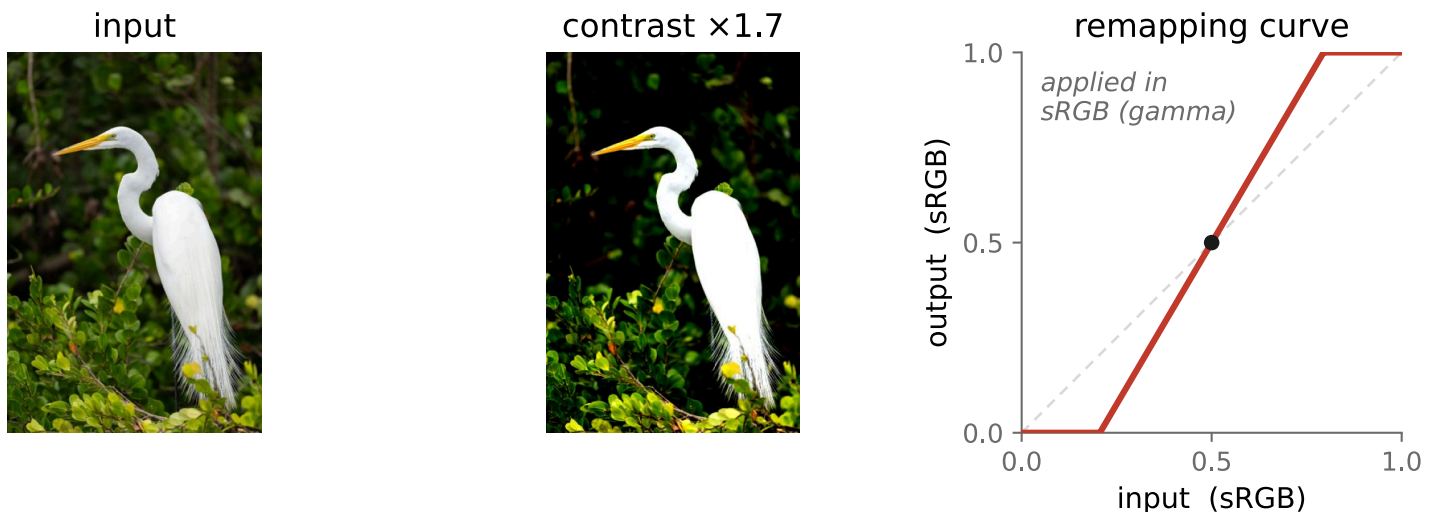


Figure 8.3.5 (reproduced). Contrast as a steepening about a pivot. Left, the input; center, increased contrast (gain > 1): shadows deepen, highlights brighten, midtones near the pivot barely move; right, the transfer curve, a straight line through the mid-gray pivot p whose slope is the gain. Steepening drives the line into the floor (0) and ceiling (1), clipping the darkest and lightest tones. The 45° diagonal (gain 1, no change) is shown for reference.

💡 **BIG LESSON (L8.6) — STORE THE RECIPE, NOT THE RESULT**

Preserve original samples and store the operations and parameters separately. This makes adjustments revisable without repeatedly quantizing intermediate images. Also preserve the processing version and dependencies; a recipe is reproducible only while its ingredients and interpretation remain available.

💡 **BIG LESSON (L2.46)**

A common sensor dynamic-range measure divides the saturation signal by the read-noise standard deviation, in consistent units. A stricter quality threshold gives a smaller usable range. Bracketing extends coverage when its exposures provide reliable, registered measurements of the same scene.

💡 **BIG LESSON (L4.6) — ADDITIVE VS. MULTIPLICATIVE → WORK IN LOG**

Choose an encoding for the operation. Add independent light contributions in linear light. For relative-contrast decomposition, log luminance turns ratios into differences, so a base and a residual can be combined by addition.

💡 **BIG LESSON (L18.2): TONE-MAP IN THE LOG DOMAIN WHEN THE OPERATION ACTS ON RATIOS**

Log luminance is useful when the operation should act on intensity ratios, as in additive base/detail decomposition or gradient attenuation. Other operators, including Reinhard's $L/(1 + L)$, can be formulated directly in linear luminance. Match the representation to the operation; do not take logarithms of nonpositive values.

💡 **BIG LESSON (L29.4) — SEPARATE THE ALGORITHM FROM ITS EXECUTION SCHEDULE**

The algorithm specifies the result; the schedule controls loop organization, parallelism, storage, and recomputation. Changing the schedule can change both traffic and executed arithmetic, especially at tile halos. Halide makes these choices explicit so you can compare them without obscuring the algorithm.