ACRONYMS — THIS CHAPTER

ACRONYMS

ARW	Sony Alpha Raw (file format)
BM3D	Block-Matching and 3D filtering (denoising)
CA	chromatic aberration
CC	CameraCalibration matrix (in the DNG pipeline); Creative Commons in a licensing context
CC0	Creative Commons Zero (public-domain dedication)
CCD	charge-coupled device
CFA	color filter array
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
CR3	Canon Raw version 3 (file format)
CYGM	cyan-yellow-green-magenta (color filter array)
DNG	Digital Negative (raw file format)
ISP	image signal processor
MATLAB	Matrix Laboratory (software)
NEF	Nikon Electronic Format (file format)
PSNR	peak signal-to-noise ratio
SDK	software development kit
SNR	signal-to-noise ratio
SSD	sum of squared differences
SSIM	structural similarity (index)
UV	ultraviolet

Course notes · 34 sections from the book · 26 optional hide optional sections the full book ↗

LECTURE NOTES 5 — DEMOSAICKING

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

▸ Contents · 27 sections on this page, from 12 chapters of the book

Sensing color: multiplexing strategies

in the book ↗

A bare photosite usually reports one integrated charge measurement, with a wavelength-dependent quantum efficiency, so to capture color we must somehow take *three* (or more) measurements where a given photosite provides only one. Every color camera multiplexes those measurements across some axis, and there are several ways to do it, each spending a different axis (Figure 4.6.1).

BIG LESSON (L4.11) — ONE VALUE PER PHOTOSITE; COLOR MUST BE MULTIPLEXED

Strip away the color filter and a sensor is **monochrome at the pixel**: each photosite reports a **single number**, a signal proportional to detected photoelectrons within its operating range, not a color. The three (or more) values that color needs must therefore be **multiplexed** along some other axis: over **space** (a color-filter array such as Bayer, paying with resolution and a demosaicking step), over **time** (sequential color filters, paying with motion robustness: the old color-wheel and many scientific cameras), across **multiple sensors** (a beam-splitter prism feeding three chips, the **3-CCD** video camera, paying with bulk and cost), or over **depth** (wavelength-dependent absorption in stacked photodiodes, the **Foveon**, paying with noise). The choice determines which measurements are simultaneous and co-located. The same constraint shapes the eye, whose three cone types sample color over space at the retina (L3.1 show, L3.6 show).

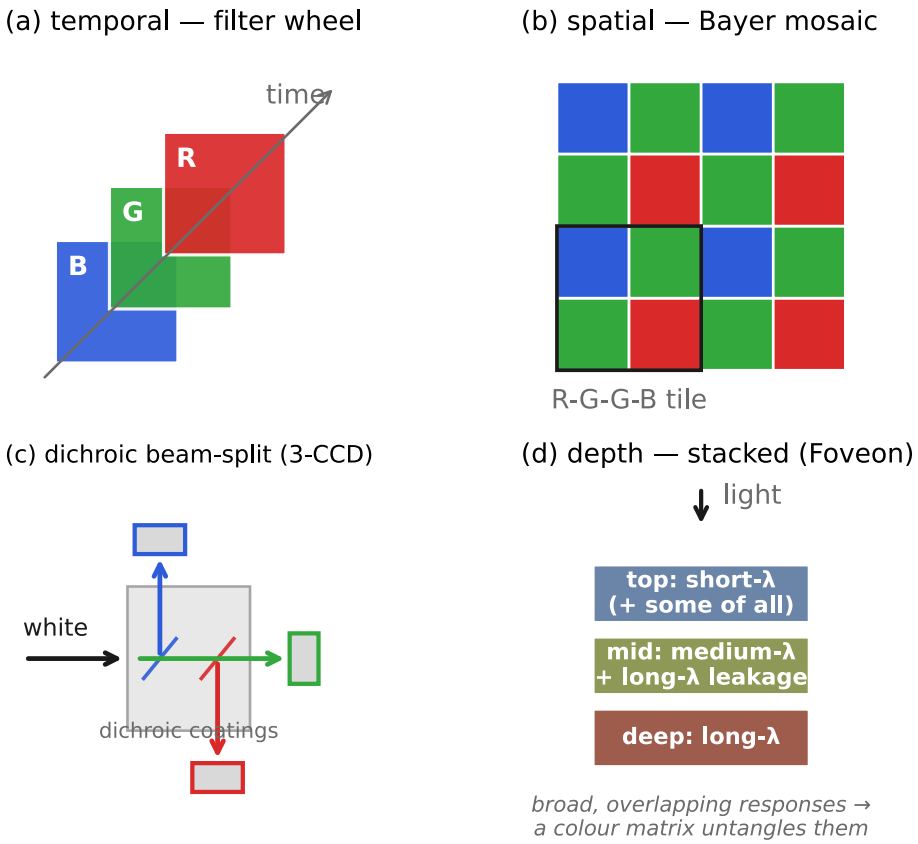


Figure 4.6.1 (reproduced). *Four color-sensing multiplexing strategies. Temporal: shoot R, G, B frames in sequence through a filter wheel. Spatial: a color filter array (the Bayer mosaic) puts one color over each photosite. Beam-splitter: a prism sends the rays to three separate sensors. Depth: stacked layers (Foveon) absorb different wavelengths at different depths. Each trades resolution, light efficiency, motion robustness, and cost differently.*

Three-channel color requires three distinct spectral measurements. Distributing them across space or time introduces assumptions about how the scene varies. Multiplex in time and you must assume the scene holds still between frames; multiplex in space and you must assume color varies smoothly enough to interpolate the samples you skipped; multiplex in depth and you inherit badly overlapping bands you then have to un-mix. A beam-splitter can send co-located, simultaneous images to several sensors, at the cost of alignment and optical complexity. A nanoscale router that sends wavelengths to neighboring photodiodes still uses spatial sampling.

Multiplexing in time

OPTIONAL
in the book ↗

Shoot red, green, and blue frames sequentially through colored filters, or a rotating filter wheel. This is the principle behind Maxwell's early color-photography demonstration (Maxwell 1861) and how Prokudin-Gorskii (Prokudin-Gorskii) captured the Russian Empire in three exposures. It is still how flatbed scanners, astronomy and fluorescence-microscopy filter wheels, telescope narrowband imaging, sequential-color machine-vision cameras, and many multispectral instruments work, because the same mechanism extends naturally from three filters to tens of narrow bands.

The implementations differ mainly in what does the sequencing. The oldest is a **rotating filter wheel** (or three separate plates) in front of a single sensor: three exposures through red, green, and blue filters, combined afterward. Maxwell's 1861 tartan-ribbon demonstration and Prokudin-Gorskii's later survey are early examples. **Astrophotography** still leans on it heavily: a cooled *monochrome* sensor (a mono sensor avoids the color filter array's light loss and keeps full resolution) sits behind a motorized filter wheel and stacks many long sub-exposures through each filter, red / green / blue for natural color or narrowband hydrogen, oxygen, and sulfur filters for nebulae, with tracking and registration to account for camera and celestial motion. Many flatbed scanners also scan spatially over time: a sensor *bar* is swept mechanically down the page, which is itself how a *one-dimensional line sensor becomes a two-dimensional image* (scanning spends time to buy the spatial axis the sensor lacks), and color can come from sequential illumination during a scan, separate passes, or spatially offset filtered sensor rows. Field-sequential color television and some endoscopes ran the same spinning-color-wheel trick in real time.

A scanner's linear sensor bar

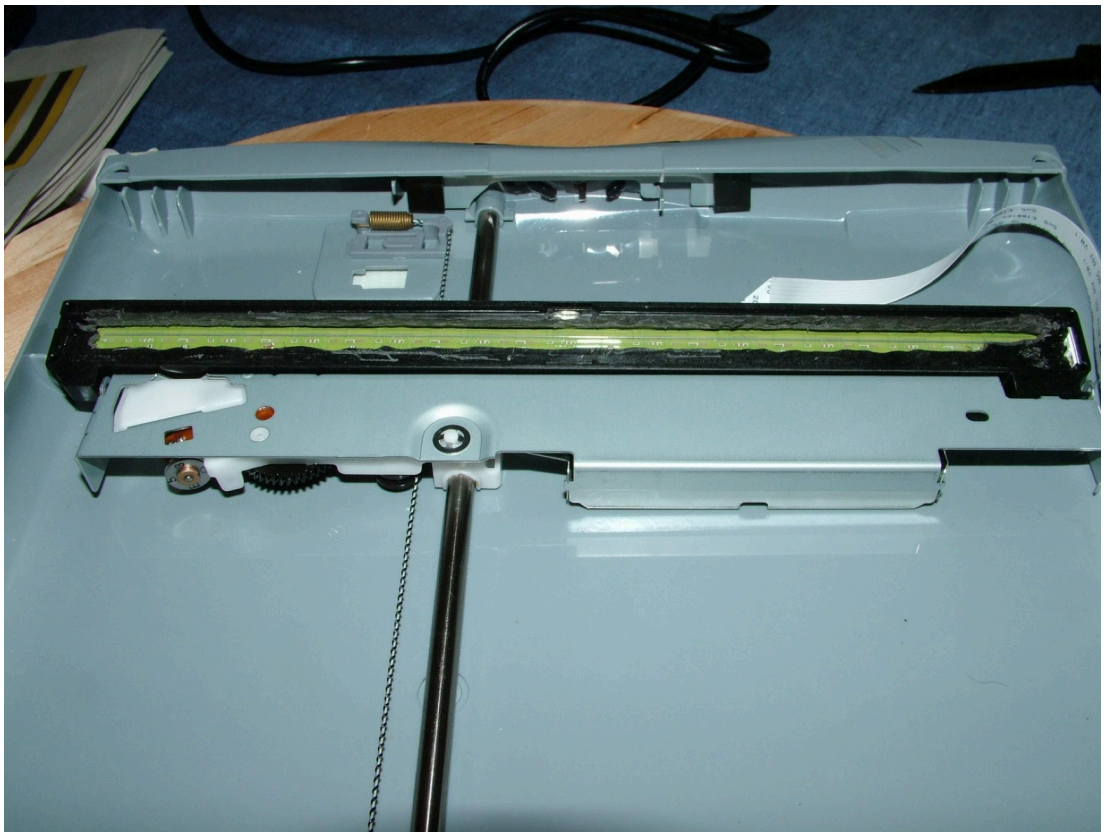


Figure 4.6.2 (reproduced). *A scanner's linear sensor bar*. The contact-image-sensor (CIS) bar of a flatbed scanner reads one line of the page at a time; the carriage sweeps it down the page, spending *time* to build the second (vertical) spatial dimension a one-dimensional sensor lacks, and color comes from three RGB-lit passes or a tri-linear sensor with three colored rows. The purest everyday example of trading a scene axis for the measurements the hardware cannot make at once. (Photo: Michele M. F. / Latente, CC BY-SA 2.0, via Wikimedia Commons.)

The trade-off is discrimination in *time*: the method assumes the scene and camera hold still across the exposures. For a static scene, each band uses the full sensor grid. The filters still reject light, and a fixed total capture time must be divided among bands. If the scene moves, its changing parts splits into colored fringes (Figure 4.6.6), and every

temporal system must first *register* the plates to undo the small shifts between them (Figure 4.6.5), the simplest image-alignment problem in the book.

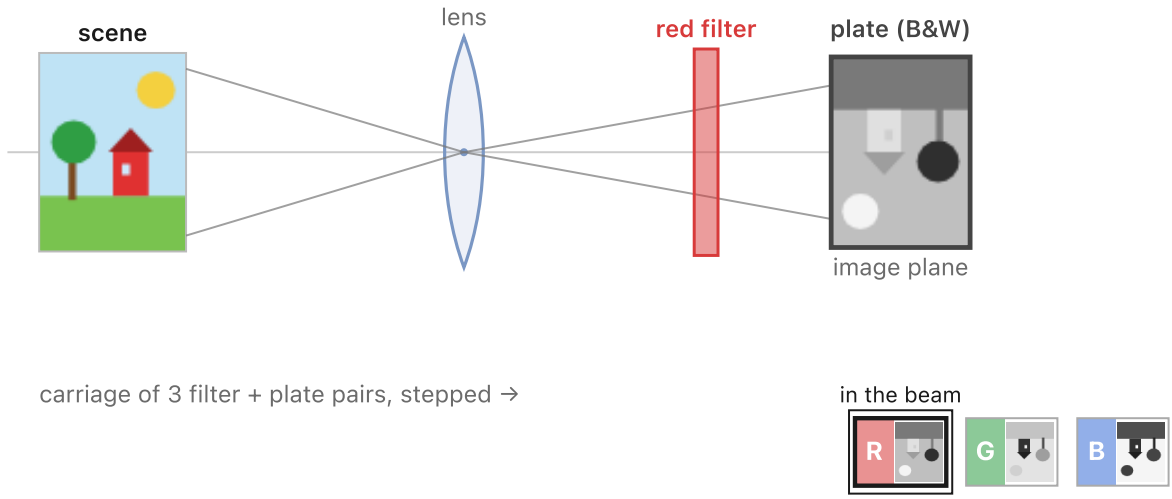


Figure 4.6.3 (reproduced). *Color multiplexing in time, following Maxwell's 1861 experiment (interactive). The lens images a simple scene onto a photographic plate at the focal plane. Three separate plates, each paired with its own red, green, or blue filter mounted between the lens and the plate, are stepped into the beam in turn, so the camera records the scene's red, then green, then blue component as three sequential black-and-white exposures (only the filter is colored; the plate records intensity, and the image is inverted, as any lens inverts). Stack the three gray records back through their filter colors and the color image reappears. No color filter array and full sensor resolution, but it works only if the scene holds still between the three shots.* Interactive: watch the carriage step each filter-and-plate pair through the beam and the three black-and-white records combine into color, or pick a single exposure.

Sidebar 1: James Clerk Maxwell and the first color photograph

The three-color principle was demonstrated in 1861 by **James Clerk Maxwell**, the physicist who also unified electricity, magnetism, and light into the equations that bear his name. To demonstrate three-primary additive color reproduction, he had the photographer Thomas Sutton shoot a tartan ribbon three times through red, green, and blue liquid filters, then projected the three black-and-white positives back through the same three filters, superimposed, onto a screen. The overlap showed color in the celebrated 1861 demonstration, and a live demonstration of trichromacy, the same three-number logic the rest of this book rests on. It very nearly should not have worked: the wet-collodion

plates of the day were almost blind to red and green, and the "red" record came mostly from ultraviolet that the ribbon's red dye happened to reflect. A lucky accident, but the principle was exactly right, with limitations imposed by the plates' spectral sensitivity. (Maxwell 1861)

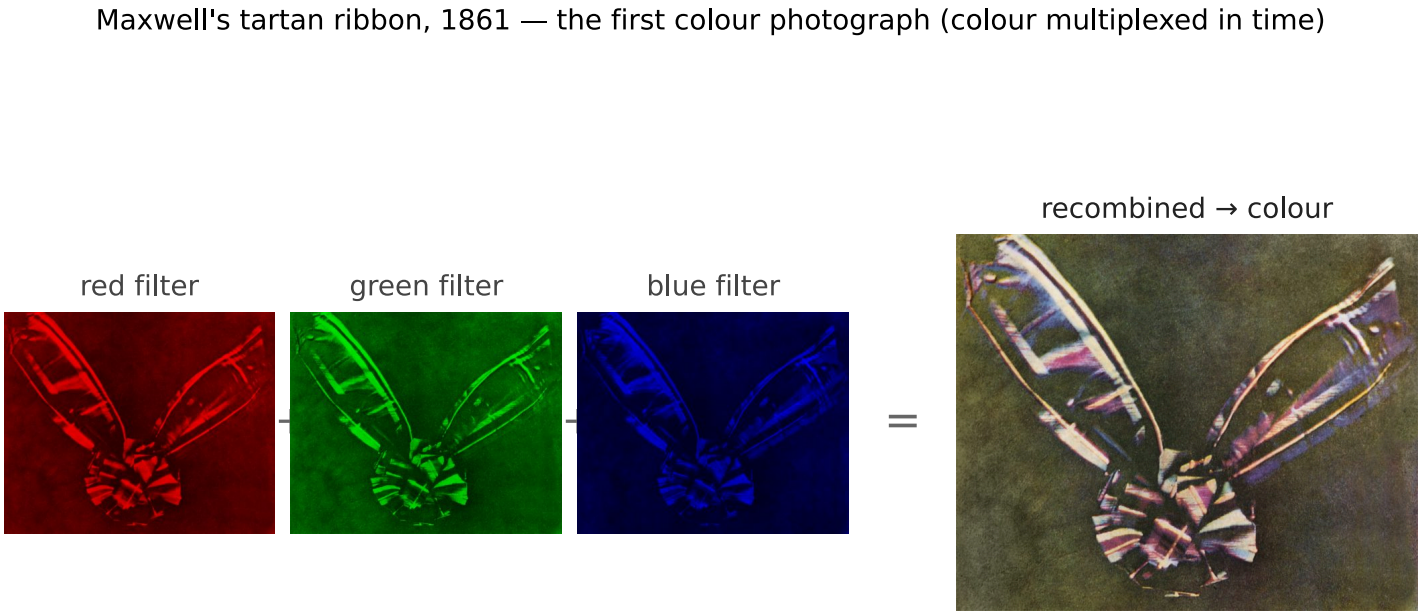
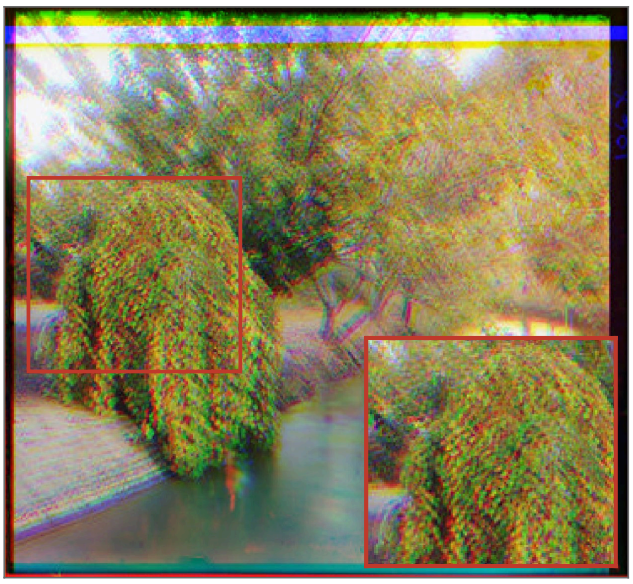


Figure 4.6.4 (reproduced). *Maxwell's tartan ribbon (1861), the first color photograph. The ribbon was shot three times through red, green, and blue filters, and the three black-and-white positives projected back through the same filters, superimposed, to reconstruct color: color multiplexed in time, the earliest example on this page. (Public domain, via Wikimedia Commons.)*

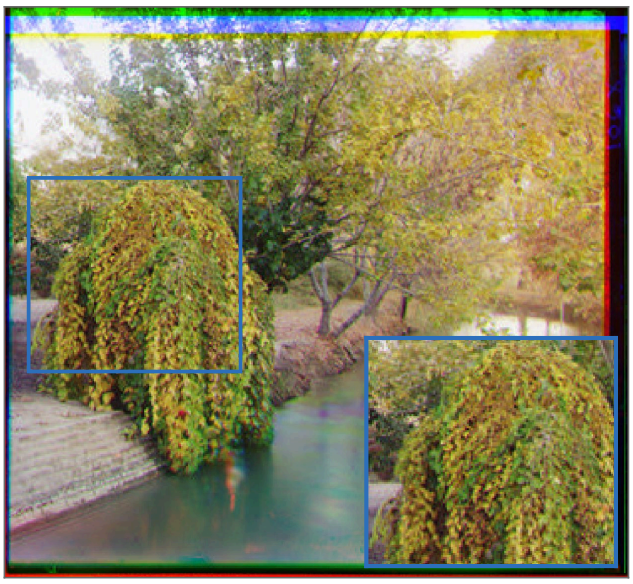
Temporal colour multiplexing (Prokudin-Gorskii, c.1910): three exposures, one per filter — then register them

(a) three plates stacked, NOT aligned



channels offset → red / blue fringes, smeared edges

(b) after registering the three plates



one true R, G, B per pixel → sharp full-colour photo

Figure 4.6.5 (reproduced). *Recovering color from the three plates: alignment is the catch. Because the red, green, and blue exposures were shot sequentially, the plates are slightly shifted relative to one another; stacked as-is (left) the color image is fringed and ghosted along every edge. Searching for the per-channel shift that minimizes the sum-of-squared differences (SSD) between channels and re-stacking (right) snaps them into registration and the true color appears, the simplest image-alignment problem in the book, and exactly the registration every temporal-multiplexing system (scanner, filter wheel, Prokudin-Gorskii plate) must perform. (Prokudin-Gorskii.)*

[figure fig-prokudin-gorskii not built]

Figure 4.6.6 (reproduced). *Temporal multiplexing fails on motion. A Prokudin-Gorskii three-filter sequential exposure: static scenery registers perfectly, but moving water appears as separated red, green, and blue color fringes, because each channel was captured a moment apart. The artifact is the price of spreading color across time. (Sourced: Prokudin-Gorskii collection, Library of Congress, public domain.)*

Multiplexing in space: the color filter array

in the book ↗

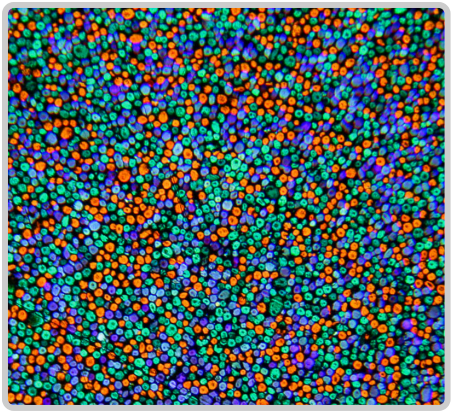
The dominant choice by far. Lay a color filter array (CFA) directly over the sensor so each photosite records one color and the missing two are filled in afterward by demosaicking (Figure 15.0.1, at the start of Part 15, Demosaicking and joint methods). The archetype is the Bayer mosaic, two green photosites for each red and blue, its doubled green mirroring the eye's green-weighted luminance (the eye is itself a spatial CFA, its interleaved cone mosaic). The family is large: Fujifilm's X-Trans uses a repeating 6×6 tile that redistributes sampling artifacts; it does not eliminate aliasing; complementary CMY or CYGM (cyan-yellow-green-magenta) arrays pass more light through each filter than the primary RGB dyes; RGBW arrays add an unfiltered panchromatic "white" pixel for low-light sensitivity; and Huawei's RYYB swaps green for yellow to gather still more light. Phone sensors with many

small pixels also group same-color photosites under one filter color. Quad-Bayer puts each color on a 2 × 2 block of photosites, and Nona-Bayer (Samsung's Nonacell) on a 3 × 3 block, so that a 108-megapixel sensor can combine each block of nine into one photosite and deliver a 12-megapixel image with less noise in low light, or read every photosite in good light. Chapter 15.5, Alternative color filter arrays covers how these layouts are demosaicked.

The reconstruction uses spatial and cross-channel regularity to estimate missing samples. Fine colored detail can violate those assumptions, producing false color and zipper artifacts. An optical low-pass filter can reduce aliasing at the cost of detail; not every camera uses one. A CFA also loses photons in its absorbing filters. Its practical advantage is simultaneous capture on one compact sensor.

Color film reached the same spatial idea decades before silicon, by literally the same trick. The **additive screen processes**, the Lumière brothers' **Autochrome** (1907) and later Dufaycolor, Finlay, and Paget, coated the plate with a fine mosaic of transparent grains or patterned filters (Autochrome used orange-red, green, and violet-blue potato-starch grains; Dufaycolor printed a regular réseau grid) over a single panchromatic black-and-white emulsion. Each grain passed only its own color to the silver behind it, so the developed plate, viewed back through the same mosaic, rebuilt color from a spatial patchwork of filtered samples. This is a **color filter array in film**, the direct ancestor of the Bayer mosaic, decades before electronic CFAs, with each dyed grain filtering the emulsion behind it and vision integrating the fine mosaic at viewing distance. (The *other* film tradition, the stacked-dye *integral tripack* of Kodachrome and modern color film, multiplexes color in *depth* instead, and so belongs with the Foveon below.)

The Lumière Autochrome (1907): a random-tiled colour filter array of dyed starch grains, an early additive-screen colour process. **Click a photo to enlarge.**



a real Autochrome plate (photomicrograph of the dyed grains)



an original box of Lumière Autochrome plates

Photos credited in the book's image credits.

Figure 4.6.7 (reproduced). *Autochrome: a color filter array made in film, a century before Bayer.* Left, the additive-screen process: white light from the scene passes through a random mosaic of transparent grains dyed red, green, and blue (scattered dyed potato-starch, in the Lumière process) onto a single panchromatic black-and-white emulsion behind, so each grain acts as one photosite's color filter; viewed back through the same mosaic, the plate rebuilds color from the spatial patchwork, the eye doing the demosaicking. Center, a photomicrograph of a real Autochrome plate showing the actual random dyed-grain mosaic. Right, an original box of Lumière autochrome plates. A spatial color filter array in film, random-tiled, the direct ancestor of the silicon Bayer mosaic. (Micrograph: Janke, CC0; box: Anne Jea, CC BY-SA 4.0; both via Wikimedia Commons.)

Multiplexing across separate sensors: the beam-splitter

OPTIONAL
in the book ↗

Send the whole image to more than one sensor and give each its own color. A dichroic prism block reflects each band onto its own chip (Figure 4.6.10): the 3-CCD / 3-chip design long standard in broadcast video cameras, three-chip cinema and telecine cameras, and scientific and machine-vision multi-sensor rigs. Because the dichroic coatings *split* the light rather than *absorb* it, the separation can retain more useful light than absorbing filters, and each channel has its own full sensor grid. (It is a *dichroic* beam-splitter, not a dispersing prism: it peels off three clean bands, it does not smear a spectrum.)

This strategy spends no scene axis, so it makes none of the constancy or smoothness assumptions of the other two: simultaneous sampling on three full grids without CFA demosaicking. Instead you pay hardware complexity: three sensors registered to sub-pixel accuracy behind a bulky, expensive prism. Bulk, optical tolerances, and cost limit its use. Adding bands increases the complexity.

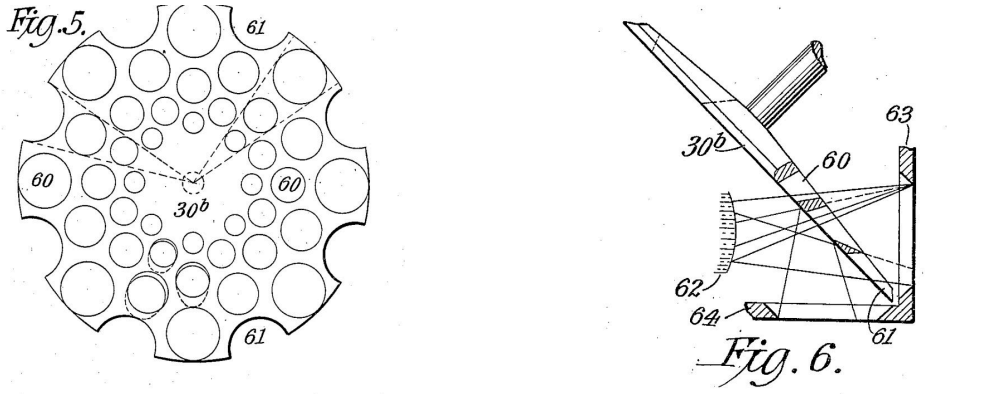
The hardware-splitting idea predates its dichroic form. Before interference coatings existed, P. D. Brewster's historical beam splitter divided a movie camera's beam with a polished metal disc drilled full of holes (Figure 4.6.8): light passing through the holes went straight to one color film gate, while light striking the solid metal between

them reflected off to a second. It is the cheese-grater ancestor of the 3-CCD prism, the same bargain of spending hardware rather than a scene axis so that each color gets its own sensor. It was cruder in two ways: the fixed metal-to-hole ratio sets a single reflect/transmit split rather than a wavelength-selective peel, and because the disc splits by geometry and not by color, each path still needs its own color filter. So it lacks the dichroic prism's wavelength-selective separation, but the core move is identical.

Brewster's “cheese-grater” beam splitter: color by a mirror full of holes (1921)

The perforated disc, face-on

At 45°, splitting the beam to two gates



Light hitting the solid metal (61) reflects to one gate; light through the holes (60) passes to the other, no coating needed.

Figure 4.6.8 (reproduced). *Brewster's "cheese-grater" beam splitter (1921): color by a mirror full of holes.* Before dichroic coatings, P. D. Brewster split a movie camera's beam with a polished metal disc perforated with holes. Left, the disc face-on: light passing through the holes (60) is transmitted, light striking the solid metal (61) is reflected. Right, set at 45° in the camera it routes the reflected beam to one color film gate and the transmitted beam straight on to another, a beam split by nothing but metal and holes, and an early ancestor of the dichroic 3-CCD prism. The fixed metal-to-hole ratio sets only one reflect/transmit split and each path still needs its own color filter, so it lacks the dichroic prism's clean, lossless separation, but it is the same core idea: spend hardware, not a scene axis, to give each color its own sensor. (US Patent 1,752,477, public domain.)

A dichroic coating separates bands through interference. The answer is the same interference that colors a soap bubble (Figure 2.1.33), run deliberately and many times over. The coating is a **quarter-wave stack**: alternating layers of a high- and a low-index transparent material, each with optical thickness $n_j d_j = \lambda_0/4$ at normal incidence in the simple quarter-wave design. Every interface reflects a little, and at λ_0 those weak reflections all come back *in phase* and add into a near-perfect mirror; outside the reflection band the coating transmits more light, with the detailed spectrum set by its complete layer design. In an ideal lossless coating, reflectance plus transmittance is one. Real stacks also have absorption, scattering, angular dependence, and polarization dependence (Figure 4.6.9).

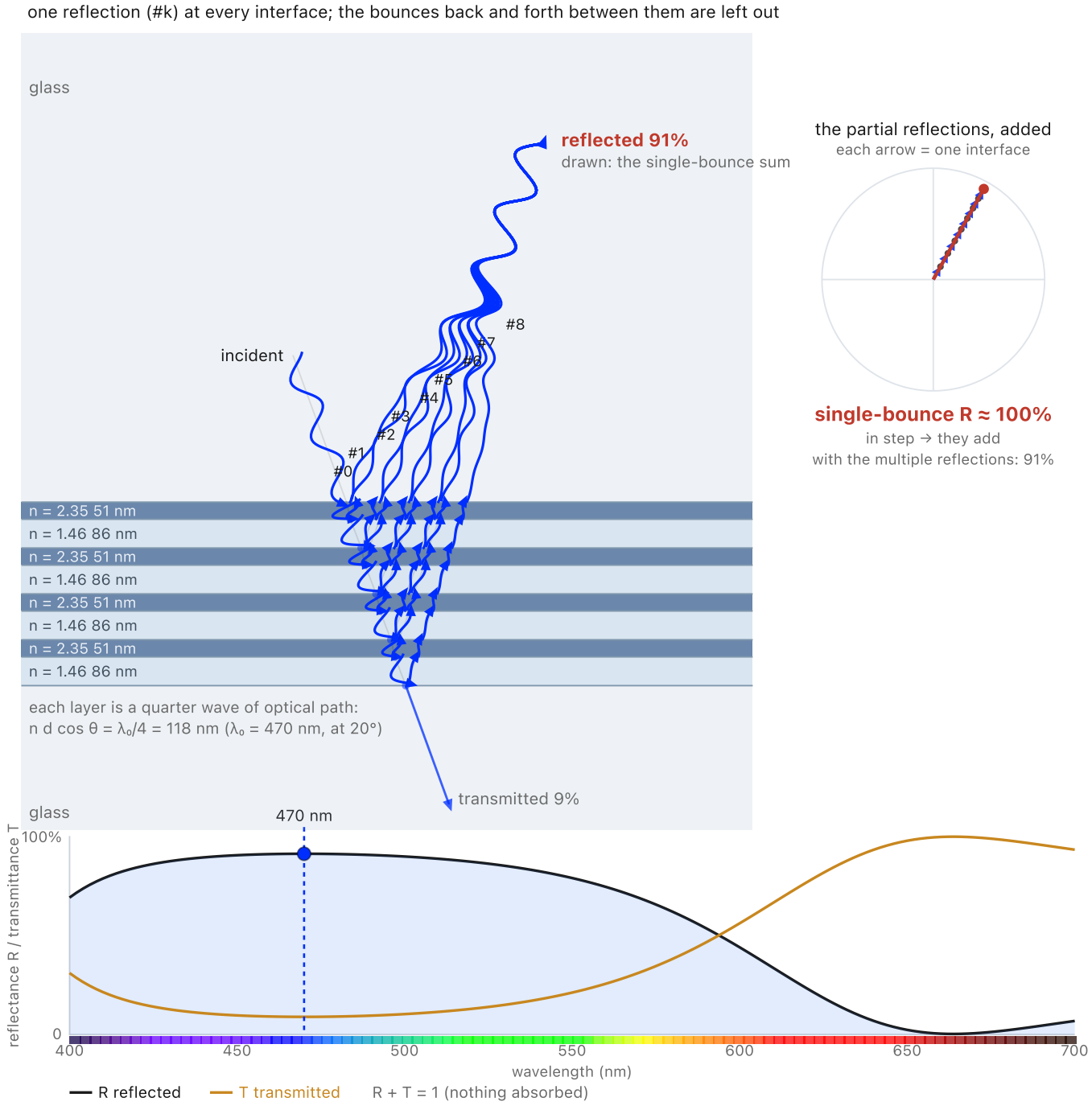


Figure 4.6.9 (reproduced). *What the dichroic coating does, at the interface. The beam-splitter figures above trace whole beams through the block; here we zoom in on the coating that does the work, and cut it down to an eight-layer stack so the mechanism is visible, each layer a quarter wave at 470 nm for the stack's 20° diagonal ($nd\cos\theta = \lambda_0/4$). A single wavelength, the one control, meets the stack and you can follow it interface by interface: at each one a little reflects and the rest is transmitted, so N layers send back $N + 1$ partial waves, each deeper one having made an extra round trip and so returning shifted in phase. The* ***phasor sum*** *beside the diagram adds them tip to tail: change the wavelength, and when the round trip works out to a half wave the arrows line up and the reflections reinforce, while away from it they curl round and cancel. Even eight layers reflect 91% at 470 nm, computed exactly with the transfer-matrix method and printed beside the beams with the transmitted share; real dichroics stack*

dozens for a sharper, flatter band. Both spectra are plotted, reflected and transmitted, and they sum to one at every wavelength because nothing is absorbed. The phasor sum counts each partial wave reflecting once and leaves out the bounces between interfaces, the same simplification the thin-film figure makes, which is why the exact value is shown under it.

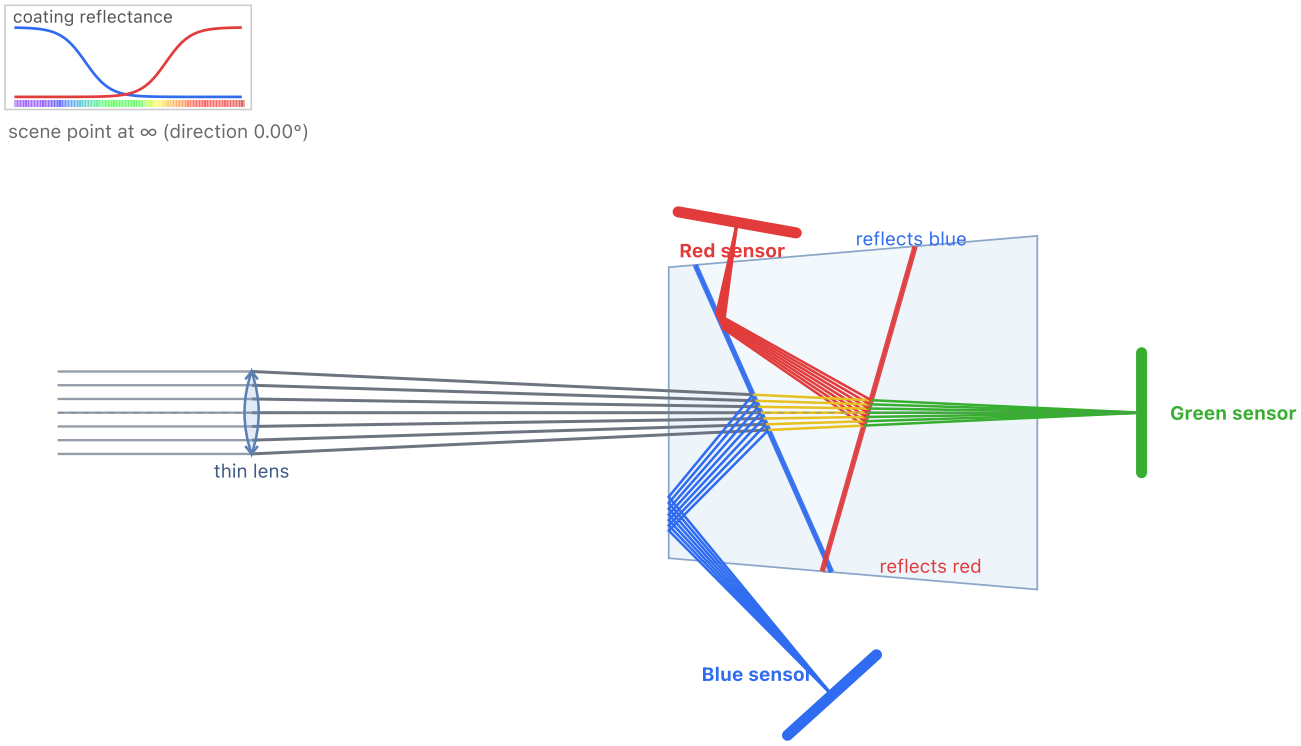
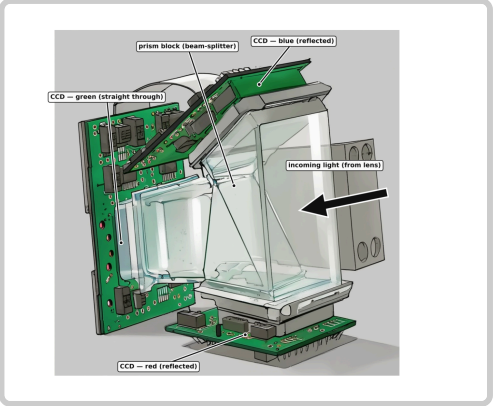


Figure 4.6.10 (reproduced). *The trichroic (3-CCD) prism (interactive ray trace).* A Philips-type beam-splitter sends one lens image to three sensors at once. The collimated beam enters a glass block and meets two tilted dichroic coatings: the first *reflects* blue and passes the rest, the second reflects red, and green goes straight through to the sensor behind. Because each band is peeled off by reflection rather than by an absorbing filter, essentially no light is wasted and every sensor keeps full resolution. Drag the wavelength to send one color through and see which coating catches it, or switch to white light and watch all three split at once. Real coatings do not switch at a hard edge: their reflectance rolls over a transition band tens of nanometers wide, so a wavelength near a crossover splits between two sensors: near 495 nm it divides between blue and green, near 580 nm between green and red. A "camera" view shows a thin lens imaging a scene point through the whole assembly onto the three chips; switch to "prism only" for just the coating-and-fold schematic. (Real Philips prisms add a thin air gap so the reflected bands totally-internally-reflect and the three chips pack compactly against the block; the loss-free dichroic split is the essential physics.) Interactive: pick a single wavelength (short → blue path, long → red path, middle → green) or white light to see the beam split three ways at the dichroic coatings.

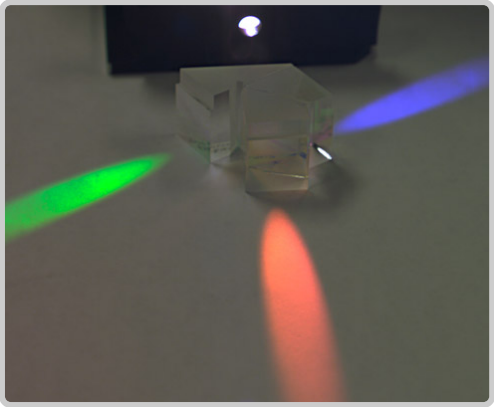
Real trichroic (3-CCD) colour-separation prisms: dichroic coatings split the light into red, green, and blue for three sensors. **Click a photo to enlarge.**



the real 3-CCD prism block



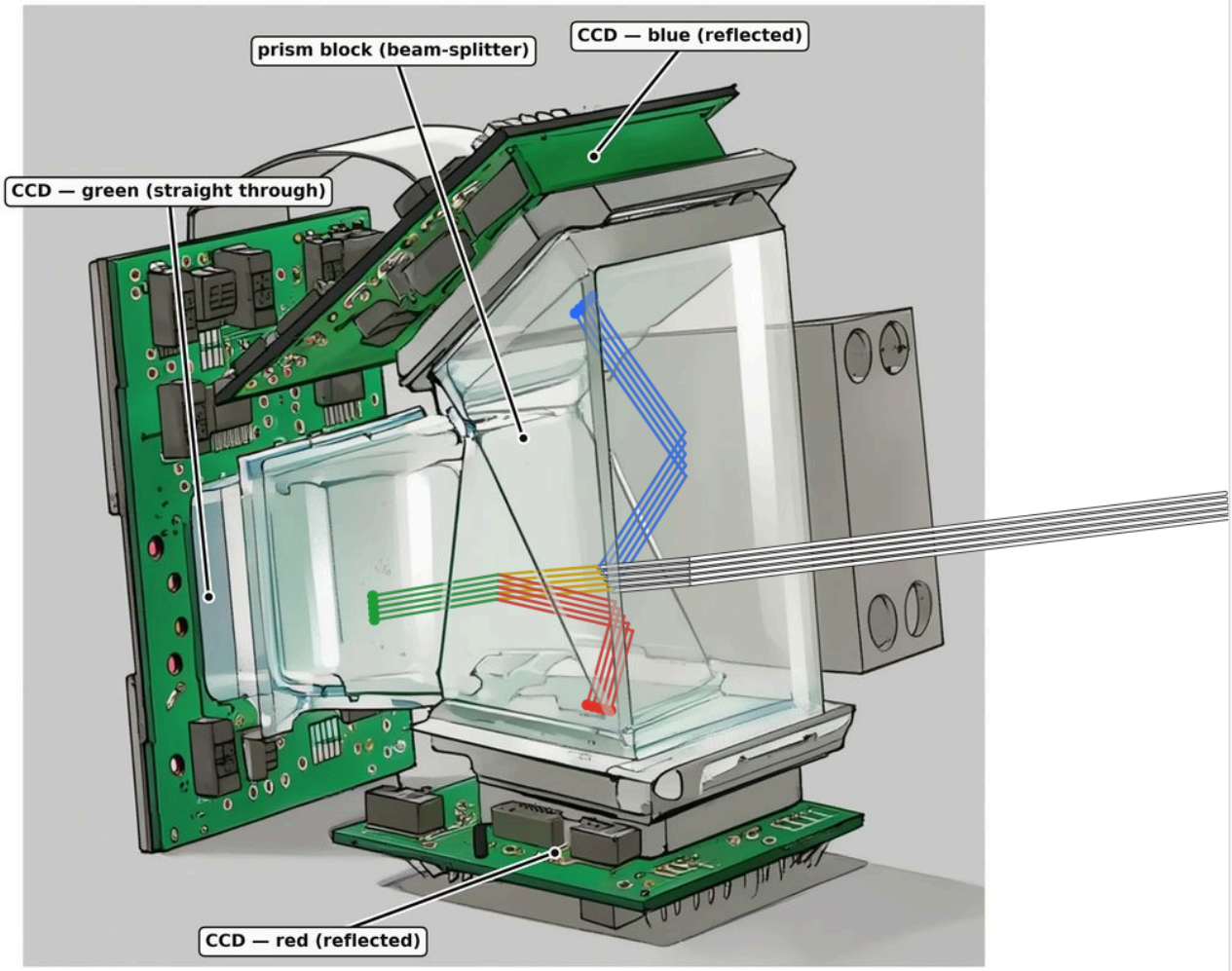
the same block, clarified and labeled



a bare prism splitting light by colour

Photos credited in the book's image credits.

Figure 4.6.11 (reproduced). *A trichroic (3-CCD) color-separation prism, and a bare prism splitting light.* The physical counterpart to the ray trace. (a) The real thing: a broadcast camera's Philips-type prism block, a solid glass block whose internal dichroic coatings split the image into red, green, and blue and route each to its own 2/3-inch CCD bolted to a face; the bulk and the sub-pixel alignment of three chips are exactly the cost that kept this design out of consumer cameras. (b) The same block clarified and labeled (redrawn from the photo, since the real thing is dark and hard to read), so the glass prism block, the three CCD sensors, and the front aperture where the lens's light enters are easy to pick out. (c) A bare separation prism with no sensors: white light enters and leaves as three clean red, green, and blue beams, the loss-free dichroic split made visible. This is a generic demonstrator, not a Philips block: here the beam that passes straight through, least deviated, is red, with green and blue folded aside. In the Philips arrangement of the ray trace above it is instead green that goes straight through to the rear sensor, while red and blue are the reflected channels. Which color is kept inline is a coating design choice, not a fixed rule; green is the usual pick in a 3-CCD camera because it carries most of the luminance detail. (Panel (a) photo Xingbo, CC BY-SA 3.0; (b) our own annotated redrawing of (a); (c) photo Dick Lyon, public domain; via Wikimedia Commons.)



white light wavelength λ 550 nm

height 470 px direction 7.0° beam width 20 px

Figure 4.6.12 (reproduced). *The 3-CCD prism's light paths, traced on the real block. The same beam-splitter as the schematic above (Figure 4.6.10), with a 2-D ray trace laid over the labeled illustration of the actual prism. White light enters from the right and refracts into the glass; the first dichroic coating reflects the blue band up to the blue CCD, the second reflects red down to the red CCD, and green passes straight through to the green CCD, each color landing on its own chip. The rays turn slightly gray once inside the block so they read against the busy hardware, and pass behind the front glass and boards that stand in front of them. (Ray trace ours, drawn over the labeled illustration from the previous figure; underlying art a CC BY-SA 3.0 derivative of Xingbo's photo.)* Interactive: switch between white light and a single wavelength, and set the incoming ray's height, direction, and beam width, to watch blue peel up, red down, and green pass straight through onto the three chips.

Multiplexing per pixel by dispersion: color routers and nano-prisms

OPTIONAL
in the book ↗

A color router directs wavelength bands toward different photodiodes instead of rejecting unwanted bands with absorbing filters. Demonstrations range from dielectric splitters Nishiwaki et al. 2013 to inverse-designed structures Zou et al. 2022. The target is better photon utilization, but reported efficiency depends on wavelength, incidence angle, polarization, aperture, and the definition of useful detector area. One-third transmission is a teaching approximation for idealized equal-width RGB filters, not a universal CFA ceiling.

Routing to neighboring photodiodes still multiplexes color in space. The spatial response must be included in reconstruction; spreading light is not automatically a correctly designed anti-aliasing filter. Fabrication tolerances and off-axis response also matter. Commercial microlens or nanostructure enhancements that retain color filters should be distinguished from filter-free research routers.

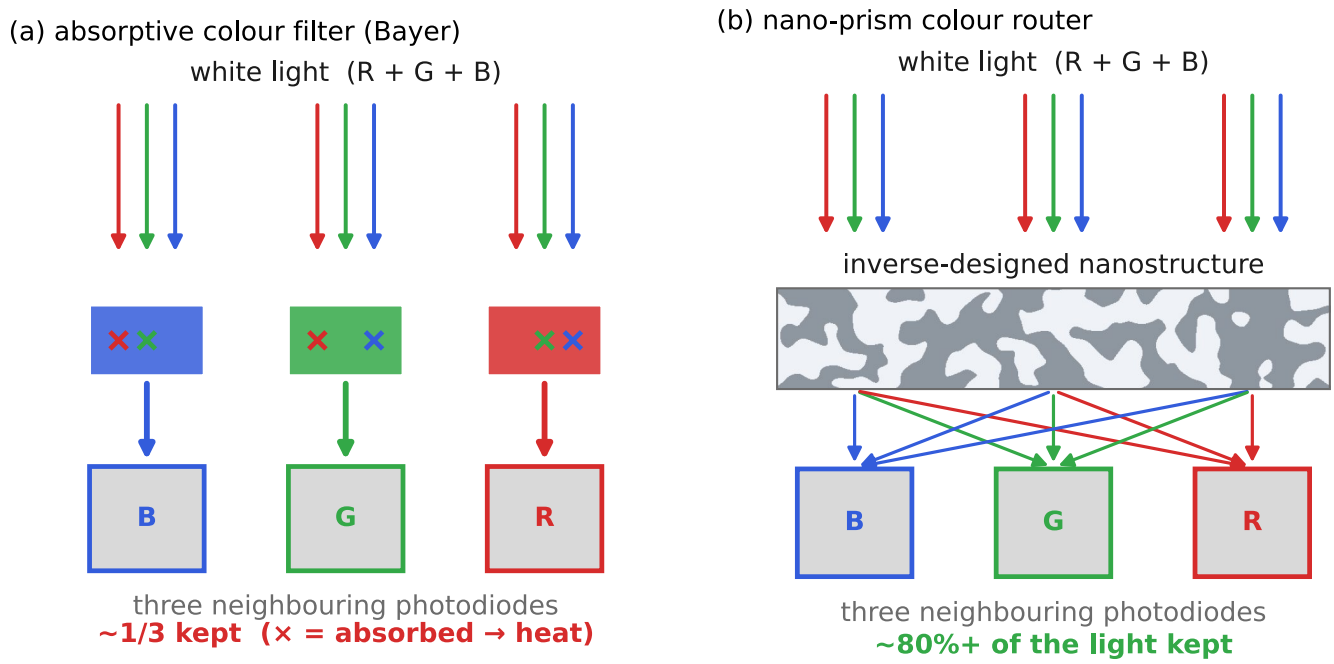
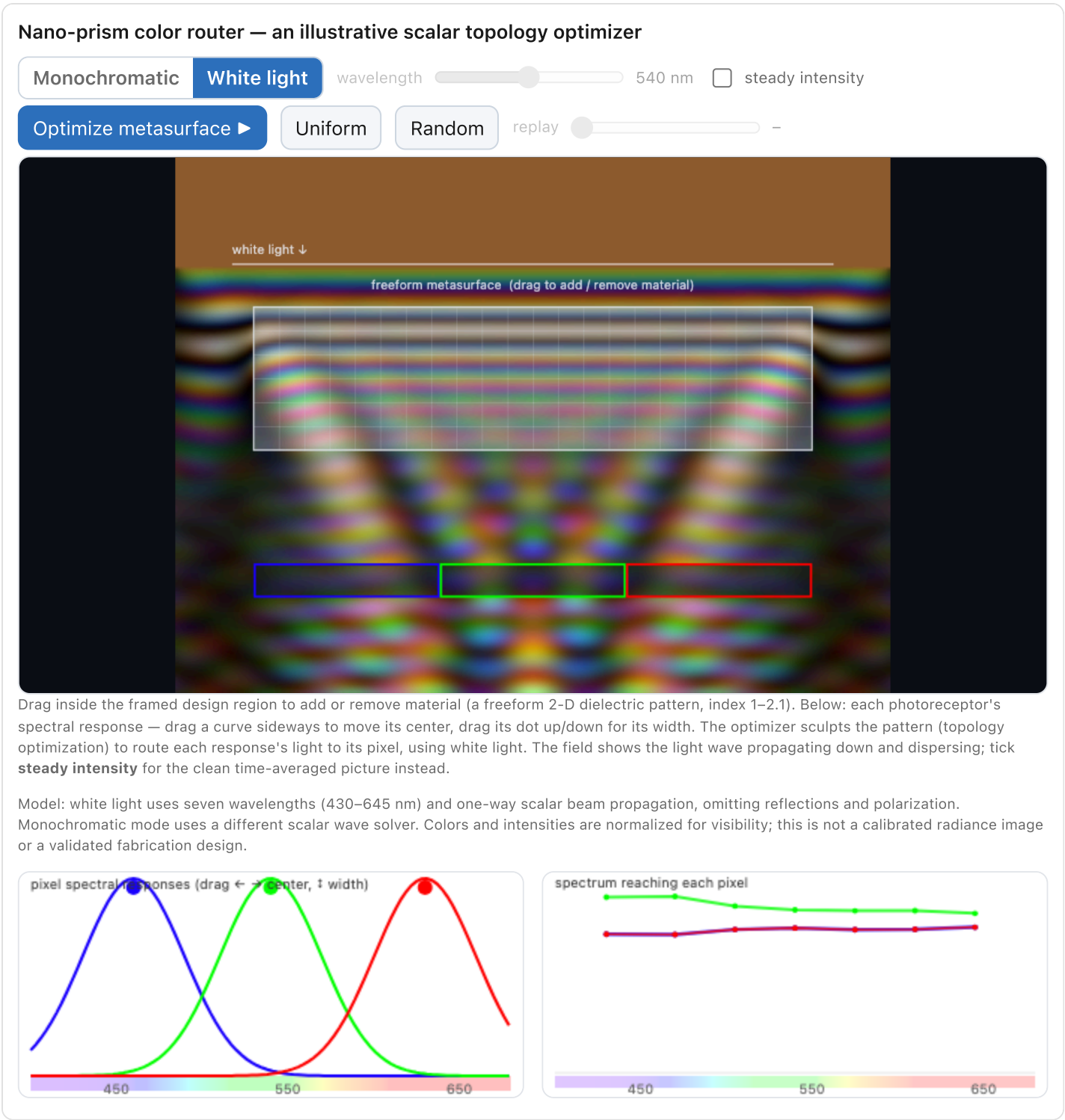


Figure 4.6.13 (reproduced). *The nano-prism color router versus the absorbing Bayer filter*. Left, an absorptive dye filter passes its own color to the photosite and throws away the other two-thirds of the light (the CFA's ~33% ceiling). Right, an inverse-designed nanostructure disperses the incident light instead, steering each wavelength toward a *different* neighboring photodiode, so most of the light is kept (efficiency climbs toward 80% and beyond). Color is still multiplexed across space, Bayer-style, but by dispersion rather than masking, the broadcast prism shrunk to a per-pixel scale and laid flat on the sensor. The same spreading smears each point's colors onto different neighbors, a built-in mild anti-alias that becomes resolution-killing crosstalk if pushed too far. (The idea runs from Panasonic's 2013 color splitters to inverse-designed metasurface routers; Samsung's "Nanoprism" is the first shipping cousin.)



Drag inside the framed design region to add or remove material (a freeform 2-D dielectric pattern, index 1–2.1). Below: each photoreceptor's spectral response — drag a curve sideways to move its center, drag its dot up/down for its width. The optimizer sculpts the pattern (topology optimization) to route each response's light to its pixel, using white light. The field shows the light wave propagating down and dispersing; tick **steady intensity** for the clean time-averaged picture instead.

Model: white light uses seven wavelengths (430–645 nm) and one-way scalar beam propagation, omitting reflections and polarization. Monochromatic mode uses a different scalar wave solver. Colors and intensities are normalized for visibility; this is not a calibrated radiance image or a validated fabrication design.

Figure 4.6.14 (reproduced). *Design your own nano-prism. Interactive: paint a **freeform 2-D dielectric pattern** into a rectangular design region (drag to add or remove high-index material, index n ≈ 2.1, the way a real metasurface router is built) and watch light disperse through it onto three photoreceptors, each with its own tunable Gaussian **spectral response** (drag to set its center and width). Toggle between a single wavelength and white light: the monochromatic view shows a live 2-D **wave simulation** (scalar FDTD), while white light is propagated with the same **beam-propagation method** the optimizer uses, so what you see is what it optimizes. Press **optimize** to inverse-design the pattern by **topology optimization** (gradient ascent with basin hopping, run on white light) so it routes each pixel's target color onto it, and drag the slider to replay the design search. The color-*

router idea made tangible: with the full 2-D freedom all three pixels sort their color reasonably cleanly, though the separation is never quite perfect, the residual crosstalk that limits real routers. The simulation is an illustrative two-dimensional scalar model with a continuous index, seven wavelengths in white-light mode and normalized false-color intensity, not a quantitative electromagnetic simulation or a design ready to fabricate.

The comparison with a dichroic prism is useful, provided we keep track of the image coordinates. A prism can send complete registered images to separate sensors. A small router sends different wavelengths toward different sites within a local footprint. The latter can avoid absorbing some of the photons that a dye would reject, but neighboring detectors now receive contributions shaped by the router's wavelength-dependent spatial response. The reconstruction must know both which wavelengths and which scene locations contributed. Improving light collection and preserving spatial detail are therefore coupled design tasks, not two independent benefits of replacing a filter.

Multiplexing per pixel by tunable absorbers: quantum dots

OPTIONAL
in the book ↗

Colloidal quantum dots offer size- and material-dependent bandgaps and can form photosensitive layers above a readout circuit. They are useful research and device materials for spectral ranges and integration schemes that differ from silicon. Color-selective layers still need multiple independent measurements, whether distributed spatially or stacked.

A tunable bandgap does not imply an arbitrarily narrow absorption spectrum. Quantum dots can absorb broadly at energies above the band edge, even when their fluorescence is narrow. Nor does replacing a dye with another absorber automatically recover rejected photons. Evaluate absorption, charge extraction, dark current, stability, and cross-channel separation for the proposed architecture.

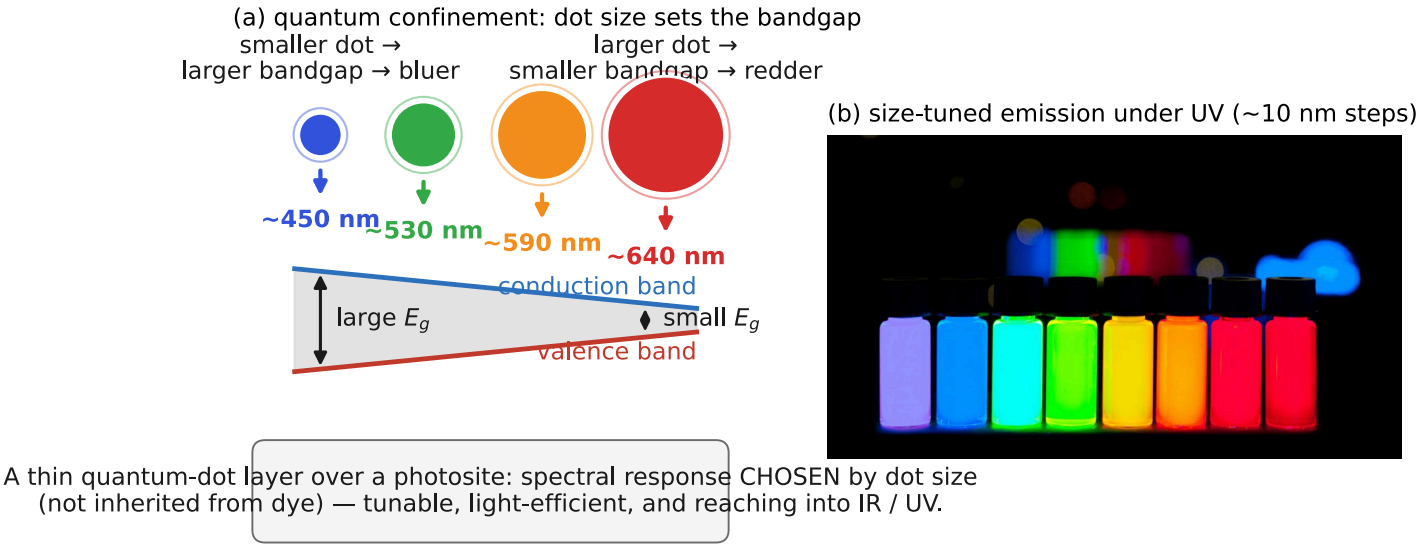


Figure 4.6.15 (reproduced). *Quantum dots: a color-sensing material you tune by size.* Left, quantum confinement: a semiconductor nanocrystal's absorbed (and emitted) wavelength is set by its *diameter*: a smaller dot has a larger bandgap and works bluer, a larger dot a smaller gap and redder, so a per-pixel quantum-dot layer has a spectral response the designer *chooses*, rather than inheriting it from dye chemistry, and can reach into the infrared and ultraviolet that silicon handles poorly; as an absorber it still discards the light outside its band, so it does not by itself recover what a dye filter rejects. Right, the canonical demonstration: vials of colloidal quantum dots fluorescing violet through red under ultraviolet light, in even size steps. (Photo: Alexei Antipov / PlasmaChem GmbH, CC BY-SA 3.0, via Wikimedia Commons.)

Multiplexing in depth: stacked photodiodes

OPTIONAL
in the book ↗

Stacked photodiodes use silicon's wavelength-dependent absorption depth. Short wavelengths are absorbed near the surface; longer wavelengths penetrate farther. The layer signals overlap spectrally, so they require a calibrated color transform rather than being direct red, green, and blue measurements. Foveon designs use this principle; some variants sample deeper layers at lower spatial resolution and still require reconstruction. Noise after color correction depends on the measured layer sensitivities, collection efficiency, and covariance.

Few production cameras have ever used a Foveon sensor; the Sigma dp series is the best known (Figure 4.6.16), sold on exactly the trade above: unusually crisp, demosaicking-free luminance detail in exchange for noisier color, device-dependent sensitivity and readout limits.

A stacked-photodiode (Foveon) sensor camera



Figure 4.6.16 (reproduced). *A camera built around a stacked-photodiode (Foveon) sensor: the Sigma dp2. Rather than a Bayer mosaic that samples one color per pixel across space, the Foveon stacks three photodiodes at different depths in the silicon and reads all three colors at every pixel: longer wavelengths penetrate deeper, so the shallow layer sees mostly blue, the middle green, the deep red. Color is multiplexed in depth, so there is no demosaicking and little color moiré, though the luminance can still alias, at the price of noisier, less cleanly separated channels.*

For a simplified collection interval from depths z_1 to z_2, Beer–Lambert absorption gives the fraction $e^{-\alpha(\lambda)z_1} - e^{-\alpha(\lambda)z_2}$, where α is silicon's absorption coefficient. This illustrates overlapping depth responses (Figure 4.6.17). Real charge collection also depends on device structure and carrier transport. A broad overlap can make some color distinctions noisy; quantify that through the calibrated transform rather than an invented "unmixing" matrix.

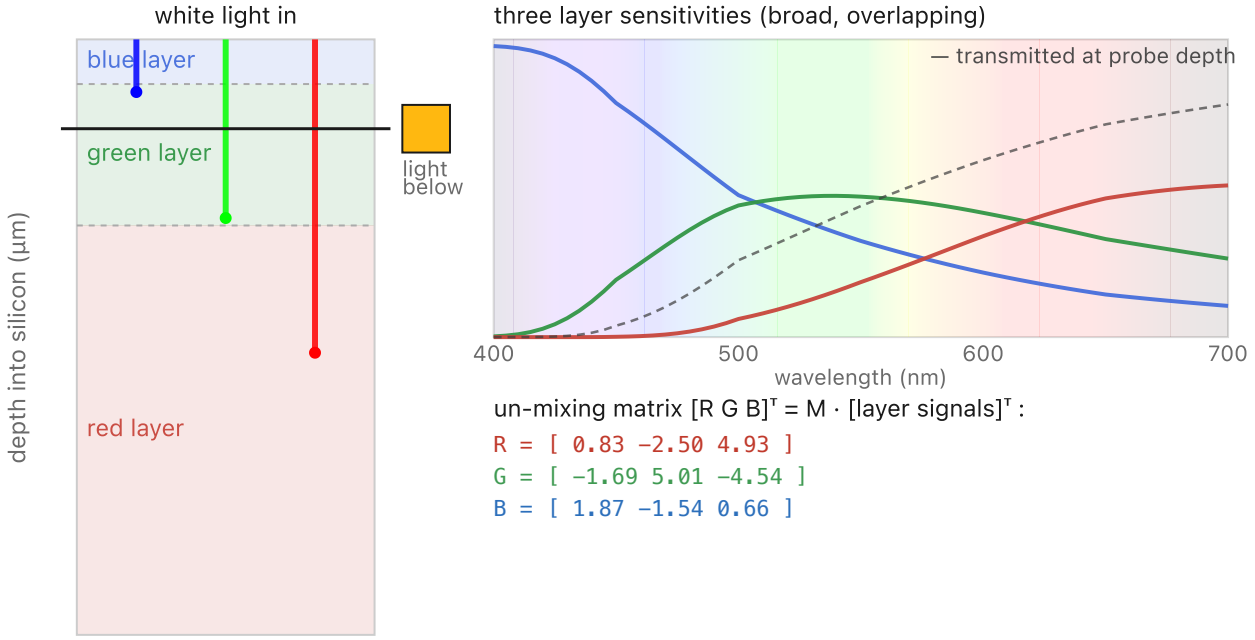


Figure 4.6.17 (reproduced). *Silicon absorption by depth and the Foveon's overlapping color channels (interactive). Blue wavelengths are absorbed within about the first half-micron of silicon, green within a couple of microns, and red only several microns down, since silicon's absorption length climbs steeply toward the red. Left: a cross-section of the silicon with the three stacked photodiode layers (blue near the surface, then green, then red); drag the* ***probe depth*** *and watch the light remaining below it redden as blue and then green are absorbed above. Right: the three resulting* ***layer sensitivities****, broad and heavily overlapping, nothing like clean, well-separated sensitivity bands; and the* ***un-mixing matrix*** M *that recovers RGB from the layer signals, whose large off-diagonal terms make it ill-conditioned and noise-amplifying, the reason Foveon images carry crisp luminance but noisy chroma in the shadows. (Silicon absorption from tabulated penetration depths; the matrix is representative, not a specific device's calibration.)* Interactive: drag the probe depth to redden the transmitted spectrum with depth; the layer sensitivities and the un-mixing matrix are shown alongside.

Depth multiplexing is not a silicon invention; color film got there first, chemically. The dominant color-film design, the **integral tripack** of Kodachrome, Ektachrome, and typical C-41 color negatives, stacks three color-sensitive layer groups, often containing several emulsions, on one base: a blue-sensitive layer on top, then a green-sensitive layer, then a red-sensitive layer, with a thin **yellow filter layer** just below the top one to stop stray blue (to which all silver halides respond) from leaking into the lower layers. White light enters the top and is caught band-by-band with depth, a similar stacked layout, though spectral sensitization in film differs from absorption-depth separation in silicon, and after development each layer forms its own complementary dye so the stack yields full color at every point with no spatial mosaic (Figure 4.6.18). It is the same trade the Foveon makes and the same one the Autochrome does *not*: color multiplexed in depth (stacked, full-resolution, no demosaicking) rather than across space (a mosaic that must be interpolated). Silicon's Foveon is the integral tripack rebuilt in a semiconductor, absorption depth standing in for the sensitized layers.

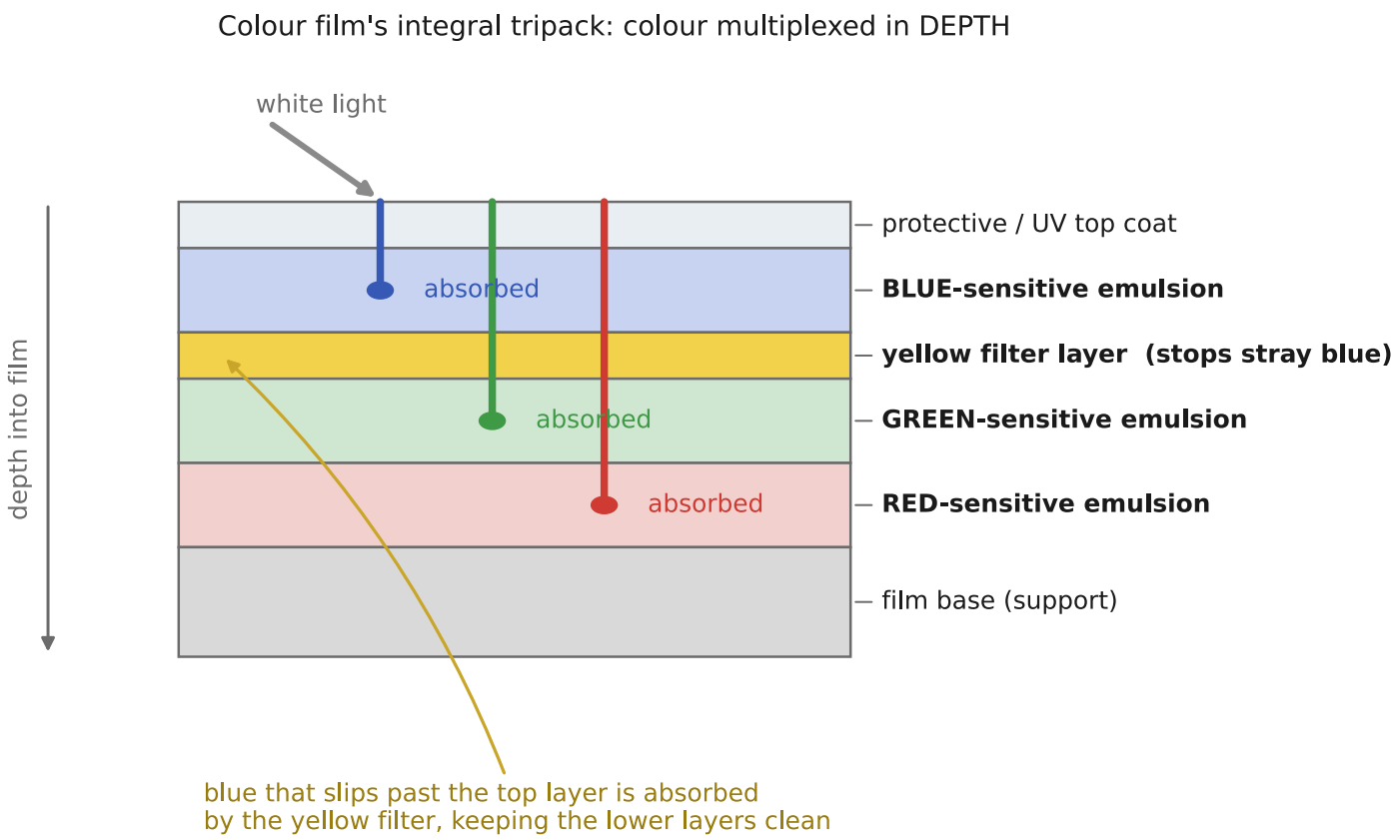


Figure 4.6.18 (reproduced). *Color film multiplexes color in depth.* Cross-section of an integral-tripack color film: light enters the top and is caught band by band as it descends: blue in the top emulsion, green in the middle, red at the bottom, with a yellow filter layer below the blue-sensitive layer to keep stray blue out of the lower two. Each layer develops its own dye, so the stack records full color at every point with no spatial mosaic and no demosaicking: the chemical ancestor of the Foveon stacked-photodiode sensor, and the depth-multiplexing counterpart to the Autochrome's spatial screen.

The depth layers are not three bins labeled blue, green, and red. Absorption is probabilistic: at a given wavelength, some photons are absorbed near the surface and others penetrate farther. A shallow collection region therefore receives longer-wavelength contributions as well as short-wavelength ones; deeper regions measure the surviving mixture. The equation above computes the difference between the fractions that reach the top and bottom of an idealized collection interval. This is why three layers need a calibration rather than merely being renamed RGB. When a desired color distinction changes the layer signals only slightly, recovering it can require a large difference of noisy measurements, connecting the sensor architecture to the matrix-noise calculation in Chapter 4.4, RGB color spaces.

Hybrid strategies

OPTIONAL
in the book ↗

Real cameras often combine strategies, and the most useful hybrid is Bayer plus pixel-shift (Figure 4.6.19). A sensor-shift actuator moves through a specified sequence of one-photosite offsets, so that over four exposures every output location is sampled through a red, a blue, *and* both green filters in turn; merge the four and you get true R, G, B at every pixel with no demosaicking guesswork (full color resolution and additional samples for noise reduction) at the cost of needing a static scene on a tripod. (It is the spatial Bayer trick borrowing the *temporal* axis to fill its own holes.) Subpixel offsets can support super-resolution if the optics retain the detail and registration is accurate. Motion and occlusion require detection or fallback processing; more frames also mean more capture time. Video pipelines hybridise the other way, leaning on motion across frames; astrophotography filter wheels are temporal multiplexing with cooled sensors.

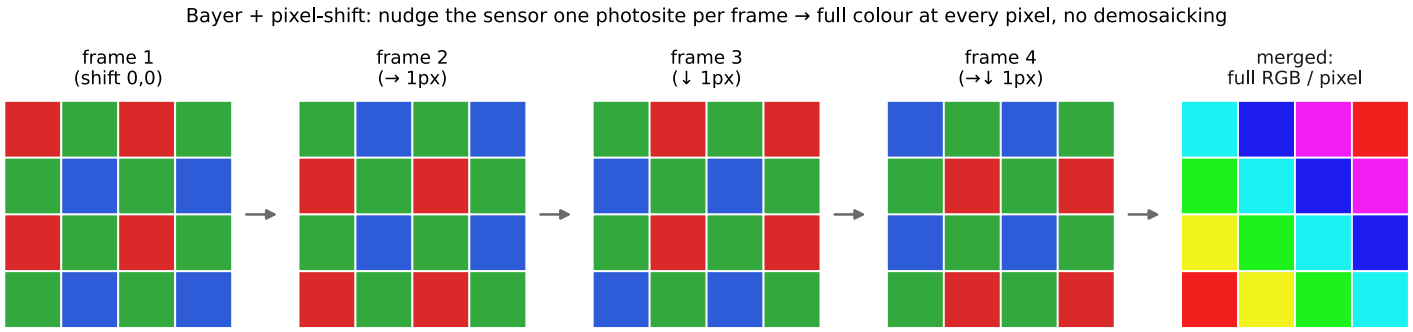


Figure 4.6.19 (reproduced). *Bayer + pixel-shift. A piezo nudges the sensor by one photosite between frames, so the four shifted Bayer frames together cover every output location with a red, a blue, and both green filters; merging them yields full R, G, B at every pixel with no demosaicking: true color resolution and lower noise, for static scenes only. Shifting by half a photosite instead reaches past the sensor's nominal resolution (multi-frame super-resolution).*

Beyond trichromatic capture: full spectrum and multispectral

OPTIONAL
in the book ↗

Multispectral instruments acquire more than three bands; hyperspectral systems sample many closely spaced bands. Filter wheels, scanned spectrometers, filter mosaics, and snapshot spectral encodings trade spectral detail against spatial sampling, time, and noise. A spectral cube is still a finite set of measurements with a calibrated spectral response, not an exact continuous spectrum.

Lippmann photography records spectral information through interference along the depth of an emulsion Lippmann 1891 Lippmann color photography (Alternative Photography) Bjelkhagen, "Lippmann photography". Incident light and its reflection from a backing mirror form depth-dependent intensity fringes. For vacuum wavelength λ, refractive index n, and internal angle θ, adjacent intensity maxima are separated by

$$\Delta z = \frac{\lambda}{2n\cos\theta}. \tag{1}$$

After development, the layered structure selectively reflects wavelengths under suitable viewing illumination. Finite thickness limits spectral resolution; the mirror's phase response, emulsion response, processing shrinkage, and viewing angle affect reproduction. It preserves more spectral information than an RGB photograph, but not a

perfect, unlimited record of the original spectrum. Modern recovery methods model these distortions Baechler et al. 2021.

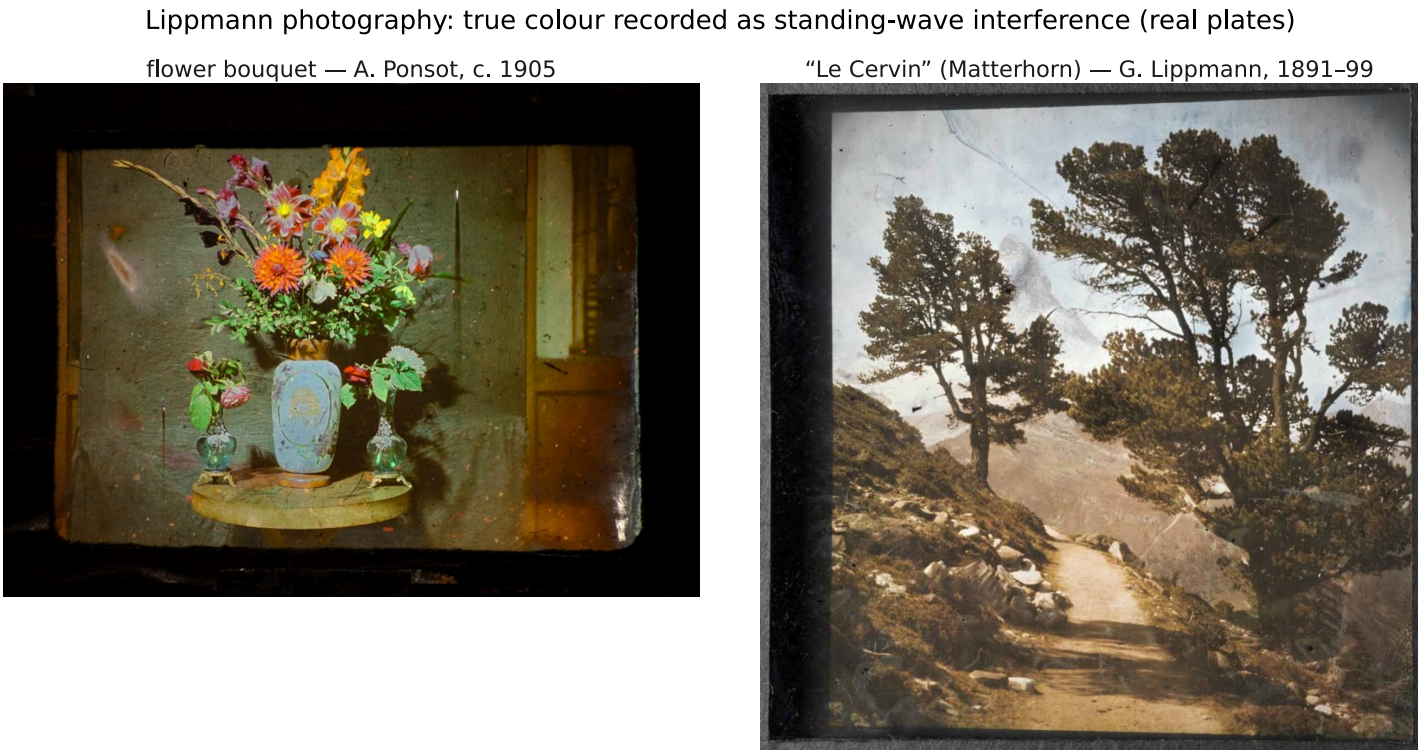


Figure 4.6.20 (reproduced). *Lippmann interference color photography. Recording (top-left): light enters a transparent emulsion backed by a liquid-mercury mirror; the incident and reflected waves form a standing wave whose antinodes expose layered fringe planes that encode the wavelength. Inside an emulsion of refractive index n, for light arriving at angle θ within it, the planes are $\lambda/(2n\cos\theta)$ apart, where λ is the wavelength in air: about 0.18 μm for green light at normal incidence with $n \approx 1.5$. Playback (top-right): under white light the recorded layers act as a Bragg reflector and send back mostly the original wavelength. A real emulsion holds only a finite number of layers, a few micrometers deep, so the reflected band has a finite width, and the color shifts when the plate is viewed at another angle. It records a spectrum rather than three numbers, within those limits. Below: two real Lippmann interference-color plates, a flower bouquet (A. Ponsot, c. 1905) and "Le Cervin" / the Matterhorn (Gabriel Lippmann, 1891–99), true color recorded with no dyes or filters. The 1908-Nobel limiting case of color capture, and a precursor of holography. (Plates via Wikimedia Commons: bouquet © Univ. de Lille / A. Ponsot, CC0; "Le Cervin", G. Lippmann, public domain.)*

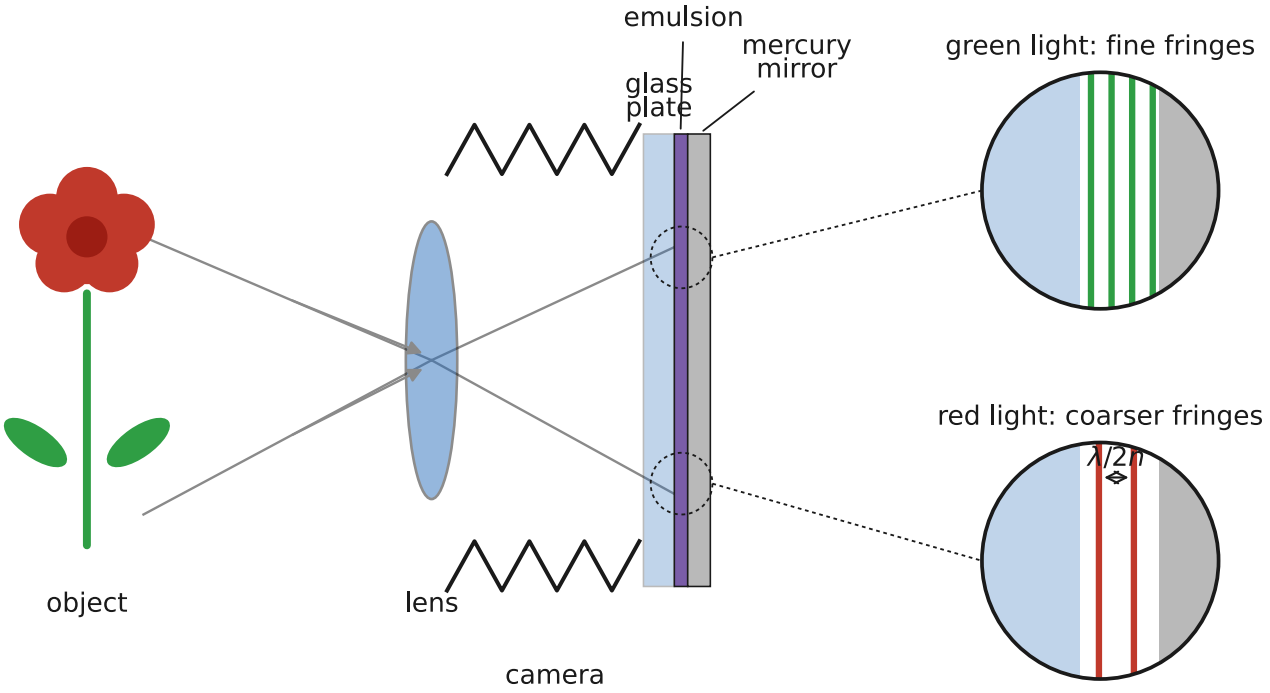


Figure 4.6.21 (reproduced). *Recording a Lippmann photograph. The lens images the scene onto a fine-grain transparent emulsion backed by a liquid-mercury mirror. At each point the incoming light and its reflection off the mercury superpose into a standing wave whose antinodes, spaced $\lambda/2n$ apart, are recorded as stacked fringe planes inside the emulsion (insets: green light lays down fine fringes, red light coarser ones). Viewed later in white light, each recorded stack Bragg-reflects exactly the wavelength that made it, so the plate replays the true spectrum rather than three color numbers. (Diagram redrawn after an illustration by Hans Bjelkhagen.)*

A whole spectrum recorded by a Lippmann plate: each colour its own fringe spacing

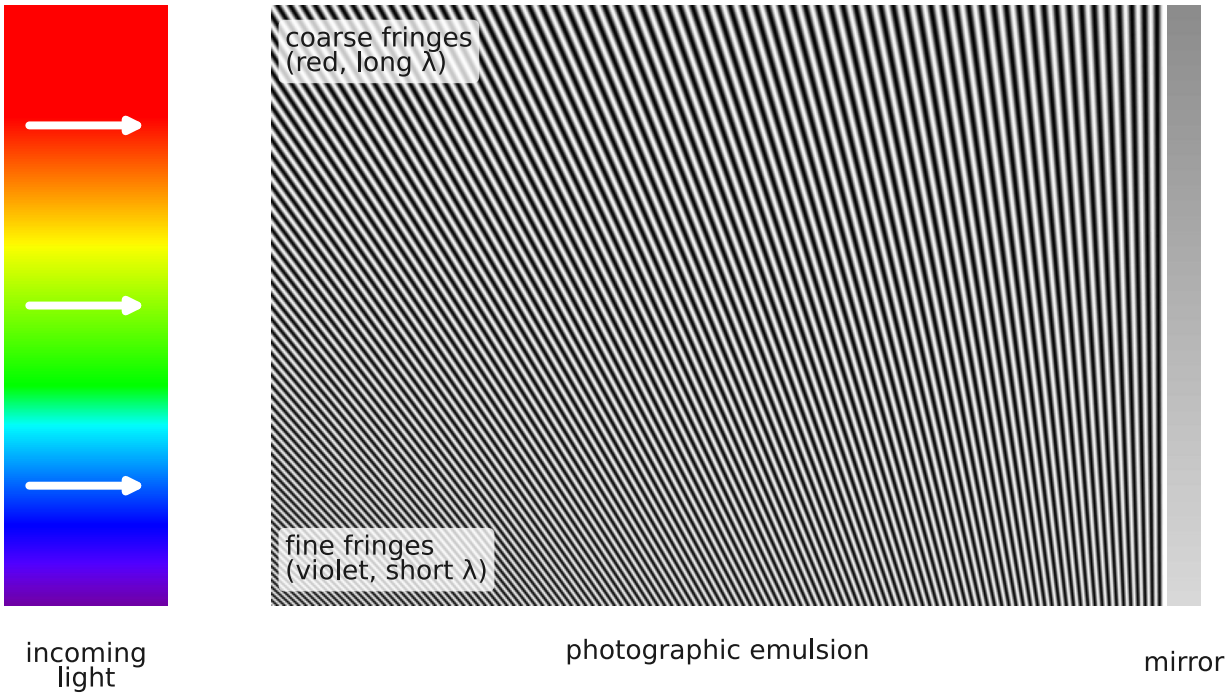


Figure 4.6.22 (reproduced). *A whole spectrum recorded at once. Incoming light spanning the visible range (left) enters the emulsion and reflects off the mirror at the right; every wavelength forms a standing wave that shares a node at the mirror, so each color records its own stack of fringe planes spaced $\lambda/2n$ apart: coarse for red, fine for violet. The developed emulsion is colorless, just black-and-white silver density: the color is not stored as pigment but purely as the fringe spacing (spatial frequency), and it reappears only when white light later Bragg-reflects each stack back at its own wavelength. (Our own illustration, inspired by a standard Lippmann diagram.)*

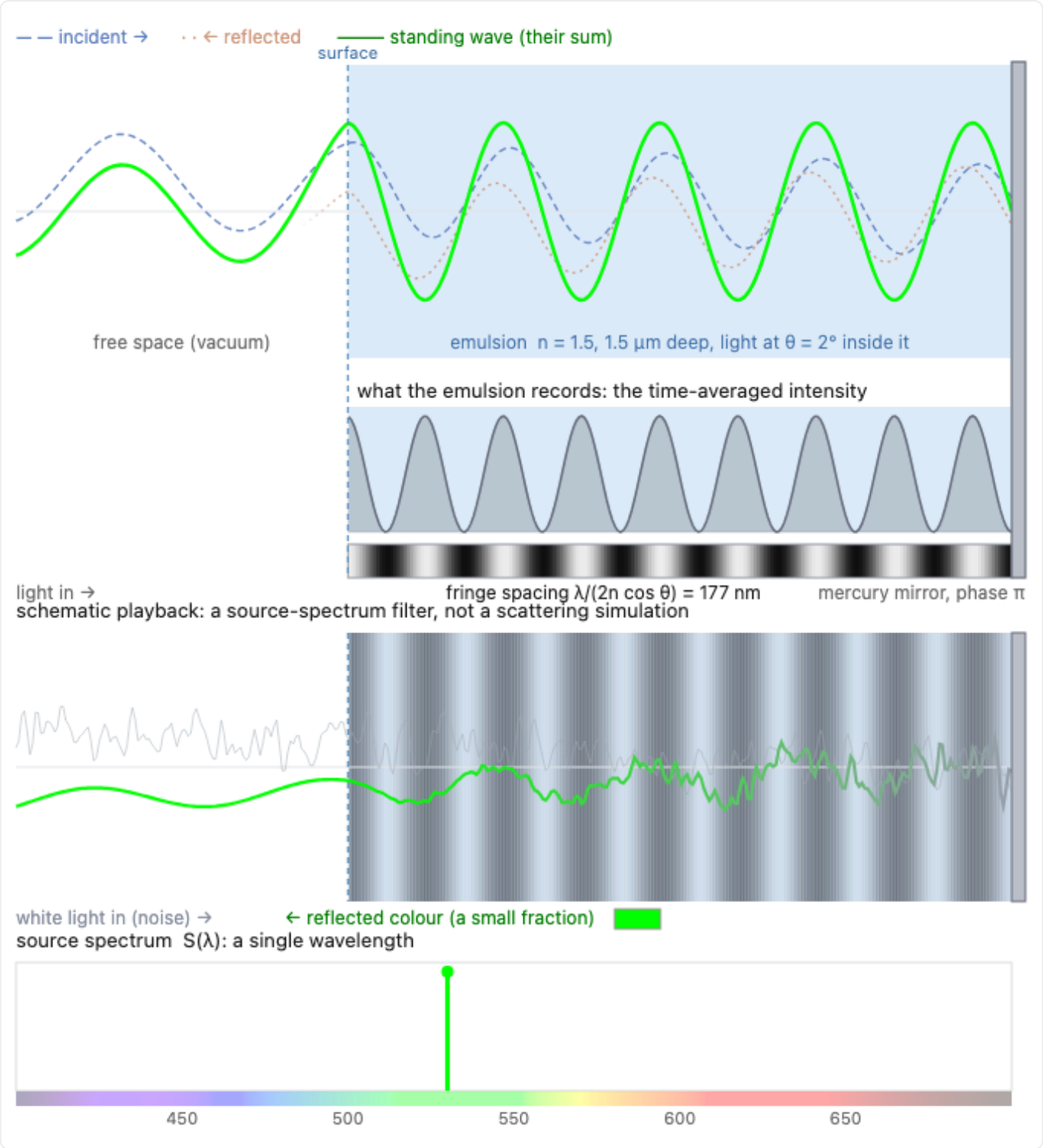
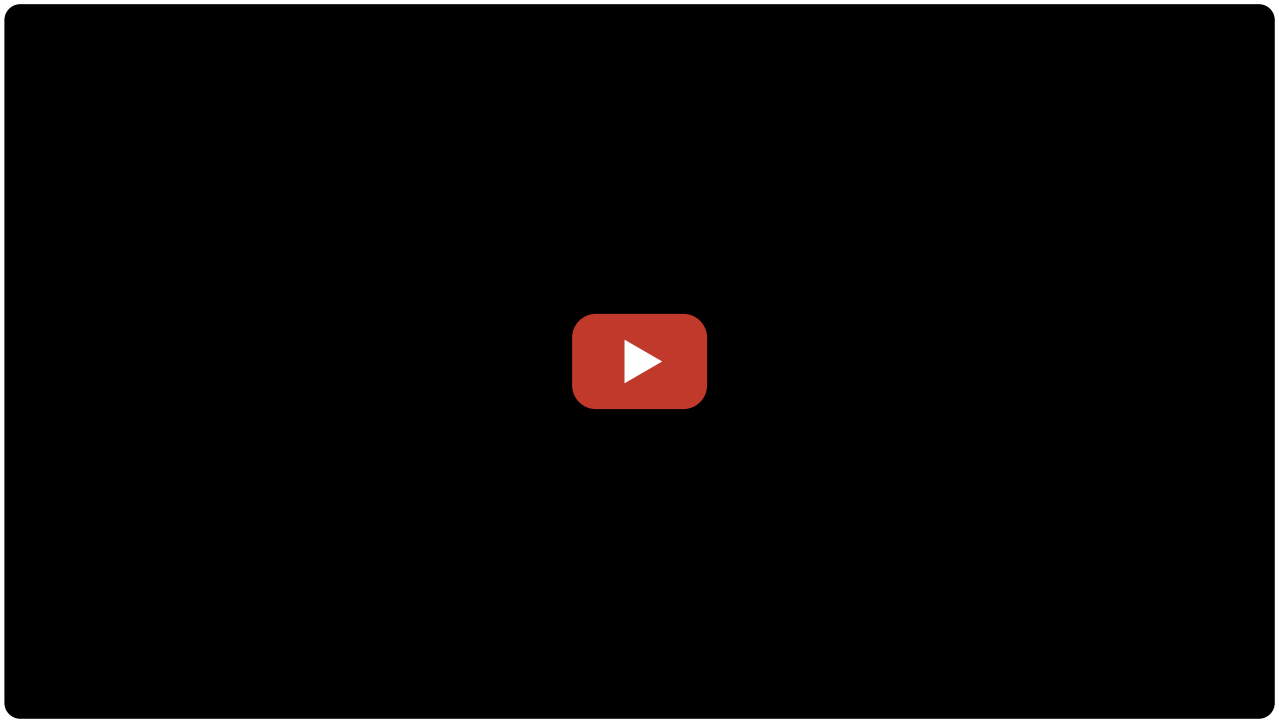


Figure 4.6.23 (reproduced). *The standing wave behind Lippmann photography (interactive, 1-D).* Light enters from the left through free space, crosses into the emulsion (where its wavelength shortens by the refractive index), and reflects off the mercury mirror; the incident and reflected waves add to a standing wave with a node fixed at the mirror; its time-averaged intensity $\sin^2(k(L-x))$ is the density the silver records, bright planes half a wavelength apart ($\lambda/2n$). An angle-of-incidence control tilts the incoming beam so the reflection separates from it and is easy to tell apart (only the depth-axis component of the wavevector drives the fringes, so they coarsen slightly); the pattern is recorded only inside the emulsion, and the reflection that leaks back into free space fades quickly. Pick a single wavelength and watch the two traveling waves slide through each other while the standing wave pulses in place (bluer light gives finer

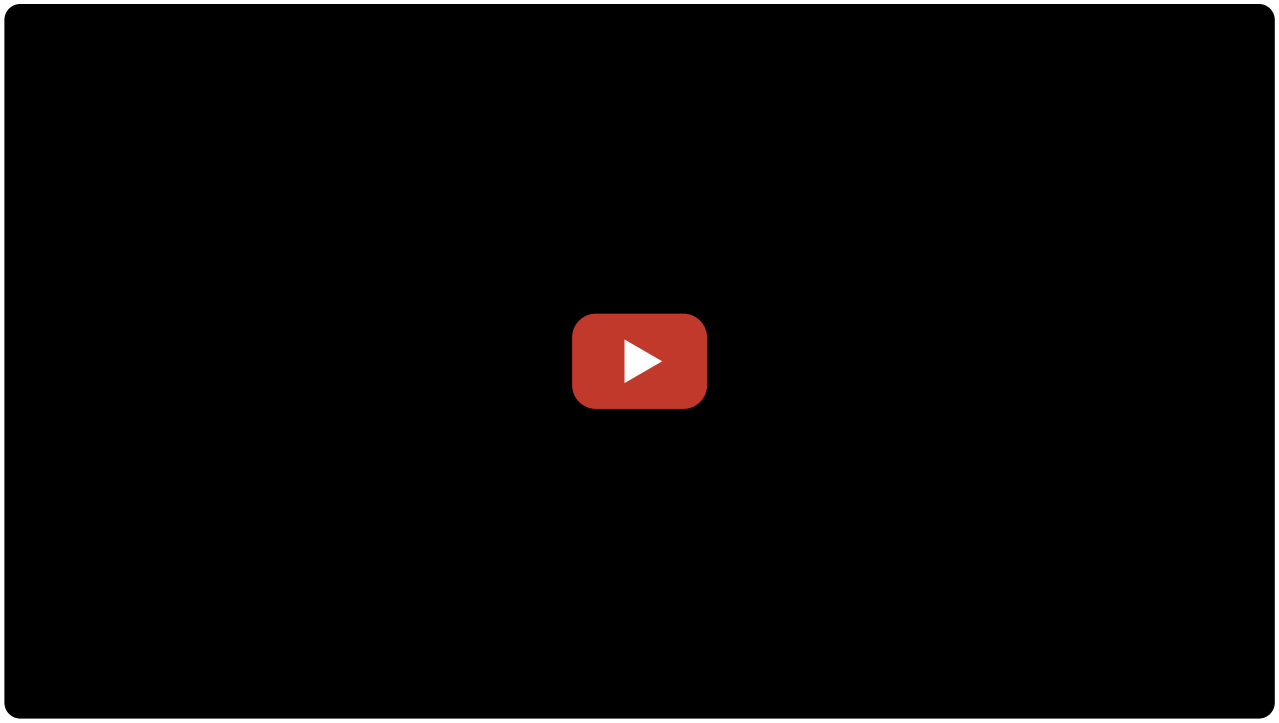
fringes); or set a full spectrum (drawn by hand on the editor, or chosen from standard sources) and see every wavelength pin its own node at the mirror, so the fringes reinforce near the mirror and dephase with depth. A narrow line keeps sharp fringes all the way through; a broad spectrum washes them out within microns. The spectrum is stored in how fast the fringes fade. A separate playback panel runs the process in reverse: shine broadband white light on the developed plate, and the recorded fringe stack acts as a wavelength filter: deep at the mirror only a few planes have formed, so the reflection there is still broadband, but working back out through more and more planes it is progressively filtered into the pure recorded color, so only a small, spectrally clean fraction of the light leaves the surface. Interactive: choose a wavelength or draw/select a source spectrum, and watch the incident, reflected, and standing waves and the recorded fringe pattern they leave.

Sidebar 2: Gabriel Lippmann

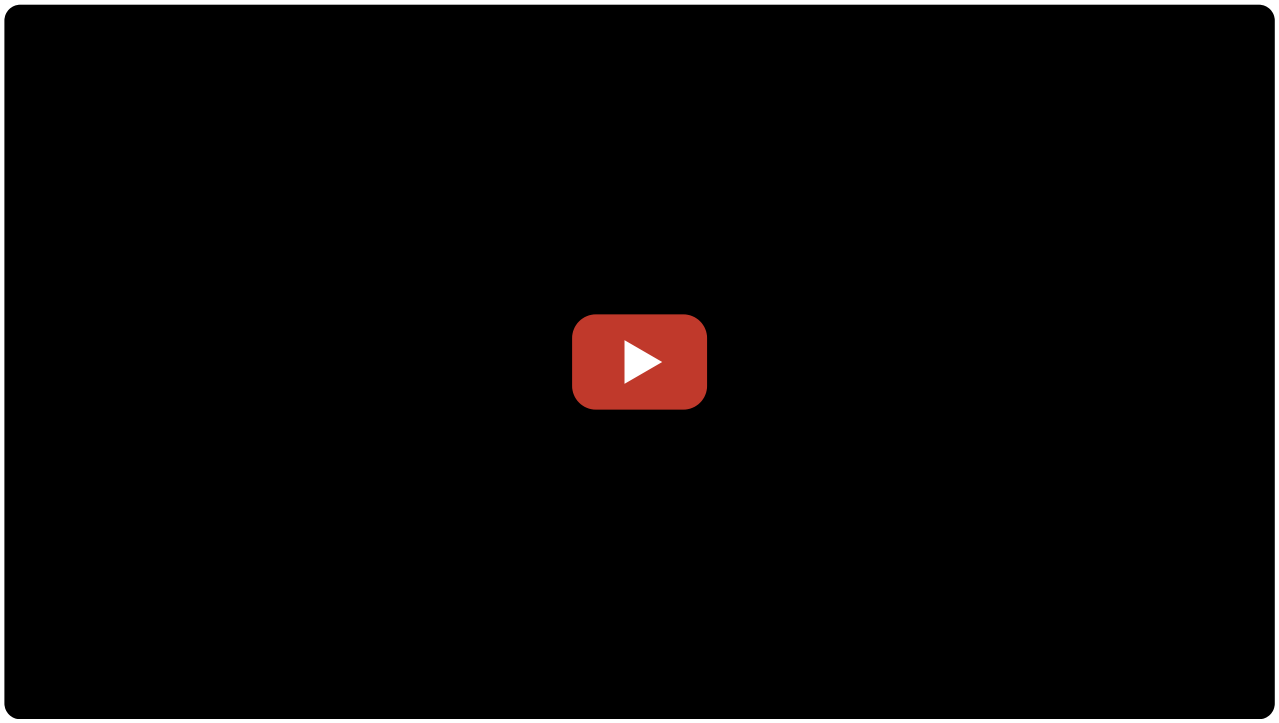
Gabriel Lippmann (1845–1921) received the 1908 Nobel Prize in Physics for interference color photography, the only Nobel awarded for a color-photography process. He was a physicist's physicist: his capillary electrometer was used to record the first human electrocardiogram, he predicted the converse piezoelectric effect that the Curies then confirmed, and he proposed integral photography, an ancestor of light-field imaging. Long exposures and restrictive viewing conditions limited the color process's practical adoption, but its spectral recording remains an instructive alternative to trichromatic capture.



Video 1. Steve Mold demonstrates interference color photography.



Video 2. Nick Brandreth, “Lippmann Plates, Still Life Series Vol. 1.”



Video 3. ?t=315 Martin Vetterli, “Lippmann Photography: the Art and Science of Multispectral Imagery.”

For a monochromatic exposure, the incident and reflected waves establish a repeated intensity pattern through the depth of the emulsion. Development records that pattern as a layered optical structure. When white light later illuminates it, reflections from successive layers reinforce one another near the recorded wavelength under the appropriate viewing geometry. A more complicated incident spectrum leaves a more complicated depth pattern. The layer spacing carries spectral information even though the emulsion is not divided into conventional red, green,

and blue channels. Lippmann's method is therefore a useful counterexample to the assumption that a color photograph must store three numbers: it records another kind of optical measurement, whose reconstruction still has finite bandwidth and physical imperfections.

Spectral sensitivity: matching the eye, catching photons, and white balance

OPTIONAL
in the book ↗

With photon spectral exposure $q(\lambda)$ and effective channel detection probability $\eta_i(\lambda)$, the expected signal electrons are

$$\mu_i = \int q(\lambda)\eta_i(\lambda)\, d\lambda. \tag{2}$$

The response includes filters, optics, and quantum efficiency under the stated convention. Sampled implementation needs wavelength-bin weights. Shot noise is determined by detected electrons, while read noise is added by the readout; the full model is in Chapter 2.8, Noise, signal-to-noise ratio and dynamic range. A broad response alone is not a color measurement: distinct response shapes supply the information that separates spectra.

The exact shape of a color sensor's three spectral-sensitivity curves (the $r(\lambda)$ of the red, green, and blue channels, built earlier by the color-filter array on the photosites of Chapter 2.7, Sensors: photosites, CCD vs CMOS) is one of the camera's most *constrained* choices, because it is pulled in three directions at once that cannot all be satisfied.

- **Fidelity to human vision.** For a camera to record the color a *human* would see, its three curves should be a fixed linear combination of the eye's cone fundamentals (equivalently, of the CIE color-matching functions), the Luther–Ives condition. A sensor that met it could be mapped to human color by a single 3×3 matrix with *no* error. Practical camera sensitivities generally only approximate it, because silicon-plus-dye curves are simply the wrong shape, and the penalty is a metameric error no color matrix can remove: two spectra that look identical to the eye read as different RGB, and two that look different can read the same. The color-correction matrix in Chapter 8.10, Auto-exposure and auto white balance is a least-squares *compromise*: right on average, wrong on the hard cases (saturated reds, narrow-band LEDs, foliage).
- **Quantum efficiency and noise.** Broad transmission bands can increase total photon collection; this is distinct from quantum efficiency at a specified wavelength: the wider each filter, the more photons reach the well and the higher the SNR. Strong overlap can make color separation poorly conditioned, because separating similar channel responses can require large signed coefficients that amplify measurement noise (compare the distinct physiological opponent processing in Chapter 3.2, Perceptual color and trichromatic vision). Narrow, well-separated filters give clean color but throw photons away and read noisier. The green-doubled Bayer is one point on this curve: spend photons and resolution where the eye cares most (luminance), economize on chroma.
- **White balance across illuminants.** Because each channel integrates *illuminant* × *reflectance* × *sensitivity*, changing the light changes the raw RGB of a fixed object. Undoing that, white balance often approximates the change with per-channel gains. Matching cone sensitivities does not make this diagonal relighting model exact;

a response set that matches human colorimetry well need not be the best basis for diagonal adaptation. Good spectral design makes white balance more reliable across illuminants (→ Chapter 8.10, Auto-exposure and auto white balance).

These three pull against each other: the curves that best match the eye are broad and overlapping (bad for separation), the curves that separate color cleanly are narrow (bad for QE and for matching the eye), and keeping white balance stable across odd illuminants wants yet another shape. Every color sensor is a negotiated settlement among them (Figure 4.6.24).

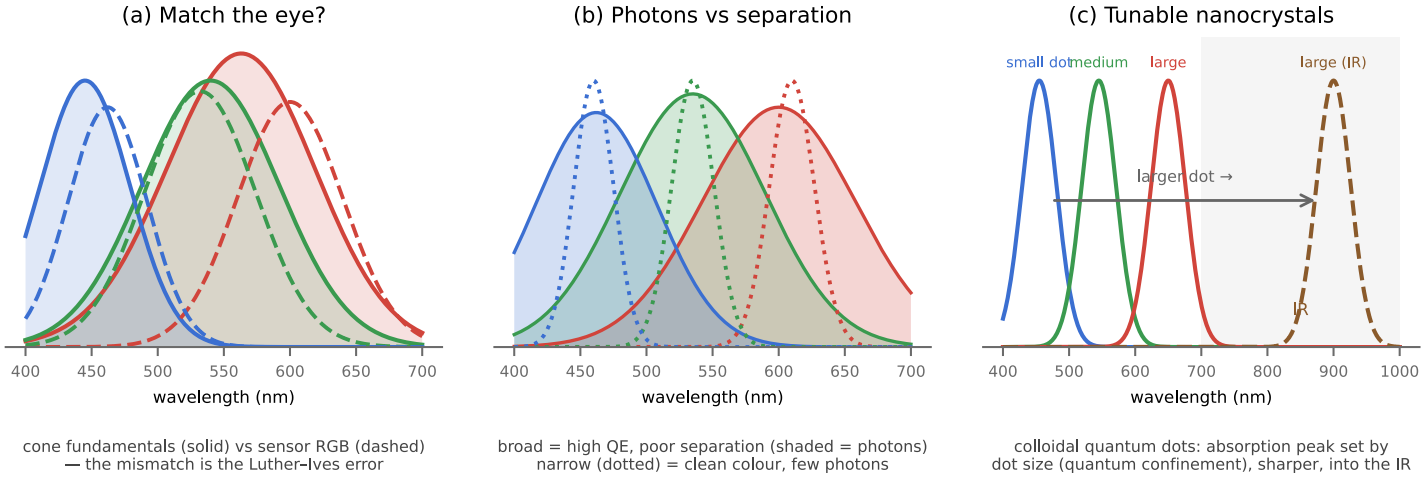


Figure 4.6.24 (reproduced). *The spectral-sensitivity trilemma. (a) Match the eye: the human cone fundamentals (smooth, heavily overlapping) beside a real sensor's RGB, the mismatch (the sensor is not a linear transform of the cones, the Luther–Ives condition) is the metameric error no color matrix can undo. (b) Photons vs separation: narrow filters separate color cleanly but pass few photons (low quantum efficiency, noisy); broad filters catch far more light (shaded area) but overlap so heavily that recovering color means differencing correlated channels and amplifying noise. (c) Tunable nanocrystals: a colloidal quantum-dot layer's absorption peak shifts with dot size (quantum confinement), so spectral sensitivity becomes a design parameter: sharper, higher-absorption, and extendable into the infrared and ultraviolet. Schematic.*

File formats and compression

RAW files: before the cooking

OPTIONAL
in the book ↗

Raw storage retains data from an earlier stage of the camera pipeline, leaving more reconstruction and rendering choices to the developer. The file may contain a CFA mosaic, demosaicked channels, or partly processed measurements. Raw capture offers latitude, but does not guarantee unmodified or losslessly stored sensor data.

To obtain scene-linear working values, decode the stored representation, apply any linearization table, subtract calibrated black levels, and use the metadata-defined scale. A linear DNG can already contain reconstructed channels. Check processing tags and profiles rather than assuming the extension establishes the numerical domain.

Here is the subtlety, stated plainly: "RAW" is not always purely raw. The name suggests an untouched sensor dump, but in practice most cameras pre-cook the data to some degree: subtracting the sensor's black level, correcting defective and hot pixels, sometimes applying lens-shading or vignetting correction or even mild denoising. Some "RAW" formats are themselves lossy-compressed or non-linearly encoded to save space. So RAW is best thought of as a spectrum of how cooked the data is, from a near-literal readout to something already partly processed, not a single guarantee of pristineness.

There is also a practical headache: RAW is largely proprietary. Nearly every manufacturer ships its own format and extension: Canon's (CR3 files), Nikon's (NEF files), Sony's (ARW files), and so on, each subtly different. That creates software and archival problems, since a format only one company supports may not be readable in thirty years. Adobe's Digital Negative (DNG) format is an open, documented, standardized RAW container meant to fix exactly this (its specification Adobe DNG specification is the reference here); many photographers convert their proprietary RAWs to DNG for safekeeping, and some cameras shoot DNG directly.



â€œIf you can read this, congratulationsâ€”the archive youâ€™re using still knows about the mouseover textâ€ !

Figure 8.11.14 (reproduced). xkcd #1683 "Digital Data," Randall Munroe, xkcd.com/1683 (CC BY-NC 2.5).

Sidebar 3: reading RAW in code

You will not parse a proprietary RAW format yourself; a library decodes the mosaic and metadata for you. The classic is dcraw, a single heroic C file that reverse-engineered dozens of formats; its maintained successor is LibRaw (C++), with a friendly Python wrapper, rawpy:

```
import rawpy
with rawpy.imread("photo.dng") as raw:
    bayer = raw.raw_image.copy()  # retain samples after closing raw
    pattern = raw.raw_pattern
    pattern = None if pattern is None else pattern.copy()
    rgb   = raw.postprocess()      # or let LibRaw develop it to RGB
```

Beyond these, open-source developers darktable and RawTherapee, Adobe's DNG Converter and DNG software development kit (SDK), and `exiftool` for metadata round out the toolkit; commercial Lightroom and Capture One are the mainstream developing apps.

Demosaicking and joint methods

in the book ↗

A Bayer sensor measures one color at each photosite. A mosaic of red, green and blue filters sits over the pixels, with green on half of them and red and blue on a quarter each, so each pixel records a single filtered value (Chapter 4.6,

Sensing color: multiplexing strategies, Figure 15.0.1). Demosaicking reconstructs the two missing values at every pixel.

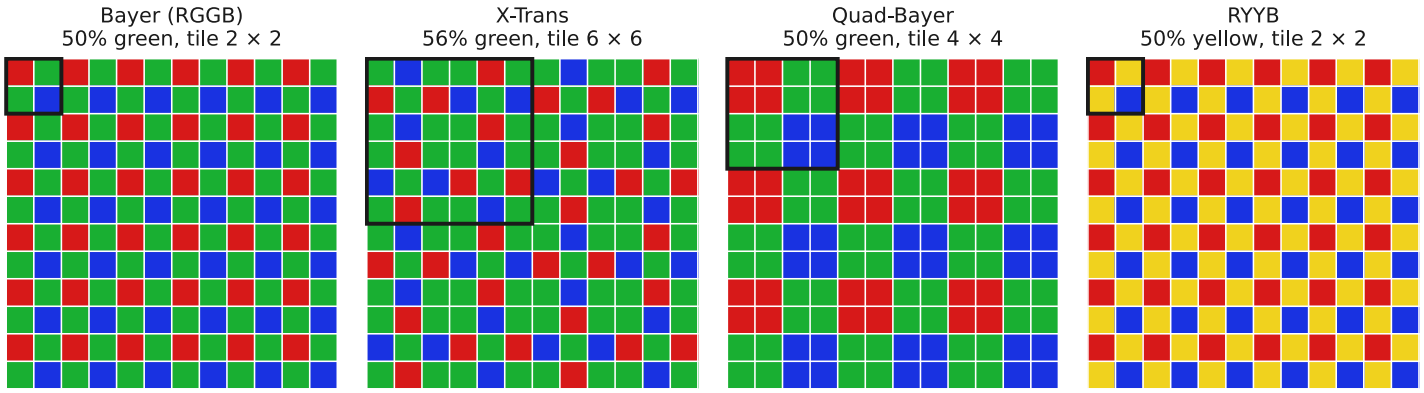


Figure 15.0.1 (reproduced). *Four color filter arrays over the same 12 × 12 photosites, each with its repeating tile outlined. Bayer (RGGB) repeats a 2 × 2 tile of two green, one red and one blue filter, so each pixel measures one color and the other two must be reconstructed; the extra green matches the eye's green-weighted luminance sensitivity. Fujifilm's X-Trans repeats a 6 × 6 tile with 20 greens in 36 and all three colors in every row and column, a less regular lattice that weakens moiré. Quad-Bayer lays the Bayer pattern out in 2 × 2 blocks of one color, which a phone sensor can merge into one larger pixel in low light (Figure 15.0.2). RYYB (used by Huawei) swaps the two greens for yellow filters: a yellow filter passes red and green together, so it lets through more light, at the cost of color accuracy and more color noise, since recovering RGB from a channel that mixes two primaries needs a stronger color-correction matrix.*

Bayer's is the most common mosaic, not the only one. Fujifilm's X-Trans repeats a 6 × 6 tile with more green than Bayer and at least one pixel of every color in every row and column (Figure 15.0.1). Its less regular layout breaks up the periodic structure that turns fine patterns into color moiré on a Bayer sensor, which lets the cameras that use it leave out an optical low-pass filter, and it needs a demosaicker of its own. Quad-Bayer, common in phone sensors with many small pixels, puts four adjacent pixels under each color filter, a Bayer pattern of 2 × 2 blocks. In low light the sensor combines each block into one larger, less noisy pixel; in good light it reads every pixel and remosaics them into a full-resolution Bayer image (Figure 15.0.2); and the four pixels of a block can be given different exposures to extend dynamic range. Nona-Bayer layouts, Samsung's Nonacell for example, push the same idea to 3 × 3 blocks: a 108-megapixel sensor that bins to 12 megapixels in low light. Other arrays, such as RYYB and RGBW, trade color accuracy for light. Chapter 15.5, Alternative color filter arrays treats these layouts, and Chapter 15.12, Hardware alternatives to the mosaic the sensors that avoid a mosaic altogether.

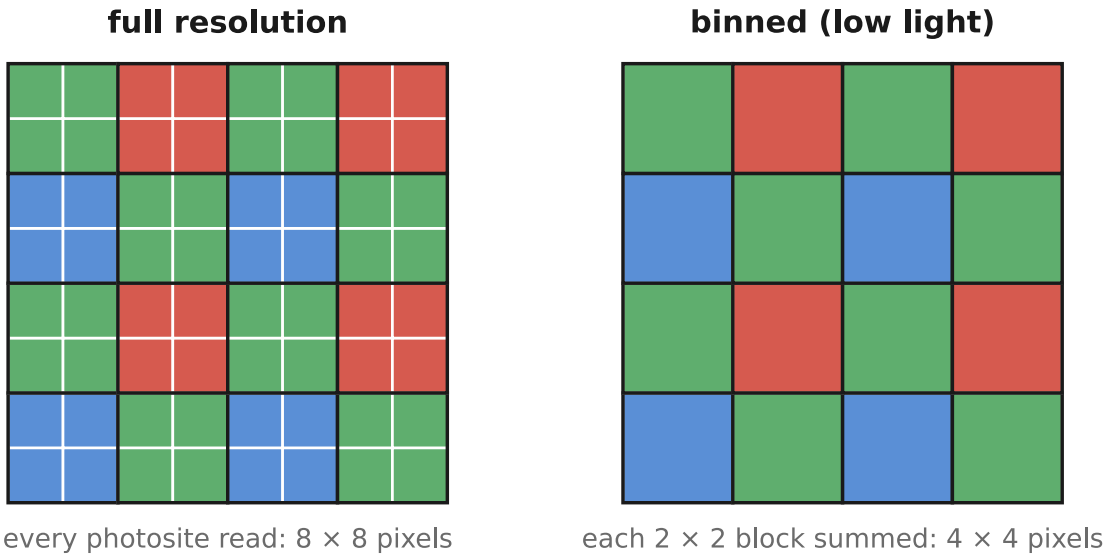


Figure 15.0.2 (reproduced). *One Quad-Bayer sensor read two ways. Left, full resolution: every photosite is its own pixel, so each 2 × 2 block of one color is four pixels, which the camera remosaics into an ordinary Bayer image before demosaicking. Right, the low-light mode: each block is summed into one larger pixel, half the resolution in each direction and a quarter of the pixel count, with four times the signal per pixel. The binned pixels form an ordinary Bayer mosaic directly.*

Demosaicking is one of the fundamentally ill-posed problems of computational photography. We need three numbers per pixel and we have measured one. Many full-color images agree exactly with the same mosaic, and they differ where it matters most, at fine texture and edges, where a wrong choice shows up on fences and fabric as false color, zippering or a maze pattern. Our only hope is to use what we know about natural images, the prior of Chapter 7.1, Imaging as Bayesian inference with probabilistic priors: for example, their color channels share edges, and color varies more slowly than brightness.

This also makes demosaicking hard to measure. On the vast majority of pixels, flat regions, gentle gradients and ordinary edges, almost any reasonable method is nearly exact. The mistakes are rare, but when they happen they are catastrophic and very visible: zippering along an edge, false color on a fence, moiré on fabric, a maze in fine texture. Averaged over a whole image, a few thousand badly wrong pixels among millions barely move a statistic such as mean squared error or PSNR, so two methods can score almost the same while one of them has artifacts that ruin the picture. Demosaickers are therefore judged on the hard cases themselves: error maps, the worst few percent of pixels, and test sets built from the patterns that fail, which is also how the learned methods of Chapter 15.6, Learned demosaicking are trained.

Demosaicking is one of several areas of computational photography, with denoising and super-resolution, where traditional techniques use these regularities implicitly, through the careful design of smart algorithms rather than an explicit model. Interpolating along an edge rather than across it assumes that edges are continuous; interpolating color differences rather than colors assumes that the channels share structure. The field is now changing, from designing algorithms that embody the regularities to learning them from data with deep networks, which can capture regularities no hand-designed rule states. Where each approach runs is set by computation. A camera's image signal processor demosaics every preview and video frame at full resolution in a few milliseconds, so it still runs hand-designed methods of the kind described in Chapter 15.1, Basic traditional demosaicking and Chapter 15.2, Advanced traditional demosaicking. Learned demosaicking has shipped mainly in desktop raw

converters, which can spend seconds on one photograph. Phones sit in between: a still photograph gets up to a second or so of processing after the shutter, enough to run networks at full resolution, though manufacturers do not say whether those networks replace the demosaicker, and some phone pipelines avoid single-frame demosaicking by merging a burst of raw frames directly onto a full-color grid.

The part follows that arc. Classical methods rest on two ideas: interpolate along an edge rather than across it, and interpolate a quantity that varies less than the channels themselves, such as the difference between two channels or the residual left after predicting one channel from another Kiku et al. 2013 (residual interpolation). Both assume the channels share structure, which colored edges, clipping and chromatic aberration can violate (Chapter 15.1, Basic traditional demosaicking, Chapter 15.2, Advanced traditional demosaicking). Chapter 15.3, Mosaics and aliasing looks at the mosaics in the Fourier domain, where false color and moiré are aliasing, and at the optical filters that try to prevent it. Chapter 15.4, Optimizing and searching classical demosaickers keeps the hand-made structure and lets data set its constants or search for its structure, Chapter 15.6, Learned demosaicking replaces the rule with a trained one, and Chapter 15.11, Synthetic data, unprocessing, and raw reconstruction asks where the training pairs come from. Demosaicking also interacts with noise: interpolation turns independent sensor errors into correlated color errors, and a denoiser that ignores the mosaic mixes unlike measurements, so Chapter 15.8, Joint demosaicking and denoising and Chapter 15.9, Joint methods and the end-to-end pipeline estimate them together, which helps without automatically beating a well-designed sequence. Chapter 15.10, Quad-Bayer and Nona-Bayer processing covers the grouped layouts of phone sensors, where the industry's remosaicking detour meets learned direct reconstruction. The remaining chapters cover alternative arrays, hardware that avoids the mosaic, and neighboring raw-stage corrections. Throughout, keep the raw measurements and their metadata, the mosaic layout, units, noise and clipping, available to whatever reconstructs the image.

Basic traditional demosaicking

in the book ↗

This chapter covers the hand-designed reconstructions, the ones that encode their assumptions about images in the algorithm itself. They start from the layout of the Bayer tile Bayer 1976: the two green-filtered photosites of each 2×2 block sit on one diagonal, red and blue on the other (Figure 15.0.1, at the start of Part 15, Demosaicking and joint methods). Each filter has a broad spectral response, so a photosite records a camera-color component, not a wavelength.

Green gets the extra samples because camera green correlates strongly with brightness, where the eye wants the most detail. It is not colorimetric luminance, though; Chapter 15.3, Mosaics and aliasing makes the split between a combined brightness signal and color explicit.

The crux of the whole problem is that the three colors are measured in different places. Red sits on one sparse grid, blue on another shifted by (1,1) pixel diagonally, green on a denser checkerboard, and no two of them coincide. When the scene has a sharp edge, the three channels cross it at slightly different pixels, and such misregistration is the source of most demosaicking artifacts: zippering, false color and color moiré all come from it, while noise amplification and aliasing already present in the samples have other causes.

We will illustrate demosaicking with a scene composed of one dark corner on a light background (Figure 15.1.1). It starts from the mosaic itself.

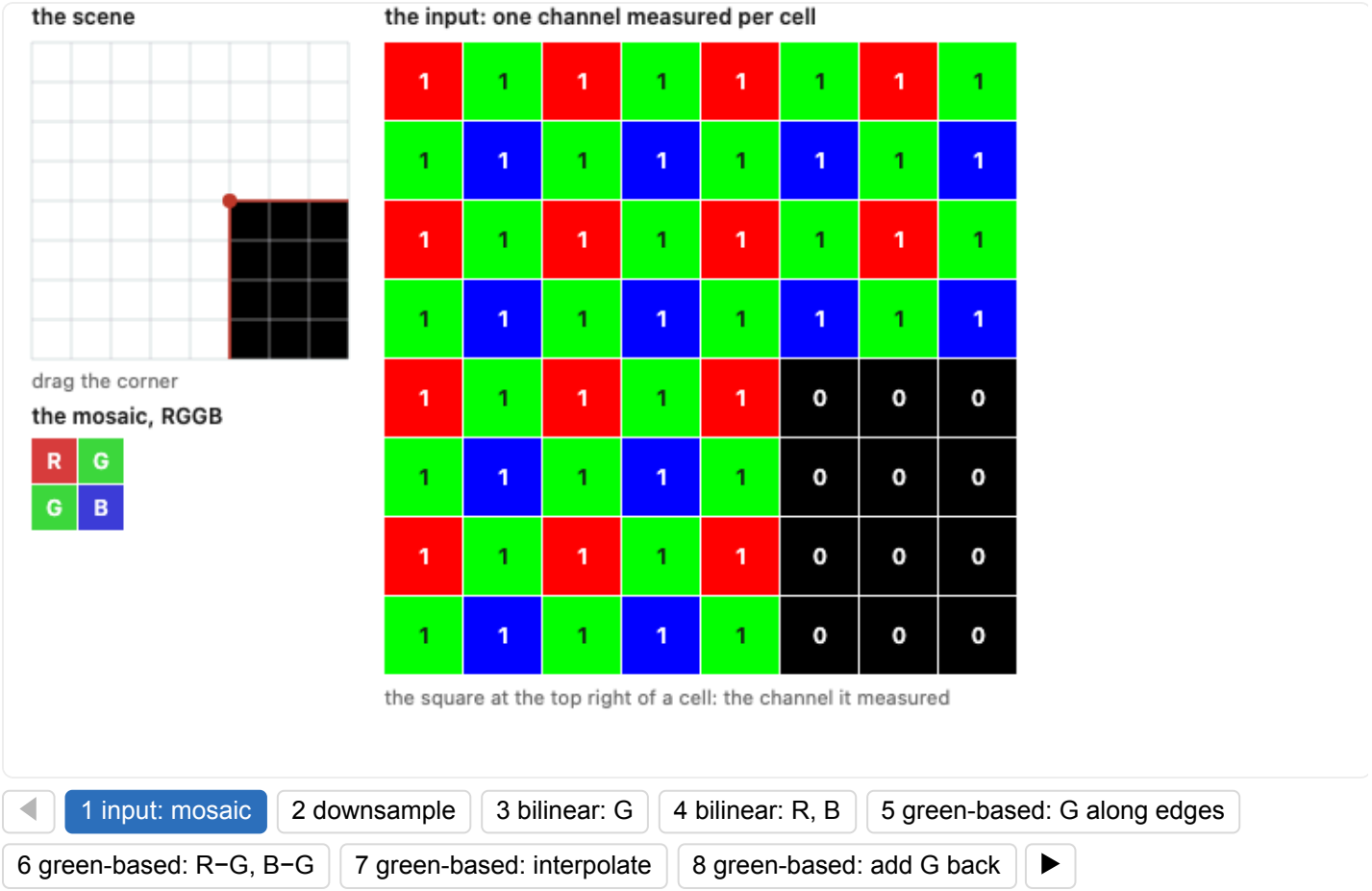


Figure 15.1.1, stage 1 (reproduced). *Interactive. One corner through three ways of demosaicking, cell by cell, shown in stages through this chapter. A dark corner on a light background is area-sampled onto 8 × 8 cells and kept through an RGGB mosaic; by default its first dark cell is a measured green, and every cell prints its value. Each stage is one technique and one step in it, and the three techniques' final results (downsampling, bilinear and green-based) sit at the right with their largest error, the current one framed, each appearing once its last step has been reached. The first stage is the input: each cell keeps only its own channel, so the three colors see the corner on three different grids.* Interactive: step through the stages, drag and turn the corner, and switch the scene between black on white, red on white and blue on yellow.

Sidebar 4: RAW is the mosaic, in linear light

A raw file may contain a CFA mosaic, linear RGB, or partially processed data. Decode its packing and any linearization table, subtract the calibrated black level, and read the CFA pattern and active-area origin. Do not infer these from the extension. Chapter 8.11, File formats and compression discusses storage and metadata.

Test a synthetic constant field first, including unequal constants such as $(0.2, 0.4, 0.7)$: a correct reconstruction returns each channel's own constant, including at the borders. A real neutral target need not give equal raw channel values before white balance. Then use an achromatic rectangle and a colored edge, and test all four Bayer phases. Those cases expose indexing errors, zippering and failures of the cross-channel assumption.

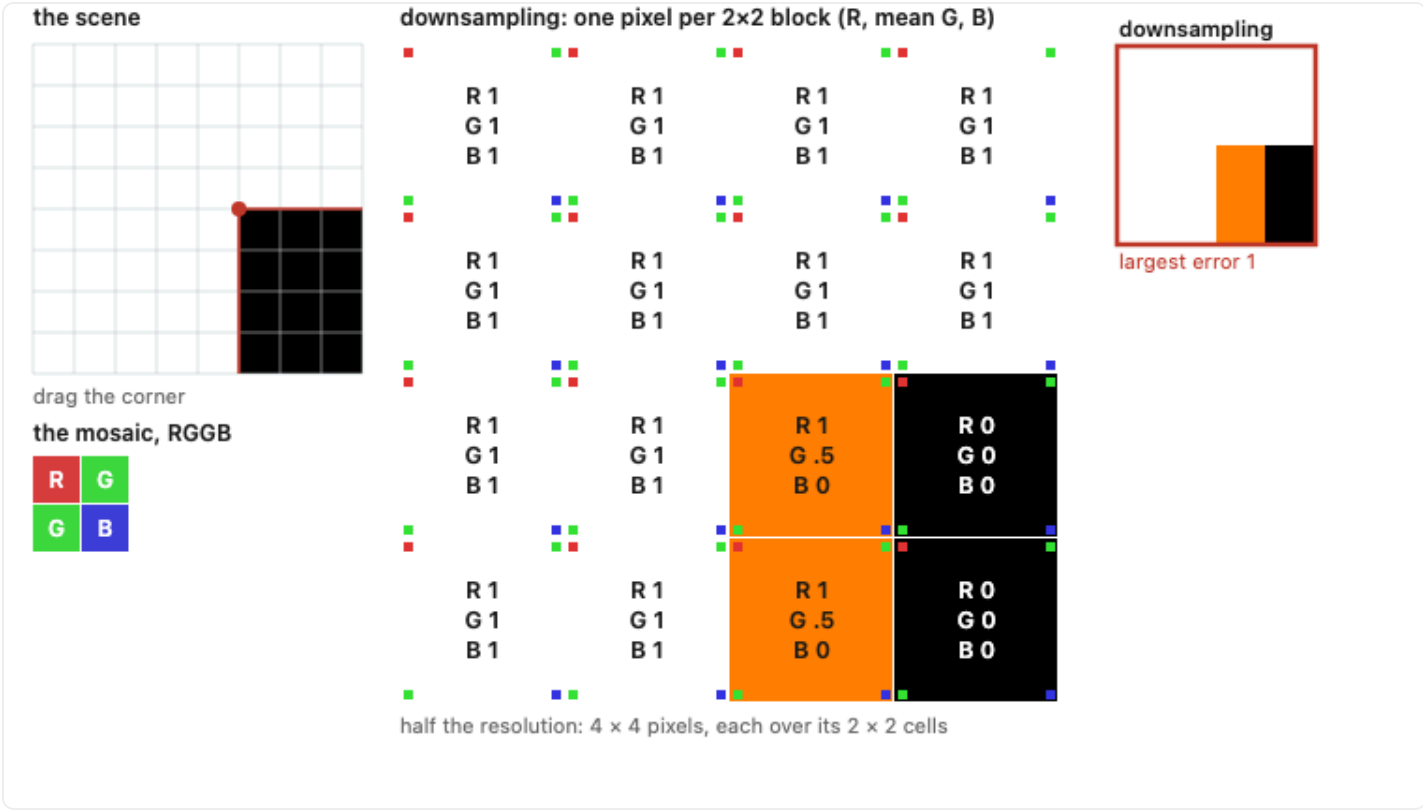
Keep in mind where demosaicking sits in the pipeline. Raw values are linear in light, and demosaicking works on them in that form; the color-difference methods below depend on it. Its output is therefore still linear camera RGB, not a picture ready for a screen. Before display, it still needs the color matrix to a standard RGB space, usually a tone curve that gives the camera its look (the beauty-curve chapter), and the gamma encoding the display system expects (Chapter 4.3, Encoding: linear, gamma or log). Shown directly, a linear demosaicked image looks dark and flat.

Downsampling and bilinear interpolation, and how they fail

in the book ↗

Two simple reconstructions come first, and their failure illustrates the demosaicking challenge.

The simplest is not to interpolate at all. Make each 2 × 2 tile one pixel, taking its red, the mean of its two greens and its blue. Every output value is a measurement, the two greens are averaged, which lowers their noise, and the operation costs almost nothing, which is why raw decoders such as dcraw offer it as a half-size mode and fast previews often use something like it. The first price is resolution: the image has half as many pixels in each direction, so detail that the sensor recorded is thrown away. And downsampling unfortunately does not remove the underlying problem: The red, green and blue of a tile were still measured at different positions, so a tile that an edge cuts in two combines red from one side with blue from the other and gets a color the scene does not have (Figure 15.1.1, stage downsample).



1 input: mosaic | 2 downsample | 3 bilinear: G | 4 bilinear: R, B | 5 green-based: G along edges | 6 green-based: R−G, B−G | 7 green-based: interpolate | 8 green-based: add G back

↓ settings from the previous stage

Figure 15.1.1, stage 2 (reproduced). *The same corner, downsampled. Each 2 × 2 tile becomes one pixel with its red, the mean of its two greens and its blue. Nothing is interpolated, but the resolution halves, and a tile cut by the edge gets a color the scene does not have: orange here, largest error 1.* Interactive: the same controls at every stage.

The same happens on real pictures and on test charts, where the comparison with the scene is side by side (Figure 15.1.2, stage downsample).

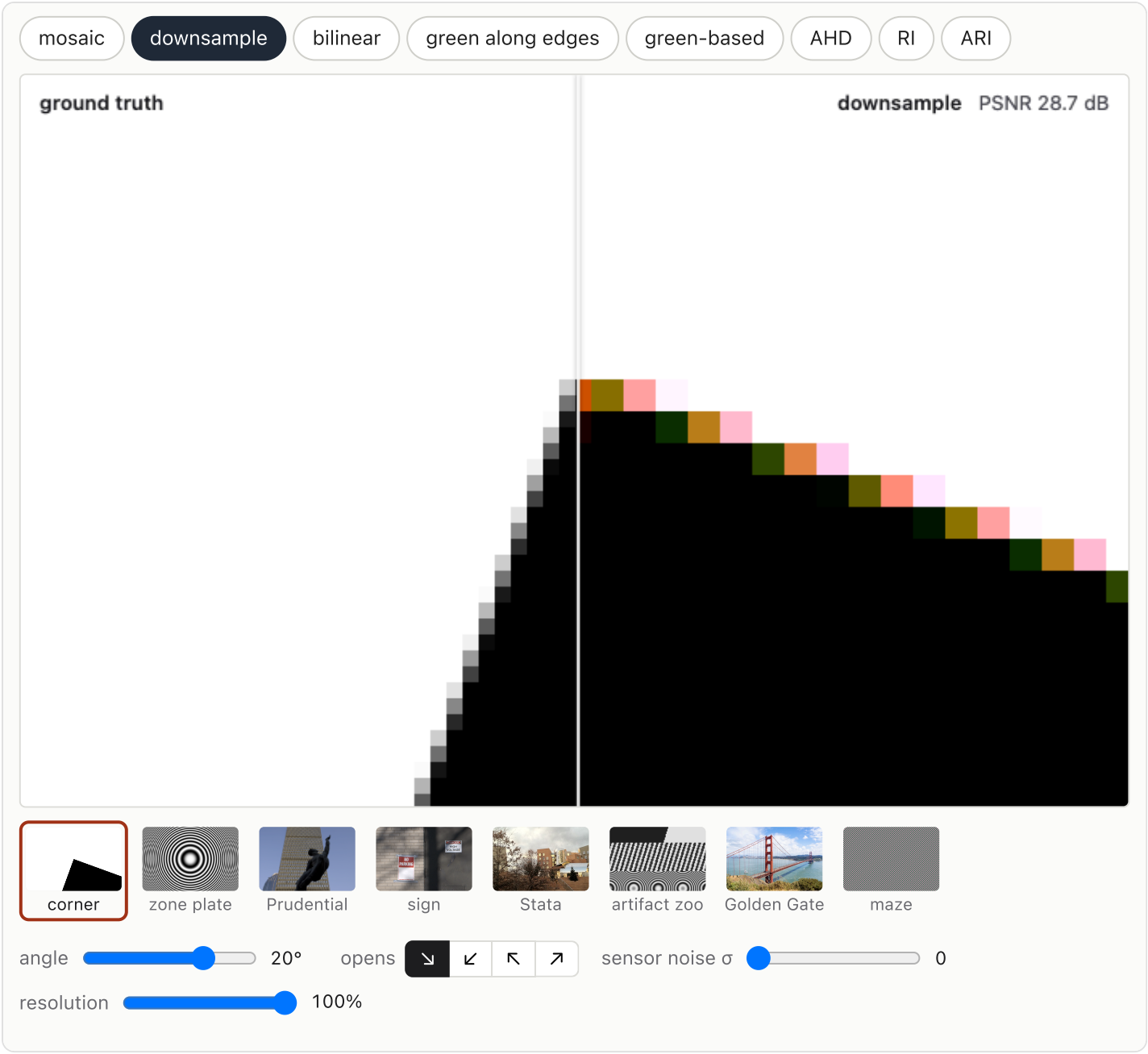


Figure 15.1.2, stage 1 (reproduced). *Downsampling against the scene. Left of the split, the scene; right of it, the scene through a Bayer mosaic and reconstructed by one method, here downsampling, with its PSNR. The corner of Figure 15.1.1 opens turned by 20 degrees and zoomed in on its tip; the other inputs are a zone plate, the Prudential tower, a NO PARKING sign from a real raw file, Stata, the artifact target of Figure 15.1.6, the Golden Gate, and fine colored stripes that make mazes. The zone plate and the star are band-limited: their finest rings and spokes fade to gray at the sensor's sampling limit, so the ground truth carries no aliasing of its own. The buttons above switch the method: the mosaic itself, downsampling, bilinear, green interpolated along edges, green-based, AHD, residual interpolation (RI) and adaptive residual interpolation (ARI).* Interactive: the split follows the pointer and a click parks it, with a magnifier on the pointer; pick the input by its thumbnail, set the corner's angle and orientation, the sensor noise, or the resolution, which gives the sensor fewer pixels over the same scene; pinch or hold ⌘ or ctrl while scrolling to zoom, drag to pan, double-click for the whole picture, and switch the method with the buttons.

Bilinear interpolation keeps the full resolution instead. It treats the three color channels as three independent, sparsely sampled images and interpolates each one. For a red pixel missing green, average the four green neighbors; for a green pixel missing red, average the two horizontal or two vertical red neighbors, and so on.

It is a useful inexpensive baseline. It reproduces some smooth signals well, but its errors on fine structure illustrate the consequences of sparse color sampling.

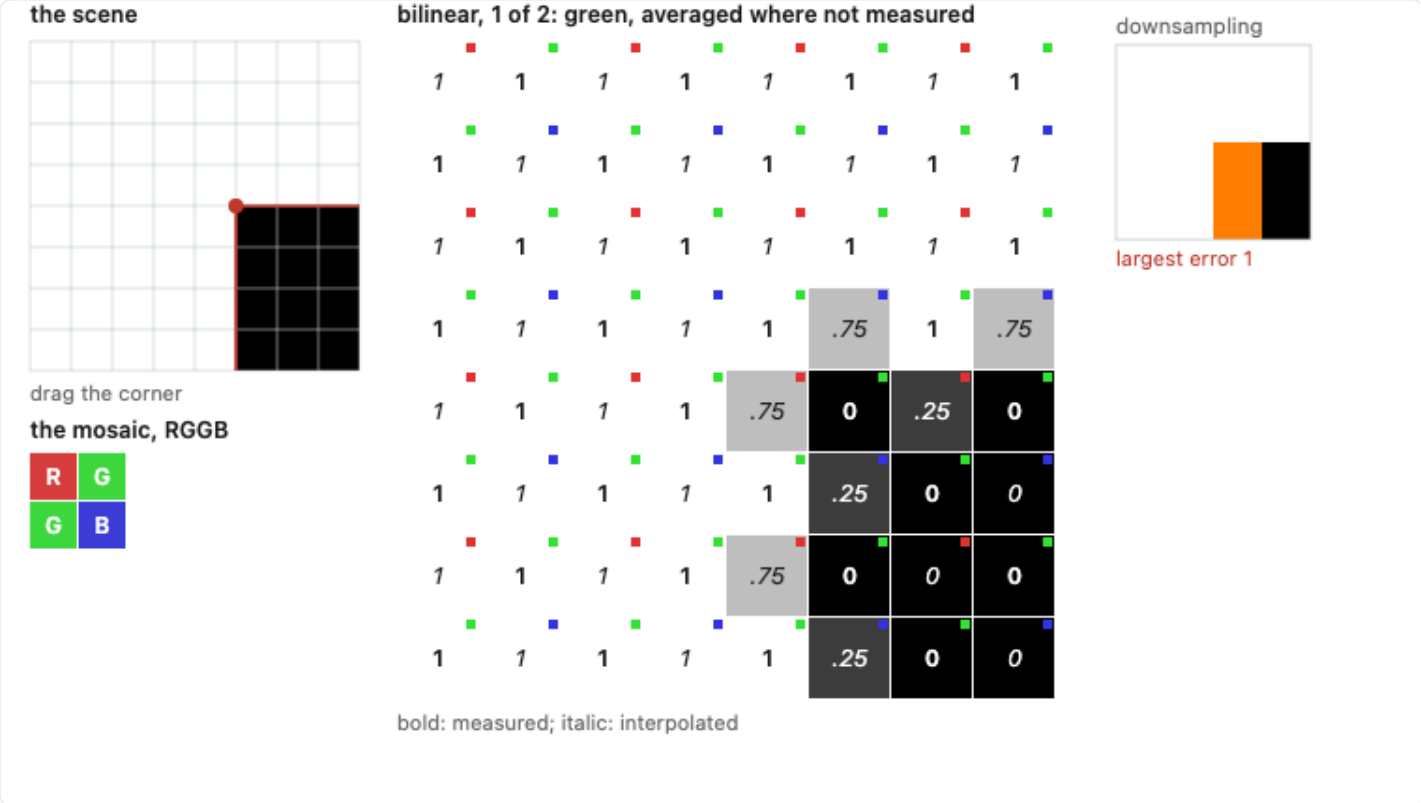


Figure 15.1.1, stage 3 (reproduced). *Bilinear green. Each missing green is the average of its four green neighbors, which gives .25 and .75 along the edge of a scene that holds only 0s and 1s.* Interactive: the same controls at every stage.

Zippering is an alternating light-dark pattern along a sharp horizontal or vertical edge. It comes from averaging across the edge: at one pixel the four neighbors are two on each side, at the next they are three and one, so the interpolated value alternates. The name comes from what it looks like. The zipper is a form of aliasing (Figure 15.1.3). Green's samples lie on a checkerboard, and a checkerboard is itself a regular square grid, of spacing $\sqrt{2}$ pixels and turned by 45 degrees, a pattern called a quincunx. To that turned grid, a horizontal or vertical edge of the scene is a diagonal line, and a regular grid samples a diagonal line as a staircase, the same jaggies a diagonal shows on a screen. Bilinear interpolation reconstructs the staircase rather than the edge, and the steps of the staircase are the zipper's teeth.

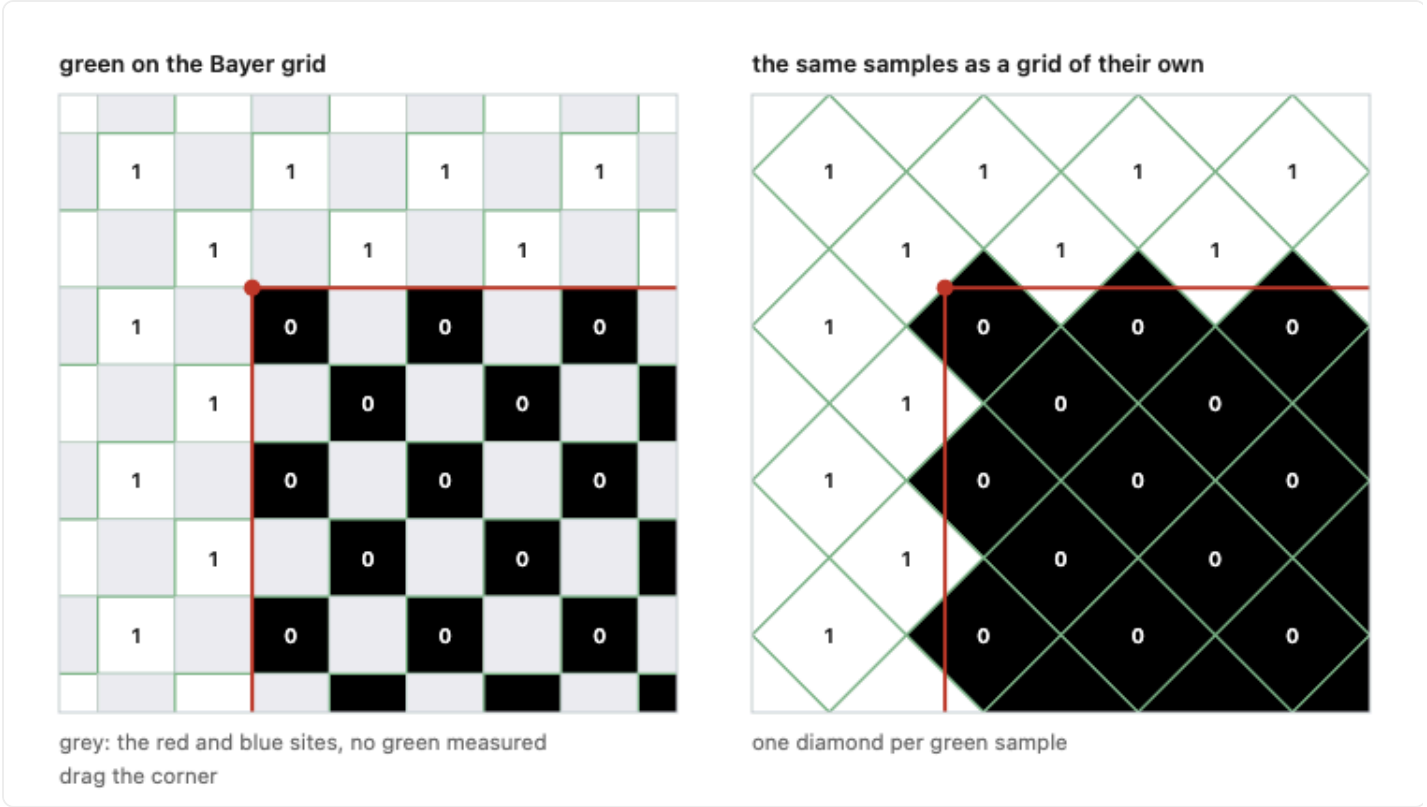
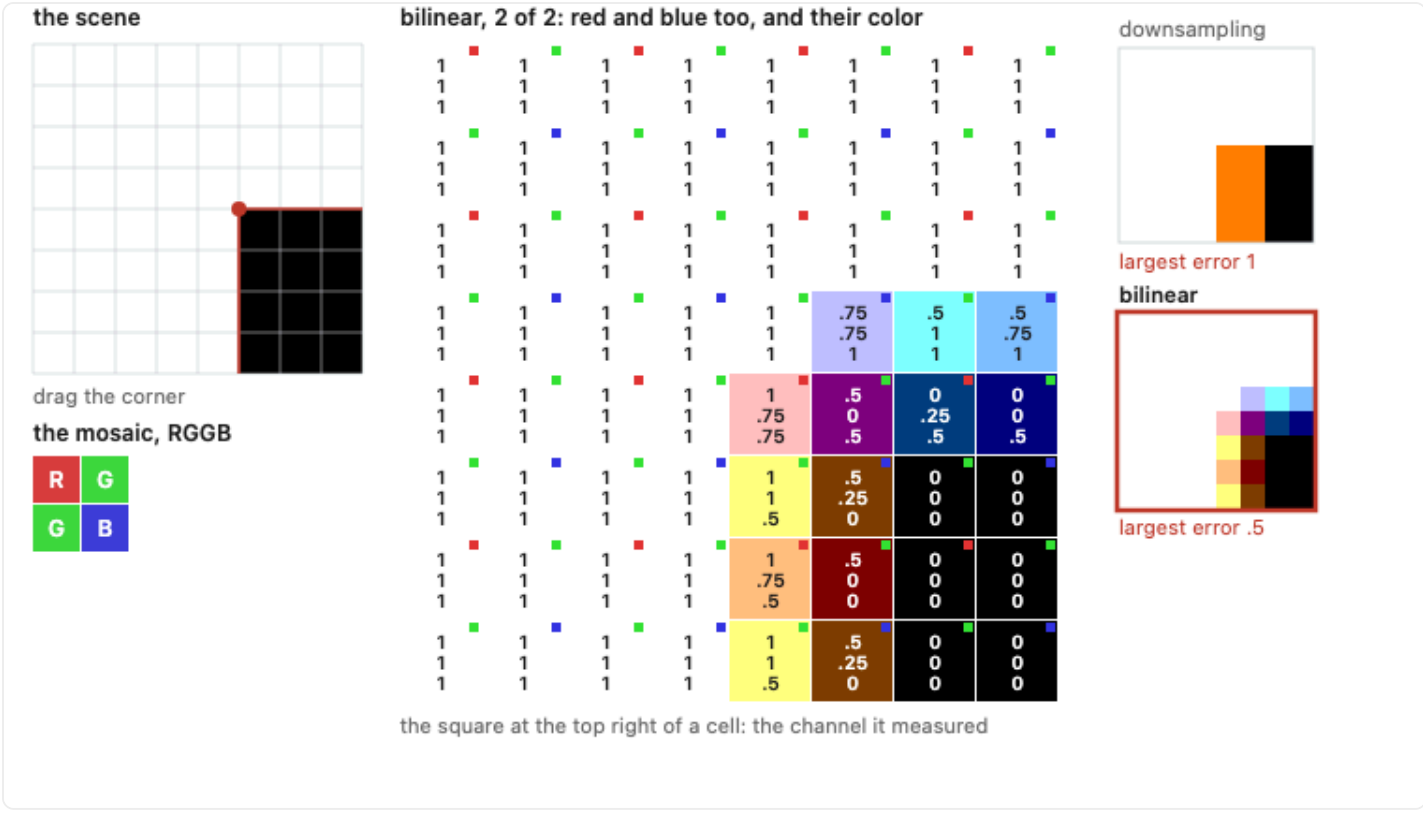


Figure 15.1.3 (reproduced). *Interactive. The green channel is a turned grid. Left, the corner's green samples on the Bayer grid: square cells, green on the checkerboard, the red and blue sites empty. Right, the same samples as a grid of their own: each green sample owns the diamond whose corners are the four red and blue sites around it, and the diamonds tile the plane with nothing missing. The numbers are the same in both panels. Turned 45 degrees, the green lattice is an ordinary square grid, and the corner's straight edges run diagonally across it as a staircase: to the green channel, the horizontal and vertical edges of the scene are 45-degree lines, and they alias like one.* Interactive: turn both panels from 0 degrees, where the Bayer grid is axis-aligned, to 45 degrees, where the green lattice is; drag the corner; hide or show the values.

False color is a colored fringe on fine detail that is gray in the scene. It happens because the three channels are sampled at different positions, so a thin structure lands on a red pixel and misses the blue ones, and the interpolation reports a color the scene did not have.



◀ | 1 input: mosaic | 2 downsample | 3 bilinear: G | 4 bilinear: R, B | 5 green-based: G along edges | 6 green-based: R−G, B−G | 7 green-based: interpolate | 8 green-based: add G back | ▶

⤓ settings from the previous stage

Figure 15.1.1, stage 4 (reproduced). *Bilinear red and blue: from two neighbors at a green site, from the four diagonal ones at the other color's site. The three planes are interpolated from different cells, so they disagree at the edge and the corner comes out fringed, largest error .5.* Interactive: the same controls at every stage.

Maze-like patterns often involve unstable direction choices in adaptive methods. They should not be attributed to a bilinear filter changing its interpolation direction.

These artifacts can involve aliasing, cross-channel disagreement and interpolation rules. Looking at their appearance helps diagnosis, but does not uniquely identify the algorithm or cause.

Concretely, for green, whose measured samples lie on a checkerboard, the four nearest measured neighbors sit above, below, left and right:

$$G_{\text{new}} = \tfrac{1}{4}\left(G_{\text{up}} + G_{\text{down}} + G_{\text{left}} + G_{\text{right}}\right). \tag{3}$$

Red and blue are sparser, so the geometry varies: at some empty sites the nearest red samples are the four diagonal ones, at others the two horizontal or two vertical. The principle is the same, and it is a small tent kernel applied per channel.

The zipper is worth doing with numbers once. Take a vertical black-and-white edge, value 0 against value 1, and look only at green. At a missing green site just on the white side, three green neighbors can be white and one black, giving 0.75 instead of 1, while the measured green site beside it stays at 1. The reconstruction therefore alternates between measured and interpolated positions, which is exactly the run of teeth. On the black side the same averaging lifts missing values above zero.

The corner showed the same numbers along a whole edge (Figure 15.1.1, stages bilinear-green and bilinear-rb).

◀ 1 bilinear 2 and green-based ▶

Click any picture to enlarge it; the arrow keys step through them.

The whole sign



raw Bayer mosaic, one colour per pixel



naive bilinear: false colour

The lettering, enlarged (the red box)



naive: orange and cyan on the letter edges

Input: problem set 3's raw mosaic signs-small.png (BGGR), © Frédo Durand.

Figure 15.1.4, stage 1 (reproduced). *A real mosaic, demosaicked by bilinear interpolation: the NO PARKING sign from problem set 3, a genuine BGGR mosaic straight from the sensor. First, the mosaic, each pixel painted in its filter's color. Then naive per-channel bilinear interpolation, with colored fringes along every letter, color that is not in the sign, because red, green and blue are sampled at different positions. The last picture enlarges the lettering. The sign is black on white, so the correct answer is known without a reference image: gray everywhere. Each picture enlarges on its own.*

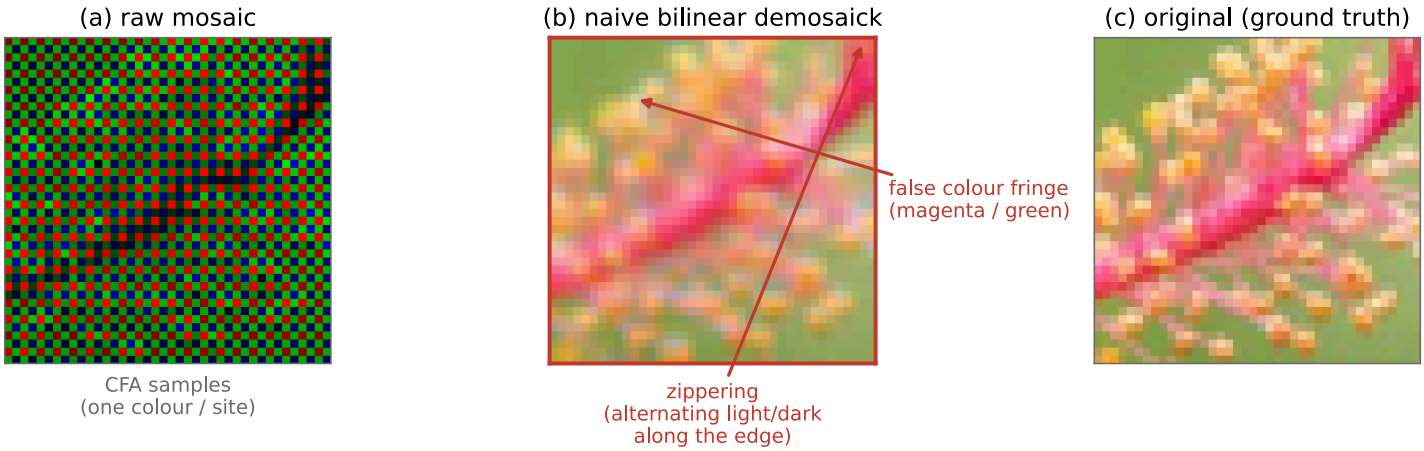


Figure 15.1.5 (reproduced). *Demosaicking, naive versus edge-aware. Left, the Bayer mosaic input. Center, per-channel linear interpolation: colors are right in smooth areas but strong edges show two artifacts: a zipper (the alternating light/dark teeth running along a near-horizontal or near-vertical edge) and color fringing (spurious colored speckles, e.g. orange and cyan, straddling a black-and-white boundary). Right, an edge-aware, green-guided result: the zipper and most of the fringing are gone and the edge stays crisp. The artifacts come from interpolating across an edge instead of along it, and from the three channels being sampled at different locations.*

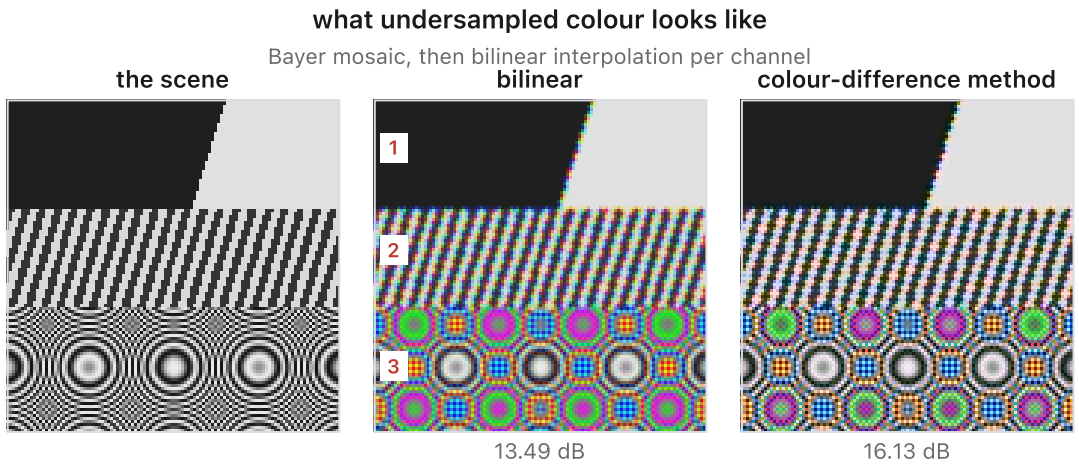


Figure 15.1.6 (reproduced). *Interactive. The three artifacts, labeled and reproducible. A synthetic target with a sharp edge, a thin gray line and a fine radial pattern is mosaicked and interpolated bilinearly, at a zoom where zippering, false color and the maze are each unmistakable, each marked by a numbered marker with a legend beside the image. A scale slider shrinks the target toward the sampling limit and the reader watches each artifact appear at the spatial frequency that produces it. Real photographic crops with the same three artifacts sit beside the synthetic ones.* Interactive: choose the target or a real crop, set the feature scale and the edge orientation, toggle each artifact's marker, and switch between the mosaic, the interpolation and the error.

Interpolate along the edge, not across it

in the book ↗

The first classical idea: decide, per pixel, which direction is safe to interpolate in.

For a missing green value, compute a horizontal and a vertical gradient from the available samples, and interpolate along whichever direction varies less. The test goes back to Robert Hibbard's Kodak patent Hibbard 1995 (Kodak patent): compare the differences between the green neighbors horizontally and vertically, and interpolate horizontally, vertically or from all four neighbors depending on which gradient falls below a threshold. Hamilton and Adams's method (Hamilton & Adams 1997) does this with a test that uses both the green neighbors and the same-color neighbors two pixels away, which is what lets a red pixel estimate the second derivative of the red channel and use it to correct the green interpolation.

$$\hat{G} = \frac{G_{\text{left}} + G_{\text{right}}}{2} + \frac{2R_{\text{center}} - R_{\text{left2}} - R_{\text{right2}}}{4} \quad \text{when the horizontal gradient is smaller.} \tag{4}$$

The second term is the correction that makes this work, and it is the second classical idea arriving early: it uses the red channel's curvature at the center to sharpen the green estimate, on the assumption that the channels share their structure.

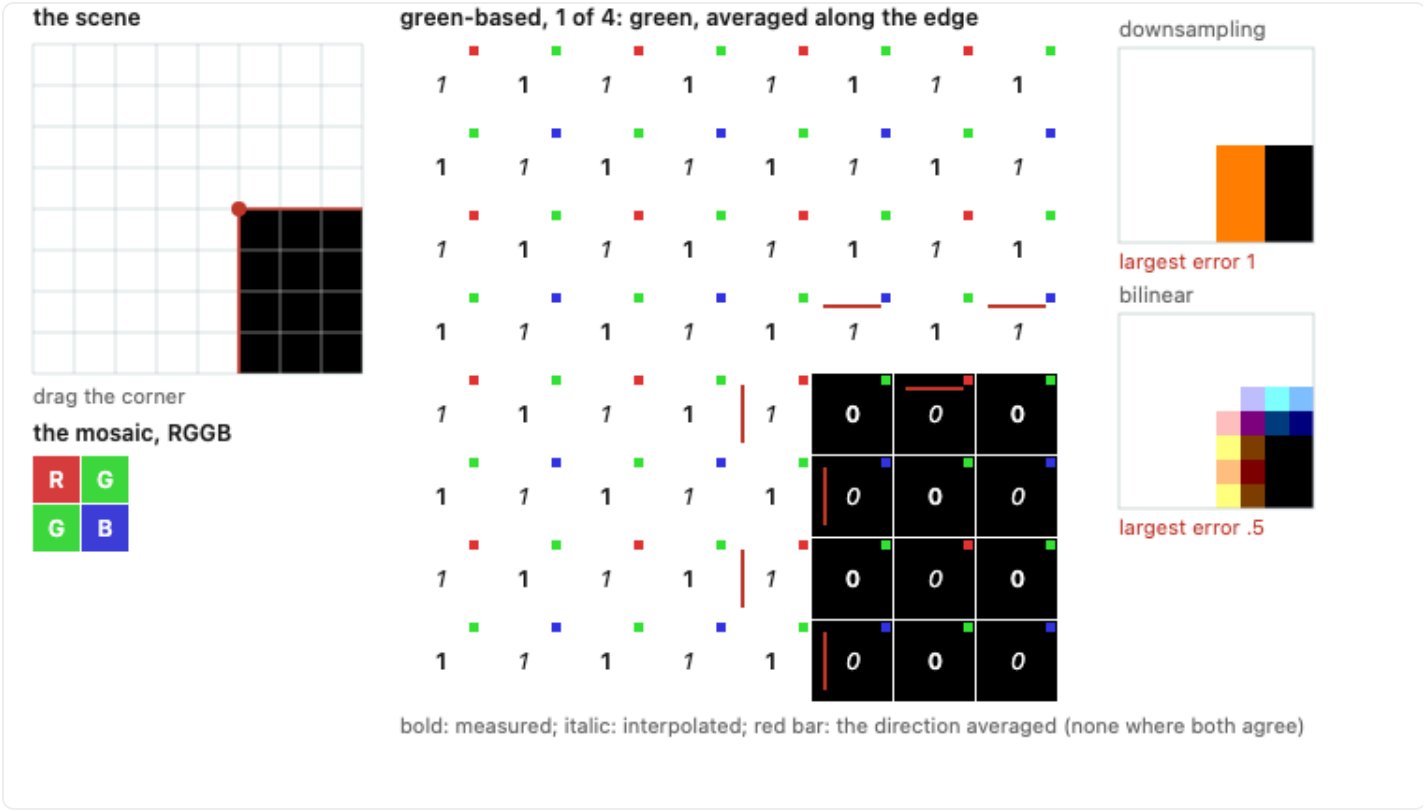
Variants differ in how the direction is chosen. Variable number of gradients (Chang, Cheung & Pan 1999 (VNG)) computes gradients in eight directions and averages the ones below a threshold, which handles corners and diagonals better than a two-way choice. Adaptive homogeneity-directed interpolation (Hirakawa & Parks 2005 (AHD)) reconstructs the image both ways, converts both to a perceptual color space, and keeps the version that is locally more homogeneous in color, which is a defensible criterion because the eye tolerates luminance detail and not chrominance noise.

The plain two-direction form of the same idea is worth seeing before the correction term arrives:

$$G_{\text{new}} = \begin{cases} \frac{1}{2}\left(G_{\text{up}} + G_{\text{down}}\right) & \text{if } |G_{\text{up}} - G_{\text{down}}| \le |G_{\text{left}} - G_{\text{right}}| \\ \frac{1}{2}\left(G_{\text{left}} + G_{\text{right}}\right) & \text{otherwise.} \end{cases} \tag{5}$$

Interpolate along whichever axis its two neighbors look most alike, on the bet that the other axis crosses an edge. The bet rests on the world being piecewise smooth: mostly flat regions separated by a relatively small number of strong, roughly one-dimensional edges. Get those right and the cases the eye notices are handled. A tie can take the four-neighbor average rather than preferring an axis, and continuous directional weights avoid abrupt switches. Fine periodic texture can make both directional differences misleading, and noise can flip the choice.

On the corner, the rule picks the right pair at every cell (Figure 15.1.1, stage edge-green).



◀ | 1 input: mosaic | 2 downsample | 3 bilinear: G | 4 bilinear: R, B | 5 green-based: G along edges | 6 green-based: R−G, B−G | 7 green-based: interpolate | 8 green-based: add G back | ▶

⤓ settings from the previous stage

Figure 15.1.1, stage 5 (reproduced). *Edge-directed green on the corner. At each red or blue site the rule averages the green pair across which green changes least; the mark in each cell shows the direction it chose. On this corner the estimate is exact everywhere.* Interactive: drag the corner so its first dark cell is red or blue, and the estimate misses at the corner.

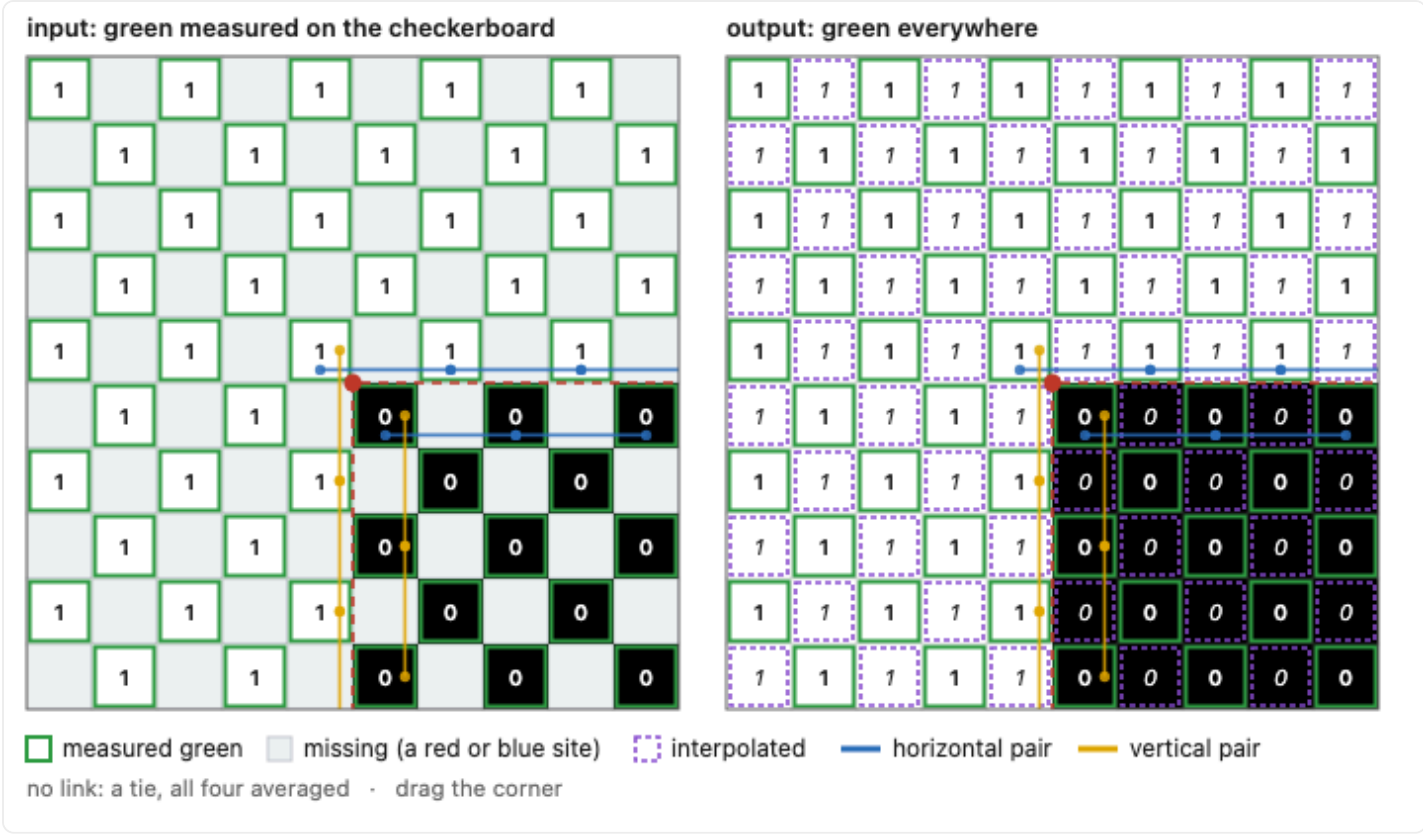


Figure 15.1.7 (reproduced). *Edge-directed green, input beside output. Left, green as measured: the green-ringed checkerboard, and the red and blue sites where it is missing. Right, green everywhere: each missing cell is filled with the average of the green pair across which green changes least, along the edge rather than across it, and marked as interpolated; where the two differences are equal, in flat areas, all four neighbors are averaged and no pair is drawn. The pairs are drawn in both panels, blue for horizontal and orange for vertical. The input can be the corner of Figure 15.1.1, black on white or on colored backgrounds and turned by any angle, or the problem-set raw mosaic of the NO PARKING sign, where the pairs run along the edge of the sign's border.* Interactive: choose the input, drag the corner or pan the sign, turn the corner, change the zoom, and hide or show the values.

◀ 1 bilinear 2 and green-based ▶

Click any picture to enlarge it; the arrow keys step through them.

The whole sign



raw Bayer mosaic, one colour per pixel



naive bilinear: false colour



green-based: clean

The lettering, enlarged (the red box)



naive: orange and cyan on the letter edges



green-based: the edges stay grey

Input: problem set 3's raw mosaic signs-small.png (BGGR), © Frédo Durand.

Figure 15.1.4, stage 2 (reproduced). *The NO PARKING sign again, now with green-based (color-difference) demosaicking beside bilinear: green first, interpolated along edges, then red and blue as differences from it. The fringes on the lettering are almost gone and the edges stay gray, which is what the sign's black and white require.*

> 💡 **BIG LESSON (L15.1) — LET THE DATA PICK THE DIRECTION**
>
> Edge-directed demosaicking is a first, very concrete instance of a theme that returns throughout the book: adapt the operation to the local structure of the image rather than applying the same fixed filter everywhere. Here the structure is an edge and the adaptation is choosing the interpolation direction; later, in edge-preserving filtering and denoising, the same instinct (use the similarity between pixels to decide how much they should influence each other) becomes the bilateral filter and the idea of affinity. Whenever a fixed linear filter blurs across something it should have respected, the cure is to look at the data first.

Algorithm 1

```
# edge-directed green interpolation, the Hamilton-Adams test
dH = abs(G[x-1,y] - G[x+1,y]) + abs(2*R[x,y] - R[x-2,y] - R[x+2,y])
dV = abs(G[x,y-1] - G[x,y+1]) + abs(2*R[x,y] - R[x,y-2] - R[x,y+2])
if   dH < dV: G_est = (G[x-1,y]+G[x+1,y])/2 + (2*R[x,y]-R[x-2,y]-R[x+2,y])/4
elif dV < dH: G_est = (G[x,y-1]+G[x,y+1])/2 + (2*R[x,y]-R[x,y-2]-R[x,y+2])/4
else:         G_est = average_of_both
```

This pseudocode is for a red center; exchange red and blue at a blue center. Copy measured green samples in noiseless interpolation and use a CFA-aware boundary extension that preserves color parity. Malvar–He–Cutler provides a useful fixed linear alternative with cross-channel Laplacian corrections (Malvar et al. 2004); it differs from selecting a direction per pixel.

Edge-directed green is the simplest case of a more general idea, steered reconstruction (Chapter 6.5, Resampling and upsampling). Reconstructing a sampled image means choosing, at each output pixel, weights for the nearby samples: a reconstruction kernel. A fixed kernel, like bilinear's, uses the same weights everywhere, so it blurs across edges and aliases along them. A steered kernel is shaped by the local image, long along an edge and narrow across it, so it averages where the signal is smooth and keeps the change sharp (Figure 6.5.19). The two-direction green rule is its coarsest version: two candidate kernels, one horizontal and one vertical, and a gradient test to choose between them. The variants above refine the choice. VNG considers eight directions, AHD compares whole reconstructions, and the Hamilton–Adams correction adds information from the other channels to the kernel it chose.

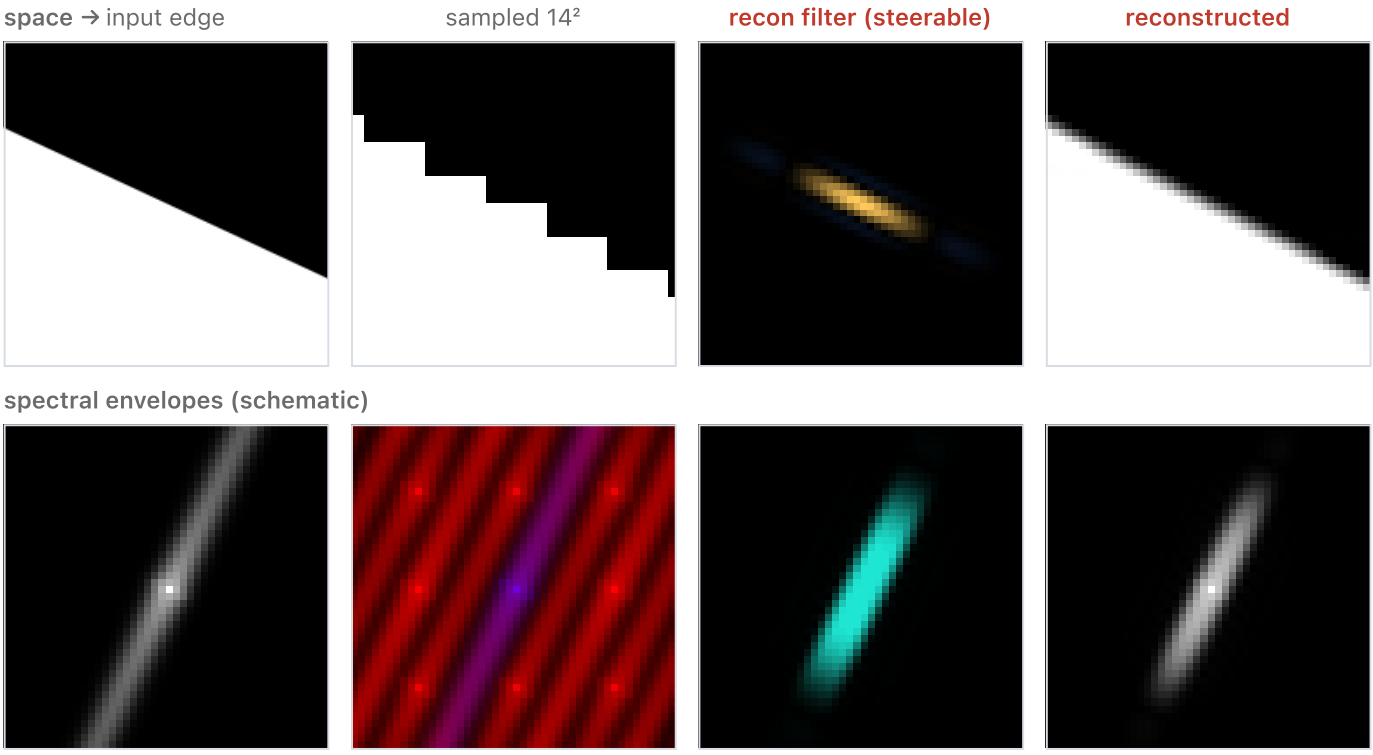


Figure 6.5.19 (reproduced). *Steered reconstruction, from the resampling chapter. The reconstruction kernel is an ellipse you aim by hand: aligned along an undersampled edge, it removes most of the staircase; misaligned, it brings the aliasing back; a round kernel only blurs it. Edge-directed green interpolation is the coarsest version of the same idea, a choice between a horizontal and a vertical kernel made by a gradient test.*

Working out the local reconstruction weights from the local pattern is the key idea of traditional demosaicking, and it reaches beyond it. Kernel regression makes the steering continuous, estimating an orientation and an elongation at every pixel from the local gradients (Chapter 10.7, Locally adaptive regression kernel (LARK)). RAISR learns a bank of filters and picks one per pixel by hashing the local gradient pattern (Chapter 14.4, Single-image super-resolution: classical methods); applied to a mosaic, the same scheme would amount to a learned gradient test. None of these methods writes down an image prior, yet each encodes one in the rule that chooses the weights: that the image is smooth along some direction at every pixel. Chapter 15.6, Learned demosaicking replaces the hand-written rule with one fitted to data.

Interpolate differences, not channels

in the book ↗

The second classical idea does more work than the first and is less often stated plainly.

Color differences can vary more slowly than individual channels, especially near neutral structure. They are not invariant to shading: if $R = rI$ and $G = gI$ for a common shading signal I, then $G - R = (g - r)I$, which varies unless $g = r$. The ratio $R/G = r/g$ is constant under this simplified model, but becomes unstable near zero green and changes at material or illumination boundaries.

So interpolate the difference. Estimate green everywhere first, since it is sampled twice as densely, then form $G - R$ at the red pixels, interpolate that smooth quantity, and recover red as $\hat{R} = \hat{G} - \widehat{(G - R)}$. The hard, high-frequency part of the problem is solved once, in the channel with the best sampling, and the other two channels inherit it.

On the corner, the differences are formed at the red and blue sites (Figure 15.1.1, stage differences).

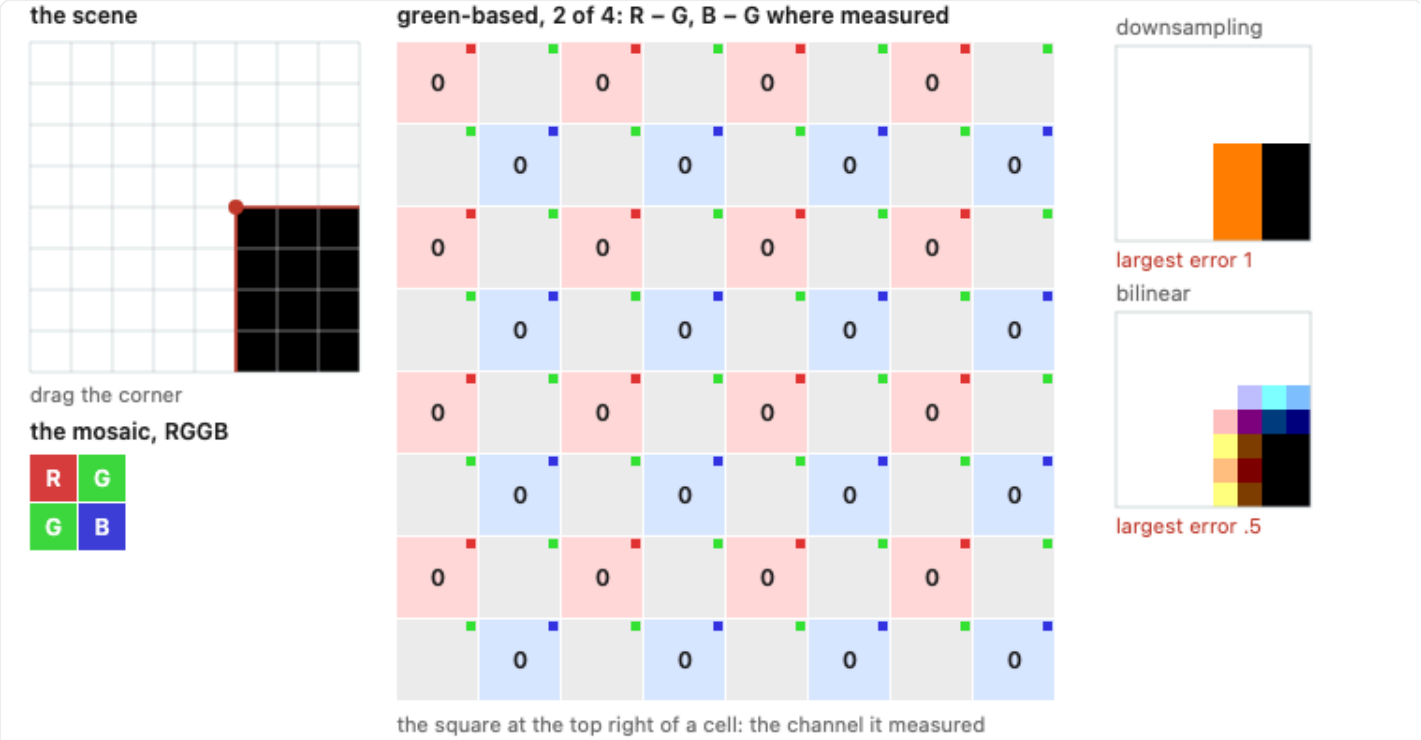


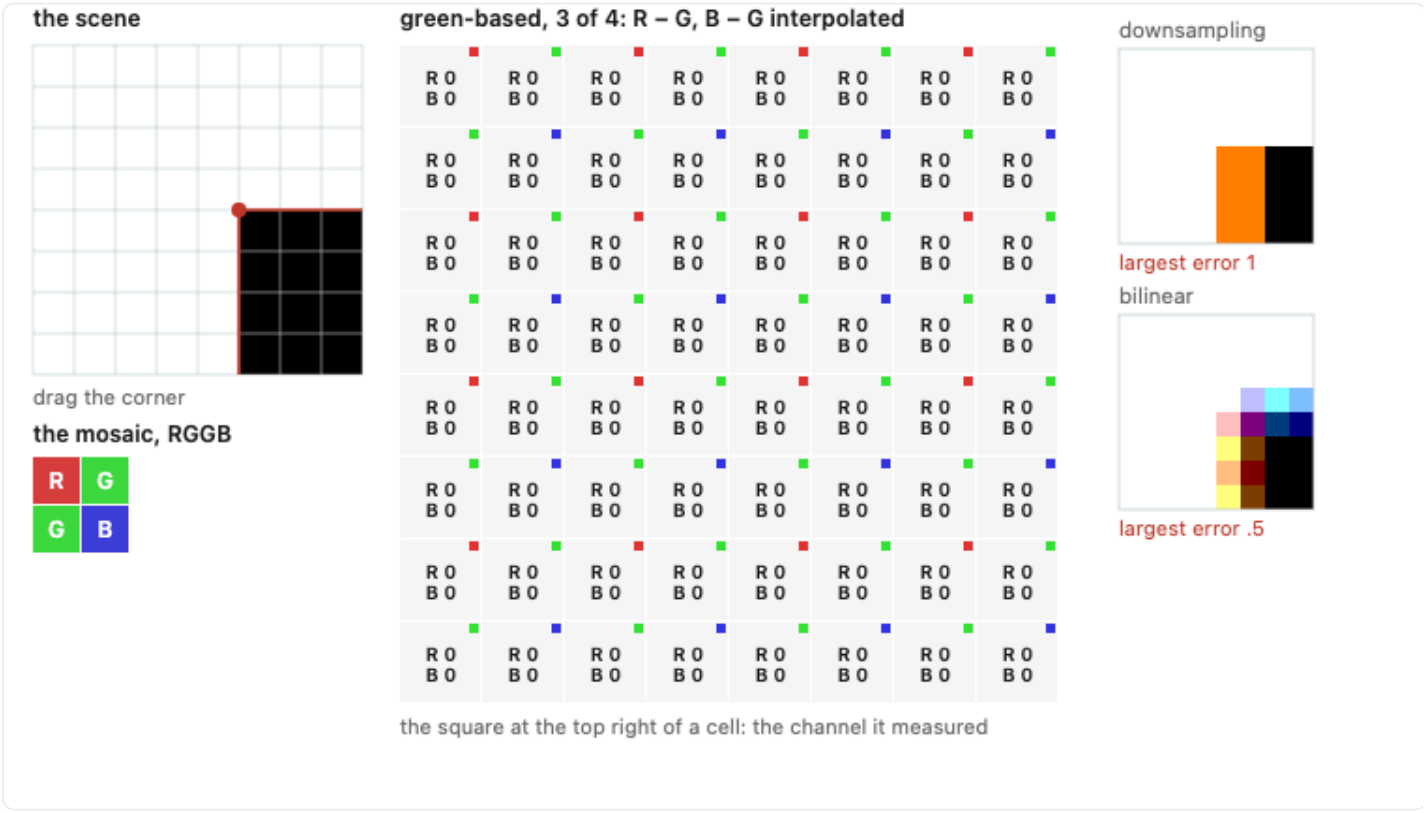
Figure 15.1.1, stage 6 (reproduced). *The color differences $R - G$ and $B - G$ at the red and blue sites, against the edge-directed green. On the gray scene they are zero wherever that green is exact.* Interactive: switch to a colored edge and the differences stop being zero.

Green-first reconstruction is common, but other joint and frequency-domain methods do not follow that ordering. Difference interpolation fails when the assumed shared structure is weak: strongly colored edges, channel clipping and chromatic displacement are important cases.

As a recipe, in four steps:

1. **Reconstruct green everywhere**, using the edge-directed method above, since green has the densest sampling.
2. **Form the difference** $R - G$ at every pixel where red was measured, and $B - G$ at every measured blue pixel, which is possible now that green exists everywhere.

3. **Interpolate the difference** with plain linear interpolation and no edge logic, because the difference is usually the smoother quantity. Sharp color boundaries remain the failure case.
4. **Add green back**: $R = (R - G) + G$, and likewise for blue.

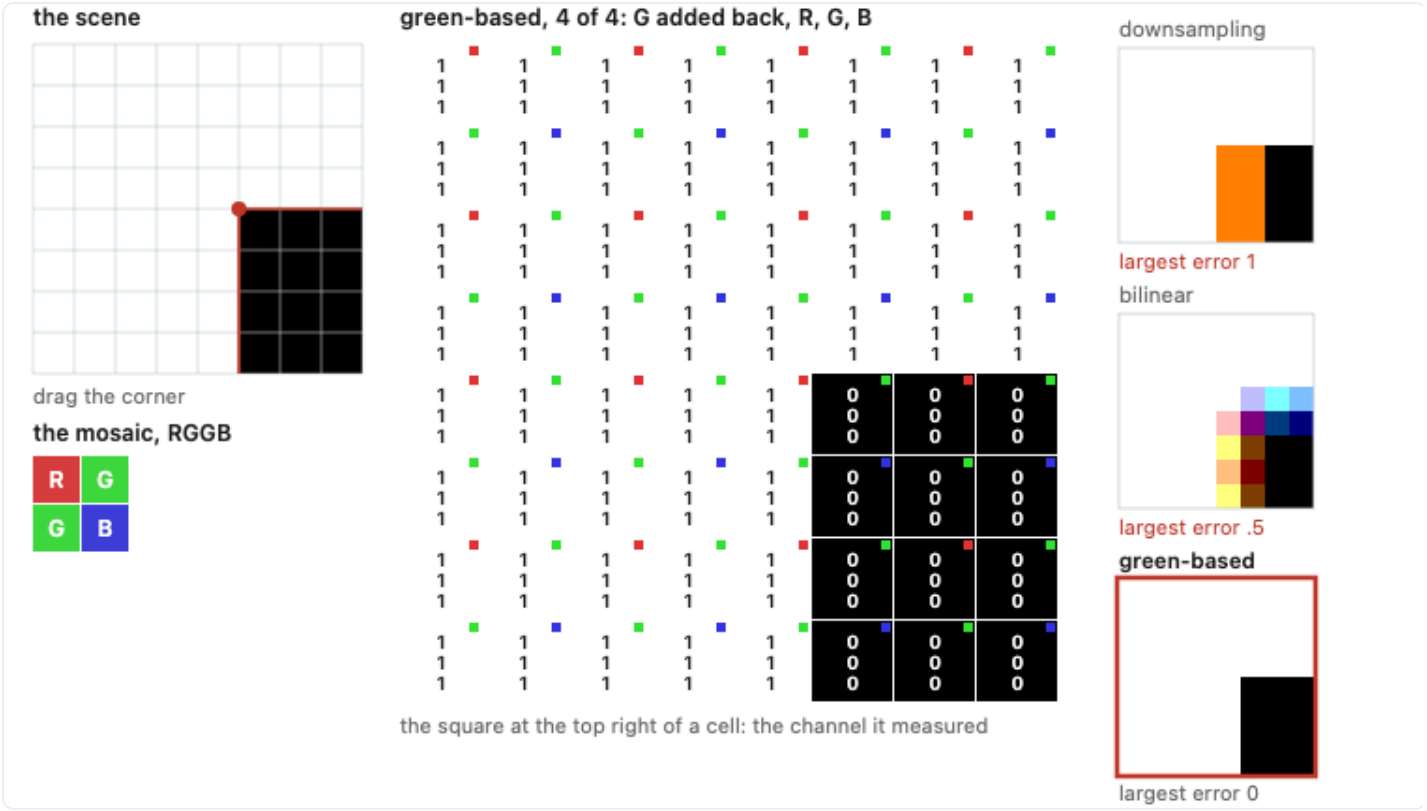


◀ | 1 input: mosaic | 2 downsample | 3 bilinear: G | 4 bilinear: R, B | 5 green-based: G along edges | 6 green-based: R−G, B−G | 7 green-based: interpolate | 8 green-based: add G back | ▶

⤓ settings from the previous stage

Figure 15.1.1, stage 7 (reproduced). *The differences interpolated bilinearly to every cell, the same rule as bilinear green, now applied to a quantity that is nearly constant across the edge.* Interactive: the same controls at every stage.

An achromatic edge has zero true color differences, so this keeps it gray, but only if the reconstructed green is also accurate at the measured red and blue sites. Green errors leave residual false color even on that simple target.



◀ | 1 input: mosaic | 2 downsample | 3 bilinear: G | 4 bilinear: R, B | 5 green-based: G along edges | 6 green-based: R−G, B−G | 7 green-based: interpolate | 8 green-based: add G back | ▶

⤓ settings from the previous stage

Figure 15.1.1, stage 8 (reproduced). *Green added back, the end of green-based (color-difference) demosaicking. On the gray scene the differences are zero wherever green is exact, so the result is exact: largest error 0, against .5 for bilinear and 1 for downsampling. Drag the corner so its first dark cell is red or blue and the green estimate misses at the corner, and the error follows it into all three channels. On a colored edge the differences are no longer constant across it: on red on white the method still beats bilinear (.35 against .45), on blue on yellow it does worse (.63 against .43).* Interactive: step back through the stages to see where each number comes from, drag and turn the corner, and change the scene.

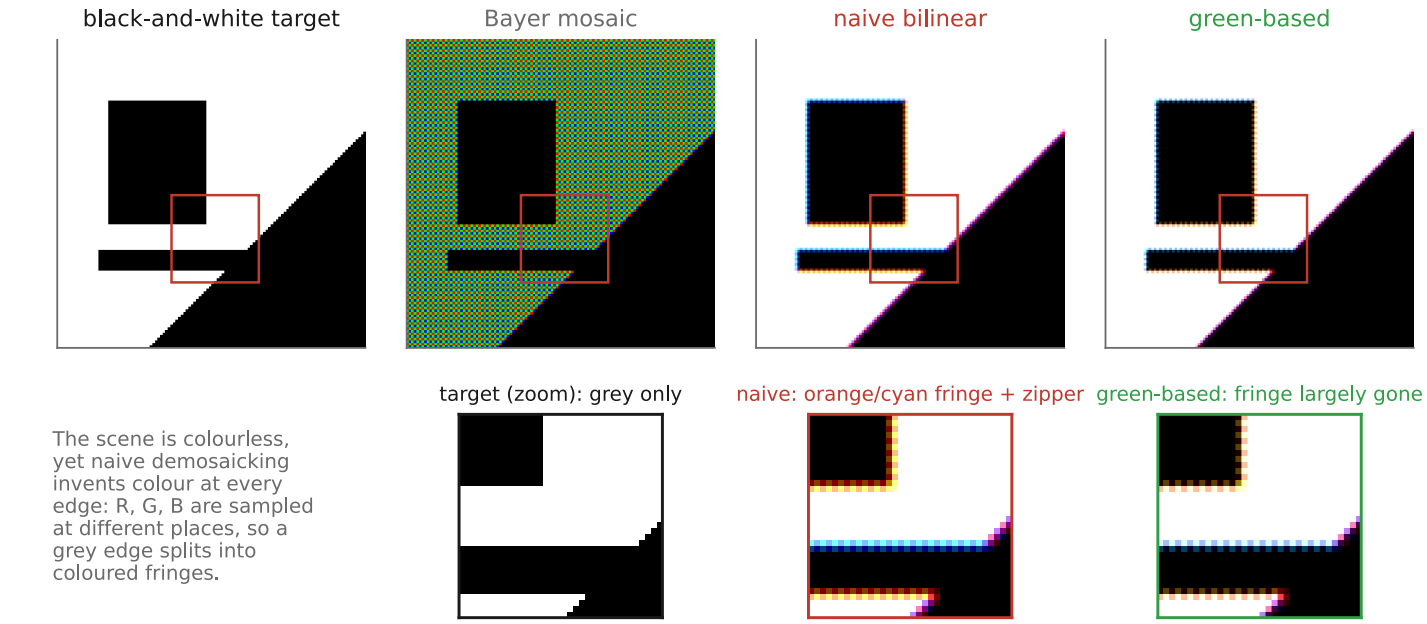


Figure 15.1.8 (reproduced). *Color out of a black-and-white scene: the cleanest way to see the problem and the fix. The target is purely achromatic (black shapes on white), so a correct demosaicker must keep it gray. Naive bilinear does not: orange and cyan fringes appear along every edge, plus zipper teeth: color invented from nothing, because red, green, and blue are sampled at different pixel positions and a sharp gray edge lands a little differently in each channel. Green-based (color-difference) demosaicking keeps it almost perfectly gray: with $R - G$ and $B - G$ constant across the achromatic edges, interpolating them is exact and the fringe collapses. The zoom row is the same corner enlarged.*

BIG LESSON (L15.2) — LUMINANCE CARRIES DETAIL, CHROMINANCE IS SMOOTH

Channel differences are often smoother than luminance-like structure, which makes guided reconstruction useful. Preserve that distinction as an image model rather than a law: fine color texture and saturated boundaries can break it.

Sidebar 5: difference or ratio?

We interpolated the *difference* $R - G$. A natural alternative is the *ratio* R/G, which encodes the assumption that hue (the proportion of red to green) is locally constant (the "constant-hue" model), and which can behave better than the difference when brightness varies a lot within a region. Ratios bring their own headache: G can be near zero in the shadows, and dividing by a small noisy number is asking for trouble. In practice the difference is simpler and more robust, the ratio can be sharper on saturated colors, and real pipelines mix or switch between them. The shared idea, interpolate *color* (a relationship between channels), not the raw channels, is what matters.

Even green-based, edge-directed demosaicking is not perfect. Fine, high-frequency color texture, a tiled roof, a striped shirt, a picket fence, breaks the smooth-color assumption and produces the residual false color often called maze or labyrinth artifacts, because at those scales the channels really are doing different things. That is where the modern methods earn their keep.

Where the maze comes from shows up best on a small patch of fine texture (Figure 15.1.9). At a pixel where the horizontal and vertical tests come out nearly equal, a little noise decides which one wins, so neighboring pixels choose differently, and the pattern of choices is the labyrinth.

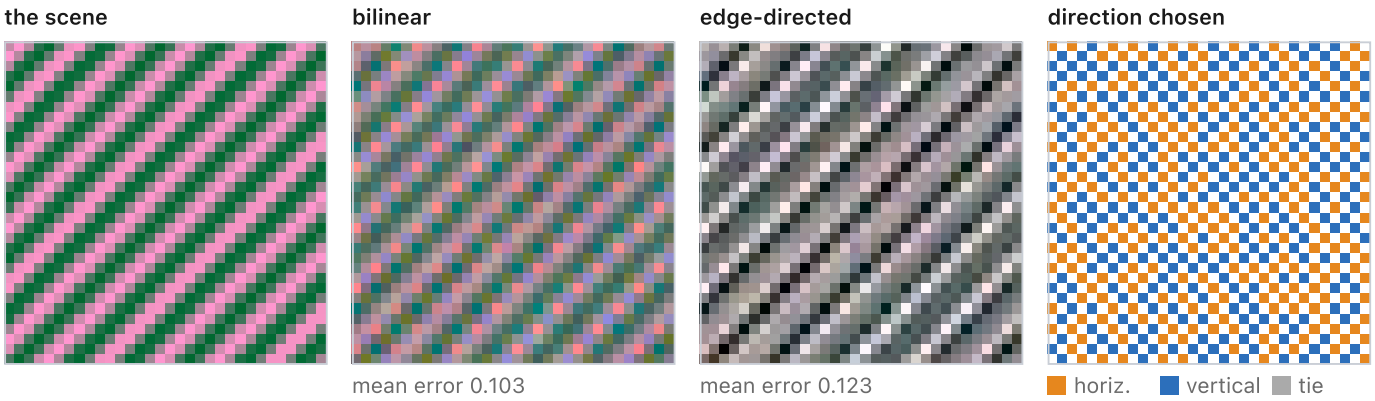
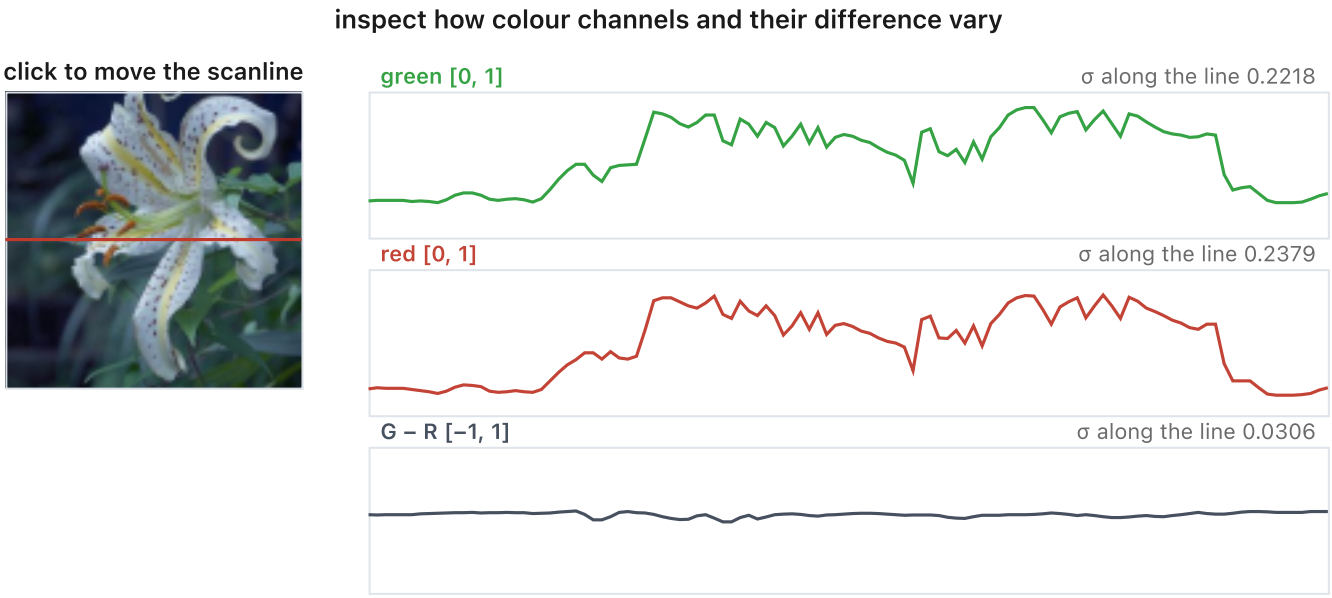


Figure 15.1.9 (reproduced). *Interactive. Fine colored stripes, 4 pixels apart at 45 degrees, kept through an RGGB mosaic with a little sensor noise, and reconstructed by bilinear interpolation and by edge-directed green (Hamilton and Adams' test, with the curvature correction) followed by color differences. The right panel is the direction each red and blue site chose: at 45 degrees the two tests are equal by symmetry, so without noise every site ties, and with noise the choices split about evenly and flip from one site to the next, a labyrinth. In the picture it shows as broken, blotchy stripes and lost color, and the edge-directed result ends up worse than bilinear (mean error .123 against .103). Take the color out and the edge-directed error falls to under a third (.037): when the channels agree, the curvature correction makes either direction nearly right, and the maze stays in the choices.* Interactive: set the stripe period and angle, the color in the stripes and the sensor noise; the patch is 64 pixels square, the central 32 shown.



Line SD: green 0.222, difference 0.031. SD measures variation, not spatial smoothness.

Figure 15.1.10 (reproduced). *Interactive. The assumption, measured on a real photograph. Three panels: the red channel, the green channel, and their difference, each with a scanline plot under it. The channels are full of structure and the difference varies far less, until the reader drags the scanline across a saturated color boundary, where it does not. Each plot shows the standard deviation along the line for its channel or difference, on the full range of values; a small spread for the difference is evidence for the assumption along that line, not proof that the difference is smooth everywhere. A second view shows the demosaicking error with and without the difference trick on the same crop, which is the same information as a result rather than as an argument.* Interactive: move the scanline, switch between $G - R$ and $G - B$, toggle the difference-based reconstruction, and hunt for the regions where the assumption breaks.

A red flower against green leaves is a useful counterexample to the shared-edge assumption: the boundary can have very different contrast in red and green. A clipped specular highlight breaks the linear relationship for another reason, because one channel stops increasing while another still responds. Lateral chromatic aberration breaks it geometrically by placing the same boundary at different coordinates. These cases need different remedies even when their demosaicked outputs all show colored fringes.

Demosaicking and other aspects of imaging

OPTIONAL
in the book ↗

Demosaicking is one part of an imaging system. The optics and the sensor decide what the mosaic records, and the stages of the camera pipeline around it (Part 8, Basic image processing and ISP) decide what it is handed and what happens to its output. Each method in this chapter and in Chapter 15.2, Advanced traditional demosaicking assumes something about its input, and these other parts decide whether the assumption holds.

The anti-aliasing filter. Many cameras place an optical low-pass filter in front of the sensor. It blurs the image slightly before sampling, so that less fine detail aliases into color moiré, the failure no demosaicker can fully undo because the mosaic cannot tell it from real color. Many recent cameras have weakened or dropped it and leave moiré to the demosaicker. Chapter 15.3, Mosaics and aliasing explains what the filter can and cannot remove, and why it has been disappearing.

Grouped layouts: Quad and Nona. Phone sensors that group same-color photosites in 2 × 2 (Quad-Bayer) or 3 × 3 (Nona-Bayer) blocks change what a demosaicker receives. In low light the sensor bins each block and hands over an ordinary Bayer mosaic at a quarter or a ninth of the pixel count, and the methods of this chapter apply unchanged. At full resolution the same color is repeated over neighboring sites, so red, for example, is missing from larger regions than in a Bayer mosaic, and the nearest red sample can be three photosites away. A Bayer demosaicker applied directly to that layout uses the wrong colors at most sites; the usual route is to remosaic the grouped layout into an estimated Bayer mosaic first, which is itself an interpolation, then demosaick, or to demosaic the grouped layout directly (Chapter 15.10, Quad-Bayer and Nona-Bayer processing).

Color routers. A nano-prism or metasurface color router keeps the Bayer layout but steers each band of light toward the photosite that measures it instead of absorbing it (Chapter 4.6, Sensing color: multiplexing strategies). Each photosite then collects light from a wider area than its own, which acts as a mild anti-aliasing filter (Chapter 15.3, Mosaics and aliasing), and part of each band lands on neighbors of the wrong color, which broadens and mixes the channels' spectral responses. The demosaicking algorithms still apply, but the color correction that follows has to undo more mixing.

Black level. The sensor adds an offset to every sample, and it has to be subtracted first. Color differences and ratios are meaningful only on values proportional to light: with the offset left in, a gray edge no longer has a constant $R - G$ across it, and the ratio model of the "difference or ratio?" sidebar is wrong everywhere.

Defects and lens shading. A hot or dead photosite is one wrong sample, and interpolation copies it into its neighbors' missing colors, so a single bad value becomes a colored speck several pixels across; repair it on the mosaic. Lens shading darkens the corners by a different amount in each channel, which demosaicking would read as color. Both are covered in Chapter 15.7, Neighboring problems in the raw stage.

White balance. The color-difference methods work best when a neutral surface has equal values in all three channels, because then the difference across a gray edge is zero on both sides and interpolating it is exact. Raw values under most lights are far from equal for a neutral, so many pipelines apply the white-balance gains to the mosaic before demosaicking (Chapter 8.10, Auto-exposure and auto white balance). The gains then scale clipped samples too: a highlight that clipped in green is no longer at a common ceiling once red and blue are multiplied, which is where pink highlights come from, so saturated samples need to be marked before the gains are applied.

Chromatic aberration. Lateral chromatic aberration shifts the color channels relative to one another toward the edges of the frame. The cross-channel assumption then fails exactly at edges, where it is needed, and a demosaicker reads the misregistration as a colored edge. Correcting it before or during demosaicking avoids fringes that are much harder to remove afterward (Chapter 22.2, Aberrations correction).

Noise. Interpolation mixes neighboring samples, so independent sensor noise becomes spatially correlated color noise, which is harder for a denoiser to recognize. Noise also flips the edge-direction decisions of this chapter. Denoising the mosaic first, or demosaicking and denoising together, avoids both (Chapter 15.8, Joint demosaicking and denoising).

After demosaicking. The color matrix, tone curve and sharpening all act on the reconstructed image, and sharpening in particular amplifies zippering and maze patterns along with real detail (Chapter 8.8, Sharpening). A

demosaicker that leaves small artifacts can look acceptable until the rest of the pipeline has run, so judge it on the finished rendering as well as on its direct output.

The practical rule is to know which of these steps have been applied to the data a demosaicker receives, in what units, and to tell the algorithm.

And all of it should happen on linear values. Black-level subtraction, defect repair, lens-shading and white-balance gains, and demosaicking itself operate on numbers proportional to light, and the image becomes nonlinear only afterward, when the tone curve and the gamma encoding prepare it for display. The reasons are the ones this chapter keeps meeting. Interpolation averages, and an average of gamma-encoded values is not the encoding of the average light, so a demosaicker run on encoded values darkens fine detail and shifts colors at edges (Chapter 6.5, Resampling and upsampling). Color differences and ratios describe the scene only when the values are proportional to light. And the noise model that joint methods rely on is simple only in linear units, before a curve stretches the shadows and compresses the highlights.

Advanced traditional demosaicking

OPTIONAL
in the book ↗

The methods below, published between the late 1980s and the mid-2000s, combine the two ideas of Chapter 15.1, Basic traditional demosaicking, edge direction and color differences, in different proportions, and hand-designed methods of this family are still what most cameras and phones run in their image signal processors, where a demosaicker must run on every preview and video frame, fast, small and predictable. Learned demosaicking has shipped mainly in desktop raw converters, such as Adobe's Enhance Details and DxO's DeepPRIME, which also denoises (Chapter 15.6, Learned demosaicking). Some phone pipelines avoid single-frame demosaicking altogether by merging a burst of raw frames directly onto a full-color grid (Chapter 14.2, Multi-frame super-resolution). Figure 15.2.1 runs ten of them side by side on the same mosaic, including the residual interpolation and ARI of this chapter, so each method's assumption can be seen where it fails.

Algorithm lab — Demosaicking

ten ways to invent the two colors the sensor never measured

Task: Demosaicking



sensor noise σ: 0

resolution: 100%

1 | 2 | 4 lines

Compare | Algorithms | Pseudocode | Slides

A 26.3 dB · SSIM 0.916 · 6 ms

Nearest neighbor

B 30.5 dB · SSIM 0.962 · 14 ms

Bilinear



C 36.0 dB · SSIM 0.990 · 26 ms

Gradient-corrected (Malvar–He–Cutler)

correction: 1

then Freeman's median: none

D 36.8 dB · SSIM 0.989 · 24 ms

Edge-directed (Hamilton–Adams)

threshold: 0

then Freeman's median: none

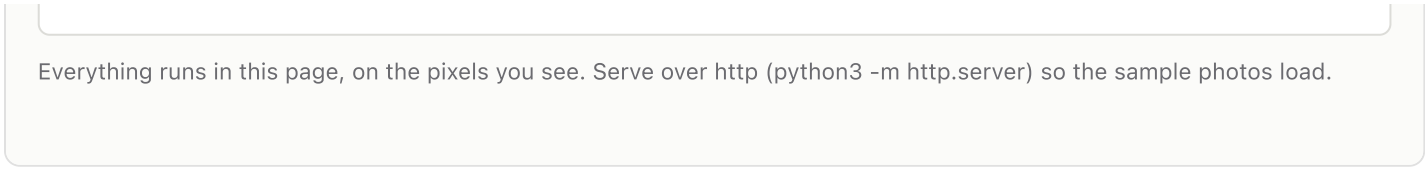
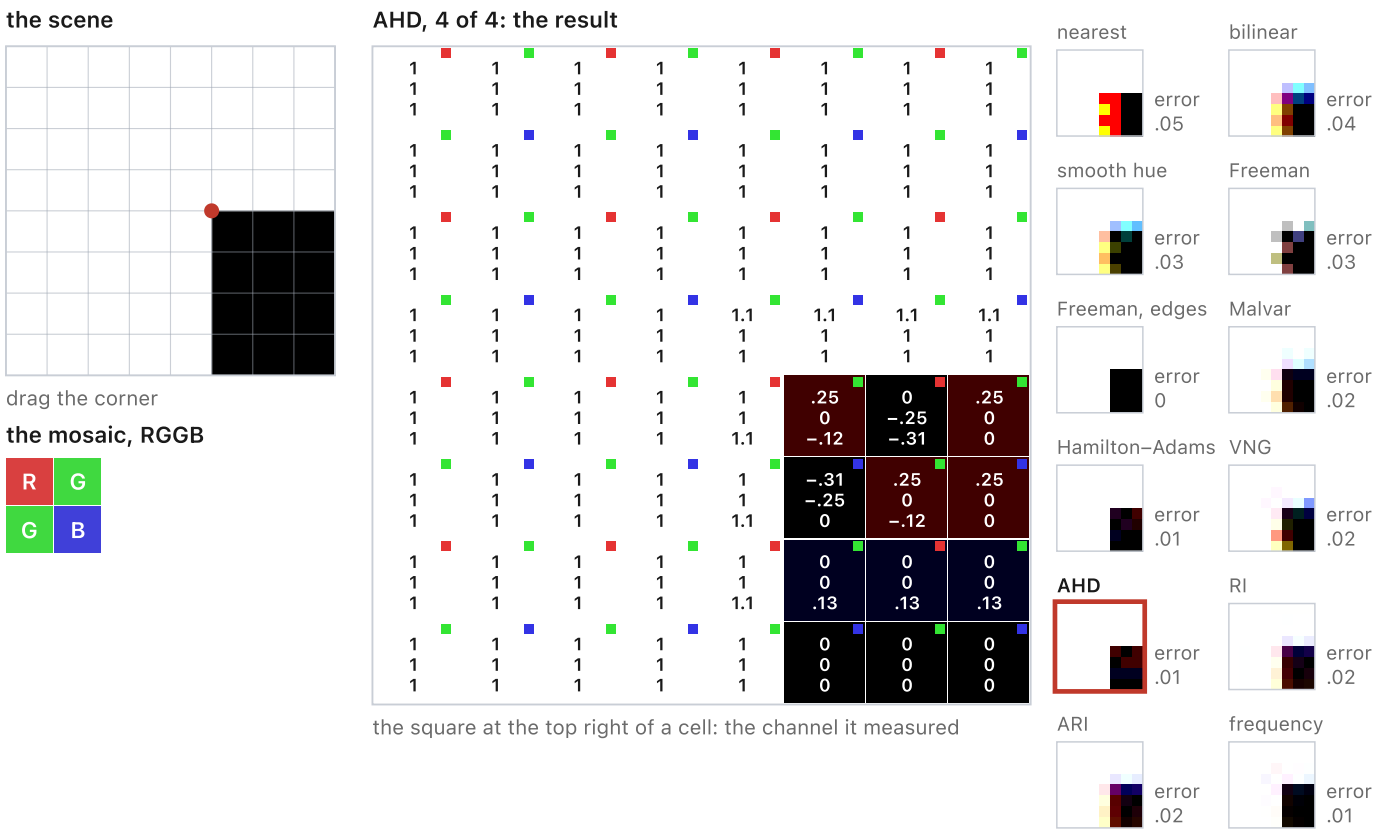


Figure 15.2.1 (reproduced). *Interactive: the demosaicking lab. One Bayer mosaic reconstructed by up to four methods at once, chosen from ten classical methods: nearest neighbor, bilinear, smooth hue transition, Freeman's median of color differences (on bilinear, or on the chapter's edge-based demosaicking), Malvar–He–Cutler, Hamilton–Adams, variable-number-of-gradients (VNG), homogeneity-directed (AHD), residual interpolation (RI) and adaptive residual interpolation (ARI), in a mouse-driven four-way split with a magnifier on the pointer, per-pane parameter sliders and live PSNR and SSIM against the original. The last six can be followed by Freeman's median, 3 × 3 or 5 × 5. A pane can instead show the ground truth, the raw mosaic, the difference of two panes, or a pane's amplified error; a sensor-noise slider corrupts the mosaic before reconstruction, a resolution slider gives the sensor fewer pixels over the same scene, and the zone-plate and Siemens-star inputs make false color and zipper unmissable; the zone plate and the star are band-limited: their finest rings and spokes fade to gray at the sensor's sampling limit, so the ground truth carries no aliasing of its own.*

Nearest neighbor copies each missing value from a nearby sample of the right color, with no arithmetic. The name is loose, because most sites have several equally near samples of a given color: a red site has four green neighbors at the same distance, and four blue ones on the diagonals. An implementation breaks the tie with a fixed rule, usually by working tile by tile. In the lab's version, every pixel of a 2 × 2 tile takes that tile's one red and one blue, a green site keeps its own green, and the other two sites take the same one of the tile's greens. That makes it a close relative of the downsampling of Downsampling and bilinear interpolation, and how they fail: downsampling turns each tile into one pixel, and nearest neighbor is almost that same small image blown back up to full size by copying each value into the four pixels of its tile. The only differences are that the measured greens stay where they were and nothing is averaged. Nearest neighbor therefore keeps the pixel count but adds almost no detail over downsampling, and it keeps downsampling's flaw: each color is attributed to a place up to a pixel away from where it was measured, so every edge becomes a colored staircase, the strongest zipper of any method here. It costs nothing, which is why it survives in live previews and thumbnails. **Bilinear** interpolation (Downsampling and bilinear interpolation, and how they fail) averages instead of copying and fixes the misregistration in smooth regions, not at edges.

Figure 15.2.2 works through every method of this section on a small mosaic, one step per stage, starting with nearest neighbor.

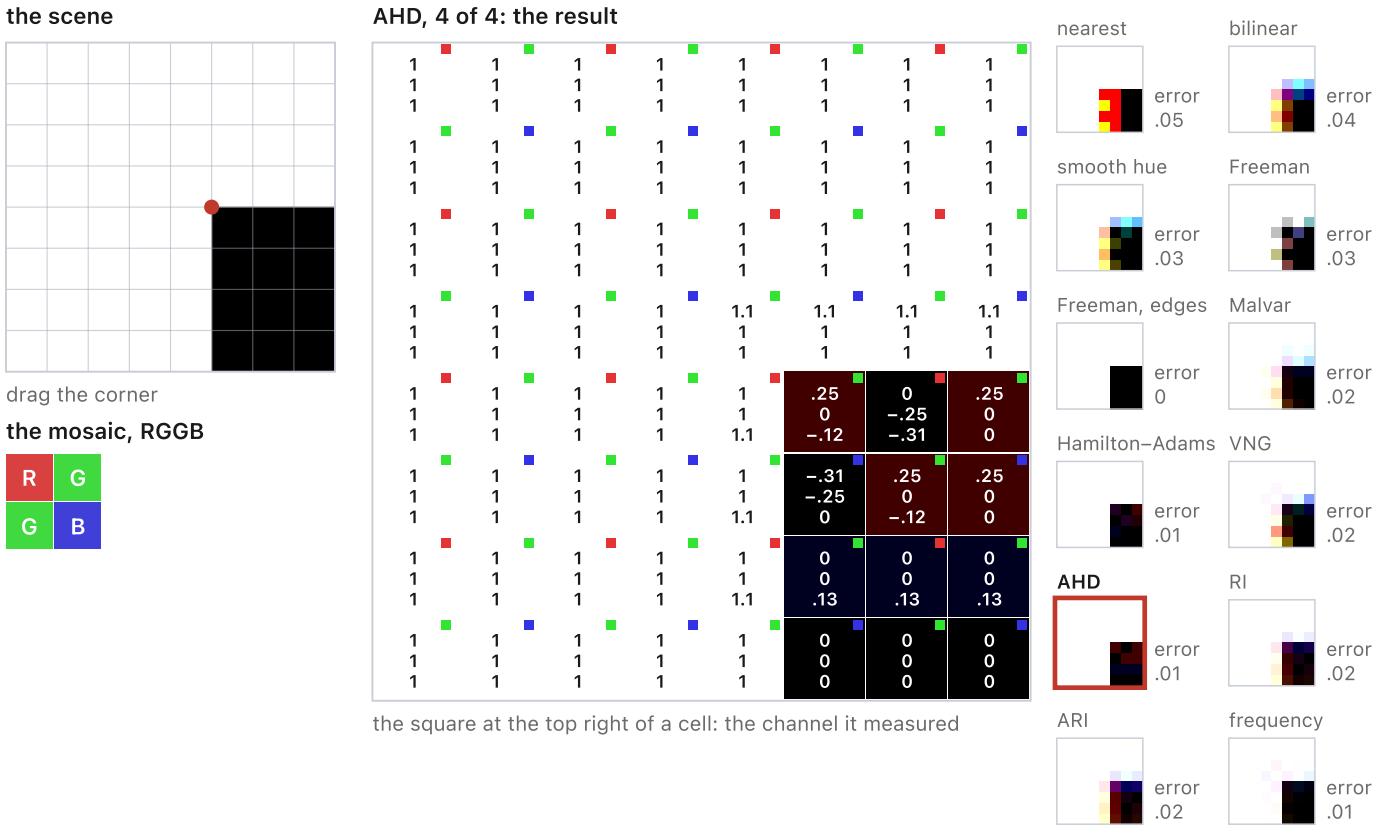


◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | 11 Malvar: correction | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | 16 VNG: directions | 17 VNG: average | 18 AHD: horizontal | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | 22 RI: G by residuals | 23 RI: tentative $\tilde{R} = aG + b$ | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | 27 frequency: spectrum | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶

Figure 15.2.2, stage 2 (reproduced). *The methods of this section, one step at a time. A small scene is area-sampled onto a grid and kept through an RGGB mosaic; the middle grid shows the numbers of the current step, bold where measured and italic where interpolated, and each method's result joins the column on the right once its last step is reached, with its mean error against the scene. This stage is nearest neighbor: each 2 × 2 tile's red and blue fill the whole tile, a green site keeps its green, and the tile's top-right green fills its red and blue sites. The input is a corner, a sine grating, a Siemens star or a zone plate, in three pairs of colors, on a grid of 8 × 8 with the values or of 16 × 16 or 32 × 32 without them, which the star and the zone plate need. The zone plate and the star are band-limited: their finest rings and spokes fade to gray at the sensor's sampling limit, so the ground truth carries no aliasing of its own.* Interactive: choose the input, its colors and the grid, set the corner's or the grating's angle, the grating's period, the star's spokes or the zone plate's highest frequency, drag the corner or the pattern's center, and step through each method with the stage buttons.

Smooth hue transition Cok 1987 (smooth hue transition), one of the first methods to use one channel to reconstruct another, interpolates green first and then interpolates hue rather than red and blue: the ratio R/G at the red sites, spread to its neighbors and multiplied back by green. It assumes that hue changes slowly across an object even where brightness does not. The choice of hue coordinate matters, and the lab lets you switch it: a ratio becomes unstable where green is near zero, a difference $R - G$ does not, and a log ratio sits in between. The difference form is the recipe of Interpolate differences, not channels.

Its three steps are the smooth-hue stages of Figure 15.2.2.

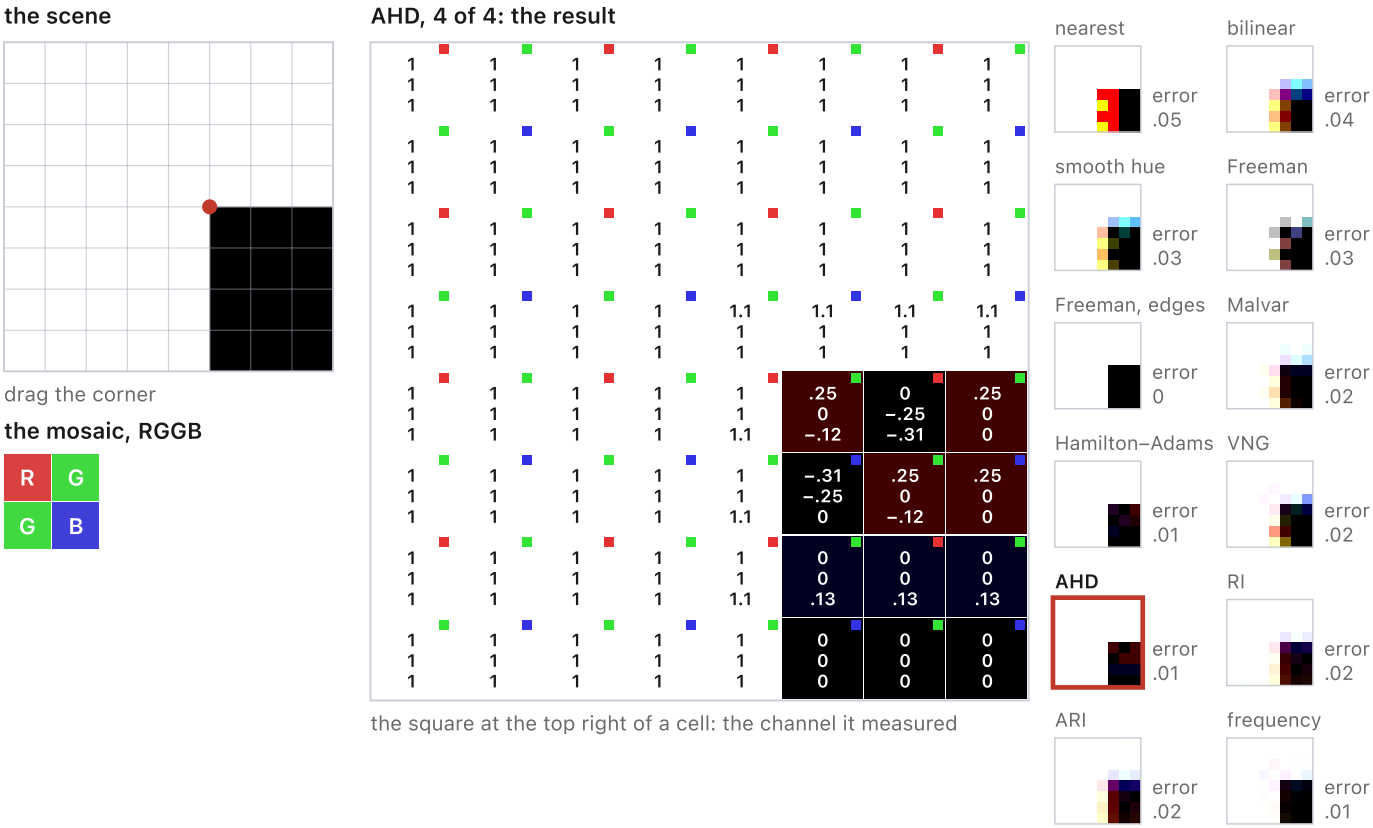


◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | 11 Malvar: correction | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | 16 VNG: directions | 17 VNG: average | 18 AHD: horizontal | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | 22 RI: G by residuals | 23 RI: tentative $\tilde{R}$ = aG + b | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | 27 frequency: spectrum | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶ | ↓ settings from the previous stage

Figure 15.2.2, stage 4 (reproduced). *Smooth hue transition: R/G and B/G at the red and blue sites, against the bilinear green; the ratios interpolated; and multiplied back by green. Beside a black region the green is near zero and the ratio is unstable; the lab's guard, one level added to both, keeps it finite.* Interactive: the same controls; the stages step through the three steps.

Freeman's method Freeman 1988 (median demosaicking) demosaicks with bilinear interpolation, then cleans up the hue: it applies a 3 × 3 median to the difference channels $R - G$ and $B - G$ and adds green back. Within an object the color difference is nearly constant, so a false-color pixel is an isolated outlier in a flat channel, which is exactly what a median removes. A 5 × 5 median often does a better job, because false color on fine texture tends to come in clusters wider than a 3 × 3 window can outvote, though a larger median also starts to carry color across object boundaries. The median does nothing for zippering in the green channel, so although the method is described on top of bilinear, it is mostly used as a clean-up step on top of a better demosaicker. Figure 15.2.1 offers both bases: on bilinear, the median removes most of the fringes and leaves the zippering; on the chapter's edge-based demosaicking, the green is already good and the median only has the remaining false color to remove. The same pass survives as the final clean-up of many later methods, AHD among them. (Freeman's patent differs in its details: it describes two colors alternating along a line, interpolates each linearly, and applies a one-dimensional median to their difference, five pixels wide in its example, with each pixel keeping its own measured value. The two-dimensional Bayer version above is the one usually implemented.)

Figure 15.2.2 shows the three steps on the grid.



◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | 11 Malvar: correction | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | 16 VNG: directions | 17 VNG: average | 18 AHD: horizontal | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | 22 RI: G by residuals | 23 RI: tentative $\tilde{R} = aG + b$ | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | 27 frequency: spectrum | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶ | ↓ settings from the previous stage

Figure 15.2.2, stage 7 (reproduced). *Freeman's method in the Bayer form the text describes: $R - G$ and $B - G$ from the bilinear planes at every pixel, drawn as the color they add to gray; their 3 × 3 median; and green added back. A fourth stage puts the median on the chapter's edge-based demosaicking instead of bilinear and adds that green back. There the false color comes in bands along slanted edges and in patches on fine patterns rather than in lone pixels, so the median there is 5 × 5, which removes most of what a 3 × 3 median keeps.* Interactive: the same controls; the stages step through the three steps, then the median on the edge-based result.

Gradient-corrected linear interpolation Malvar et al. 2004, by Malvar, He and Cutler, keeps a fixed linear filter but adds the Laplacian of the channel measured at the center pixel, scaled by a gain, to the bilinear estimate of each missing channel. A red pixel's curvature, for example, corrects the green estimated there, on the assumption that the channels vary together. The result is a set of fixed 5 × 5 filters with no decisions, so it is fast, easy to run on

a GPU, and free of the unstable direction choices that make mazes; it is, for example, what MATLAB's `demosaic` function implements. Because it does not adapt, it still blurs across edges where an adaptive method would not.

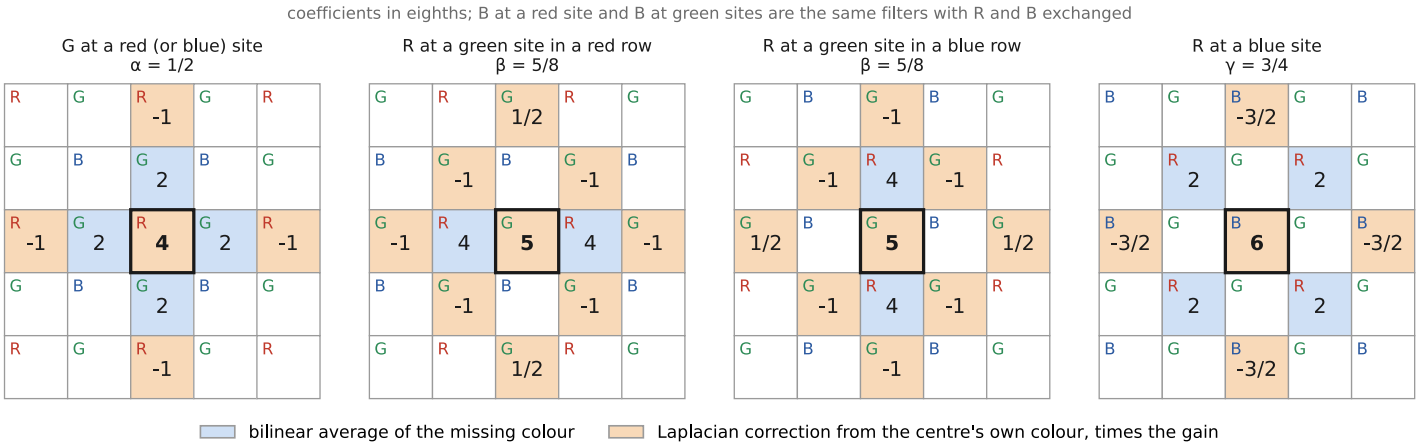
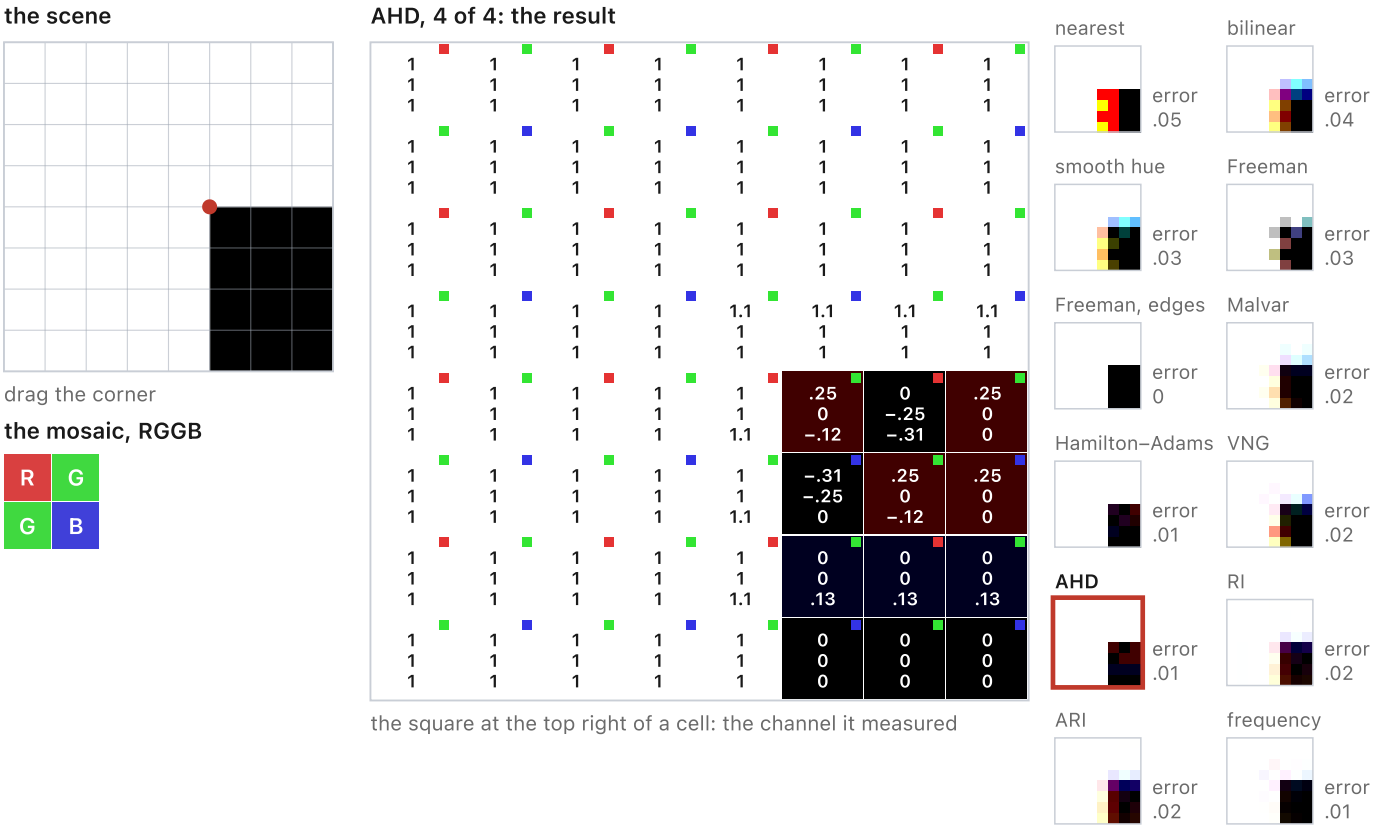


Figure 15.2.3 (reproduced). *The four Malvar–He–Cutler filters, in eighths: green at a red or blue site, red at a green site in a red row and in a blue row, and red at a blue site (blue at a red site is the same by symmetry). Each cell shows its weight and the mosaic color it applies to. The bilinear part is shaded blue and the Laplacian correction from the center channel orange, with the published gains α = 1/2, β = 5/8 and γ = 3/4.*

The same correction, pixel by pixel, is in Figure 15.2.2.



◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | **11 Malvar: correction** | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | 16 VNG: directions | 17 VNG: average | 18 AHD: horizontal | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | 22 RI: G by residuals | 23 RI: tentative $\tilde{R} = aG + b$ | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | 27 frequency: spectrum | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶ | ↓ settings from the previous stage

Figure 15.2.2, stage 11 (reproduced). *Malvar–He–Cutler: the correction each missing channel receives, the Laplacian of the measured channel scaled by the published gains; then the bilinear estimate plus that correction. Near an edge the correction overshoots, and the values leave the range 0 to 1.* Interactive: the same controls, and a checkbox that follows the result with Freeman's 5 × 5 median; the stages step through the two steps.

Hamilton–Adams Hamilton & Adams 1997, met in Interpolate along the edge, not across it, combines both ideas of the basic chapter, and it does so in three passes over the mosaic, each using the channels already filled in.

1. **Green at the red and blue sites.** Measure a horizontal and a vertical activity at the site, each the sum of a green gradient across the site and the second difference of the center color along the same line. Interpolate green along the less active direction, as the average of its two green neighbors plus a quarter of the center color's second difference, and use all four neighbors when the two activities are equal. This is the pseudocode of Interpolate along the edge, not across it.

2. **Red and blue at the green sites.** Now that green exists everywhere, a missing red at a green site is the average of its two red neighbors, horizontal or vertical depending on the row, corrected by half of green's second difference along that line. It is color-difference interpolation in disguise: the result equals the site's green plus the average of the two neighbors' $R - G$, up to the correction.
3. **Red at the blue sites, and blue at the red sites.** The four nearest samples of the missing color sit on the diagonals, so the method chooses between the two diagonals with the same kind of test, a gradient of the missing color along the diagonal plus green's second difference along it, and interpolates along the calmer one with the same half-weight correction.

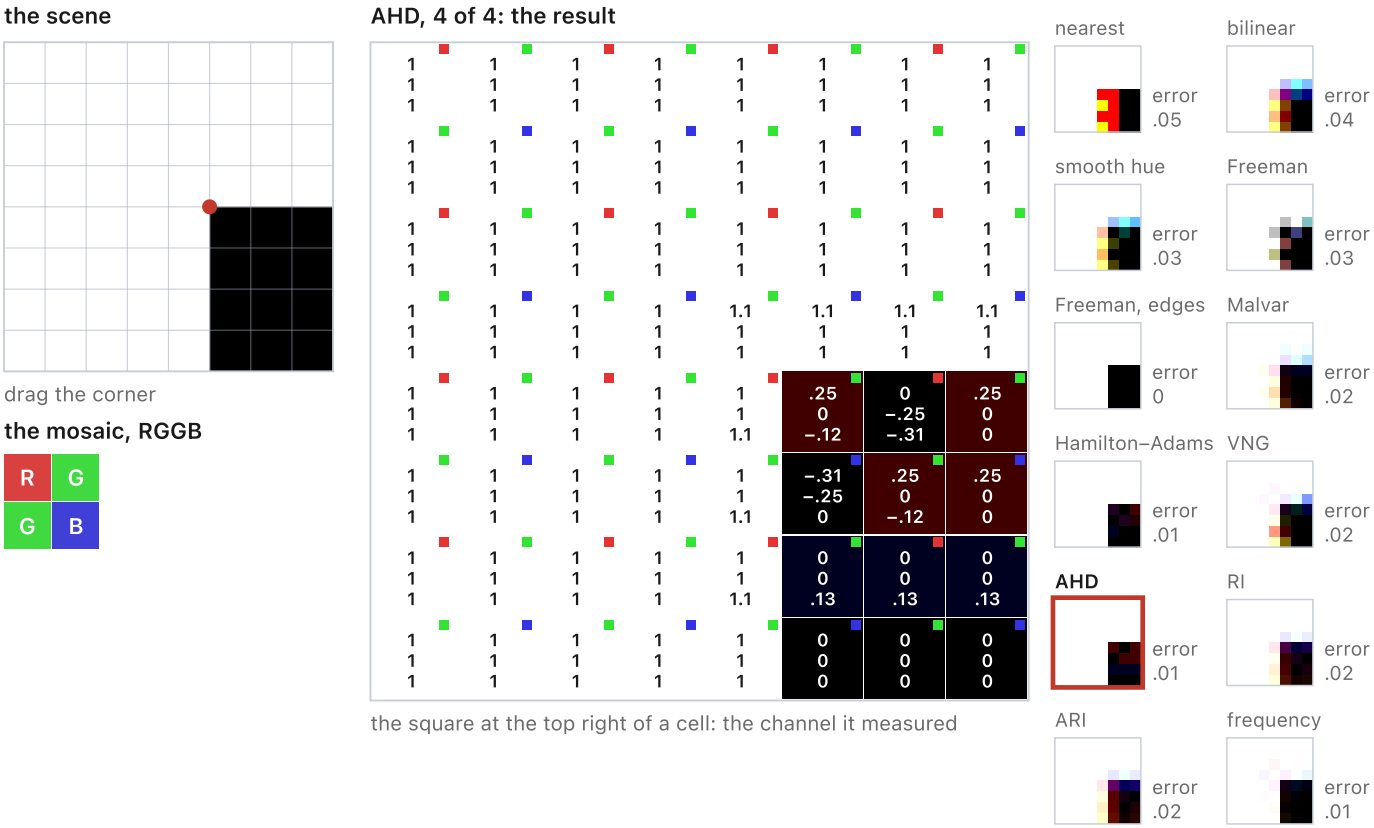
Algorithm 2

```
# Hamilton–Adams, steps 2 and 3 (step 1, green, is in the edge-directed section)
# step 2: red at a green site in a red row (blue rows: use the vertical pair)
R[x,y] = (R[x−1,y] + R[x+1,y])/2 + (2∗G[x,y] − G[x−1,y] − G[x+1,y])/2

# step 3: red at a blue site, choosing a diagonal
dN = abs(R[x−1,y−1] − R[x+1,y+1]) + abs(2∗G[x,y] − G[x−1,y−1] − G[x+1,y+1])   # NW–SE
dP = abs(R[x+1,y−1] − R[x−1,y+1]) + abs(2∗G[x,y] − G[x+1,y−1] − G[x−1,y+1])   # NE–SW
if   dN < dP: R[x,y] = (R[x−1,y−1] + R[x+1,y+1])/2 + (2∗G[x,y] − G[x−1,y−1] − G[x+1,y-
elif dP < dN: R[x,y] = (R[x+1,y−1] + R[x−1,y+1])/2 + (2∗G[x,y] − G[x+1,y−1] − G[x−1,y-
else:         R[x,y] = (R[x−1,y−1] + R[x+1,y+1] + R[x+1,y−1] + R[x−1,y+1])/4 \
                     + (4∗G[x,y] − G[x−1,y−1] − G[x+1,y+1] − G[x+1,y−1] − G[x−1,y+1
# blue at a red site: the same, with R and B exchanged
```

The corrections are what separate it from bilinear. Each one assumes that the missing channel curves the same way as the channel that was measured or already reconstructed at the site, so a sharp edge in green is copied into red and blue at the same place instead of being smeared into a fringe. The direction tests are what separate it from Malvar–He–Cutler, which applies the same kind of correction everywhere, in every direction. Hamilton–Adams costs a small, fixed amount of work per pixel, and its green step is the starting point of many later methods, including the residual interpolation of the next chapter; its failures are those of any two-way choice: diagonal texture, and fine patterns where both directions look equally active.

Figure 15.2.2 runs the three passes on the grid; the grating near 45 degrees shows the failure.



◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | 11 Malvar: correction | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | 16 VNG: directions | 17 VNG: average | 18 AHD: horizontal | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | 22 RI: G by residuals | 23 RI: tentative $\tilde{R} = aG + b$ | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | 27 frequency: spectrum | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶ | ↓ settings from the previous stage

Figure 15.2.2, stage 13 (reproduced). *Hamilton–Adams in its three passes: green along the calmer axis with the curvature correction, the bar marking the axis averaged; red and blue at the green sites from the pair on that line, corrected by half of green's second difference; and red at the blue sites and blue at the red ones along the calmer diagonal, the bar marking it.* Interactive: the same controls, and a checkbox that follows the result with Freeman's 5 × 5 median; the stages step through the three passes.

Variable number of gradients Chang, Cheung & Pan 1999 (VNG), VNG, by Chang, Cheung and Pan, replaces the two-way choice with a vote among eight directions, and lets the image decide how many of them to trust. At each pixel it works in the 5 × 5 neighborhood around the site, in four steps.

1. **Eight gradients.** For each compass direction, north, northeast, east and so on, sum the absolute differences between pairs of same-color samples that lie along that direction on that side of the pixel. Pairing same-color

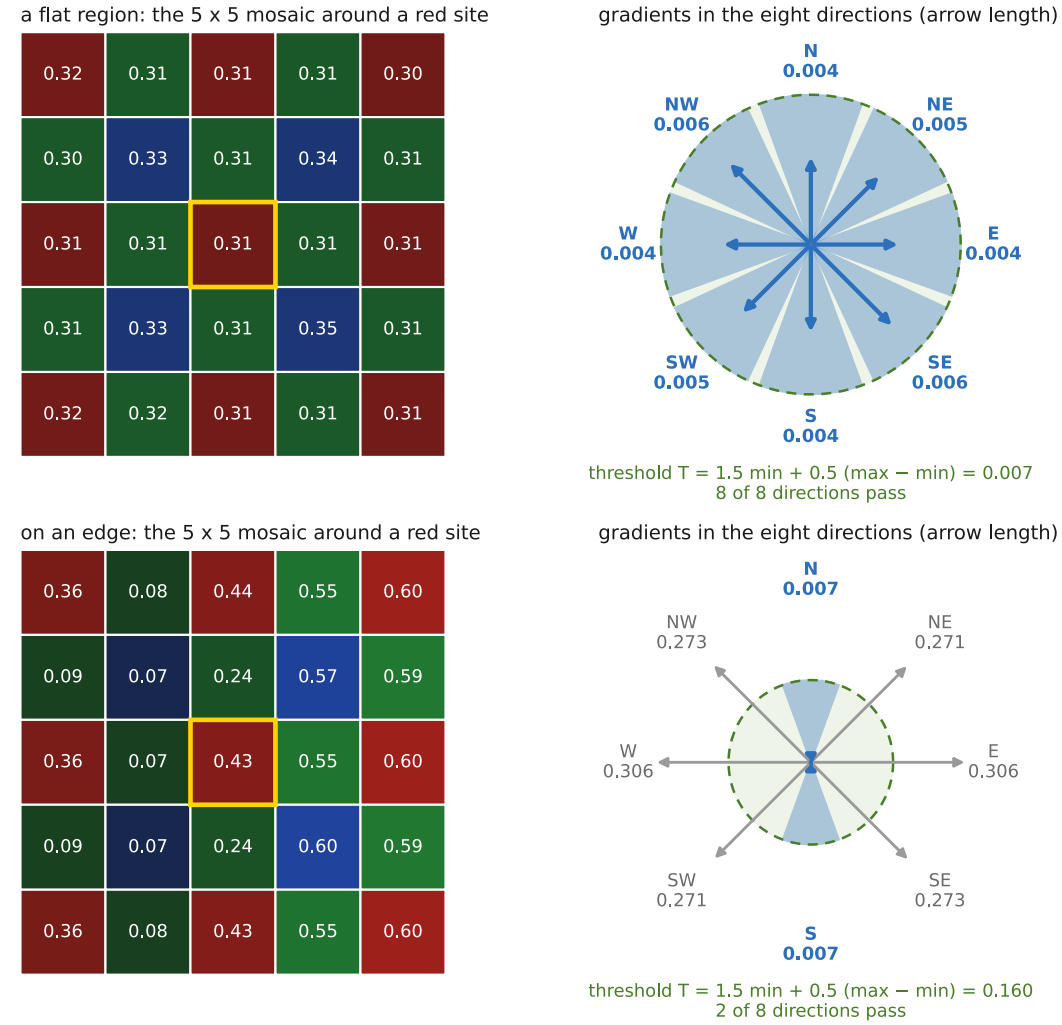
samples keeps the gradient free of the differences between channels. A direction that points into a smooth region gets a small gradient; one that crosses an edge gets a large one.

2. **A threshold.** From the smallest gradient $g_{\min}$ and the largest $g_{\max}$, set $T = 1.5\, g_{\min} + 0.5\,(g_{\max} - g_{\min})$. The threshold is relative: it asks which directions are nearly as smooth as the smoothest one here, not which are below a fixed number, so the same rule works in shadows and highlights.
3. **A vote.** Keep every direction whose gradient is below T. For each kept direction, average the samples of each color that lie on that side of the pixel, and sum these averages over the kept directions, one sum per color.
4. **A color difference.** Estimate each missing color from the center sample and the difference of the sums: at a red site, for instance, $\hat{G} = R + (S_G - S_R)/n$, where S_G and S_R are the sums for green and red over the n kept directions. This is the color-difference idea of Interpolate differences, not channels, averaged over the directions the vote allowed.

Algorithm 3

```
# VNG at one pixel, in outline (per-site sample weights omitted)
g = {d: gradient(d) for d in [N, NE, E, SE, S, SW, W, NW]}       # same-color pairs a
T = 1.5*min(g.values()) + 0.5*(max(g.values()) - min(g.values()))
keep = [d for d in g if g[d] <= T]
S = {c: sum(average_on_side(c, d) for d in keep) for c in [R, G, B]}  # one sum per c
for c in missing_colors:
    value[c] = center_value + (S[c] - S[center_color]) / len(keep)
```

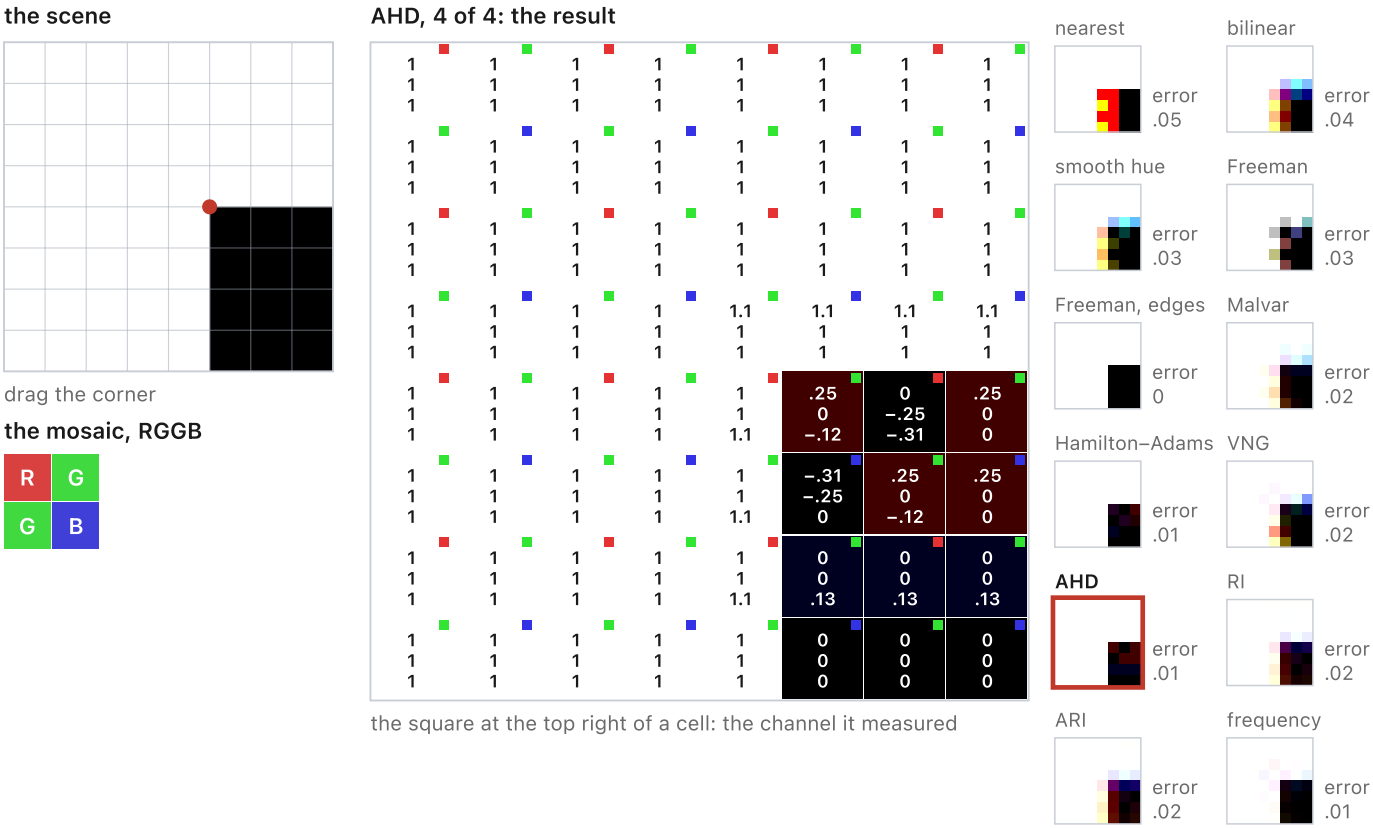
The number of directions that pass is what gives the method its name and its behavior. In a flat region all eight pass and it averages widely, which suppresses noise; along a straight edge only the two directions parallel to it pass, and it averages along the edge like Hamilton–Adams; at a corner or a line end it can keep three or four, which a two-way choice cannot express (Figure 15.2.4). The cost is about eight gradients per pixel over a 5×5 window, several times Hamilton–Adams, and the relative threshold can pass too many directions in fine, low-contrast texture, where everything looks equally smooth. VNG was long one of the modes of the dcraw raw decoder.



Samples: problem set 3's raw mosaic signs-small.png (BGGR). Dashed circle: the threshold; shaded sectors: the directions below it, which VNG averages over.

Figure 15.2.4 (reproduced). *Variable number of gradients at one red site of a real mosaic, the NO PARKING sign of problem set 3. The eight compass gradients are drawn as arrows scaled by their size, with the threshold* $T = 1.5\ \min + 0.5\,(\max - \min)$ *marked. In a flat patch all eight pass and the method averages in every direction; on a letter stroke only north and south pass, the directions along the stroke. The gradients follow the published idea, not its exact per-site weights.*

Figure 15.2.2 runs VNG on every pixel of the grid.



◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | 11 Malvar: correction | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | **16 VNG: directions** | 17 VNG: average | 18 AHD: horizontal | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | 22 RI: G by residuals | 23 RI: tentative $\tilde{R} = aG + b$ | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | 27 frequency: spectrum | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶ | ↓ settings from the previous stage

Figure 15.2.2, stage 16 (reproduced). *Variable number of gradients: at each pixel, how many of the eight directions fall below the threshold and which, drawn as spokes; then the missing colors as the center sample plus the color differences averaged over them. The gradients follow the published idea, not its exact per-site weights.* Interactive: the same controls, and a checkbox that follows the result with Freeman's 5 × 5 median; the stages step through the two steps.

Two further refinements are common. Iteration: once green and the color differences are in, re-estimate the differences against the interpolated green and repeat. Alternating projections Gunturk, Altunbasak & Mersereau 2002 (alternating projections) formalize this as projecting in turn onto two sets, the images that agree with the measured samples and the images whose red and blue have the same fine detail as green. And directional estimation with explicit noise models: Zhang and Wu's directional linear minimum mean-square-error method

Zhang & Wu 2005 (DLMMSE) fuses the horizontal and vertical estimates with weights set by how noisy each one is, rather than choosing one.

Across the eight panes, three things change from method to method. The assumption about images: smooth channels, smooth hue, correlated channels, or edges along one of a few directions. The decision made at each pixel: none for bilinear and Malvar–He–Cutler, a direction for Hamilton–Adams, VNG and AHD (#Adaptive homogeneity-directed demosaicking (AHD), next). And the cost, from nothing for nearest neighbor to two full reconstructions and a color-space conversion for AHD. The residual-interpolation methods (#Residual interpolation and ARI) take the next step by replacing the color difference with the residual of a guided fit.

For reproducible comparisons, state the implementation and demosaicking mode. LibRaw and dcraw expose several algorithms, but their defaults are not interchangeable with the Malvar–He–Cutler method. Compare against a strong classical baseline such as adaptive residual interpolation Monno et al. 2015 (adaptive residual interpolation) as well as a simple linear method.

The sections that follow take up three families that go further: AHD, which builds whole reconstructions in two directions and keeps the more homogeneous one; residual interpolation and ARI, which interpolate what a guided prediction leaves over; and frequency-domain demultiplexing, which treats the mosaic as a modulated signal. The chapter ends with the situations that defeat all of them.

Adaptive homogeneity-directed demosaicking (AHD)

OPTIONAL
in the book ↗

If you want to implement one classical demosaicker that is simple and works well, this is probably your best bet, followed by Freeman's median of the color differences (earlier in this chapter) to remove the false color that remains. Every method so far decides the interpolation direction before interpolating, from gradients in the mosaic. Adaptive homogeneity-directed demosaicking Hirakawa & Parks 2005 (AHD), by Hirakawa and Parks, decides afterward. It reconstructs the whole image twice, once interpolating only horizontally and once only vertically, and at each pixel keeps whichever reconstruction looks more plausible there. The criterion is the key idea: interpolating in the wrong direction averages across an edge and creates colors that exist on neither side of it, so the wrong candidate is less homogeneous, locally, than the right one.

1. **Two candidates.** Build a full image with every interpolation along rows: green with a Hamilton–Adams-style correction, then red and blue by interpolating color differences. Build a second with every interpolation along columns.
2. **A perceptual space.** Convert both candidates to CIELAB, so that lightness and color can be compared separately and a distance means roughly what a viewer would notice (Chapter 4.5, Advanced color: opponents, perceptually-uniform, and preferred).
3. **Homogeneity.** For each pixel of each candidate, count the neighbors in a small window that are close to it in both lightness and color, within thresholds ε_L and ε_C. The thresholds are set at every pixel from the differences to its nearest neighbors along each candidate's own direction, so the test adapts to local contrast rather than using one number for shadows and highlights.
4. **The choice.** Sum each candidate's counts over a 3 × 3 window and keep, at each pixel, the candidate with the larger sum; average the two on a tie.

5. **Clean-up.** Finish with a few passes of a median over the color differences $R - G$ and $B - G$, Freeman's clean-up.

Algorithm 4

```
# AHD, in outline
Ih, Iv = demosaic_along_rows(mosaic), demosaic_along_columns(mosaic)
Lh, Lv = to_cielab(Ih), to_cielab(Iv)
eps_L, eps_C = adaptive_thresholds(Lh, Lv)            # per pixel, from nearest-neigh
Hh = count_close_neighbors(Lh, eps_L, eps_C)          # lightness and color both with
Hv = count_close_neighbors(Lv, eps_L, eps_C)
Hh, Hv = box_sum_3x3(Hh), box_sum_3x3(Hv)
out = where(Hh > Hv, Ih, where(Hv > Hh, Iv, (Ih + Iv) / 2))
out = median_color_differences(out, passes=3)
```

Figure 15.2.5 shows the homogeneity test at one pixel.

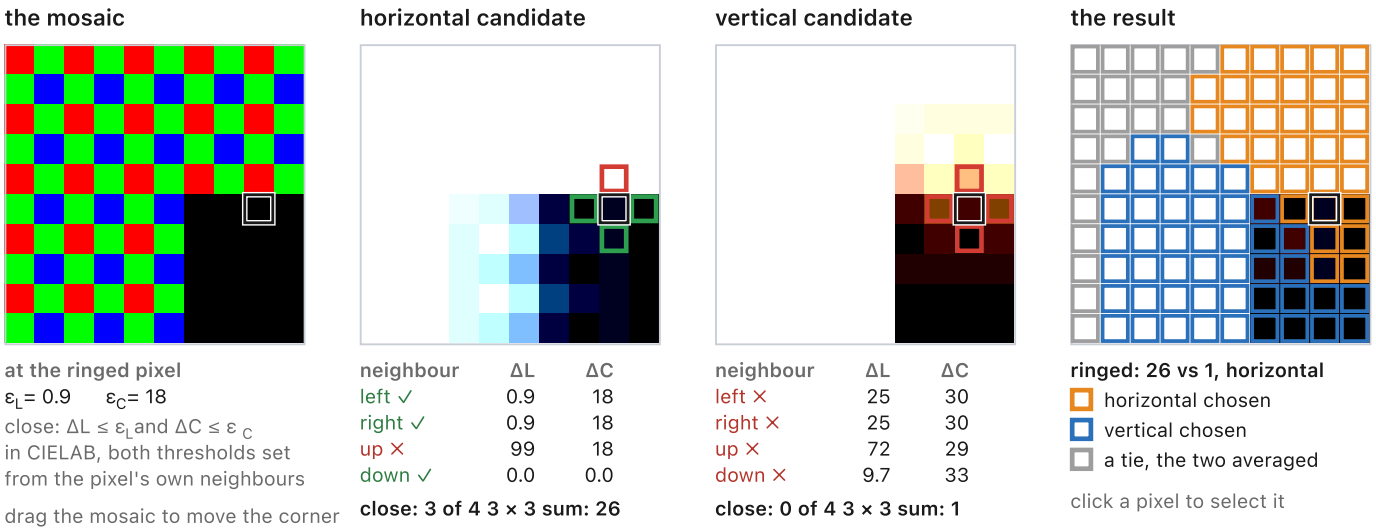
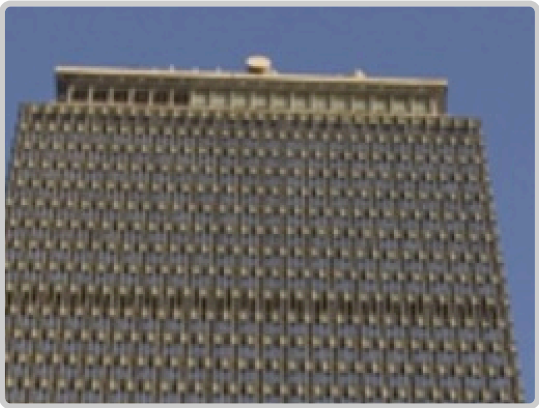


Figure 15.2.5 (reproduced). *AHD's homogeneity criterion at one pixel. Left, the mosaic. Then the horizontal and the vertical candidate, the ringed pixel's four neighbors outlined green when they are within both thresholds, ε_L in lightness and ε_C in color (CIELAB), and red when they are not; beneath each, the differences, the count and its sum over 3 × 3. The thresholds are the same for both candidates, set from the pixel's own neighbors along each candidate's direction. Right, the result: each pixel framed orange where the horizontal candidate won, blue where the vertical one did, gray on a tie. Next to an edge, the candidate that interpolated across it has made colors that neither side has, and its neighbors stop matching. The inputs are the corner, a sine grating, a Siemens star and a zone plate (both band-limited), the maze's colored stripes, and crops of four photographs, on a grid of 10 × 10 or 16 × 16.* Interactive: choose the input and the grid, set the pattern's angle, period, spokes or frequency, drag on the mosaic to move the pattern or pan the photograph, and click a pixel to test it.

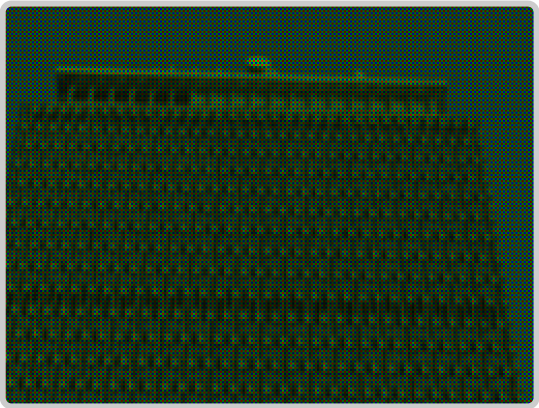
The figure below runs it on a real facade, where the cornice's horizontal lines choose the horizontal candidate and the window columns the vertical one (Figure 15.2.6), and Figure 15.2.2 makes the same choice cell by cell on the chapter's small inputs.

Click any picture to enlarge it. The choice picks the more accurate candidate at 68% of the pixels where the two differ.

The input

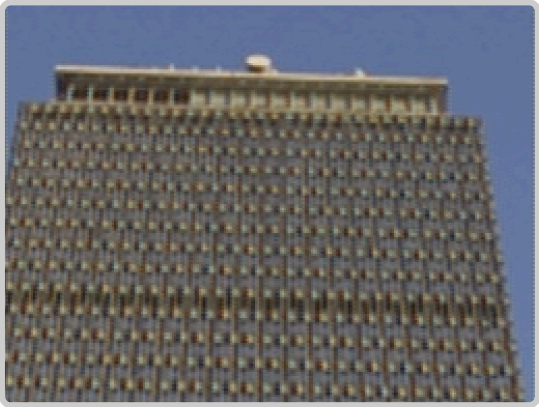


the photograph (a window grid)



its RGGB mosaic

Two full reconstructions

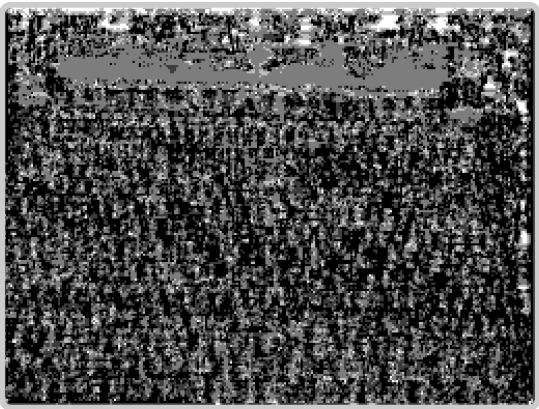


interpolating only horizontally:
error 2.4

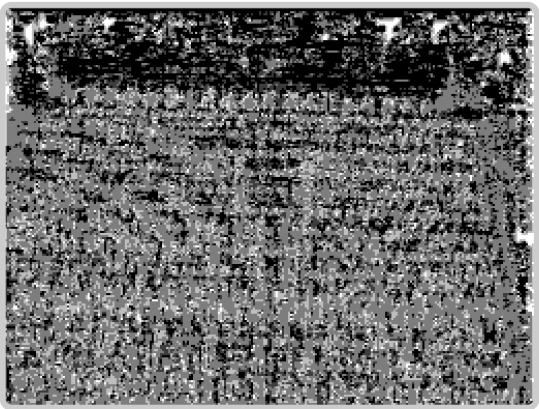


interpolating only vertically:
error 1.9

Homogeneity in CIELAB: neighbours within tolerance, white = all four

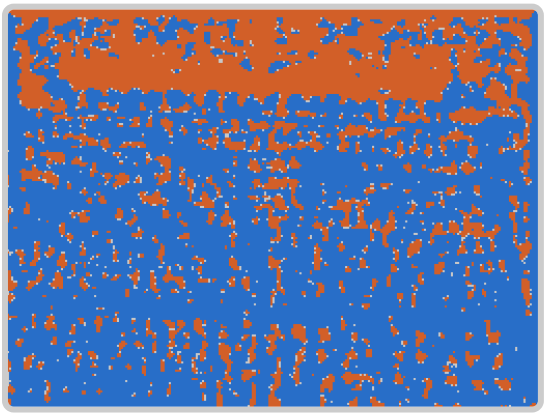


homogeneity of the horizontal candidate



homogeneity of the vertical candidate

The decision and the result



per pixel: orange horizontal, blue vertical, grey a tie

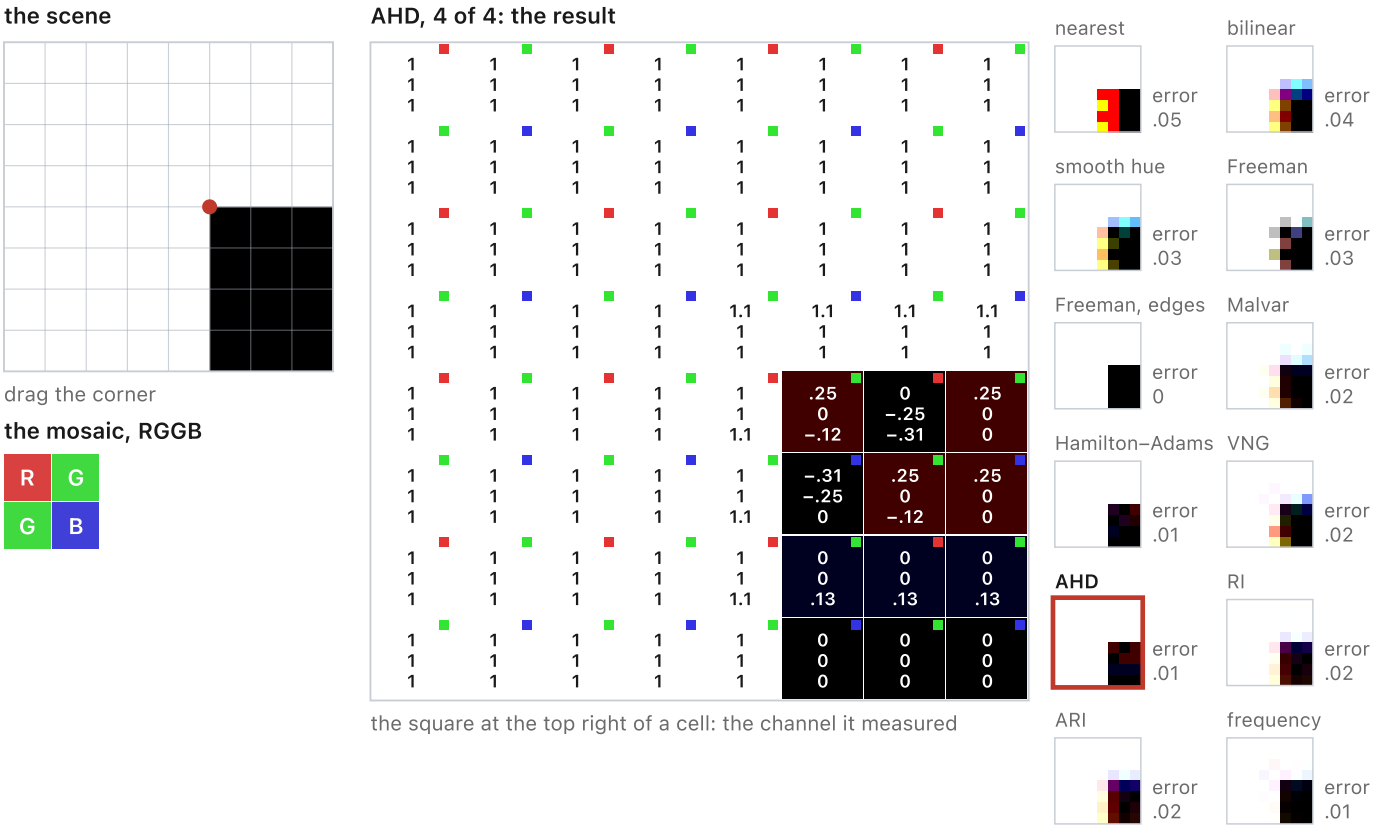


AHD: error 1.7

Photograph © Frédo Durand; the mosaic is sampled from it, as in the demosaicking lab.

Figure 15.2.6 (reproduced). *AHD on a crop of the Prudential facade, from a mosaic sampled from the photograph as in the lab: the horizontal candidate, the vertical candidate, their homogeneity maps, the per-pixel choice and the result. The cornice's horizontal lines choose the horizontal candidate and the window columns the vertical one. Where the candidates differ, the choice picks the more accurate one at about two pixels in three, and the result's mean error is below either candidate's. Each picture enlarges on its own.*

Figure 15.2.2 steps through the same algorithm on the small inputs.



◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | 11 Malvar: correction | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | 16 VNG: directions | 17 VNG: average | **18 AHD: horizontal** | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | 22 RI: G by residuals | 23 RI: tentative $\tilde{R} = aG + b$ | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | 27 frequency: spectrum | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶ | ↓ settings from the previous stage

Figure 15.2.2, stage 18 (reproduced). *AHD: the horizontal candidate, the vertical candidate, the choice at each pixel from their homogeneity counts summed over 3 × 3, and the result. The published method finishes with a median of the color differences; the checkbox adds one, Freeman's 5 × 5.* Interactive: the same controls, and a checkbox that follows the result with Freeman's 5 × 5 median; the stages step through the four steps.

AHD costs two full reconstructions, two color-space conversions and the counting, several times Hamilton–Adams. It was the default of the dcraw raw decoder and of many converters built on it for years, and it is still a strong classical baseline. Its limits are the other methods' limits. It still chooses between two directions, so diagonal texture gives both candidates errors and the choice cannot fix them. In fine texture near the sampling limit both candidates are inhomogeneous and the choice becomes erratic, which is where mazes come from. And noise lowers the homogeneity of both candidates everywhere, so the choice degrades as the noise rises, one reason to denoise first or jointly (Chapter 15.8, Joint demosaicking and denoising).

Residual interpolation and ARI

OPTIONAL
in the book ↗

The best-performing classical demosaickers, and the ones to compare against today, come from the residual-interpolation family, and adaptive residual interpolation (ARI) is its strongest member. It appeared a decade after AHD, outperforms it clearly in published comparisons on the standard test sets, and needs no training. It is also built directly on the guided filter of Chapter 10.6, Guided image filtering, so it is a payoff of that chapter rather than a new tool. It takes more work to implement than AHD, which remains the simpler choice when that matters.

The idea changes what gets interpolated. Every classical method interpolates something, and each step of the classical line chose a flatter quantity. Bilinear interpolates the channel itself and fails at edges. Hamilton–Adams and its successors interpolate the color difference $G - R$, which is smooth wherever the two channels differ by an offset, and fails where that assumption does: at a saturated color boundary, at a specular highlight, wherever the ratio between the channels changes rather than their offset. Residual interpolation Kiku et al. 2013 (residual interpolation), by Kiku, Monno, Tanaka and Okutomi, goes one step further. Rather than hope that some quantity is flat, it constructs one: it fits a local prediction of the sparse channel from the dense one and interpolates only what the prediction gets wrong.

$$\underbrace{R}_{\text{full of structure}} \quad \rightarrow \quad \underbrace{G - R}_{\text{smoother, if the offset model holds}} \quad \rightarrow \quad \underbrace{R - \tilde{R}}_{\text{potentially smoother}} \tag{6}$$

Take red, sampled on a quarter of the pixels, and green, already reconstructed everywhere because it is sampled twice as densely. In a small window, model red as a linear function of green,

$$\tilde{R}(\mathbf{p}) = a\, G(\mathbf{p}) + b, \tag{7}$$

with a and b estimated from the red samples that exist in that window. This is a guided filter with green as the guide He, Sun & Tang 2013 (guided filter), restricted to the measured sites. In a window Ω, with $\mathcal{R}$ the set of sites where red was measured, solve

$$\min_{a,b} \sum_{q \in \Omega \cap \mathcal{R}} (R_q - a G_q - b)^2 + \epsilon a^2, \tag{8}$$

where $\epsilon > 0$ stabilizes the slope. Unmeasured sites must not enter the fit as zeros; a window with no red sample has to grow or fall back to a simpler estimate; and since every pixel lies in many windows, the overlapping predictions are averaged, which is the guided filter's own rule. The tentative estimate $\tilde{R}$ is a full-resolution red built entirely out of green's structure, and it follows a change of contrast between the channels that a difference cannot. The residual is the part of red that green cannot explain,

$$\Delta R = R - \tilde{R}, \qquad \text{known only where red was measured,} \tag{9}$$

and the reconstruction interpolates it, by any simple method, and adds it back:

$$\hat{R} = \tilde{R} + \mathrm{interp}(\Delta R). \tag{10}$$

The prediction carries the shared detail, and the residual corrects its errors where measurements constrain them. When the local model explains the structure the channels share, the residual is smaller and flatter than the color difference, and the same cheap interpolator does much less damage to it. That flatness is an assumption to test, not a property the fit enforces: a small residual at the measured sites does not prove that the residual between them is small or smooth, and a misleading guide, a sharp color boundary or a poorly conditioned fit can transfer the wrong structure into the result.

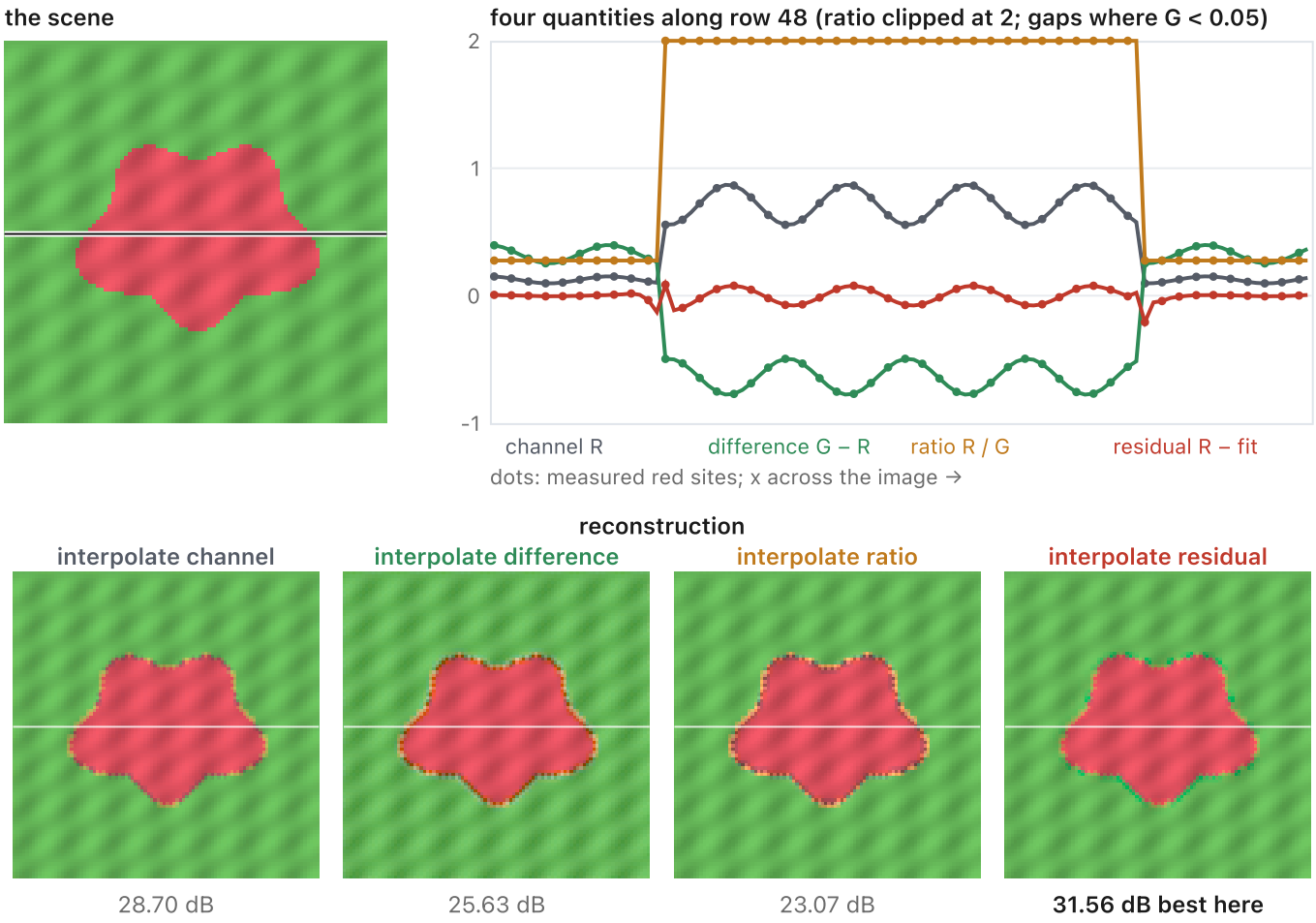


Figure 15.2.7 (reproduced). *Interactive. One scanline across a hard color boundary, with four candidate quantities plotted on the same axes: the red channel, the color difference $G - R$, the ratio R/G, and the residual $R - \tilde{R}$ against a fitted guide. The channel swings, the difference is flat until the boundary and then jumps, the ratio is worse where green is small, and the residual stays flat across the boundary. Under the plot, the reconstruction each choice produces on the full image, with its error map, so the flatness of a curve and the quality of a result are on screen together. A scene selector moves the scanline onto a saturated flower, a specular highlight and a neutral edge, which are the three places the older assumptions fail.* Interactive: choose the scene and the scanline, toggle the four quantities, set the guide window size, and compare reconstructions and error maps.

The full algorithm applies the same idea twice, first to green and then to red and blue.

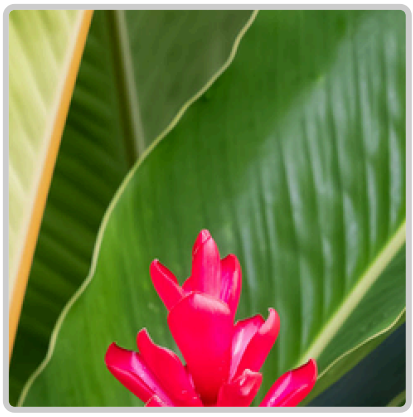
1. **Green first.** Estimate green along rows and along columns separately, each time guiding by the measured red or blue along that line and interpolating the residual along it, then combine the two directional estimates with weights that favor the direction whose color differences vary less locally.
2. **Tentative red.** With the full green as the guide, fit a local linear model $\tilde{R} = aG + b$ in each window, using only the sites where red was measured, and evaluate it everywhere.
3. **Residual.** At the measured red sites, form $R - \tilde{R}$.

4. **Interpolate and add back.** Interpolate the residual bilinearly to every site and add it to $\tilde{R}$. Blue is the same with blue's samples.

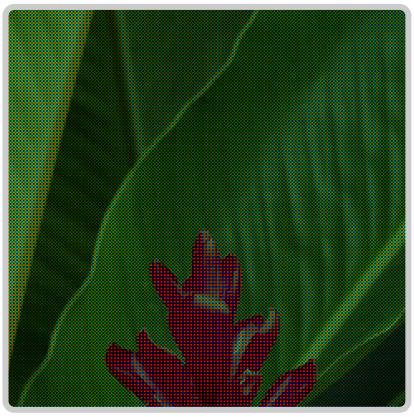
On a red flower against green leaves, the residual is close to zero where the color difference is large, and residual interpolation scores 3.3 dB above AHD (Figure 15.2.8).

Click any picture to enlarge it.

The input



the photograph: red bracts against green leaves

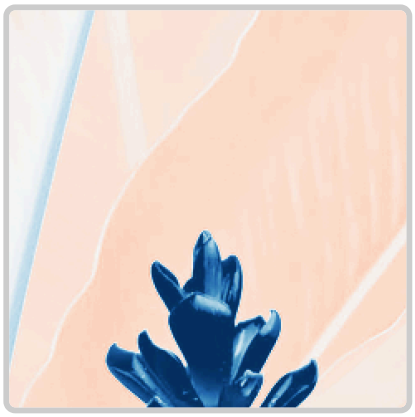


its RGGB mosaic

What gets interpolated, for red (the same colour scale for both)



tentative red R~ = aG + b, fitted on the red samples

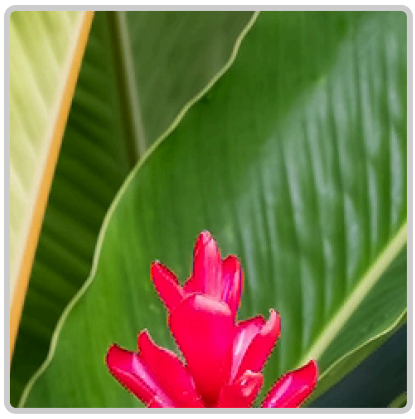


the colour difference G − R: mean |Laplacian| 9.2



the residual R − R~: mean |Laplacian| 6.0

Results, scored against the photograph



AHD: 32.8 dB

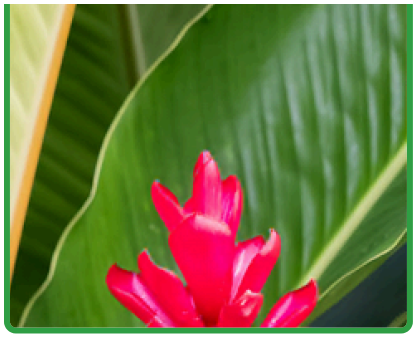


residual interpolation: 36.2 dB



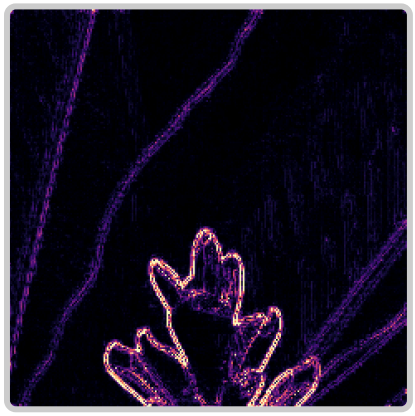
MLRI: 36.5 dB



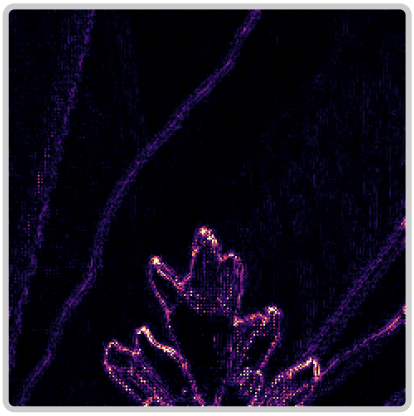


ARI: 37.7 dB

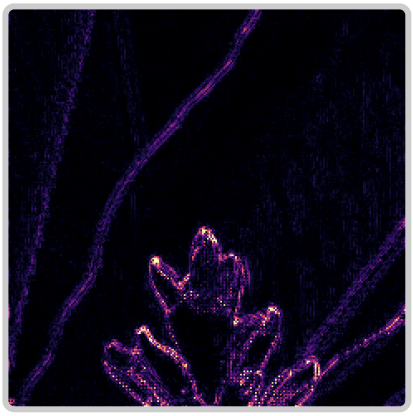
Error, on one scale (black 0, yellow 25 of 255 or more)



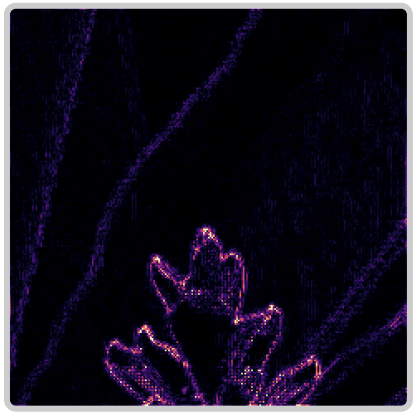
AHD



residual interpolation



MLRI



ARI

Residual interpolation and MLRI: our implementation (Kiku et al. 2013, 2014). ARI: the authors' code (Monno et al. 2015), ported. Photograph © Frédo Durand; the mosaic is sampled from it.

Figure 15.2.8 (reproduced). *Residual interpolation on saturated color edges: red bracts against green leaves, a crop of a photograph whose RGGB mosaic is sampled from it, so every result is scored against the original. For red, the tentative estimate $\tilde{R} = aG + b$ comes from a guided filter with the reconstructed green as guide, fitted on the red samples only. Beside it, on the same color scale, the color difference $G - R$ that earlier methods interpolate and the residual $R - \tilde{R}$ that residual interpolation interpolates: the difference is large wherever the flower is, while the residual is close to zero except at a few edge pixels, and its mean absolute Laplacian, the roughness a bilinear interpolator suffers from, is 6.0 against 9.2. The results: AHD 32.8 dB, residual interpolation 36.2 dB, MLRI 36.5 dB*

and ARI 37.7 dB, with error maps on one scale; the errors that remain sit on the petal edges. Residual interpolation and MLRI are our implementation from the papers; ARI is the authors' code, ported.

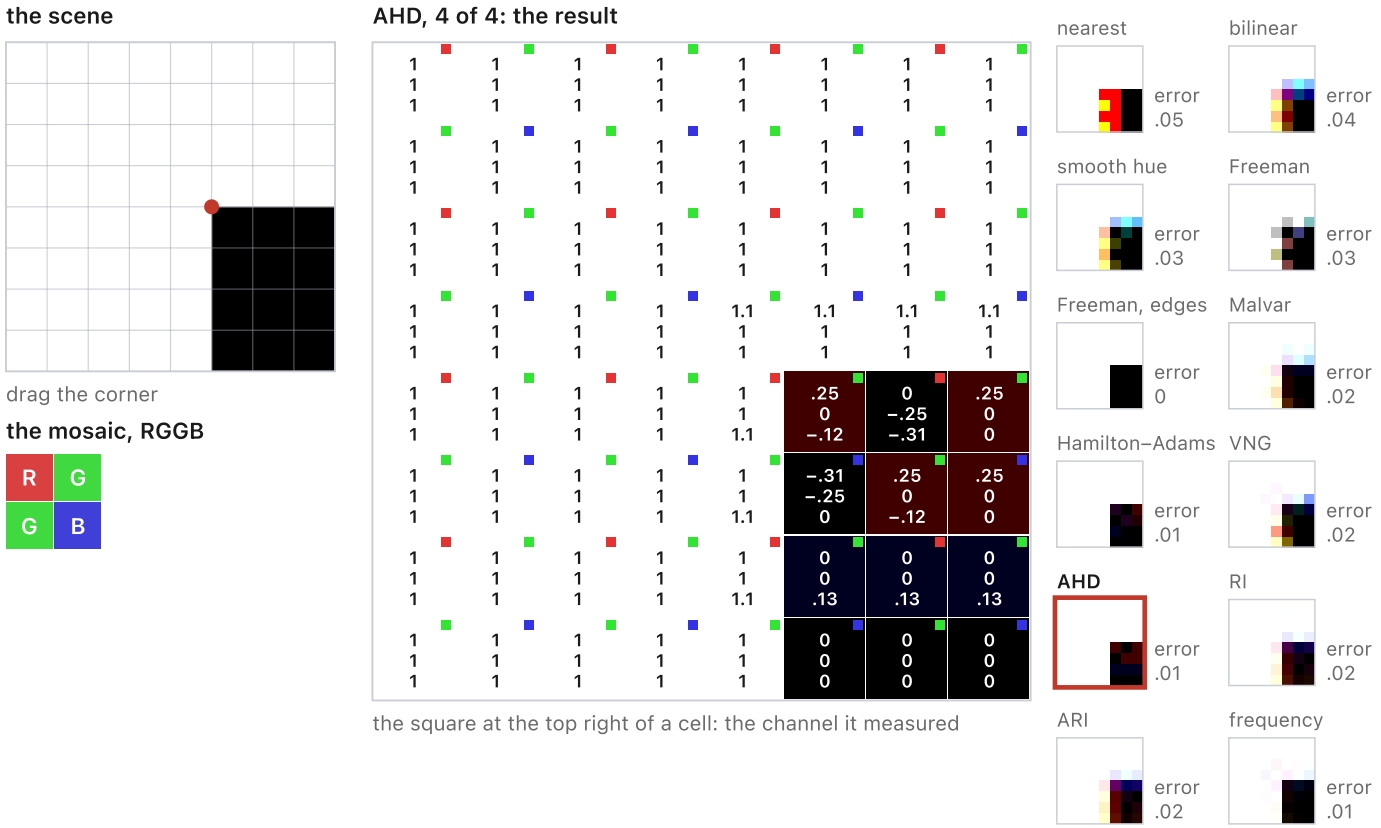
Minimized-Laplacian residual interpolation, MLRI Kiku et al. 2014 (minimized-Laplacian residual interpolation), changes only how a and b are fitted: instead of least squares on the values, it minimizes the Laplacian energy of the residual, which is what the final bilinear interpolation needs to be small. It changes how the regression coefficients are estimated, not which channel serves as the guide. ARI Monno et al. 2015 (adaptive residual interpolation), by Monno, Kiku, Tanaka and Okutomi, iterates both, using each output as the next guide, and at each pixel selects the variant and the number of iterations whose residual is smoothest, judged by the local fit. That is the direction-selection idea of Hamilton–Adams operating one level up: not which direction to interpolate in, but which method to trust here. A later version extends it to multispectral filter arrays Monno et al. 2017 (ARI, multispectral). The authors' reference code has been used widely enough in later work that comparisons against it can be trusted.

Algorithm 5

```
# Residual interpolation for red, green already reconstructed (blue is the same)
def residual_interpolation(G, R_mosaic, red_mask, radius, eps):
    a, b = masked_guided_fit(guide=G, target=R_mosaic, mask=red_mask, radius=radius, e
    R_tent = box_mean(a, radius) * G + box_mean(b, radius)     # guided filter output
    residual = where(red_mask, R_mosaic - R_tent, 0)
    return R_tent + bilinear_from_samples(residual, red_mask)

# MLRI: the same, with a and b fitted to minimize the Laplacian of the residual
# ARI, in outline: at each pixel keep the (method, iteration) whose residual is smooth
estimates = []
guide = G
for k in range(K):
    for method in [RI, MLRI]:
        R_k = method(guide, R_mosaic, red_mask)
        estimates.append((R_k, laplacian_energy(R_mosaic - R_k, red_mask)))
    guide = R_k
R = per_pixel_argmin(estimates)          # smoothest residual wins, in a local window
```

Figure 15.2.2 steps through residual interpolation cell by cell on the chapter's small inputs, from the directional green to ARI's per-pixel choice, and the demosaicking lab at the start of this chapter (Figure 15.2.1) runs the same code on photographs.



◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | 11 Malvar: correction | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | 16 VNG: directions | 17 VNG: average | 18 AHD: horizontal | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | **22 RI: G by residuals** | 23 RI: tentative $\tilde{R} = aG + b$ | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | 27 frequency: spectrum | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶ | ↓ settings from the previous stage

Figure 15.2.2, stage 22 (reproduced). *Residual interpolation on the small inputs. Green is estimated along rows and along columns from residuals, and the two are weighted by which direction's color difference is smoother; a bar marks the heavier direction. The following stages show the tentative red $\tilde{R} = aG + b$ with the fitted a and b per cell, the residual beside the color difference $G - R$ at the red and blue sites, the result, and a map of ARI's choice at each pixel between RI and MLRI and between the first and second pass. On a colored edge the residual is smaller than the color difference. ARI here is the published idea in outline, the second pass guided by the first's tentative estimate and the smoothest residual chosen, and on inputs this small it does not always beat plain residual interpolation.* Interactive: the same controls as the other stages; step through the five residual-interpolation stages, and the checkbox adds Freeman's median to the result.

The limits are the guided filter's. A local linear model cannot follow channels that are related nonlinearly, at clipping or under strong chromatic aberration, and fitting a and b from a few noisy samples is unstable, so noise

should be removed first, accounted for within the fit, or handled by joint reconstruction (Chapter 15.8, Joint demosaicking and denoising).

This family, not bilinear interpolation, is the baseline a learned demosaicker should be measured against. It embodies three separate decisions: predict one channel from a guide, interpolate what the prediction missed, and choose locally between useful variants. A learned model can change any of the three, and a comparison with bilinear alone leaves those established improvements unaccounted for, so it cannot tell which part of a network's gain required learning. Compare actual accuracy, noise assumptions and runtime; no ranking of classical methods, and no fixed advantage in decibels, carries over across datasets and settings. A network can represent similar cross-channel predictions, but it chooses what to interpolate implicitly, which is one reason it is hard to say what it learned: a benchmark gain does not show that it learned this algorithm, and attributing the gain requires controlled experiments rather than an analogy between architectures (Chapter 15.6, Learned demosaicking).

Demosaicking as demultiplexing: frequency-domain methods

OPTIONAL
in the book ↗

One family of methods treats the mosaic the way a radio engineer treats a signal. As Chapter 15.3, Mosaics and aliasing shows, a Bayer mosaic is exactly a luma component L plus two chroma components modulated onto carriers at the highest frequencies the grid supports: C_1 on the horizontal and vertical carriers, C_2 on the diagonal one,

$$y_{ij} = L_{ij} + C_{1,ij}\left[(-1)^i + (-1)^j\right] + C_{2,ij}(-1)^{i+j}. \tag{11}$$

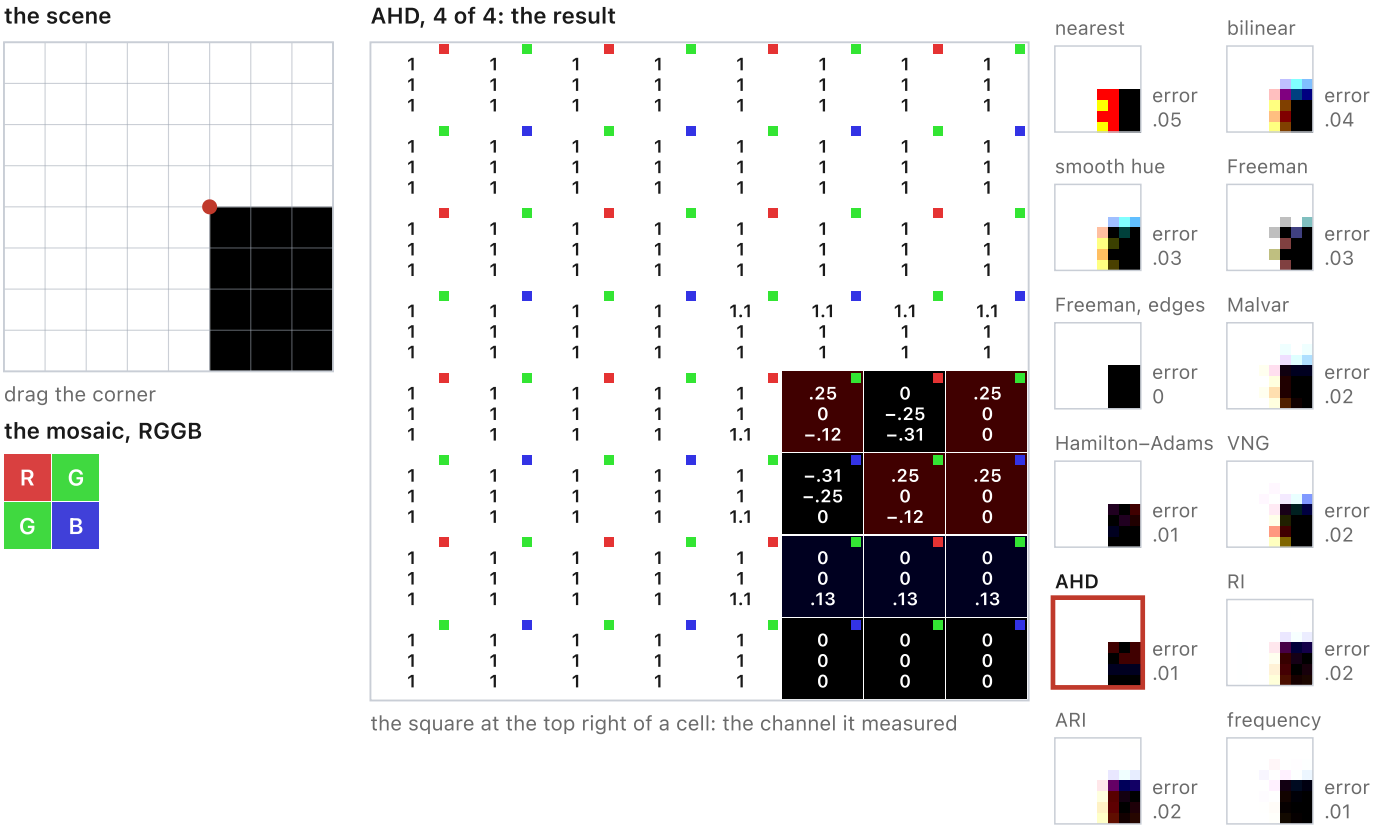
Demosaicking is then demultiplexing: isolate each modulated chroma band with a bandpass filter, shift it back to baseband by multiplying by its carrier, and take the luma as what remains. Alleysson, Süsstrunk and Hérault Alleysson, Süsstrunk & Hérault 2005 (frequency-domain demosaicing) arrived at this from the side of human vision, where luma and chroma are carried separately; Dubois Dubois 2005 (frequency-domain demosaicking) made it adaptive. Nowhere else in this chapter is the algorithm itself the frequency argument, rather than a heuristic that happens to be analyzable in frequency terms.

1. **Diagonal chroma.** Filter the mosaic with a bandpass filter centered on the diagonal carrier and multiply by $(-1)^{i+j}$ to bring C_2 back to baseband.
2. **Axial chroma, twice.** C_1 is carried twice, on the horizontal and on the vertical carrier. Extract it from each with its own bandpass filter and demodulate.
3. **Choose the cleaner copy.** Luma detail leaks into the two copies differently: a vertical texture puts its energy near the horizontal carrier and spoils that copy, while the vertical copy stays clean. Dubois weights the two estimates at each pixel by the local energy of the crosstalk in each, which is the edge-direction idea of this chapter moved into the frequency domain.
4. **Luma.** Remodulate the chroma estimates and subtract them from the mosaic; what remains is L.
5. **Back to RGB.** $G = L - C_2$, $R = L + C_2 + 2C_1$, $B = L + C_2 - 2C_1$.

Algorithm 6

```
# frequency-domain demosaicking (Dubois style), for an RGGB mosaic y
C2  = lowpass(bandpass(y, carrier=(pi, pi)) * alternate(i + j))
C1h = lowpass(bandpass(y, carrier=(pi, 0))  * alternate(i))
C1v = lowpass(bandpass(y, carrier=(0, pi))  * alternate(j))
eh, ev = local_energy(crosstalk_h), local_energy(crosstalk_v)   # luma leaking into ea
C1  = (ev * C1h + eh * C1v) / (eh + ev)                         # trust the cleaner co
L   = y - C1 * (alternate(i) + alternate(j)) - C2 * alternate(i + j)
G, R, B = L - C2, L + C2 + 2*C1, L + C2 - 2*C1
```

Figure 15.2.2 runs the same demultiplexing on the small inputs.



◀ | 1 input: mosaic | 2 nearest | 3 bilinear | 4 smooth hue: R/G, B/G | 5 smooth hue: interpolate | 6 smooth hue: times G | 7 Freeman: R−G, B−G | 8 Freeman: median | 9 Freeman: add G back | 10 Freeman on edges, 5×5: add G back | 11 Malvar: correction | 12 Malvar: add to bilinear | 13 Hamilton–Adams: G | 14 Hamilton–Adams: R, B at G | 15 Hamilton–Adams: diagonals | 16 VNG: directions | 17 VNG: average | 18 AHD: horizontal | 19 AHD: vertical | 20 AHD: choose | 21 AHD: result | 22 RI: G by residuals | 23 RI: tentative $\tilde{R}$ = aG + b | 24 RI: residual vs G − R | 25 RI: add back | 26 ARI: per-pixel choice | **27 frequency: spectrum** | 28 frequency: C_2 | 29 frequency: C_1, two copies | 30 frequency: luma | 31 frequency: R, G, B | ▶ | ↓ settings from the previous stage

Figure 15.2.2, stage 27 (reproduced). *Frequency-domain demosaicking on the small inputs, with the convention of the text: the magnitude of the mosaic's Fourier transform with the three carriers ringed, then the demodulated C_2, the two copies of C_1 with the weight given to the horizontal one, the luma left after subtracting the remodulated chroma, and the result. A vertical grating shows the point of the per-pixel choice: its luma leaks into the horizontal copy of C_1, and the weight moves to the vertical copy. The weights come from each copy's local energy of detail beyond a wider smoothing, a stand-in for Dubois's crosstalk estimate.* Interactive: the same controls; step through the five frequency-domain stages, and try the zone plate at 32 × 32 to see the carriers clearly.

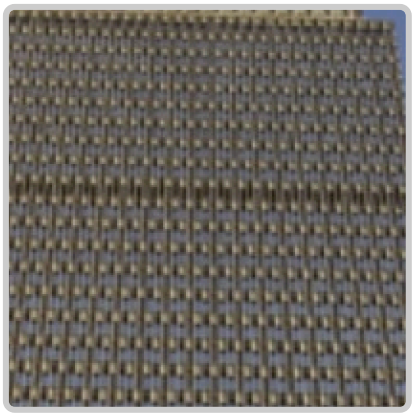
The filters are where the quality lives. Leung, Jeon and Dubois Leung, Jeon & Dubois 2011 (least-squares luma-chroma demultiplexing) design them by least squares on training images rather than by hand, the least-squares luma-chroma demultiplexing that is the version to cite for performance: a small learned element inside an entirely classical structure. The limit is the one Chapter 15.3, Mosaics and aliasing names: where luma and chroma overlap

in frequency, as on fine texture near the carriers, no linear filter can separate them, and the leftover luma shows up as false color.

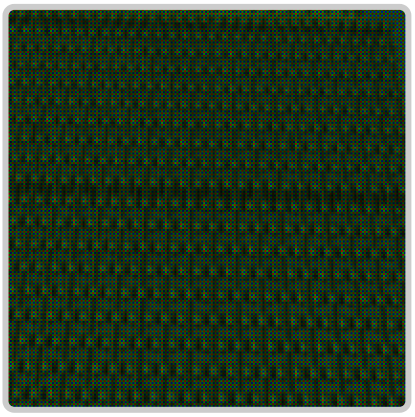
On the upper floors of a tower facade, where the window grid approaches the carriers, choosing the cleaner copy of C_1 per pixel removes much of the false color that averaging the two copies leaves, and the method scores above AHD (Figure 15.2.9). The same implementation falls below bilinear interpolation on the flower of Figure 15.2.8, whose saturated edges carry chroma at frequencies the low-pass filters remove.

Click any picture to enlarge it.

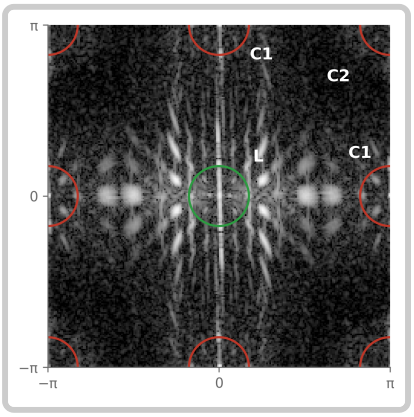
The mosaic and its spectrum



the facade: the window grid is fine near the top



its RGGB mosaic

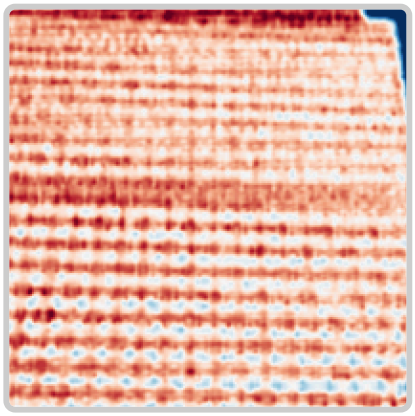


log spectrum of the mosaic: luma at the centre (green), C1 on the axes and C2 in the corners (red), each ringed at its carrier

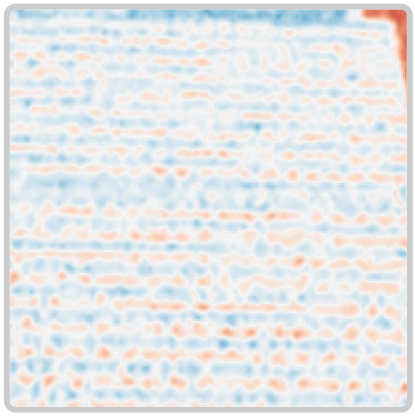
Demultiplexed (C1 and C2 on one colour scale)



luma L, what remains after the chroma is removed

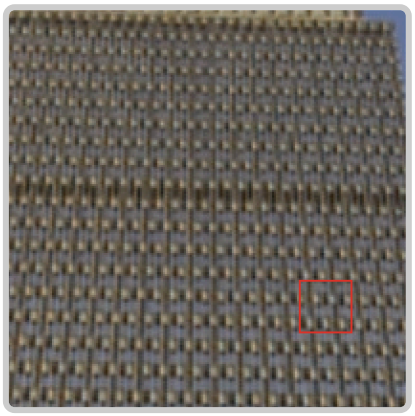


C1, demodulated from the axial carriers

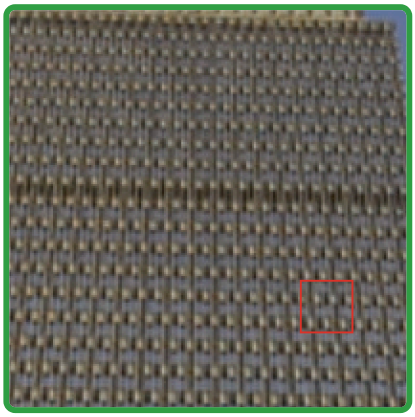


C2, demodulated from the diagonal carrier

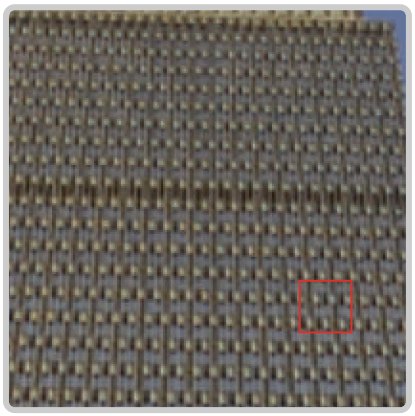
Results, scored against the photograph



the two copies of C1 averaged:



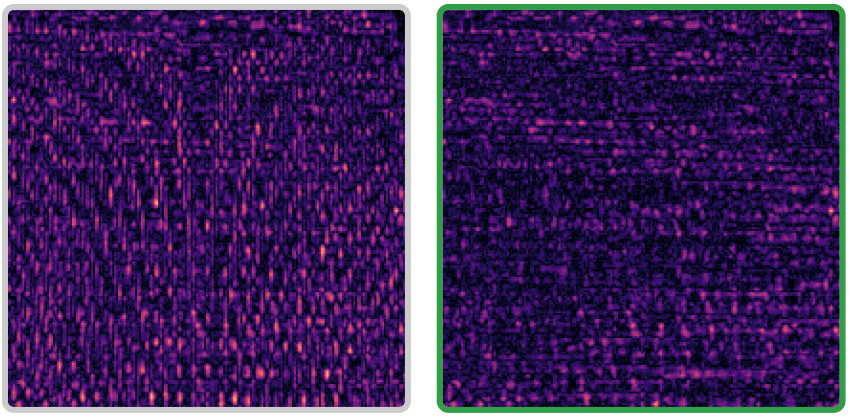
the cleaner copy chosen per



AHD, for reference: 37.6 dB

37.6 dB pixel: 39.8 dB

False colour, the error in R – G and B – G (black 0, yellow 20 of 255 or more)



copies averaged: mean 3.9 cleaner copy chosen: mean 3.1

The red box, enlarged: luma near a carrier read as colour



the photograph copies averaged: false colour cleaner copy chosen: less of it

Our implementation after Alleysson, Süsstrunk and Hérault (2005) and Dubois (2005), with our own filters. Photograph © Frédo Durand; the mosaic is sampled from it.

Figure 15.2.9 (reproduced). *Demosaicking as demultiplexing, on a facade whose window grid is fine enough to put luma detail near the carriers; the RGGB mosaic is sampled from the photograph. The mosaic's spectrum shows luma around the origin and the chroma carriers ringed: C_1 on the axes at π, C_2 in the corners. Demodulating each carrier and low-pass filtering gives C_1 and C_2; luma is what remains once they are remodulated and subtracted. Averaging the two copies of C_1 (Alleysson and colleagues' version) scores 37.6 dB, the same as AHD; choosing the copy with less luma leakage at each pixel (after Dubois) scores 39.8 dB. The false-color maps show where luma near a carrier was read as color, in vertical streaks when the copies are averaged, and the enlarged box shows it in the windows. Our implementation, with our own low-pass filters rather than the least-squares filters of Leung, Jeon and Dubois.*

Where the classical methods stop

OPTIONAL
in the book ↗

Four situations defeat this family, and they are worth listing because the learned methods of Chapter 15.6, Learned demosaicking are judged on exactly them.

Diagonal edges can challenge a horizontal/vertical decision. More directions or a broader model can help; failure is not inevitable on every diagonal.

Repetitive texture near the sampling limit can alias into apparently consistent false structure. Wider context helps some cases, but a single mosaic may leave genuine ambiguity.

Clipped samples no longer report the incident signal linearly. Mark them and avoid treating their color differences as ordinary measurements. The amount of recoverable highlight color depends on which channels remain useful.

Noise perturbs gradients and can switch interpolation decisions. The error depends on signal-to-noise ratio, not ISO alone. Chapter 15.8, Joint demosaicking and denoising incorporates this uncertainty in reconstruction.

Mosaics and aliasing

All channels at once: luma and chroma

OPTIONAL
in the book ↗

The mosaic is one image, not three, and its spectrum is the sum of the three sampled channels. Because the masks add up to one at every pixel, their impulses away from zero cancel for a gray scene: whatever sits at a carrier frequency is a difference between channels. For Bayer, with red at even-even positions and blue at odd-odd positions, the algebra is exact and short:

$$y_{ij} = L_{ij} + C_{1,ij}\left[(-1)^i + (-1)^j\right] + C_{2,ij}(-1)^{i+j}, \tag{12}$$

$$L = (R + 2G + B)/4, \quad C_1 = (R - B)/4, \quad C_2 = (R - 2G + B)/4. \tag{13}$$

At even-even positions the expression gives R, at mixed parity G, and at odd-odd positions B, which is a direct implementation check; a different origin for the array changes the signs of the carriers. The luma L is a mixture of camera channels, not CIE luminance. Multiplication by $(-1)^i$ shifts a spectrum by $1/2$ cycle per pixel, so the mosaic is the luma at baseband plus C_1 modulated onto the two axis carriers $(1/2, 0)$ and $(0, 1/2)$, plus C_2 modulated onto the diagonal carrier $(1/2, 1/2)$ (Alleysson, Süsstrunk & Hérault 2005 (frequency-domain demosaicing); Dubois 2005 (frequency-domain demosaicking)). Given separated components, $G = L - C_2$, $R = L + C_2 + 2C_1$ and $B = L + C_2 - 2C_1$.

The luma is sampled at every pixel and keeps the full band of the pixel grid. The chroma is what pays for the mosaic: it is recorded only as copies parked at the highest frequencies the grid supports. This works because natural images have much less chroma detail than luma detail, so the chroma copies are compact and mostly stay clear of the luma band. Chapter 15.2, Advanced traditional demosaicking used this model as an algorithm, filtering the chroma copies out and shifting them back; here it is a way of seeing what goes wrong.

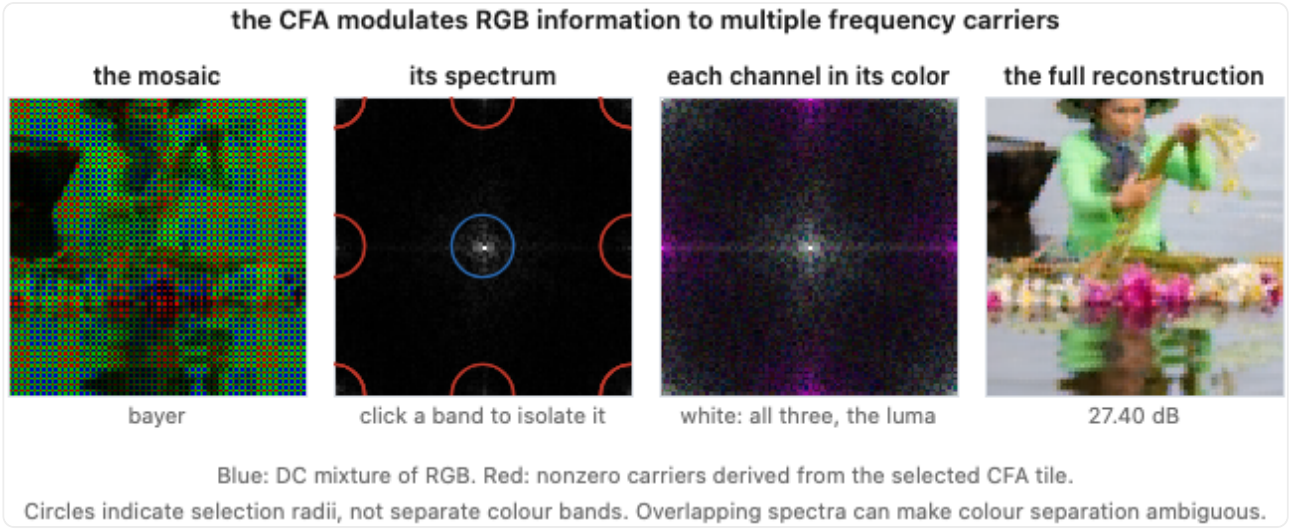


Figure 15.3.2 (reproduced). *Interactive. The spectrum of a mosaicked image, with the luminance band and the chrominance copies outlined and labeled; the copy positions are derived from the array's own tile rather than drawn by hand. A scene selector switches between a smooth image, where the copies are compact and well separated, and a finely textured one, where the luminance band spreads until it collides with them. Select a band to see the mosaic filtered to that band together with its mirror-image conjugate, which keeps the result a real image: a chrominance copy shows the color modulation it carries, and the luminance band shows the detail. The output is a rescaled, filtered mosaic, not a reconstructed color image. The array selector adds X-Trans, Quad-Bayer and Nona-Bayer, whose copies sit elsewhere, and a color view beside the gray one transforms each channel's share of the mosaic separately and draws it in its own color, divided by that channel's share of the sites, so that the luma of a gray scene is white and each chroma copy shows which channel differences it carries.* Interactive: choose the scene and the array, isolate a band with its conjugate, and see the filtered mosaic it leaves.

False color and moiré as spectral overlap

OPTIONAL
in the book ↗

Aliasing on a mosaic is the luma band reaching a chroma carrier. Fine gray detail near $1/2$ cycle per pixel, a fence, a woven fabric, the fine print on a page, puts luma energy exactly where a demosaicker looks for chroma. The demosaicker cannot tell the two apart and reports a color the scene did not have: false color. A pattern whose frequency is close to a carrier, but not on it, beats against the carrier and produces a slow colored ripple: color moiré. Where the luma band and a chroma copy overlap, the mosaic does not determine which is which, and no reconstruction can separate them without an assumption about the scene. Image context and priors, including learned ones (Chapter 15.6, Learned demosaicking), make better guesses, and extra captures (Chapter 15.12, Hardware alternatives to the mosaic) remove the ambiguity; none of this recovers arbitrary overlapping signals from one mosaic.

Foreshortening makes this common. A fine pattern on a surface that recedes from the camera, a floor, a roof, a striped shirt turned away, grows finer on the sensor with distance, so it sweeps through every frequency on its way to the horizon and passes the carriers at some distance or angle, where the colors appear (Figure 15.3.3).

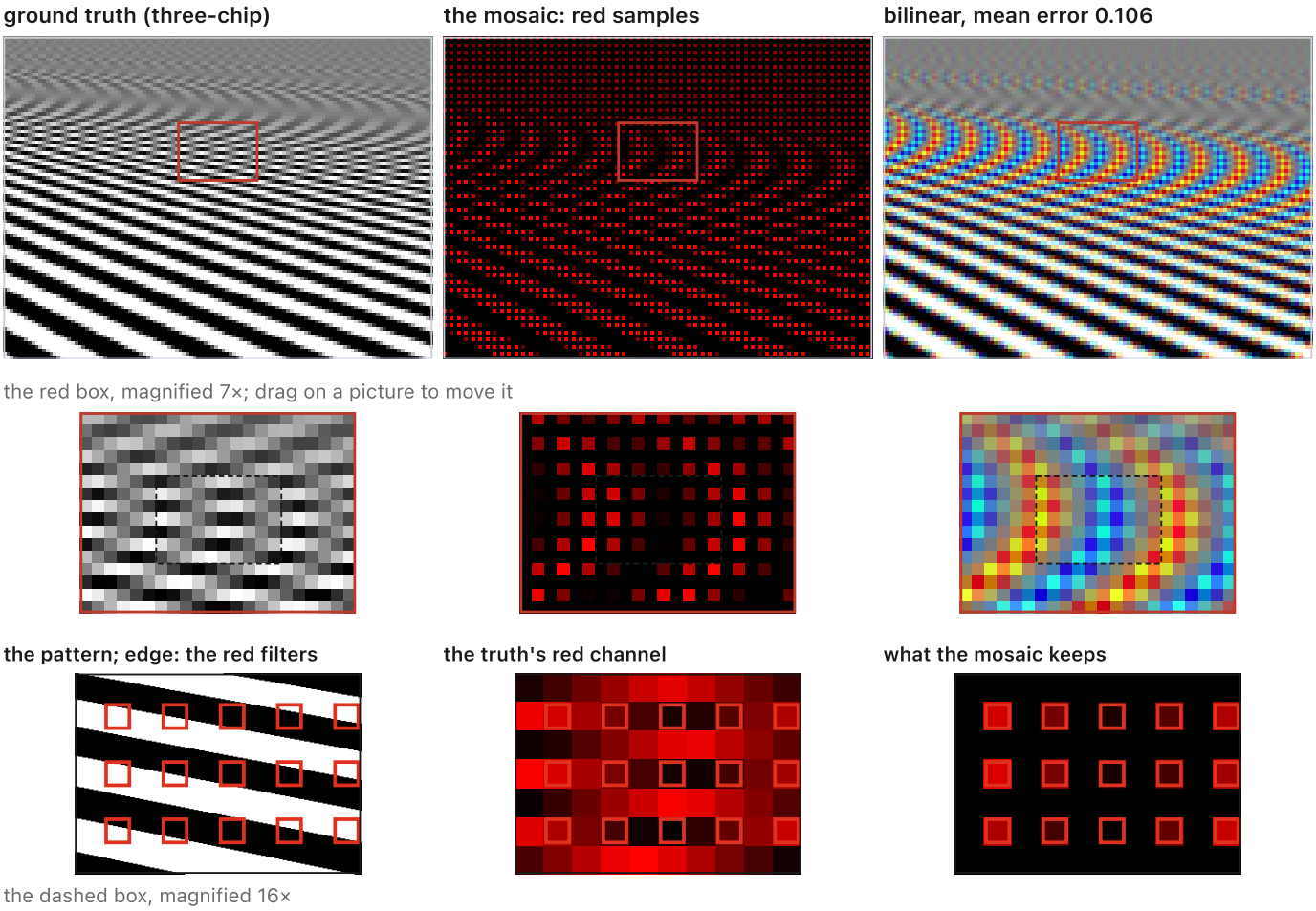


Figure 15.3.3 (reproduced). *A striped plane seen by a small camera, tilted away from it. Left, what a three-chip camera with the same pixels records: each pixel averages the pattern over its own square. Center, the Bayer mosaic of that picture, one channel at a time, red to begin with, or all three in color; with one channel chosen, the closest row below shows that channel alone. Right, its reconstruction by the chosen method, bilinear to begin with. The red box is magnified below each picture, and the center of that box is magnified again under it: on the left the pattern itself, not averaged, under the pixel squares, each square edged in the color of its filter, so the stripes can be seen falling on the red, green and blue sites at different phases; then the truth and the mosaic there. The truth itself shows moiré where the stripes pass 1/2 cycle per pixel: averaging over a pixel's square keeps 64% of the stripes' contrast at 1/2 cycle per pixel and about a fifth between 1.25 and 1.5, and that folds back. Optional dots mark where the stripes reach 1/4 cycle per pixel, the limit of the red and blue lattices, and 1/2, the carriers; false color begins between them. Tilting the plane or changing the stripes' frequency moves the band. Stripes parallel to the horizon vary only down the columns, and every method that interpolates along the rows reconstructs them exactly; lean them and none does. The anti-aliasing filter, four spots one pixel apart, removes most of the color at the price of the finer stripes.* Interactive: choose the reconstruction (bilinear, the chapter's green-based method, Hamilton–Adams, Malvar–He–Cutler, AHD), the mosaic's display, the pattern (sine, boxes, sawtooth, irregular threads), an infinite plane or a rectangle, and the colors; set the tilt and the stripes' frequency, orientation and phase; switch the anti-aliasing filter and the marks; drag on a picture to move the magnified window.

The same picture explains the two greens. Green carries half the luma, so doubling it keeps the luma band wide, while red and blue enter mostly through the chroma, whose narrow band the sparser lattice can hold. This is a statement about natural images and about the eye, which also resolves luma more finely than chroma, and not

about green light as such. A sharp colored pattern, red text on a blue sign, violates the assumption and aliases in the chroma itself.

The model is global and images are not. A photograph is smooth in some regions and finely textured in others, so a filter that separates the bands correctly in one place is wrong in another; the adaptive methods of Chapter 15.2, Advanced traditional demosaicking estimate the local structure and change the filter accordingly. The carrier positions describe an array, not the error of a particular demosaicker on a particular image Hirakawa & Wolfe 2008 (spatio-spectral sampling).

Learned demosaicking

OPTIONAL
in the book ↗

Fine fabric and thin diagonal structures can defeat local interpolation rules. A learned demosaicker uses a larger training collection to fit how sparse color samples map to full-color detail.

A synthetic training pair is easy to construct by selecting one component per pixel from a full-color image. Producing a representative clean target is harder: a processed photograph may already contain demosaicking, denoising, sharpening and compression artifacts, and its RGB channels need not be camera-linear responses.

Specify the color domain and how the target was prepared before treating remosaicking as a sensor simulation.

Joint demosaicking and denoising

OPTIONAL
in the book ↗

A noisy raw mosaic contains missing color components and uncertain measured samples. Demosaicking and denoising can therefore benefit from a shared estimator. A denoiser that ignores the CFA mixes unlike colors; a demosaicker that ignores noise can make poor direction decisions and spread their errors.

These are failures of particular models and interfaces, not proofs that either sequential order must fail. CFA-aware denoising can preserve color phases, and an RGB denoiser can model correlations introduced by demosaicking. Joint estimation retains access to the original observations and avoids treating an intermediate estimate as exact. Its benefit must still be measured against suitable sequential baselines.

For a fixed linear demosaicker A and raw noise covariance Σ, the output covariance is $A\Sigma A^{\mathsf{T}}$. Reused samples correlate output values. Simple averaging of independent equal-variance samples reduces variance, but negative correction weights or noisy adaptive choices need not. Thus post-demosaic noise is not universally quieter, and a raw-domain scalar variance estimate does not fully describe it.

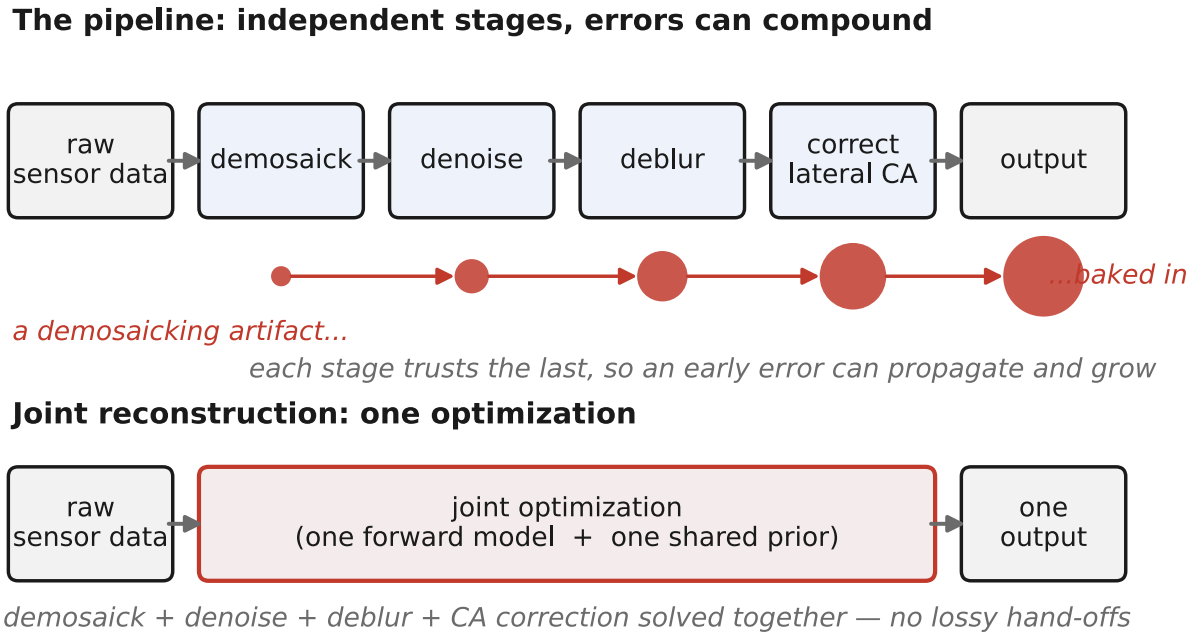


Figure 15.8.1 (reproduced). *The pipeline versus the joint solve. Top: the classical ISP as a chain of independent stages (raw → demosaick → denoise → deblur → correct lateral CA → output), each designed assuming clean input from the last. A small error introduced at demosaicking (red marker) is trusted by every downstream stage and can grow at each hand-off: the denoiser may preserve it as "signal," the deblur may sharpen it, and the CA corrector cannot undo it. That is a way errors can compound, not a measured rule that they always do. Bottom: joint reconstruction replaces the chain with a single optimization: one forward model (mosaic · blur · warp) plus one shared image prior, mapping the raw data to an image with no lossy intermediate commitments for later stages to trust by mistake. It removes that route for errors to compound; it does not by itself guarantee a clean result.*

Follow a possible error on a striped shirt through a sequential pipeline. A demosaicker produces a patch of false color where its local model cannot resolve the texture. A subsequent denoiser may preserve that patch because it is spatially coherent, even while removing independent grain around it. Sharpening can then increase its contrast. Each operation can be reasonable for its assumed input while the composite result reinforces an earlier mistake. This is a reason to retain the original mosaic and its uncertainty when designing the estimator. It is not inevitable: a sequential method whose later stages model reconstruction artifacts can behave differently.

The information-flow issue is the commitment at an intermediate image. Once a stage passes only one RGB estimate onward, a later stage may have no indication that two different textures fitted the original mosaic almost equally well. It cannot reconsider that ambiguity using the raw likelihood unless the raw data or an adequate uncertainty representation remains available. A joint objective keeps those constraints active while the estimate changes. This is the useful distinction between coordinating several reconstruction operations and merely running them once in sequence.

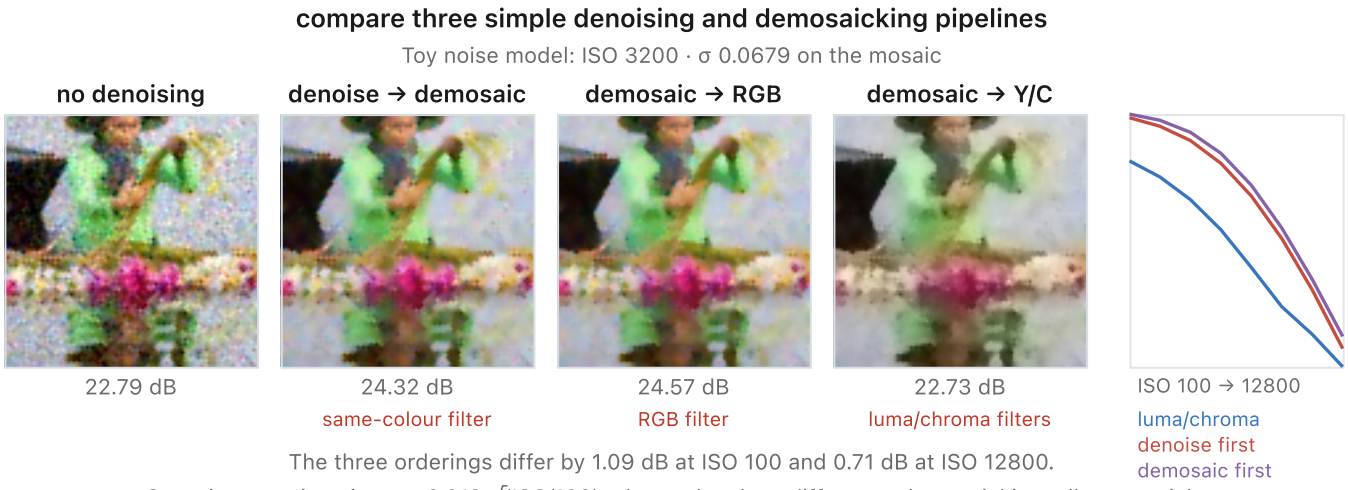
The two orderings, made visible

OPTIONAL
in the book ↗

Compare the same raw input, output domain and computation budget. Tune the sequential and joint methods separately on validation data.

Denoising before reconstruction can remove details needed for interpolation if it oversmooths them. A CFA-aware denoiser need not mix neighboring color channels, and its strength can be chosen to retain those structures.

Demosaicking first changes the noise covariance and may introduce structured artifacts. A denoiser trained or designed for this input can address some of them. Independent white-noise assumptions should not be transferred unchanged from raw samples to demosaicked RGB.



These measurements do not test a joint estimator or establish how learned methods rank.

Figure 15.8.2 (reproduced). *Interactive. One noisy raw crop through three pipelines, at matched strength, with the truth and a per-pixel error map for each: denoise then demosaick, demosaick then denoise, and demosaick then filter luma and chroma separately. All three are sequential; the third changes what gets filtered, not the order of estimation, so none of them is a joint estimator. Each ordering's damage shows in its error map: texture flattened where the denoiser ran on the mosaic, structured maze artifacts where the demosaicker ran on noise. What makes deployed joint methods win is estimating both at once, usually by training the whole chain, which no hand-built panel can show. Watch how the gap between the orderings changes with the noise level rather than reading any one setting as the verdict.* Interactive: set the noise level and the filter strength, switch the ordering, toggle error maps, and move the crop between flat, edge and texture regions.

FlexISP: one energy, one prior

OPTIONAL
in the book ↗

The idea of collapsing the pipeline into a single optimization was crystallized by Heide et al. (2014) in a system named FlexISP, a flexible image-signal processor built as one energy function rather than a fixed chain Heide et al. 2014. Its architecture is exactly the inverse-problem template we have been using. On one side, a single image-

formation model: a linear operator A that *composes* all the physical steps the sensor performed (subsampling by the color-filter mosaic, blurring by the optics' point-spread function, geometric warps for lens distortion and lateral color, even the resampling of a burst of frames) into one map from the clean latent image x to the raw measurement y. On the other side, regularization applied to the shared latent image, combining image and cross-channel structure. Recovery is the one objective $\frac{1}{2}\|Ax - y\|^2 + \lambda\,\Phi(x)$, minimized as one large optimization.

A composite operator permits the data fit to account for several physical effects together. Operator splitting decomposes the numerical solve into manageable subproblems; it does not remove the need to define each forward operation, boundary and adjoint consistently. Chapter 13.11, Denoising as a universal prior distinguishes explicit proximal priors from arbitrary inserted denoisers.

A shared final-image prior avoids some incompatible intermediate objectives, but its accuracy remains a modeling choice. FlexISP demonstrates a flexible formulation, not a theorem that one prior always beats every stagewise pipeline.



Video 4. Video: *FlexISP: A Flexible Camera Image Processing Framework*: one optimization, built around a single image prior, solving demosaicking, denoising, deblurring and more as a single fused inverse problem instead of a fixed pipeline. Heide, Steinberger, Tsai, Rouf, Pajak, Reddy, Gallo, Liu, Heidrich, Egiazarian, Kautz and Pulli 2014 (**Heide et al. 2014**), reproduced with permission.

FlexISP evaluated ADMM Boyd et al. 2011 (ADMM) and a first-order primal-dual solver, and used the latter in its reported examples. Its regularization combines BM3D Dabov et al. 2007, total variation and a cross-channel term rather than relying on a color penalty alone (FlexISP paper). Splitting the numerical work into updates does not recreate a one-pass ISP: the updates revisit one shared image and reconcile their constraints through iteration. This is why a system can contain several computational modules while still solving a joint reconstruction problem.

Joint methods and the end-to-end pipeline

Where demosaicking sits in the pipeline

OPTIONAL
in the book ↗

Demosaicking normally follows linearization, black-level correction, and defect handling. White balance and denoising may occur before it, after it, or jointly with it; a scalar CFA sample can already receive its channel's white-balance gain. A camera-to-output color matrix requires reconstructed channels. Nonlinear rendering and output encoding belong to the later appearance stages, as summarized in Chapter 8.12, Recap ISP, non-destructive editing.

Hardware alternatives to the mosaic

Pixel shift: spending time, precisely

OPTIONAL
in the book ↗

In an ideal static scene, a four-shot cycle with one-pixel offsets can sample each scene-grid position through red, both green phases and blue. This avoids interpolating missing color components at covered interior positions. Actual fusion still needs calibration, accurate offsets, valid boundaries and motion handling.

Two patterns are in use and they buy different things. A one-pixel, four-shot cycle buys color: full color at the sensor's native resolution, direct sampling of each color at covered positions, with demosaicking removed from the pipeline entirely. A half-pixel cycle, usually eight or sixteen shots combining whole- and half-pixel steps, buys resolution as well, interleaving the sampling grids finely enough to raise the effective pixel count, which is the deliberate version of the hand tremor that Chapter 14.2, Multi-frame super-resolution exploits. Sony calls it Pixel Shift Multi Shooting, Pentax calls it Pixel Shift Resolution, OM System calls it High Res Shot, and Hasselblad calls it Multi-Shot, which tells you the market: landscape, product, and cultural-heritage work.

Commanded actuator offsets do not guarantee exact scene registration. Shake, lens distortion, actuator error and subject motion can perturb them. Fusion may reject moving regions or use a single-frame reconstruction there. Controlled pixel shift and unconstrained handheld fusion therefore share sampling principles but need not use identical algorithms.

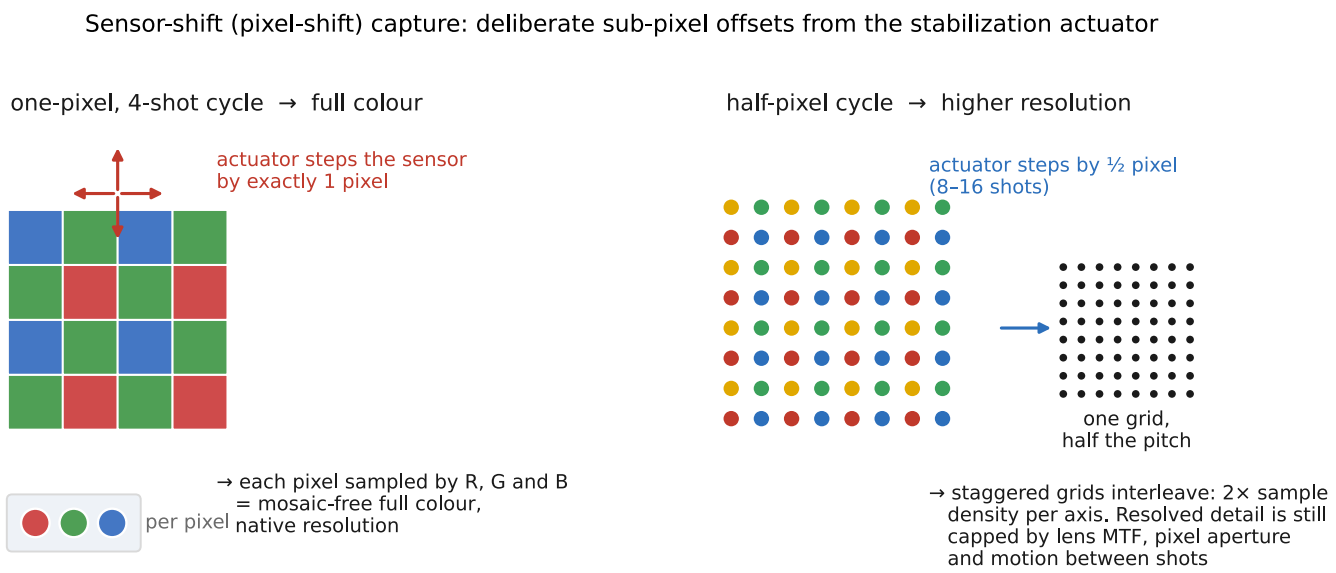


Figure 15.12.1 (reproduced). *Manufacturing the offsets. A sensor-shift mechanism, the in-body stabilization actuator, steps the sensor by precise fractions of a pixel between exposures. Left: a one-pixel, four-shot cycle, in which every pixel visits a red, green and blue mosaic position, yielding a mosaic-free full-color image at native resolution. Right: a half-pixel cycle, more shots, interleaving the sampling grids to double the sample density in each direction, which raises the resolvable detail only as far as the lens and a motionless scene allow. The same offsets can come deliberately from the actuator, on a tripod with a static scene, or accidentally from hand tremor, the handheld and phone case: the reconstruction rests on the same idea, but handheld offsets are irregular and must be estimated, and scene motion must be handled.*

Foveon and stacked photodiodes: spending silicon depth

OPTIONAL
in the book ↗

Silicon absorbs short wavelengths near its surface and long wavelengths deeper in, so three photodiodes stacked at different depths in the same pixel respond to three different, broad, overlapping spectral bands. That is the Foveon sensor (Lyon & Hubel 2002 (Foveon)): three measurements at every pixel location, no filter array, no demosaicking, with reduced CFA-specific color aliasing when all channels share the same grid. Ordinary spatial aliasing and channel-response differences remain; some stacked designs also use different sampling densities across layers. The full account of the physics is in Chapter 4.6, Sensing color: multiplexing strategies; the reason it belongs here is what it implies for reconstruction, which is that missing-color interpolation can be reduced while spectral separation and noise estimation remain necessary.

It did not displace Bayer, and the reasons are instructive because none of them is about the idea. The three depth responses overlap heavily and are not the primaries anyone would choose, so recovering colorimetric values needs a color-separation transform whose coefficients and conditioning can amplify noise: the color separation is set by silicon's absorption curve rather than by a filter someone designed. Carriers generated at one depth diffuse to another, which is crosstalk that further mixes the bands. The potential advantage is collecting useful light at several depths without a surface color mosaic. Not every incident photon is collected, however, and collection efficiency

differs by layer. The resulting benefit depends on both collection and how well the overlapping responses can be separated.

The lesson generalizes past this one sensor. A measurement that captures more per pixel is only better if the mixing it introduces can be inverted without amplifying noise, which is the same conditioning argument that governs every multiplexed measurement in Part 34, Computational sensors.

Four ways to multiplex color

OPTIONAL
in the book ↗

The mosaic is a clever answer to a hard constraint (a photosite measures one number, and color needs three) but it is not the only answer, and seeing the alternatives makes clear what the mosaic trades away. Broadly, you can multiplex the three measurements in time, in space, or in depth.

- **Temporal multiplexing**: take three exposures through three filters, one after another. Sequential filtered acquisition was used by Sergei Prokudin-Gorskii for color photographs of the Russian Empire around 1900, shooting three plates through red, green, and blue filters. (The same idea lives on in the color wheel spun in front of a monochrome sensor.) You get three real measurements at every pixel and need no interpolation, but only for a static scene; anything that moves between exposures shows color fringes.
- **Spatial multiplexing**: one measurement per location, varying the color across the grid: the Bayer mosaic itself, and, not coincidentally, the strategy your retina uses (a mosaic of long-, medium-, and short-wavelength cones). Single sensor, high resolution, mature technology, at the cost of interpolation (this whole chapter), color jaggies, the resolution lost to the OLPF, and the light absorbed by the filters.
- **Three sensors**: split the incoming light with a prism and route it to three separate sensors, one per color (the 3-chip / 3-CCD video cameras, named after the charge-coupled device (CCD) sensor). Three real values per pixel and almost no photons wasted (the prism redirects rather than absorbs), but you pay in cost, bulk, and alignment.
- **Depth multiplexing**: stack the color-sensitive layers so each pixel measures all three at once at different depths: color film (tripack emulsions) and the Foveon sensor, which exploits the fact that longer wavelengths penetrate silicon more deeply. Three numbers per pixel and good light efficiency, but it needs more color processing and tends to be noisier.

These all tie back to Chapter 4.1, Linear color reproduction and CIE XYZ of the book; the takeaway is that the Bayer mosaic wins on cost, resolution, and single-sensor simplicity, and demosaicking is the computational price we pay for that win.

Demosaicking TLDR

in the book ↗

- Preserve the raw mosaic, CFA layout and origin, black/white levels, gain and saturation mask. Cropping or rotating a mosaic changes its phase. A network expecting RGGB will not interpret BGGR correctly by itself.
- Use a demosaicker matched to the sensor and noise conditions. Compare a strong classical method with a learned or joint method on fine fabric, colored edges and small text; clean-looking detail can still be wrong

(Chapter 15.1, Basic traditional demosaicking, Chapter 15.2, Advanced traditional demosaicking, Chapter 15.6, Learned demosaicking).

- Model noise in the domain where reconstruction operates. Gains scale variance by their square; interpolation can introduce correlations. Joint demosaicking and denoising can help, but compare it with properly designed sequential baselines (Chapter 15.8, Joint demosaicking and denoising).
- Check calibration before tuning reconstruction: defects, autofocus-site correction, clipping, shading and chromatic displacement can create colored artifacts. Highlight recovery estimates missing signal; it cannot guarantee original color after saturation.
- Keep a documented color representation for later rendering. Learned reconstruction can coexist with exposure, white-balance and tone controls. End-to-end training does not require giving up editable stages (Chapter 15.9, Joint methods and the end-to-end pipeline).
- Treat synthetic remosaicking and unprocessing Brooks et al. 2019 (Unprocessing) as training models. They do not recreate the original sensor capture exactly. Validate on representative cameras and report input domain, reference preparation, color error and computation.

A Bayer photosite measures one filtered channel, leaving the other two to be estimated. Two classical ideas make a useful starting point: interpolate along an edge, and interpolate color differences rather than independently smoothing every channel. Estimate the more densely sampled green channel, then use it to guide red and blue. Color differences are often smoother than the channels, though saturated color edges can break that assumption. Residual interpolation Kiku et al. 2013 (residual interpolation) takes the same idea further: fit a local relationship between channels and interpolate what the fit leaves unexplained (Residual interpolation and ARI).

The frequency view explains why fine monochrome texture can turn into false color. A periodic color-filter array shifts combinations of color signals into different frequency bands. Reconstruction must separate them; when scene detail overlaps a chrominance band, the measurements admit competing color and texture explanations. That is a sampling ambiguity, and a cleaner-looking reconstruction can still choose incorrectly (Chapter 15.3, Mosaics and aliasing).

Array design changes the problem. X-Trans changes the pattern and distribution of aliases; it does not eliminate aliasing. Quad-Bayer groups like-colored sites, supporting binning and reconstruction modes with different resolution and noise tradeoffs. RGBW adds broad-band measurements while reducing the sites allocated to filtered color. Match the reconstruction to the actual array and operating mode (Chapter 15.5, Alternative color filter arrays).

Denoising and demosaicking affect one another. Denoising a raw mosaic must respect the different channels; indiscriminate neighborhood averaging mixes different measurements. Demosaicking first can turn independent sensor errors into correlated errors and can turn a noisy direction decision into a visible artifact. Joint methods use the mosaic and its noise model together. They can help avoid this propagation, but still impose a prior and can lose detail or fail on unfamiliar data. A sequential baseline designed for the correct domains remains a useful comparison (Chapter 15.8, Joint demosaicking and denoising).

For a learned method, realistic supervision matters as much as the network. Smooth regions dominate random crops, while the difficult cases are fine fabric, roof tiles, fences, and colored edges. Include those cases during training and evaluation. Synthetic remosaicking provides convenient pairs, but a processed reference may already contain demosaicking artifacts. Unprocessing approximately reverses encoding and rendering to construct raw-like training examples; it cannot undo clipping or retrieve information discarded by the original pipeline (Chapter 15.6, Learned demosaicking, Chapter 15.11, Synthetic data, unprocessing, and raw reconstruction).

Inspect the artifacts separately from an average score. CPSNR can summarize reference error while hiding a small region of conspicuous zippering. Color and spatially filtered color-difference measures answer complementary questions; none replaces looking at the difficult structures. Use the same color representation, crop, alignment, and reference preparation when comparing methods.

The surrounding pipeline also matters. Learned raw reconstruction can feed a documented color representation with adjustable exposure, white balance, and tone controls. Joint training does not require an uneditable raw-to-JPEG network. Preserve the intermediate quantities a photographer or later rendering stage needs (Chapter 15.9, Joint methods and the end-to-end pipeline).

Alternative capture hardware changes which colors are missing. Pixel shift can measure more color samples at corresponding scene positions when the scene stays still. Layered sensors separate spectral responses through depth but still need color unmixing. Multi-sensor optical systems trade alignment and hardware complexity for simultaneous measurements. These alternatives spend time, space, or optical complexity to reduce the spatial sampling compromise of a mosaic (Chapter 15.12, Hardware alternatives to the mosaic).

Holography

Lippmann and Gabor: recording the wave

OPTIONAL
in the book ↗

The intellectual root is again Chapter 31.2, Light field cameras, the same physicist behind integral photography. In 1891 Gabriel Lippmann invented interference color photography Lippmann 1891, which won him the 1908 Nobel Prize. He recorded color not with dyes but with *standing waves*: light reflecting off a mirror behind the emulsion interfered with the incoming light, laying down microscopic layers in the film at spacings set by the light's wavelength, so the developed plate reconstructed color by reflecting back exactly those wavelengths. It was the first photographic process to record information *in the standing-wave structure of light* rather than in intensity alone, the conceptual seed of capturing phase.

Holography proper was invented by Dennis Gabor in 1948 Gabor 1948, for which he received the 1971 Nobel Prize. (He was trying to improve electron microscopy, not to make 3-D pictures; the 3-D display application came later.) Gabor's insight: bring the light from the object together with a coherent reference beam at the recording plate, and let them interfere. Where the two waves arrive in phase they reinforce (bright); where they arrive out of phase they cancel (dark). The resulting pattern of fine fringes is an intensity pattern an ordinary plate *can* record, but because the fringe positions depend on the *phase difference* between object and reference at each point, that intensity pattern secretly encodes the object wave's phase as well as its amplitude. The phase difference appears in a recordable intensity. With object field U_o and reference field U_r,

$$I = |U_o + U_r|^2 = |U_o|^2 + |U_r|^2 + 2\,\mathrm{Re}(U_o U_r^*). \tag{14}$$

The cross term carries their relative phase. A single intensity sample does not uniquely determine a general complex field; the known reference and the holographic encoding geometry make separation and reconstruction possible.

On replay, illuminate the developed fringe pattern with the reference beam again, and the fringes act as a diffraction grating. They bend the reference light by exactly the angles that reconstruct the original object wave. The wavefront that leaves the hologram is (to good approximation) the very wavefront the object once sent. Within the reconstructed aperture, wavelength band, and viewing zone, it can reproduce the corresponding visual cues: look from the left and you see the object's left side; move your head and it shows full, continuous parallax; and your lens accommodates to the true depth of each virtual point, because the reconstructed wavefront has the correct curvature.

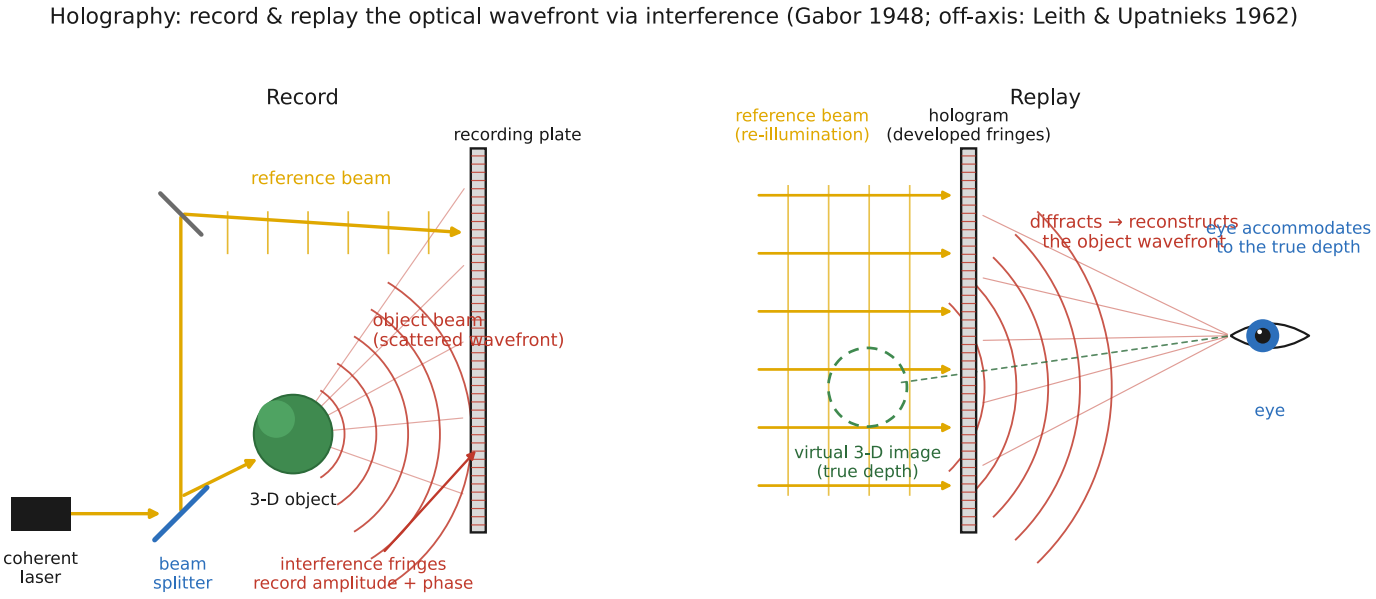


Figure 37.5.1 (reproduced). *Holography: record and replay the wavefront. Record (left): a coherent laser is split into an object beam that scatters off the 3-D object and a reference beam sent straight to the plate; the two interfere, laying down fine fringes whose positions encode the object wave's amplitude and phase. Replay (right): re-illuminating the developed fringe pattern with the reference beam makes it diffract the light back into the original object wavefront, so the eye sees a virtual 3-D image and accommodates to the true virtual depth. Because the actual wavefront is reconstructed, every cue is correct at once: stereo, full motion parallax, and accommodation. (Gabor 1948; off-axis geometry, Leith & Upatnieks 1962.)*

Recap: big lessons in this lecture

BIG LESSON (L4.11) — ONE VALUE PER PHOTOSITE; COLOR MUST BE MULTIPLEXED

Strip away the color filter and a sensor is **monochrome at the pixel**: each photosite reports a **single number**, a signal proportional to detected photoelectrons within its operating range, not a color. The three (or more) values that color needs must therefore be **multiplexed** along some other axis: over **space** (a color-filter array such as Bayer, paying with resolution and a demosaicking step), over **time** (sequential color filters, paying with motion robustness: the old color-wheel and many scientific cameras), across **multiple sensors** (a beam-splitter prism feeding three chips, the **3-CCD** video camera, paying with bulk and cost), or over **depth** (wavelength-dependent absorption in stacked photodiodes, the **Foveon**, paying with noise). The choice determines which measurements are simultaneous and co-located. The same constraint shapes the eye, whose three cone types sample color over space at the retina (L3.1 show, L3.6 show).

BIG LESSON (L15.1) — LET THE DATA PICK THE DIRECTION

Edge-directed demosaicking is a first, very concrete instance of a theme that returns throughout the book: adapt the operation to the local structure of the image rather than applying the same fixed filter everywhere. Here the structure is an edge and the adaptation is choosing the interpolation direction; later, in edge-preserving filtering and denoising, the same instinct (use the similarity between pixels to decide how much they should influence each other) becomes the bilateral filter and the idea of affinity. Whenever a fixed linear filter blurs across something it should have respected, the cure is to look at the data first.

BIG LESSON (L15.2) — LUMINANCE CARRIES DETAIL, CHROMINANCE IS SMOOTH

Channel differences are often smoother than luminance-like structure, which makes guided reconstruction useful. Preserve that distinction as an image model rather than a law: fine color texture and saturated boundaries can break it.