

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 Slopendum* 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

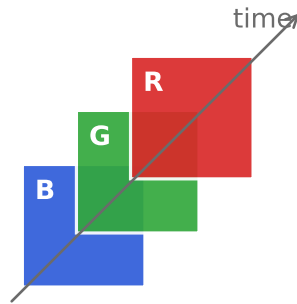
in the book [↗](#)

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](#)).

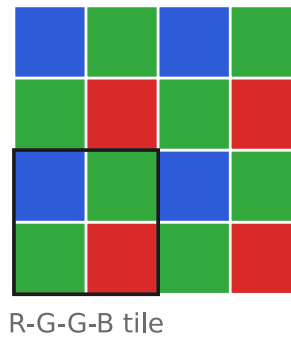
💡 BIG LESSON (L4.11) — ONE VALUE PER PHOTOSITE; COLOR MUST BE MULTIPLEXED

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](#)).

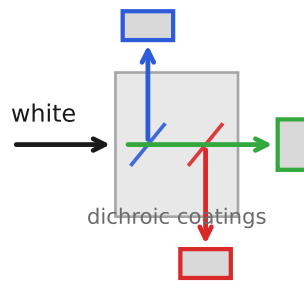
(a) temporal — filter wheel



(b) spatial — Bayer mosaic



(c) dichroic beam-split (3-CCD)



(d) depth — stacked (Foveon)

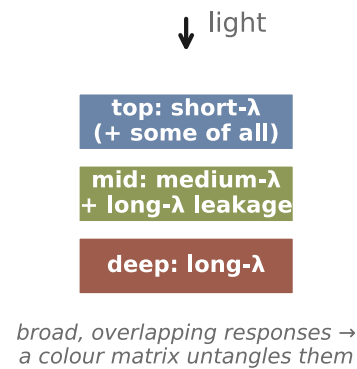


Figure 4.6.1 (reproduced). Four color-sensing multiplexing strategies. Temporal: shoot R, G, B frames in sequence through a filter wheel. Spatial: a color filter array (the Bayer mosaic) puts one color over each photosite. Beam-splitter: a prism sends the rays to three separate sensors. Depth: stacked 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 space: the color filter array

[in the book](#) ↗

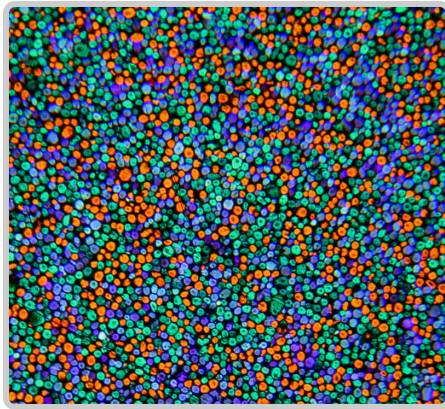
The dominant choice by far. Lay a color filter array (CFA) directly over the sensor so each photosite records one color and the missing two are filled in afterward by demosaicking ([Figure 15.0.1](#), at the start of [Part 15, Demosaicking and joint methods](#)). The archetype is the Bayer mosaic, two 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, CCo; box: Anne Jea, CC BY-SA 4.0; both via Wikimedia Commons.)

File formats and compression

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.

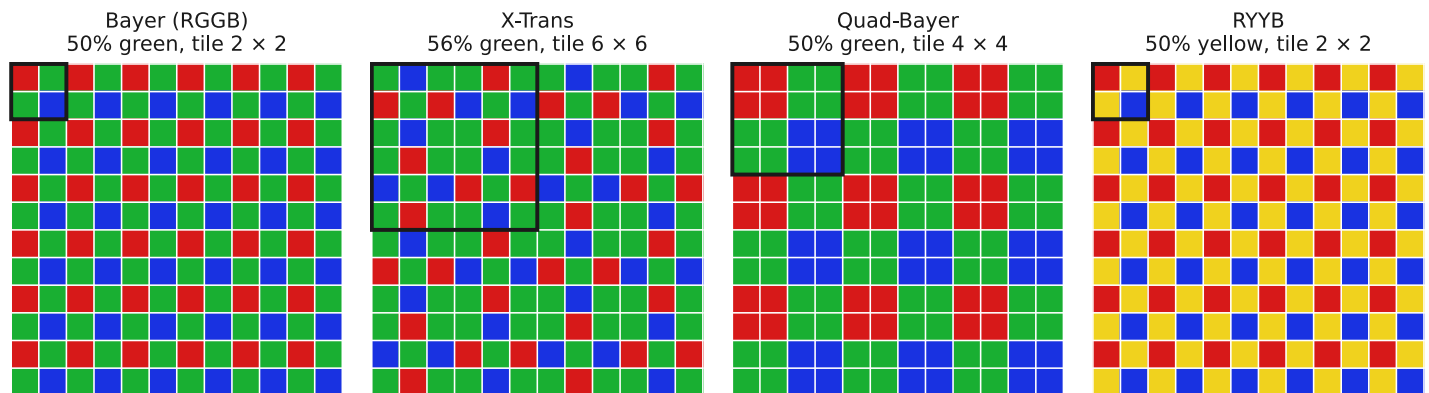


Figure 15.0.1 ([reproduced](#)). Four color filter arrays over the same 12×12 photosites, each with its repeating tile outlined. Bayer (RGGB) repeats a 2×2 tile of two green, one red and one blue filter, so each pixel measures one color and the other two must be reconstructed; the extra green matches 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.

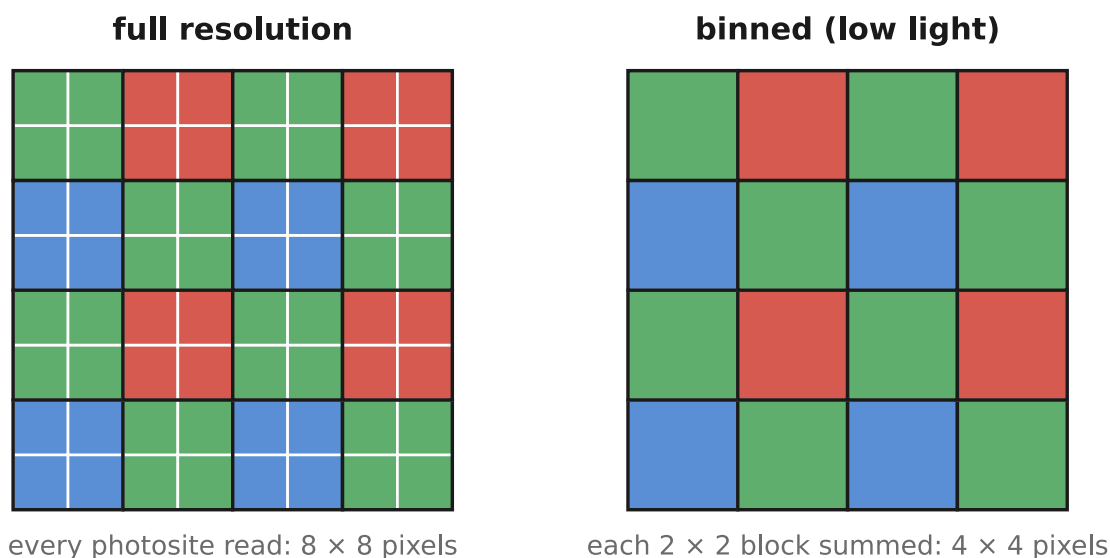


Figure 15.0.2 ([reproduced](#)). One Quad-Bayer sensor read two ways. Left, full resolution: every photosite is its own pixel, so each 2×2 block of one color is four pixels, which the camera remosaics into an ordinary Bayer image before demosaicking. Right, the low-light mode: each block is summed into 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.

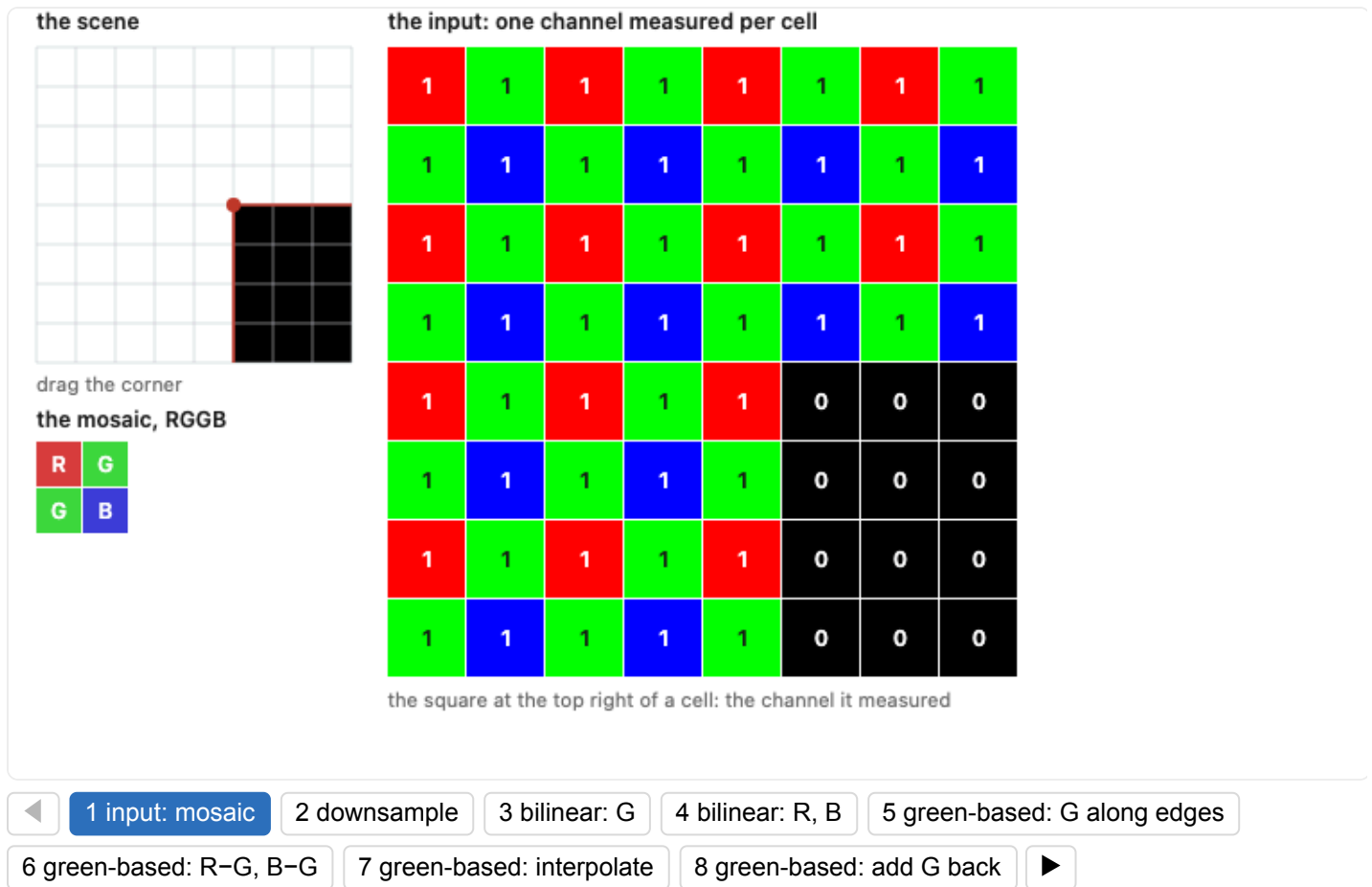


Figure 15.1.1, stage 1 (reproduced). Interactive. One corner through three ways of demosaicking, cell by cell, shown in stages through this chapter. A dark corner on a light background is area-sampled onto 8×8 cells and kept through an RGGB mosaic; by default its first dark cell is a measured green, 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 dcrw 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](#)).

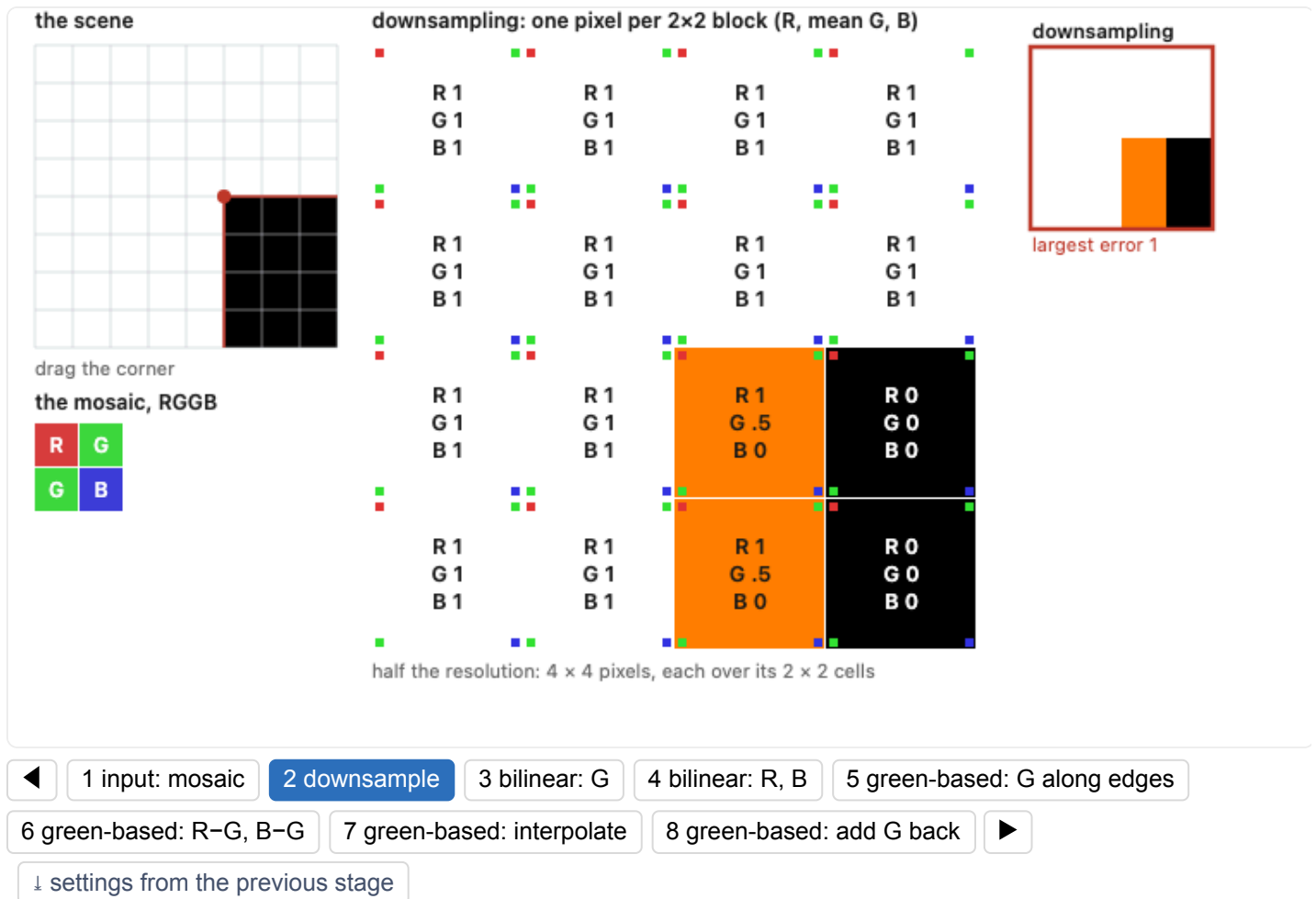


Figure 15.1.1, stage 2 ([reproduced](#)). The same corner, downsampled. Each 2×2 tile becomes one pixel with its red, the mean of its two greens and its blue. Nothing is interpolated, but the resolution halves, and a tile cut by the edge gets a color the scene does not have: orange here, largest error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

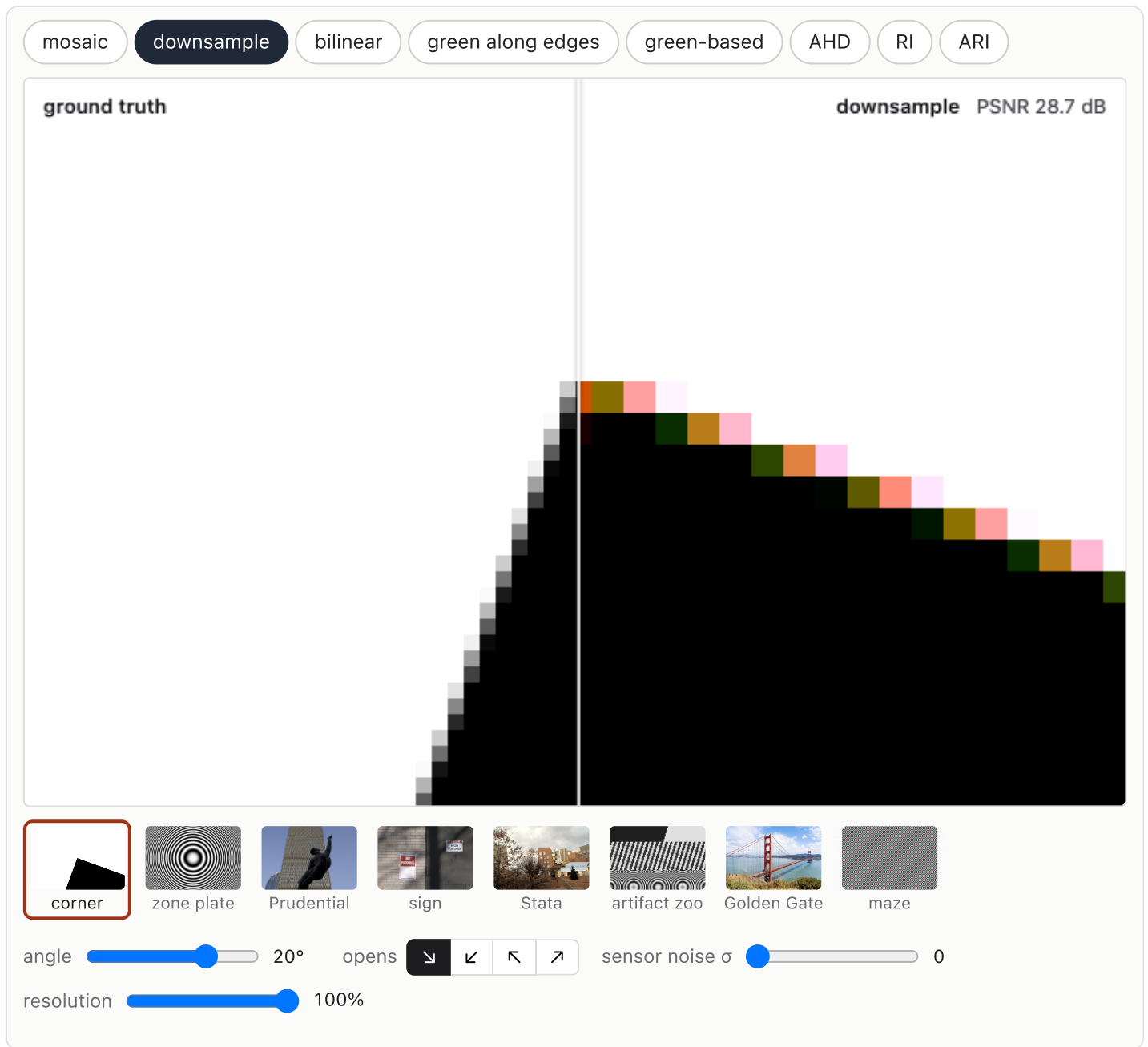


Figure 15.1.2, stage 1 ([reproduced](#)). Downsampling against the scene. Left of the split, the scene; right of it, the scene through a Bayer mosaic and reconstructed by one method, here downsampling, with its PSNR. The corner of [Figure 15.1.1](#) opens turned by 20 degrees and zoomed in on its tip; 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.

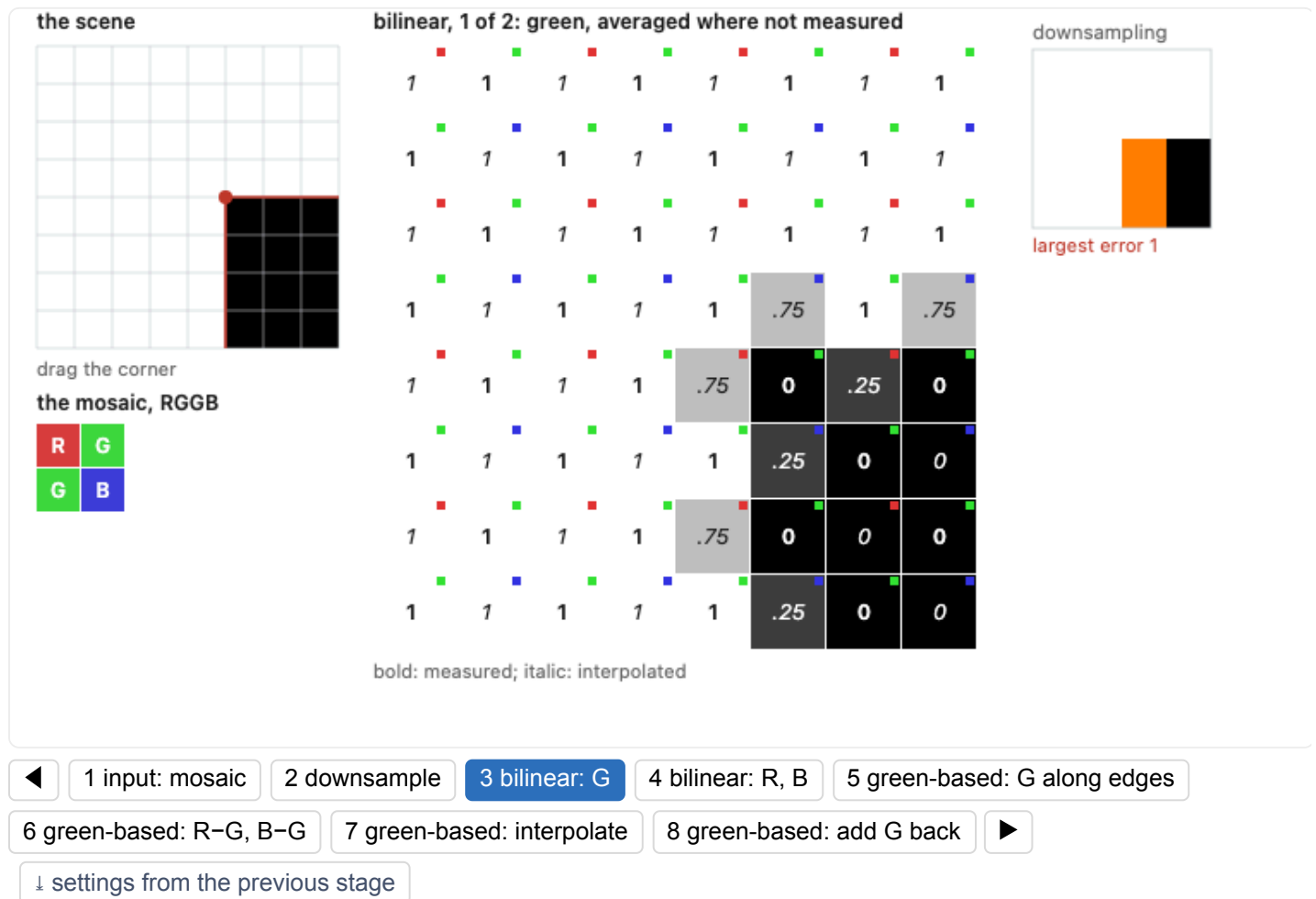
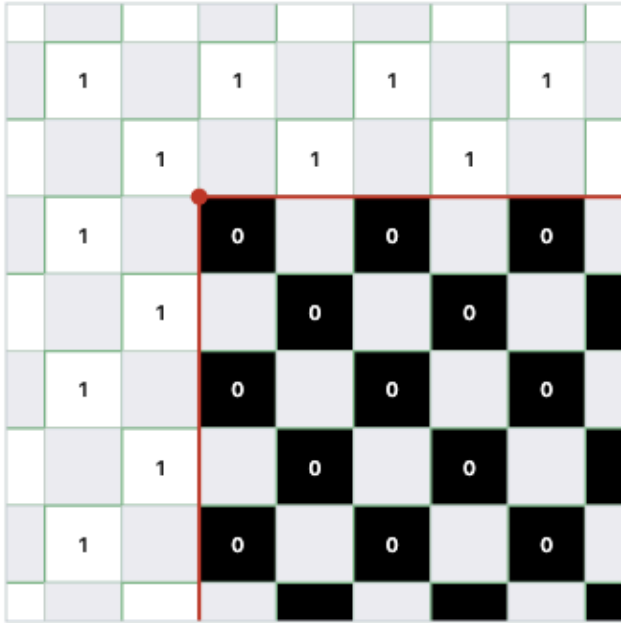


Figure 15.1.1, stage 3 (reproduced). Bilinear green. Each missing green is the average of its four green neighbors, which gives .25 and .75 along the edge of a scene that holds only 0s and 1s. Interactive: the same controls at every stage.

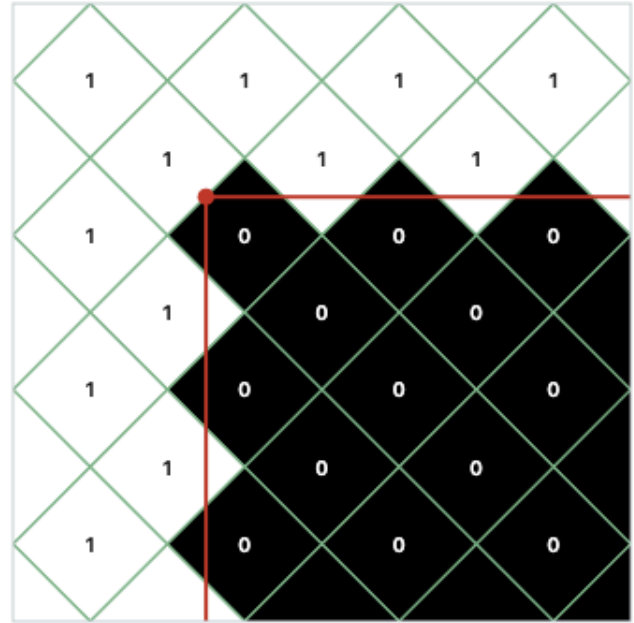
Zippering is an alternating light-dark pattern along a sharp horizontal or vertical edge. It comes from averaging across the edge: at one pixel the four neighbors are two on each side, at the next they are three and one, so the interpolated value alternates. The name comes from what it looks like. The 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.

green on the Bayer grid



grey: the red and blue sites, no green measured
drag the corner

the same samples as a grid of their own



one diamond per green sample

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.

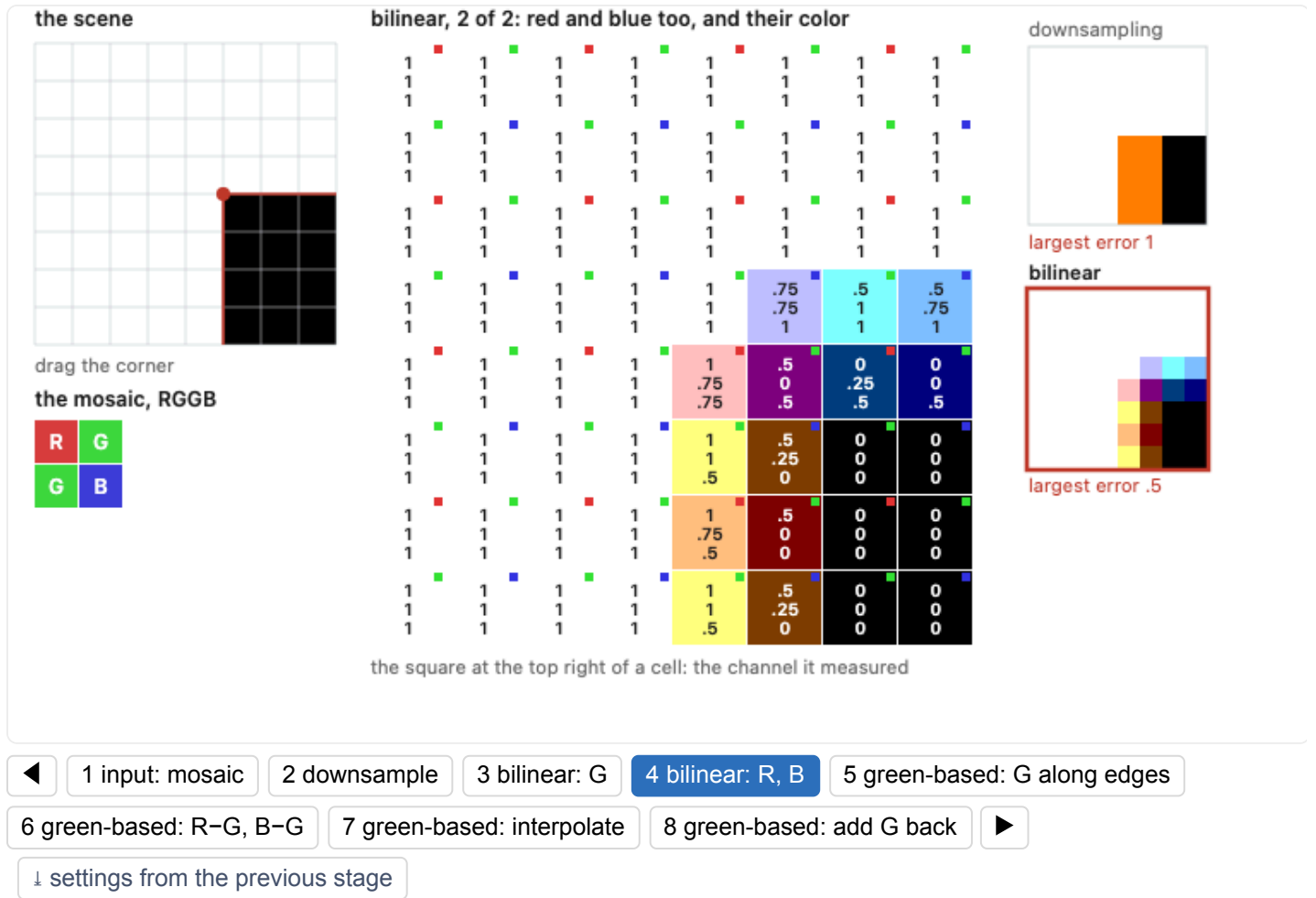


Figure 15.1.1, stage 4 ([reproduced](#)). Bilinear red and blue: from two neighbors at a green site, from the four diagonal ones at the other color's site. The three planes are interpolated from different cells, so they disagree at the edge and the corner comes out fringed, largest error .5. Interactive: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

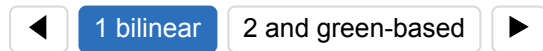
Concretely, for green, whose measured samples lie on a checkerboard, the four nearest measured neighbors sit above, below, left and right:

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

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).

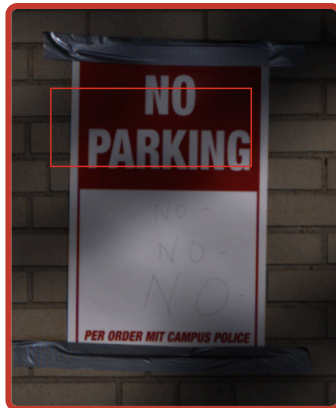


Click any picture to enlarge it; the arrow keys step through them.

The whole sign

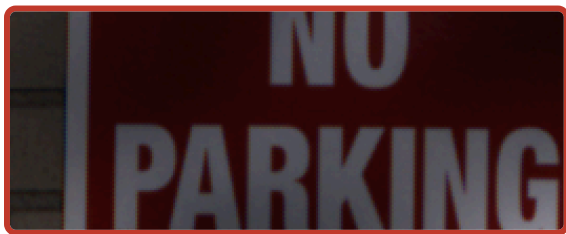


raw Bayer mosaic, one colour
per pixel



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.

Naive per-channel bilinear demosaick zippers at edges and invents colour fringes

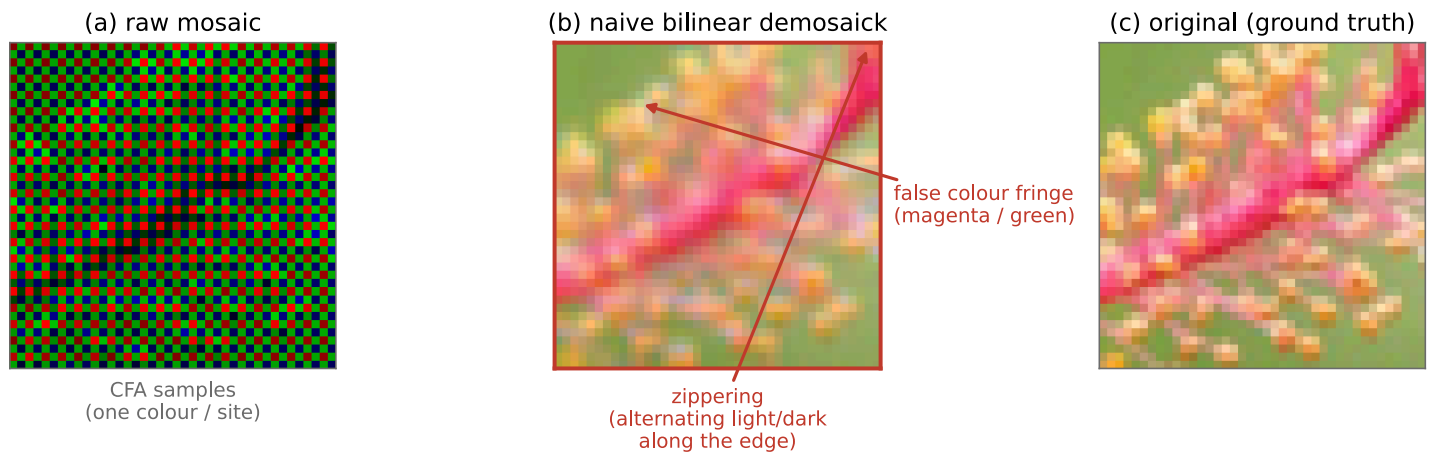
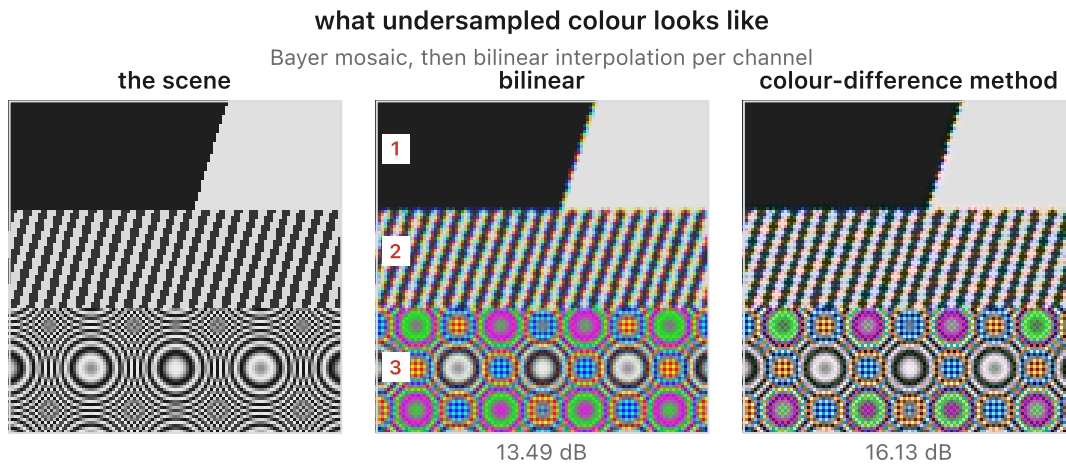


Figure 15.1.5 (reproduced). Demosaicking, naive versus edge-aware. Left, the Bayer mosaic input. Center, per-channel linear interpolation: colors are right in smooth areas but strong edges show two artifacts: a zipper (the alternating light/dark teeth running along a near-horizontal or near-vertical 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.



Shrink the feature scale toward the sampling limit and each artifact appears at the frequency that produces it. Compare the two reconstructions: colour-difference interpolation changes both the interpolated quantity and the support.

1: zippering along the edge

2: false colour on the thin lines

3: false colour and aliasing in the fine pattern

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.

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

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:

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

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).

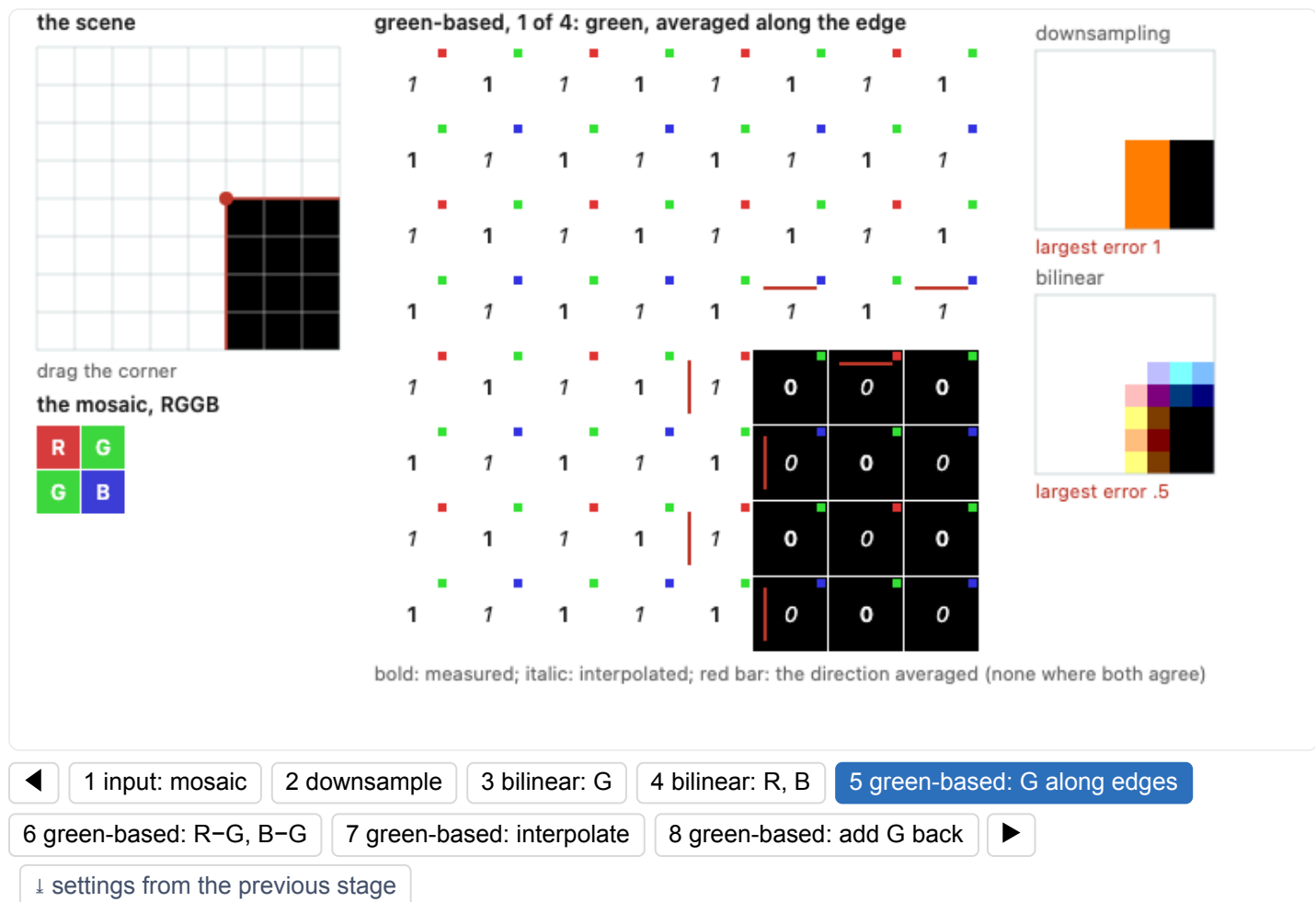


Figure 15.1.1, stage 5 ([reproduced](#)). *Edge-directed green on the corner. At each red or blue site the rule averages the green pair across which green changes least; the mark in each cell shows the direction it chose. On this corner the estimate is exact everywhere. Interactive: drag the corner so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

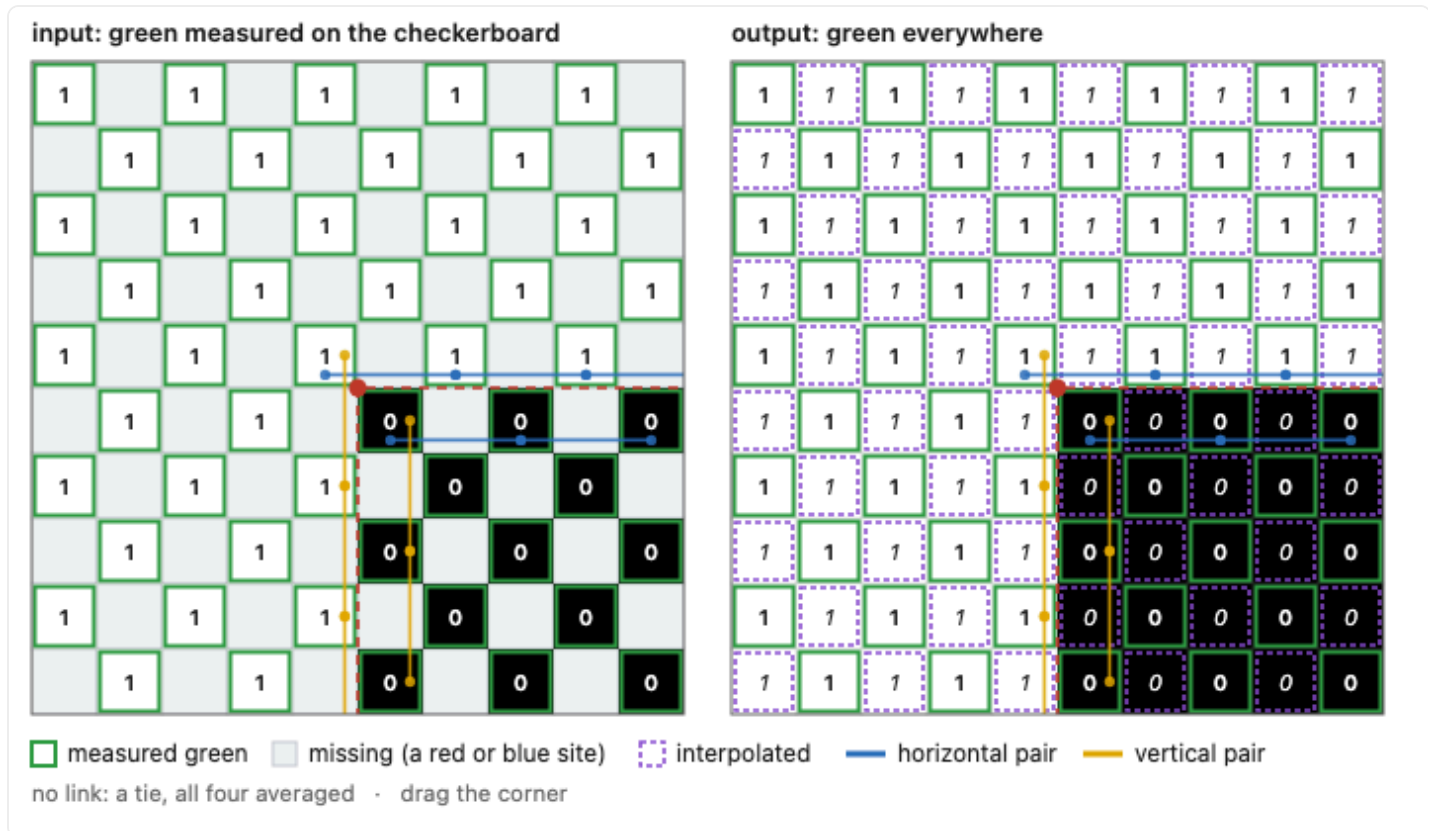


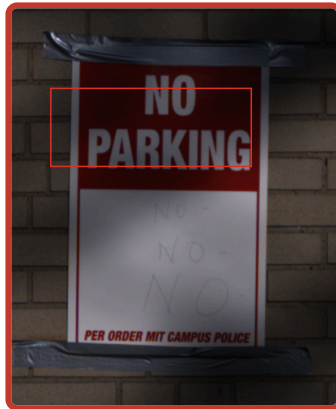
Figure 15.1.7 (reproduced). Edge-directed green, input beside output. Left, green as measured: the green-ringed checkerboard, and the red and blue sites where it is missing. Right, green everywhere: each missing cell is filled with the average of the green pair across which green changes least, along 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.

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

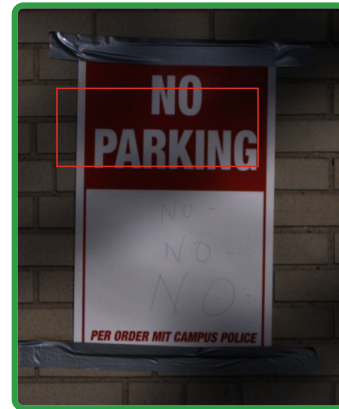
The whole sign



raw Bayer mosaic, one colour per pixel

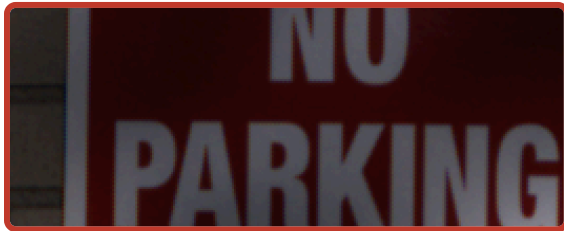


naive bilinear: false colour

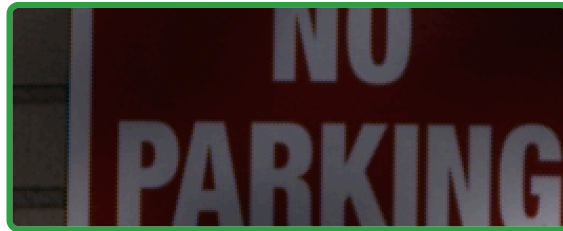


green-based: clean

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.

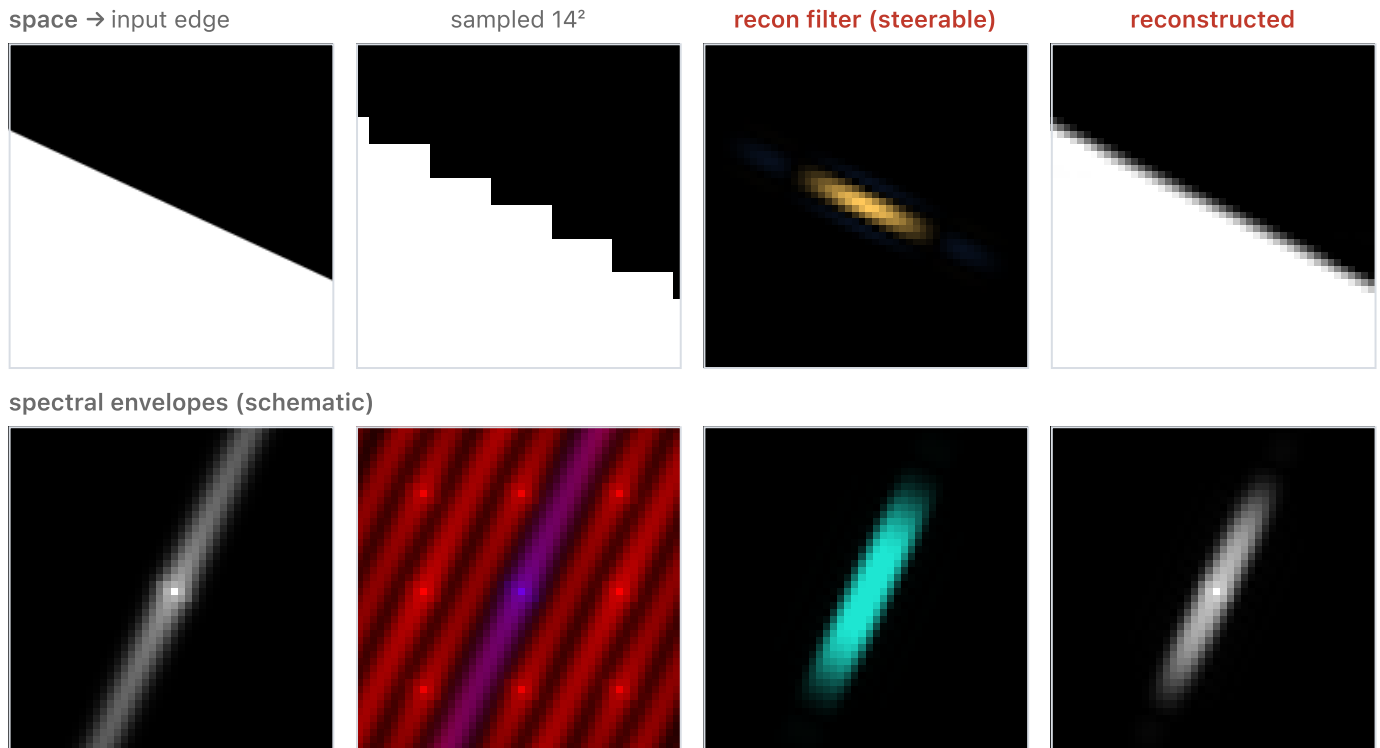


Figure 6.5.19 (reproduced). Steered reconstruction, from the resampling chapter. The reconstruction kernel is an ellipse you aim by hand: aligned along an undersampled edge, it removes most of the staircase; misaligned, it brings the aliasing back; a round kernel only blurs it. Edge-directed green interpolation 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 $\hat{R} = \hat{G} - (\hat{G} - R)$. 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).

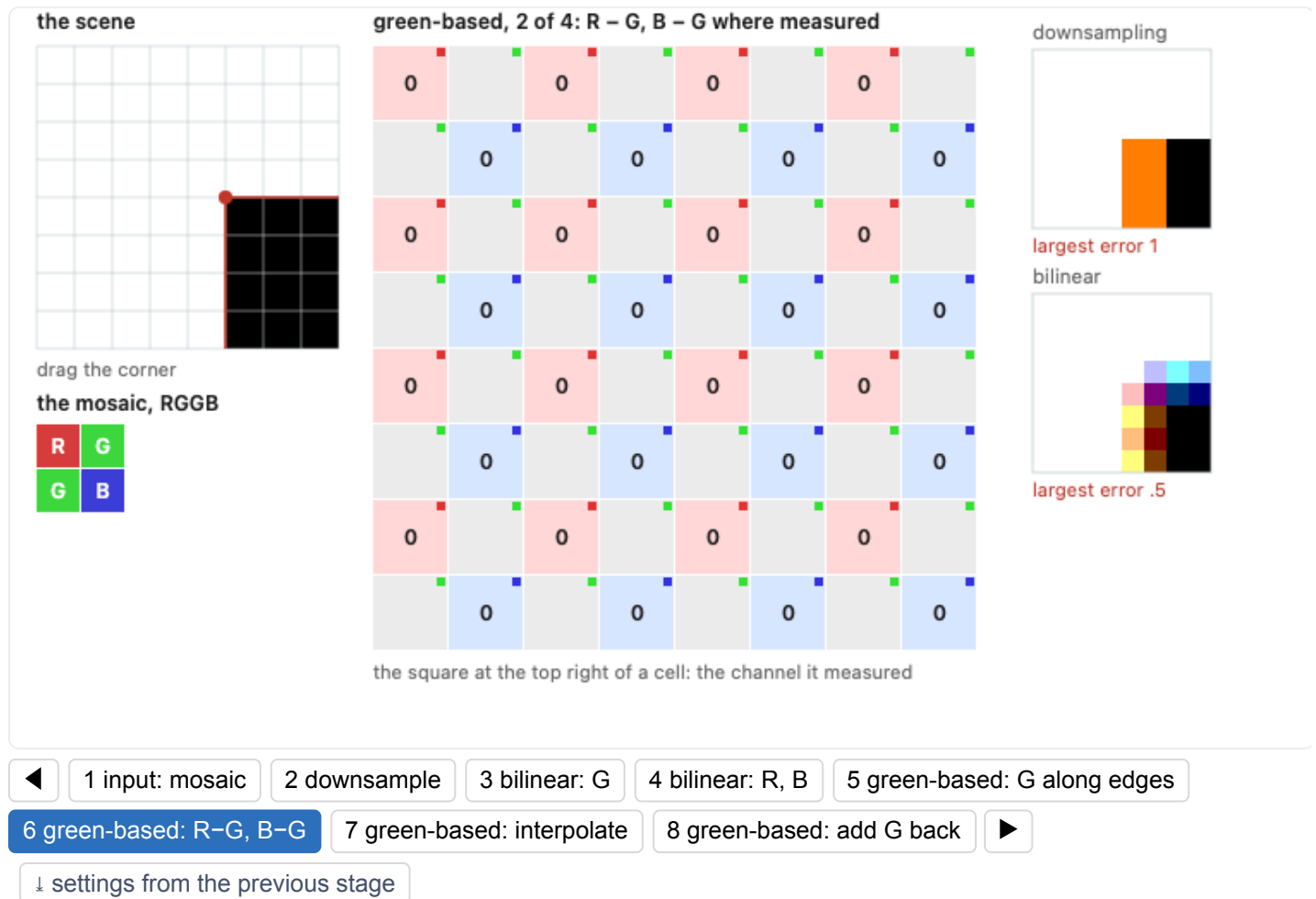


Figure 15.1.1, stage 6 ([reproduced](#)). The color differences $R - G$ and $B - G$ at the red and blue sites, against the edge-directed green. On the gray scene they are zero wherever that green is exact. Interactive: switch to a colored edge and the differences stop being zero.

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.

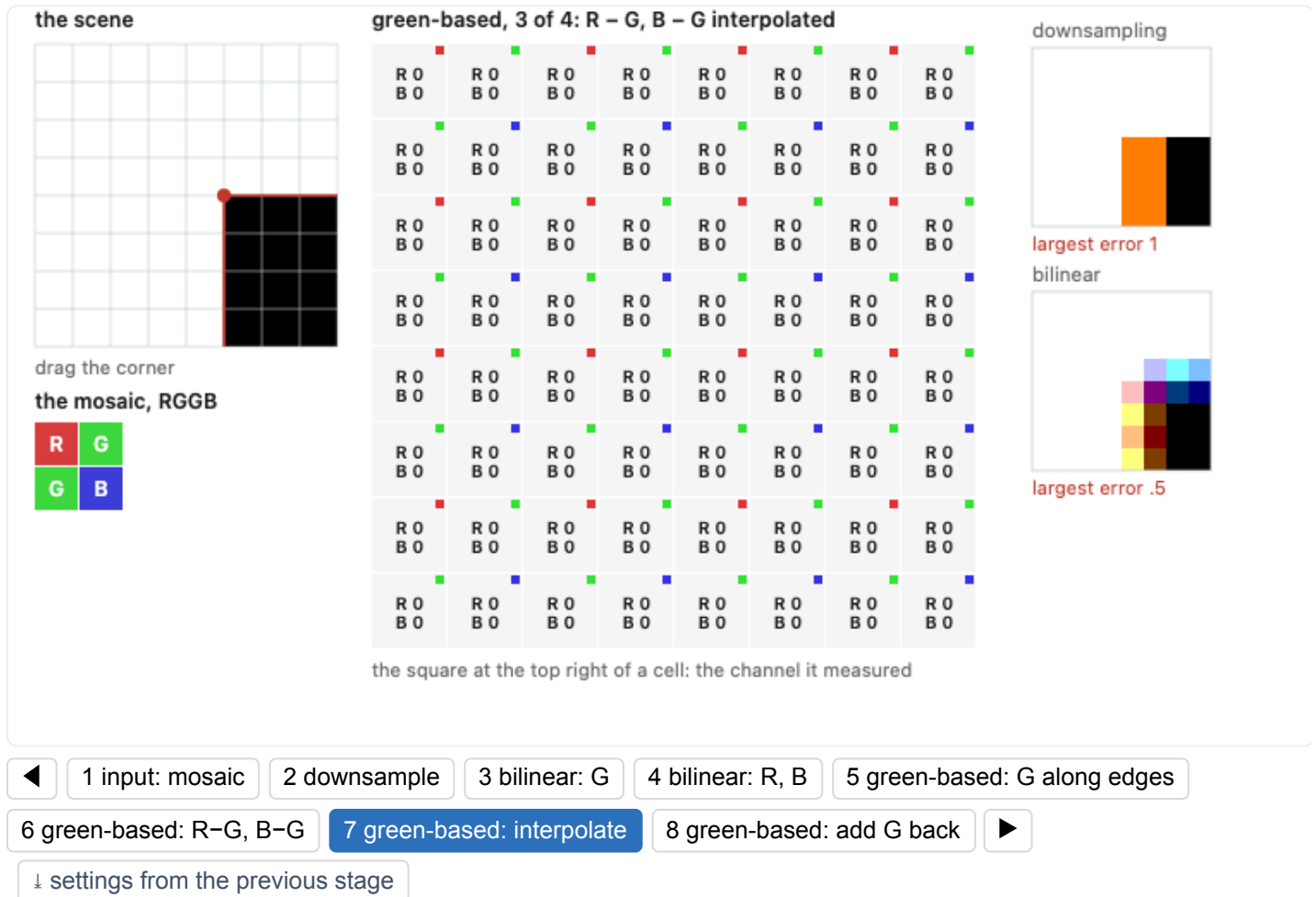


Figure 15.1.1, stage 7 (reproduced). The differences interpolated bilinearly to every cell, the same rule as bilinear green, now applied to a quantity that is nearly constant across the edge. Interactive: the same controls at every stage.

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.

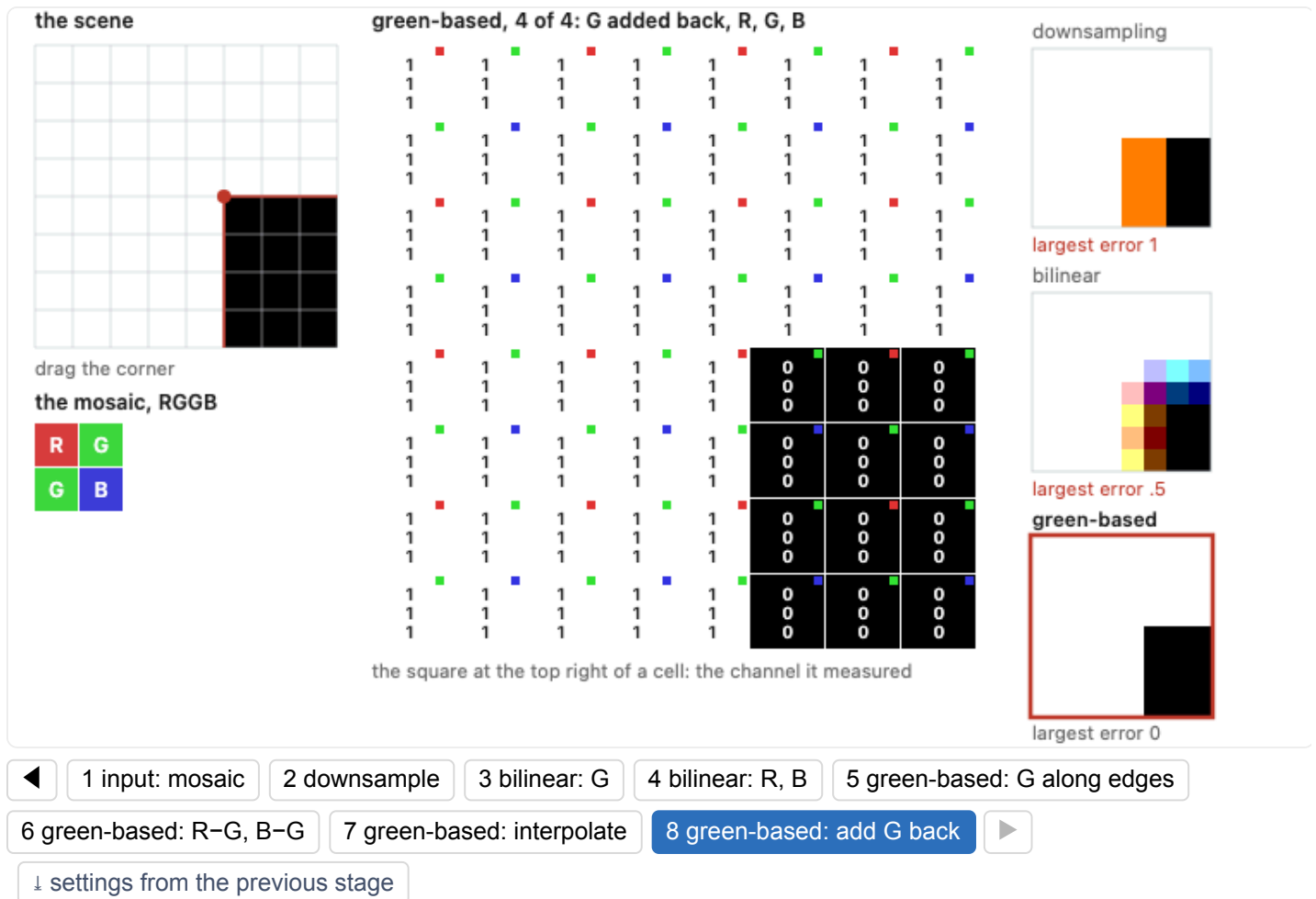


Figure 15.1.1, stage 8 ([reproduced](#)). Green added back, the end of green-based (color-difference) demosaicking. On the gray scene the differences are zero wherever green is exact, so the result is exact: largest error 0, against .5 for bilinear and 1 for downsampling. Drag the corner so its first 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.

Demosaicking a black-and-white target: per-channel interpolation invents colour at edges; green-based does not

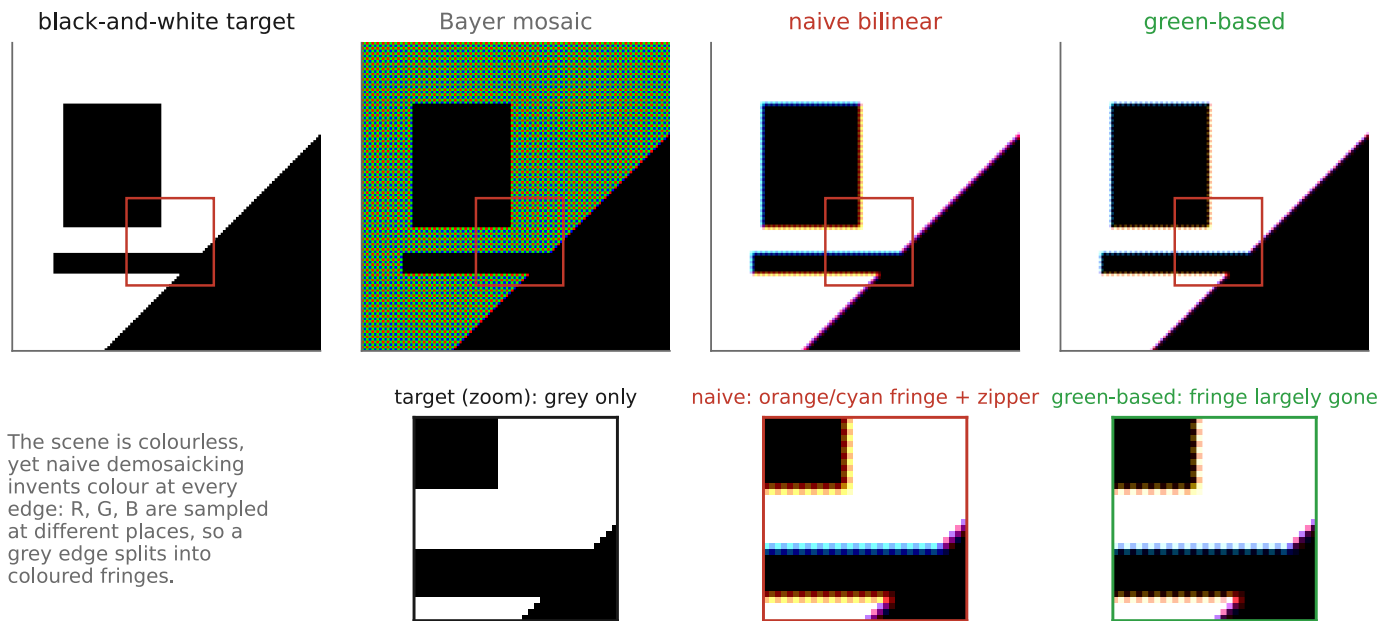


Figure 15.1.8 (reproduced). Color out of a black-and-white scene: the cleanest way to see the problem and the fix. The target is purely achromatic (black shapes on white), so a correct demosaicker must keep it gray. Naive bilinear does not: orange and cyan fringes appear along every edge, plus zipper 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.

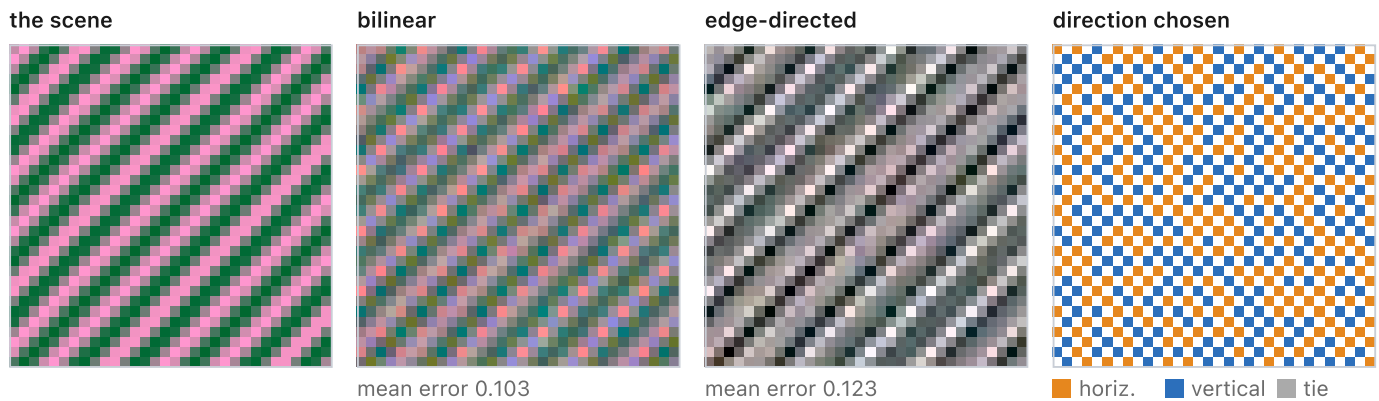
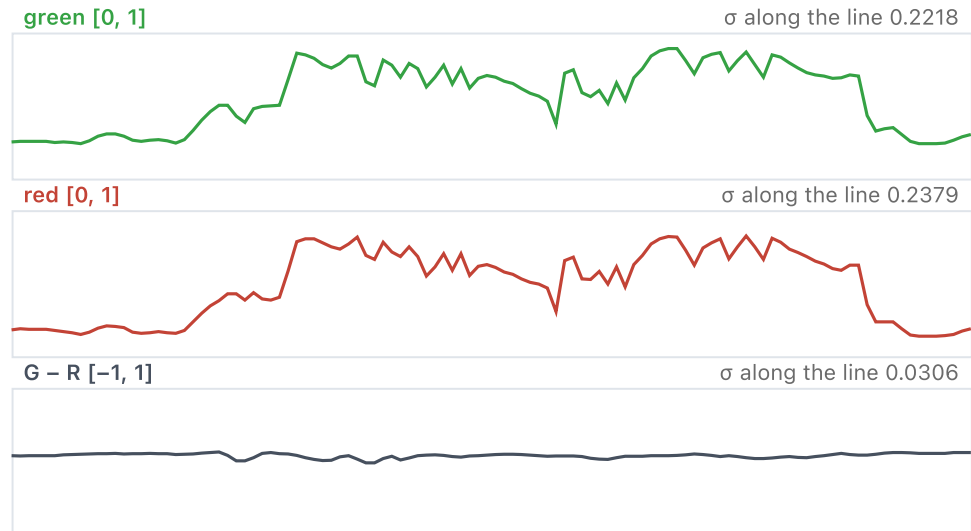
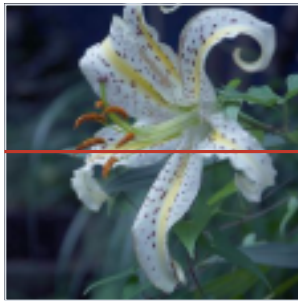


Figure 15.1.9 ([reproduced](#)). *Interactive. Fine colored stripes, 4 pixels apart at 45 degrees, kept through an RGGB mosaic with a little sensor noise, and reconstructed by bilinear interpolation and by edge-directed green (Hamilton and Adams' test, with the curvature correction) followed by color 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.*

inspect how colour channels and their difference vary

click to move the scanline



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.

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:

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

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

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, Ssstrunk & Hrault 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.

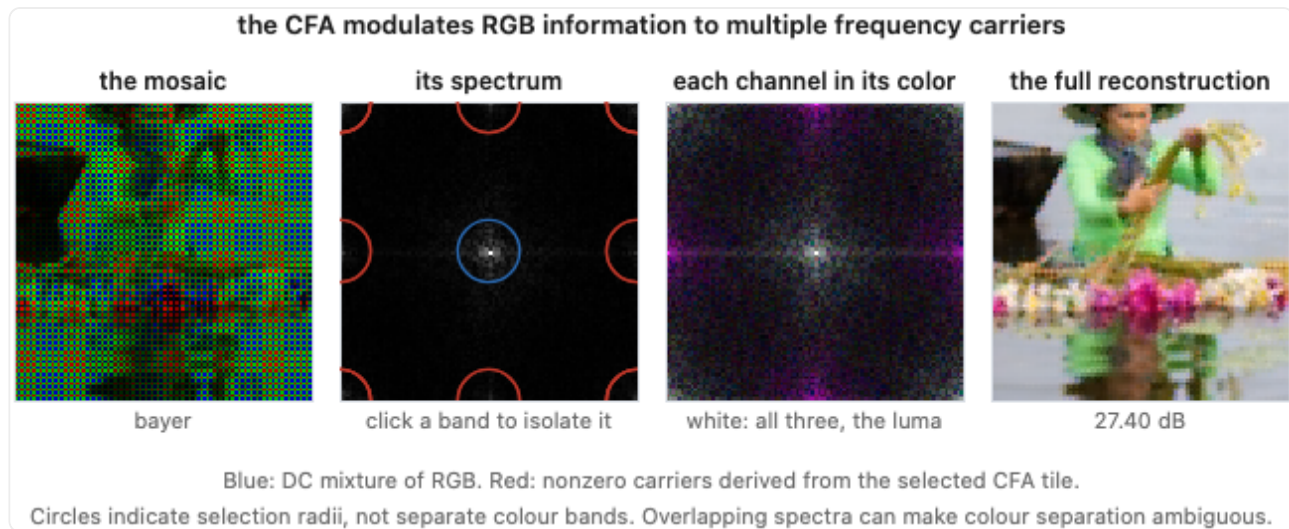


Figure 15.3.2 ([reproduced](#)). *Interactive. The spectrum of a mosaicked image, with the luminance band and the chrominance copies outlined and labeled; the copy positions are derived from the array's own tile rather than drawn by hand. A scene selector switches between a smooth image, where the copies 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.*

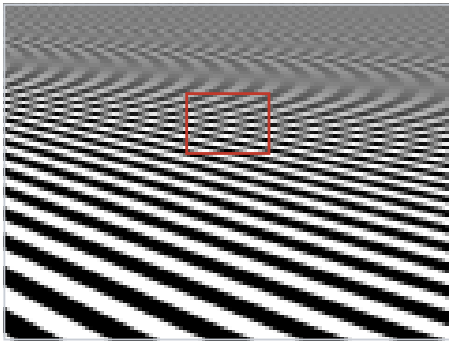
False color and moiré as spectral overlap

OPTIONAL
in the book [↗](#)

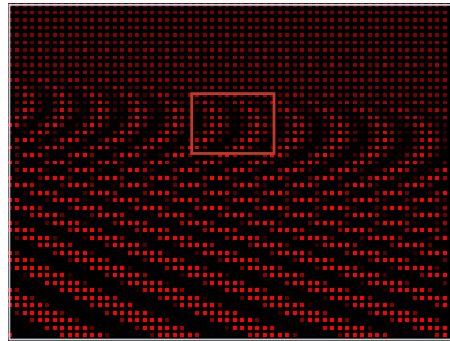
Aliasing on a mosaic is the luma band reaching a chroma carrier. Fine gray detail near $1/2$ cycle per pixel, a fence, a woven fabric, the fine print on a page, puts luma energy exactly where a demosaicker looks for chroma. The demosaicker cannot tell the two apart and reports a color the scene did not 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](#)).

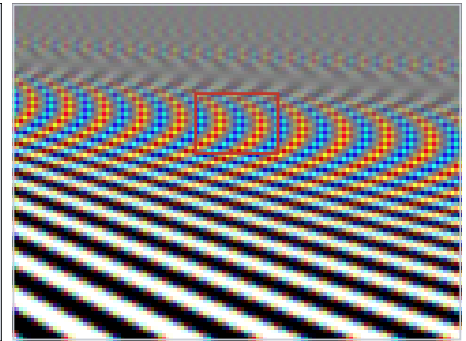
ground truth (three-chip)



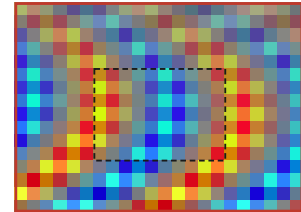
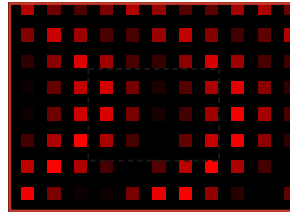
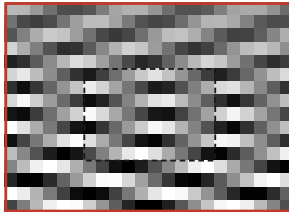
the mosaic: red samples



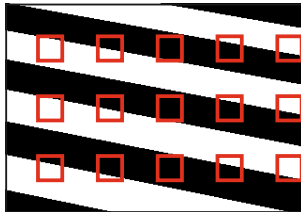
bilinear, mean error 0.106



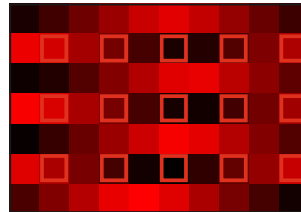
the red box, magnified 7x; drag on a picture to move it



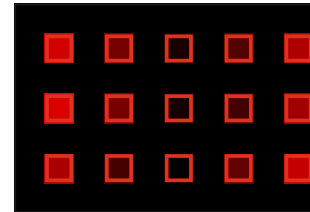
the pattern; edge: the red filters



the truth's red channel



what the mosaic keeps



the dashed box, magnified 16x

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

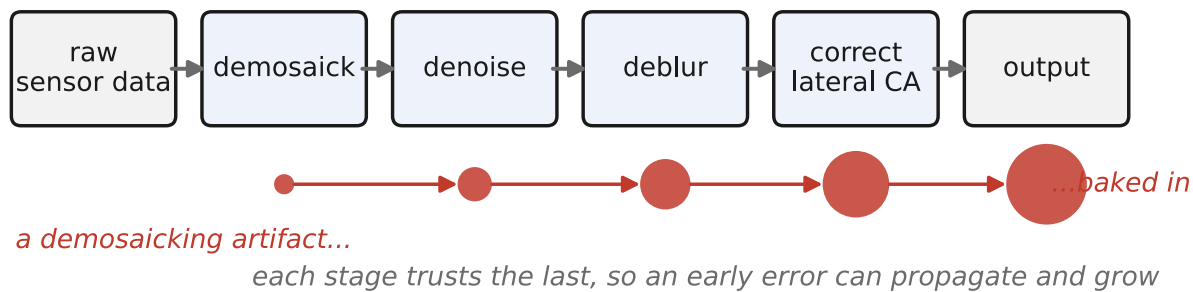
OPTIONAL
in the book [↗](#)

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 $A\Sigma A^T$. 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.

The pipeline: independent stages, errors can compound



Joint reconstruction: one optimization

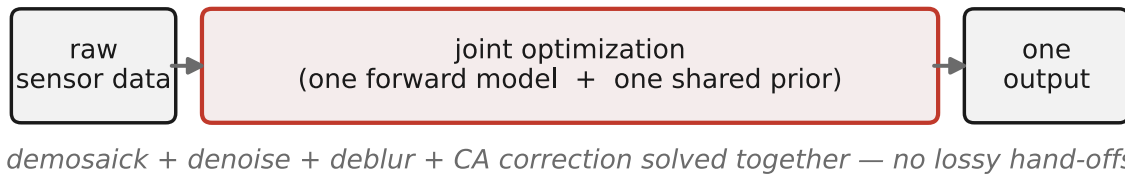


Figure 15.8.1 (reproduced). The pipeline versus the joint solve. Top: the classical ISP as a chain of independent stages (raw → demosaick → denoise → deblur → correct lateral CA → output), each designed assuming clean input from the last. A small error introduced at demosaicking (red marker) is trusted 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.

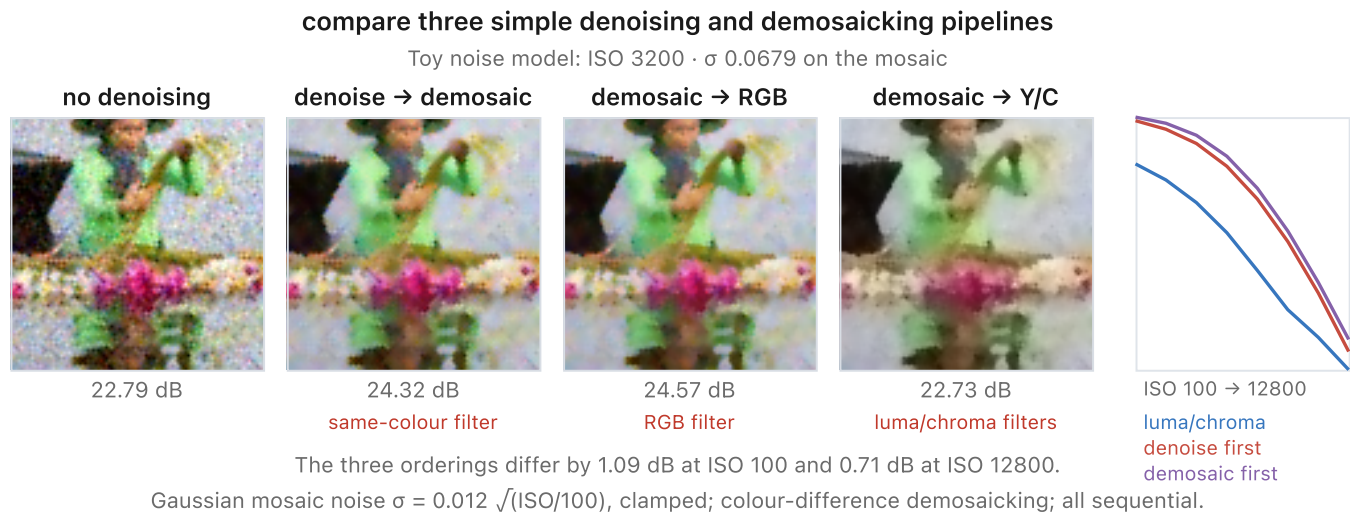
The two orderings, made visible

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

Hardware alternatives to the mosaic

Holography

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.