ACRONYMS — THIS CHAPTER

ACRONYMS

BM3D	Block-Matching and 3D filtering (denoising)
CC	CameraCalibration matrix (in the DNG pipeline); Creative Commons in a licensing context
CIELAB	CIE Lab color space (lightness L with a, b color-opponent axes)
DCT	discrete cosine transform
EV	exposure value
HDR	high dynamic range
NCC	normalized cross-correlation
NL	non-local (as in NL-means)
PSNR	peak signal-to-noise ratio
RMS	root mean square
SNR	signal-to-noise ratio
SSD	sum of squared differences
SSIM	structural similarity (index)
YUV	luma (Y) with blue- and red-difference chroma (U, V)

Course notes · 24 sections from the book · 15 optional hide optional sections the full book ↗

LECTURE NOTES 3 — NOISE, BILATERAL FILTER

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 · 17 sections on this page, from 10 chapters of the book

Noise, signal-to-noise ratio and dynamic range

Signal-to-noise ratio: it's about ratios

OPTIONAL
in the book ↗

A counterintuitive fact ties the whole part together. Since shot noise is $\sigma = \sqrt{N}$, the *absolute* noise is actually larger in the bright parts of the image than in the dark parts: a highlight collecting 10,000 photons has $\sigma = 100$, while a shadow collecting 100 photons has only $\sigma = 10$. And yet noise *looks* worst in the shadows. The resolution is

that noise visibility depends strongly on its size relative to the local signal, as well as on display brightness, spatial frequency, and image content. Signal-to-noise ratio (SNR) captures the first of these effects. For shot noise,

$$\mathrm{SNR} = \frac{N}{\sqrt{N}} = \sqrt{N}. \tag{1}$$

The SNR *grows* with brightness, so it is *worst where the light is faintest* (the shadows) even though the raw noise magnitude is smallest there (Figure 2.8.6). This emphasis on ratios also motivates gamma and log encoding (log encoding gives equal steps to equal ratios, while a power-law gamma curve approximates perceptual spacing over a useful range) and exactly why exposing to the right works: collecting more photons everywhere, especially in the shadows, raises N and therefore raises the SNR where it is worst.

One caution before we go on, because the acronyms collide: SNR is not PSNR. The signal-to-noise ratio here is a property of a *capture*, the true signal at a pixel relative to the sensor's noise there ($\sqrt{N}$ for shot noise), and it answers *how clean is this measurement?* Peak signal-to-noise ratio (PSNR), which you meet whenever an algorithm is *evaluated*, is a different animal: a full-reference quality metric that compares a processed image $\hat{I}$ against a ground-truth reference I, computed from their mean-squared error as

$$\mathrm{PSNR} = 10 \log_{10} \frac{\mathrm{peak}^2}{\mathrm{MSE}} \quad (\mathrm{dB}), \tag{2}$$

where peak is the largest possible value (e.g. often 1.0 for us when using floating point values, or 255 for 8-bit) and $\mathrm{MSE} = \frac{1}{n} \sum (\hat{I} - I)^2$. So SNR describes *a photograph* (how much noise a single measurement carries, with no reference needed), while PSNR scores *a reconstruction* against a known truth (how close a denoiser or codec got). Both compare a signal scale to an error scale, but their numerators and purposes differ. PSNR uses a fixed peak rather than the local signal. Maximizing expected PSNR through minimizing MSE favors a conditional-mean estimate, which can blur uncertain detail; that makes it useful for some comparisons but incomplete as a perceptual score.

A word on the **decibel** itself, since it recurs through this chapter and the next. A decibel is just a logarithmic way to write a ratio: for a *power* (or energy, intensity, variance) ratio it is $10 \log_{10}$, and for an *amplitude* (or voltage, signal) ratio it is $20 \log_{10}$, the factor of two coming from power $\propto$ amplitude^2. Photographers count the same ratios in **stops**, factors of two ($\log_2$), so one stop is about 6 dB of amplitude or 3 dB of power. Decibels, stops, plain linear ratios, and scientific notation are thus four scales for the same underlying number, a multiplicative factor, and it helps to move fluently between them (Figure 2.8.5).

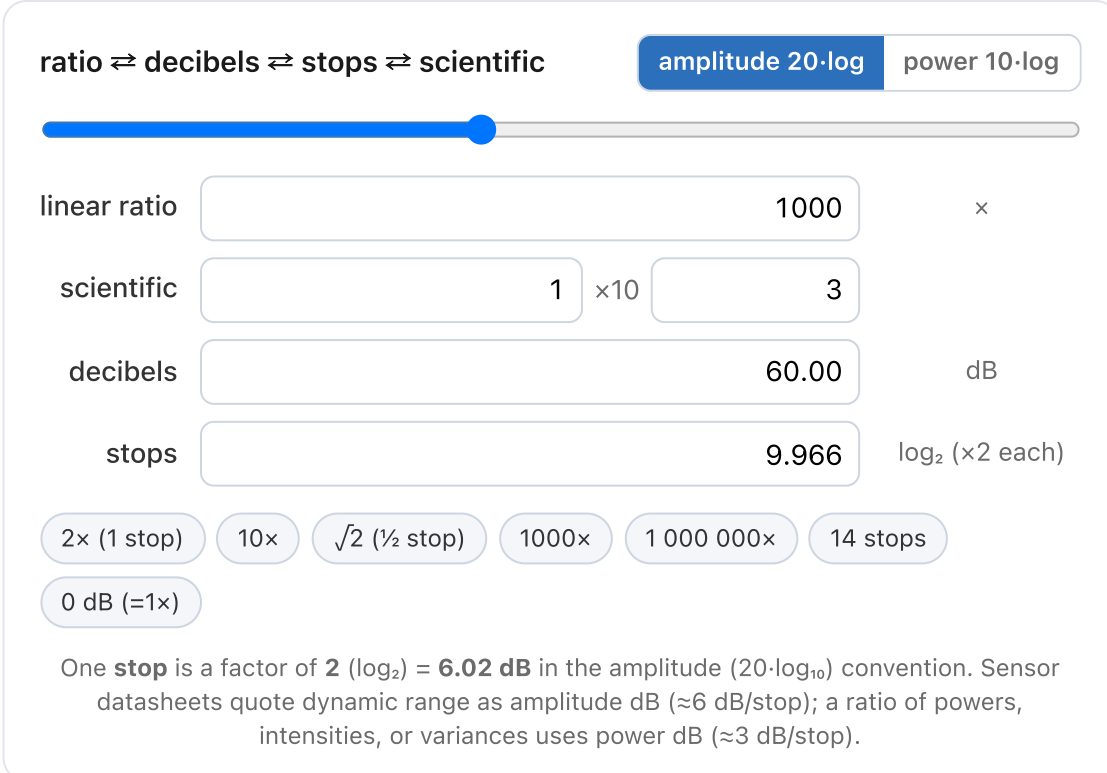


Figure 2.8.5 (reproduced). *A units converter for ratios (interactive). Decibels, photographic stops (log2), a plain linear ratio, and scientific notation are four ways of writing one multiplicative factor. Type into any field, or drag the log-progression slider, and the others update to match; a toggle switches the decibel convention between amplitude (*$20\log_{10}$*, about 6 dB per stop, the convention sensor datasheets use for dynamic range) and power or energy (*$10\log_{10}$*, about 3 dB per stop, for ratios of intensities or variances). Preset chips jump to the common landmarks: one stop, ten-fold, a million-to-one, a full-frame sensor's roughly fourteen stops.*

💡 **BIG LESSON (L2.41) — BRIGHT PIXELS ARE *NOISIER* (MORE ABSOLUTE NOISE) BUT HAVE *BETTER SNR***

Shot noise is Poisson, so its **variance grows linearly with the signal** ($\sigma^2 \propto N$) and its **standard deviation grows only as the square root** ($\sigma \propto \sqrt{N}$). Two consequences pull in opposite directions: the *absolute* noise is **larger in highlights** than in shadows (a 10,000-photon highlight has $\sigma = 100$; a 100-photon shadow has $\sigma = 10$), yet the **SNR** $= N/\sqrt{N} = \sqrt{N}$ **is higher in the highlights**: the signal outgrows the noise. So bright areas are noisier in absolute terms but **cleaner to the eye**; the shadows have the worst SNR (Figure 2.8.4, Figure 2.8.6). This is why **exposing to the right** and brighter scenes look smoother, and why we **gamma/log-encode** so equal *ratios* get equal steps.

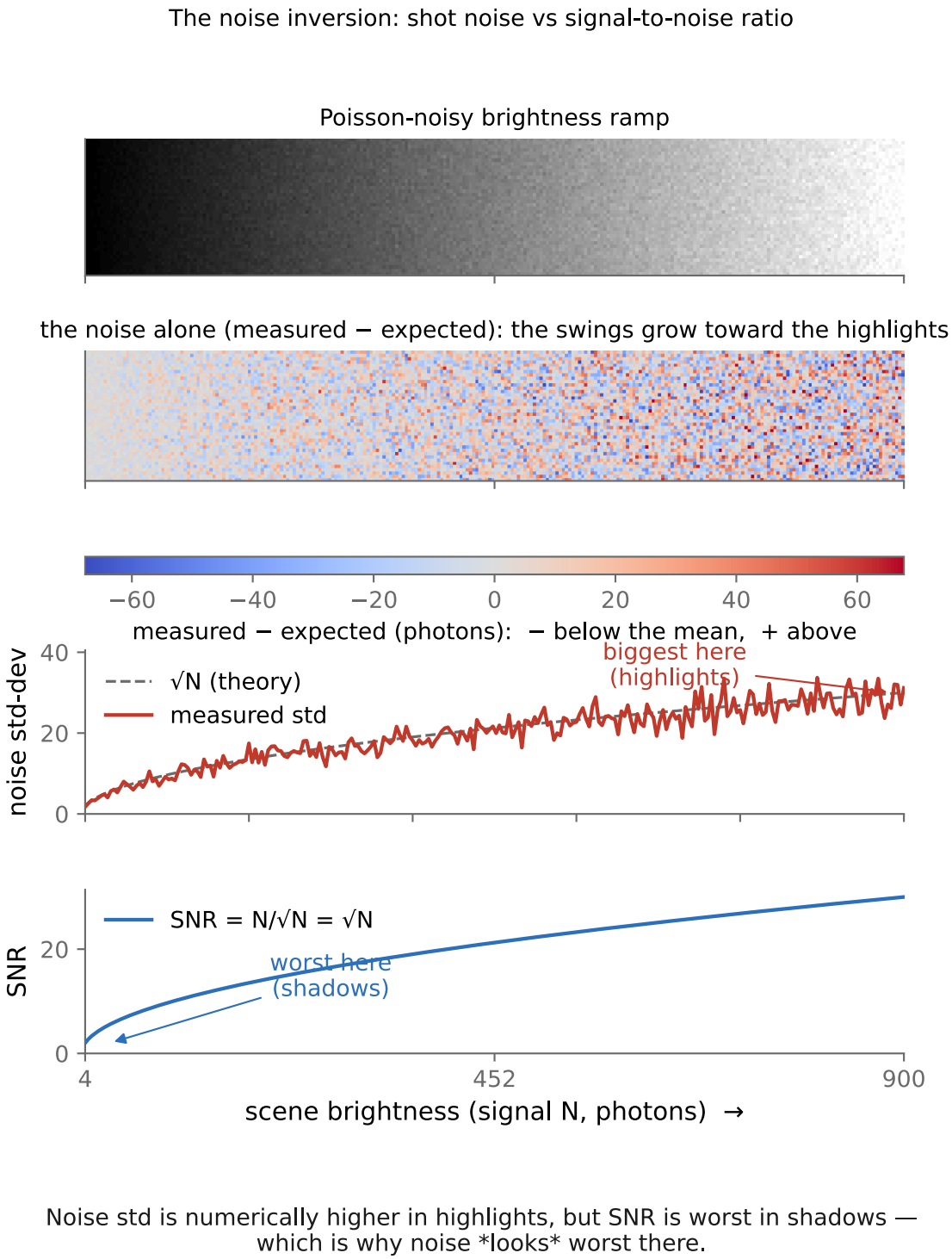


Figure 2.8.6 (reproduced). *Noise is numerically worse in highlights but looks worse in shadows. Three panels of the same brightness ramp, each labeled so the good direction is unambiguous. A standard-deviation map (absolute noise, higher* = worse*) is* brightest in the highlights*, since shot noise* $\sigma = \sqrt{N}$ *grows with brightness; an SNR map (* $= \sqrt{N}$*, higher* = better*) is* worst in the shadows*, since the ratio falls where photons are few; and a* ***noise-only*** *panel shows a single frame's residual, the measurement minus its expected value, on a diverging colormap (blue for negative excursions, red for positive, white at zero) so both signs read at once, the residual staying near zero-mean everywhere yet visibly growing in amplitude toward the highlights. Perception tracks the ratio, so shadows look noisiest, the reason for exposing-to-the-right and for gamma/log encoding. All three maps are computed in linear light and gamma-encoded only for display, so the noise magnitudes shown are faithful.*

The affine model, whose full story is the next section, settles a lot of everyday "will this help my noise?" questions at a glance. Read each move against $\sigma^2 = g\,I + \sigma_r^2$ (signal I, photon-gain slope g, read-noise floor σ_r), tracking what happens to the signal, the noise σ, and their ratio. The entries below are for the shot-noise-dominated regime (well-lit midtones); the read-noise term σ_r changes only the two rows that mention it.

scenario	signal	noise σ	SNR	what is really going on
raise ISO by 1 stop (×2 analog gain)	×2	×2	≈ same	amplifies signal and noise together and adds no photons. Read noise, applied *after* the gain, shrinks relative to the signal, so deep shadows come out a touch cleaner. Not a substitute for light.
brighten by 1 EV in software (×2)	×2	×2	unchanged	multiplies values already recorded, so it amplifies the noise baked in at capture. No shadow benefit, unlike raising ISO at capture.
darken by 1 EV in software (×½)	×½	×½	unchanged	pure scaling; the ratio is untouched (though it can still clip or quantize).
average N frames	×1	$\div\sqrt{N}$	$\times\sqrt{N}$	independent noise partly cancels while the signal stays: the $1/\sqrt{N}$ law behind burst denoising and stacking.
sum N values	$\times N$	$\times\sqrt{N}$	$\times\sqrt{N}$	variances add, so SNR grows just like averaging; but read noise is paid N times if the values are read separately.
charge-bin N pixels (combine, then read once)	$\times N$	$\times\sqrt{N}$	$\times\sqrt{N}$, cleaner shadows	same shot noise as a digital sum, but read noise is paid *once*, not N times. Check that the sensor actually combines charge before separate readout.
gather 2× the light (wider aperture or longer exposure)	×2	$\times\sqrt{2}$	$\times\sqrt{2}$	collects more photons per measurement, improving SNR before any reconstruction. Read noise unchanged.

Software rescaling preserves SNR. Analog gain can reduce the relative contribution of noise added downstream, as the ISO section explains. Combining N equally exposed independent measurements improves SNR as $\sqrt{N}$, while more photons per measurement improve the capture itself.

Image metrics

PSNR

OPTIONAL
in the book ↗

Peak signal-to-noise ratio expresses the same error in decibels:

$$\mathrm{PSNR} = 10 \log_{10}\left(\frac{L^2}{\mathrm{MSE}}\right). \tag{3}$$

Here L is the fixed reference range: 1 for normalized samples in $[0, 1]$, or 255 for 8-bit samples. It is not each image's measured maximum. For HDR values without a shared finite range, specify a normalization or use a measure designed for the intended luminance conditions. PSNR is not the sensor SNR discussed in Chapter 2.8, Noise, signal-to-noise ratio and dynamic range: the denominator here includes any discrepancy from the reference, including bias, blur, and registration errors.

Higher PSNR means lower MSE at the same L; zero error gives infinity. Halving MSE adds about 3 dB. There is no universal threshold for an acceptable image. Scores depend on the scene, encoding, crop, and degradation.

Be explicit about aggregation. Averaging image PSNRs differs from pooling squared errors and converting once, because the logarithm is nonlinear. For equal-sized images with MSEs 0.001 and 0.01 at $L = 1$, the mean PSNR is 25 dB, while the PSNR of their mean MSE is about 22.60 dB. Report the convention and inspect the distribution, not only its mean.

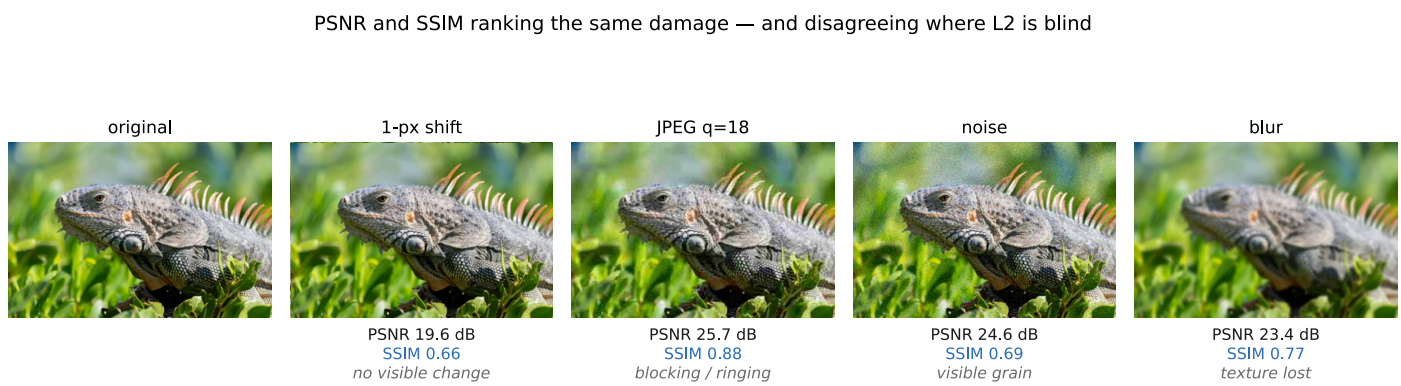


Figure 9.3.1 (reproduced). *PSNR and SSIM ranking the same damage, and disagreeing exactly where L2 is blind.* One clean photo and four common degradations of it (a 1-pixel shift, a low-quality JPEG, additive noise, and a Gaussian blur), each labeled with its PSNR and its SSIM. The shift looks identical to the original yet scores the lowest PSNR; the blur softens every scale of texture yet keeps a respectably high PSNR. SSIM, comparing local structure rather than raw pixel values, rates both far closer to how they actually look; the disagreement is the whole point.

Sidebar 1: error depends on color space

Linear RGB is useful for radiometric fidelity; a specified output encoding measures errors after rendering. Color differences such as ΔE in CIELAB address a different question, under a stated white point and viewing model; see Chapter 4.5, Advanced color: opponents, perceptually-uniform, and preferred. They do not account for spatial structure or automatically make an image-quality benchmark better. Do not apply an RGB PSNR peak to Lab channels and call the result comparable.

Bilateral filtering

in the book ↗

📎 **Problem set**

PS2 implements the bilateral filter (with a YUV chroma variant). → Chapter 1.5, Problem sets (appendix).

A blur mixes two sides of an edge and makes it soft. That is useful for some operations, but harmful when we want to remove noise while preserving sharpness or to smooth texture while keeping the boundary. The big idea of the bilateral filter is to only take into account pixels that are similar enough. This notion of affinity between pixels will come back over and over for edge-preserving techniques.

We illustrate bilateral filtering for the task of denoising, but it also underlies many algorithms because it enables the factoring of images into large-scale structure and local detail while preserving strong edges. This proves powerful for applications such as high-dynamic-range tone mapping (Chapter 18.3, Two-scale and bilateral tone mapping), upsampling low-resolution results, and combining images with different illumination, where the edges come from a second image (#Cross / joint bilateral).

Debugging: hand-checkable limits

The bilateral has two limits you can predict by hand, and together they make an excellent correctness test for an implementation (Figure 10.1.7). Push the range width to each extreme:

- Large range variance ($\sigma_r \to \infty$): the range Gaussian flattens, $g_r \to 1$, so it does nothing: every value difference counts as "similar." The weight reduces to the spatial Gaussian alone, and the bilateral collapses to a plain Gaussian blur.
- Tiny range variance ($\sigma_r \to 0$): only neighbors with the same value retain weight in the limit. They may include many pixels, but averaging equal values leaves the output unchanged. Averaging those equal values changes nothing, and the output is the identity: the image comes back untouched, edges and all.

A middle σ_r interpolates between these two: enough to smooth away noise and texture, not so much that it bleeds across real edges. If your implementation does not reproduce a Gaussian at large σ_r and the identity at small σ_r, it is wrong. These are the first two things to check (and the recipe in the debugging sidebar above).

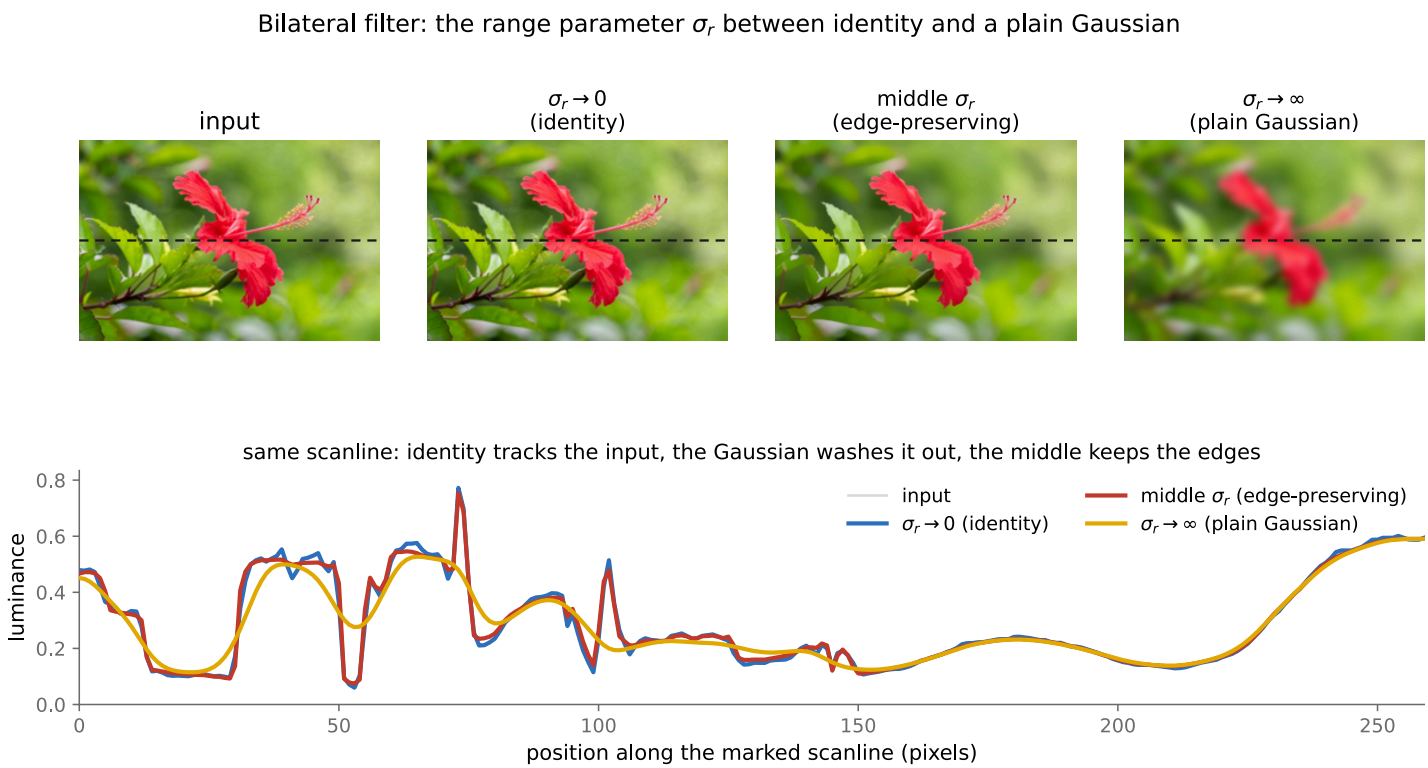


Figure 10.1.7 (reproduced). *The two debugging limits. The same input filtered across a sweep of range widths σ_r. At large σ_r the range weight does nothing and the bilateral is a plain Gaussian blur (edges washed out); at tiny σ_r only the pixel itself counts and the output is the identity (input untouched); in between, the filter smooths within regions while holding edges. The scanline plot underneath makes it concrete: the identity tracks the input exactly, the Gaussian washes it out, and a middle σ_r denoises the plateaus while keeping the step.*

The same two knobs guide tuning. Pick σ_s from the spatial scale of the structure you mean to remove (how large a region should be pooled). Pick σ_r from the contrast of the edges you mean to preserve: set it just below the value jump of a real edge, so the filter smooths everything gentler than that and stops at anything sharper. And state the encoding first: the same numeric σ_r means something completely different in linear light, in gamma, and in log luminance.

Bilateral filter

in the book ↗

Consider an input signal composed of a step edge with noise (Figure 10.1.1). The goal of bilateral filtering is to remove the noise by averaging nearby pixels without blurring the edge.

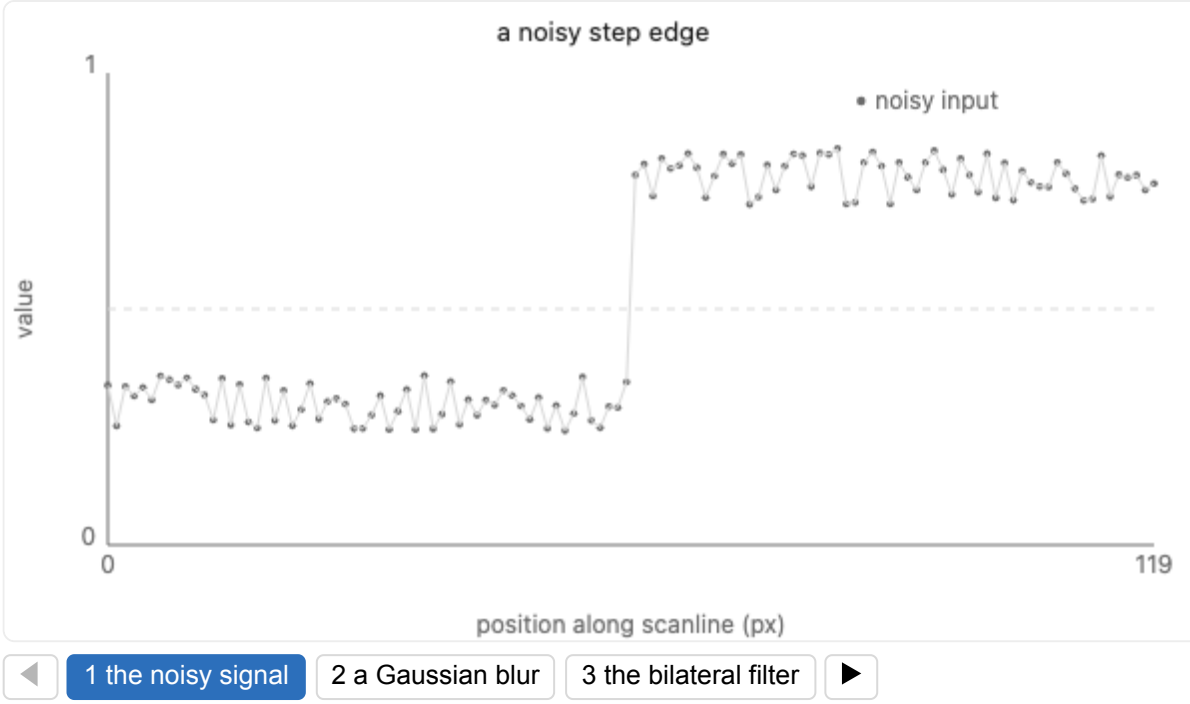
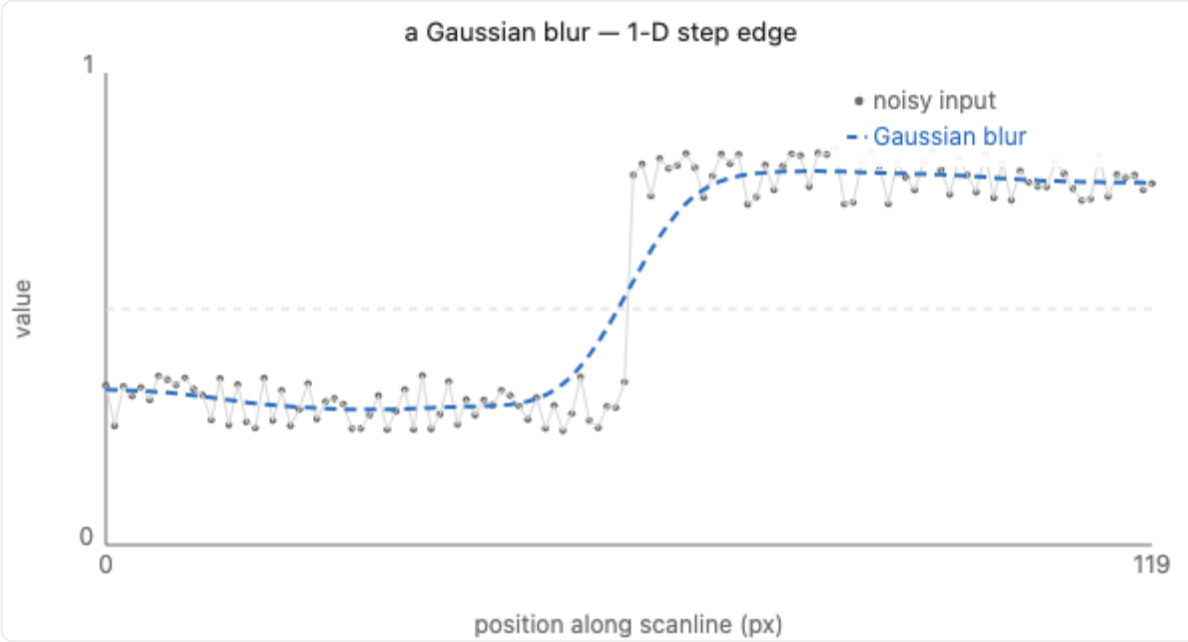


Figure 10.1.1 (reproduced). *The bilateral filter on a 1-D scanline, in three stages (interactive).* The figure builds the argument of this section one step at a time: first the noisy step edge, one row of an image; then a plain Gaussian, which averages across the step and smears it into a ramp; then the bilateral, which denoises each plateau and keeps the step crisp, because at the edge the range weight kills every neighbor on the far side. Each stage brings the controls that matter at that point, σ_s with the blur and σ_r with the bilateral.

Start from the Gaussian blur. It replaces each pixel with a weighted average of its neighbors, where the weight only depends on distance. It is a spatial Gaussian $g_s(\|p - q\|)$ of the distance between the center pixel p and a neighbor q: nearby pixels count for a lot, distant ones taper off. It is easiest to watch this on a 1-D scanline (a single row of the image), where the picture becomes a wiggly curve of value against position (Figure 10.1.1).



◀ 1 the noisy signal | 2 a Gaussian blur | 3 the bilateral filter ▶ | ↓ settings from the previous stage

Figure 10.1.1, stage 2 (reproduced). *The same figure at its second stage: a plain Gaussian blur of the noisy scanline.* The blur cleans both plateaus, and at the step it averages across the boundary, so the edge arrives as a ramp. σ_s sets how wide that reach is.

The trouble, once more, is the edge. A step edge on the scanline, with values low on the left and high on the right, gets *smeared* by the Gaussian, because right at the step the averaging window reaches across the boundary and blends the two plateaus together. The same bleed, in two dimensions and on a compressed base layer, is what paints halos along high-contrast edges in tone mapping (Chapter 18.3, Two-scale and bilateral tone mapping).

The fix is a single extra factor. Keep the spatial Gaussian, but multiply in a second Gaussian, this one on the value difference between the center pixel and the neighbor, $g_r(|I_p - I_q|)$. This range weight asks "are these two pixels similar in value?" and down-weights neighbors that are too different. A neighbor that is close in space but far in value, one on the *other* side of an edge, now earns a weight near zero, however spatially near it sits. The filter, all by itself, declines to average across the edge.

A practical wrinkle follows. The range weight asks whether two pixels are *similar*. But on a badly corrupted image the raw values are an unreliable judge: noise can make two same-region pixels look different, or two different ones look alike, so the weights themselves come out noisy. The weights need not be read from the original image at all. We can compute g_r from a pre-cleaned version (a quick median filter, say, or any cheap denoise) and then apply those steadier weights when averaging the *original* pixels. Decoupling *what decides the affinity* from *what actually gets averaged* is exactly the move the cross / joint bilateral (below) makes with a separate guide image; using a median-filtered copy of the same image as that guide is just the in-place version of the trick.

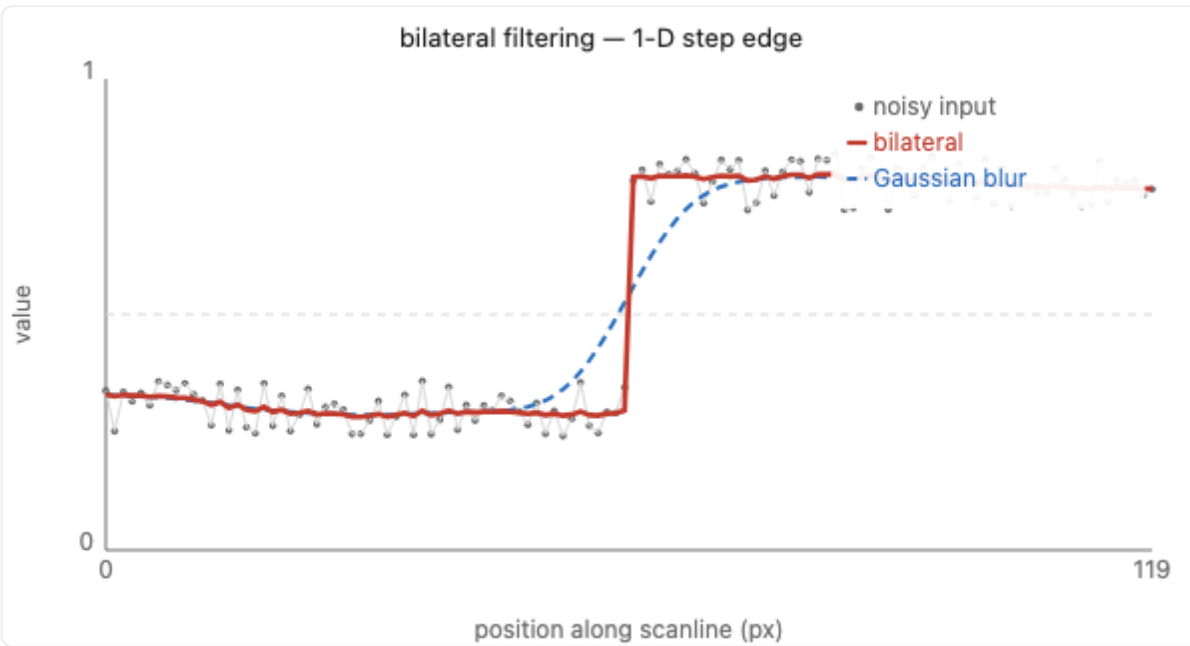
Putting the two Gaussians together gives the bilateral filter (Tomasi & Manduchi 1998). The weight a neighbor q receives at pixel p is the product of the spatial and range terms,

$$w(p,q) \;=\; g_s\big(\|p-q\|\big)\; g_r\big(|I_p - I_q|\big), \qquad g_s(d) = e^{-d^2/2\sigma_s^2}, \quad g_r(\Delta) = e^{-\Delta^2/2\sigma_r^2}, \tag{4}$$

and the output is the normalized weighted average of the neighbors,

$$I_p^{\mathrm{bf}} = \frac{1}{W_p} \sum_q w(p,q)\, I_q, \qquad W_p = \sum_q w(p,q). \tag{5}$$

In words: *"average my neighbors, but only those that are both nearby and similar to me."* The two widths do different jobs. σ_s, in pixels, sets how large a region we pool over, the reach of the spatial Gaussian. σ_r, in intensity units, sets what counts as "too different": the value gap beyond which a neighbor stops counting. Because σ_r is measured in the same units as I, we must fix the encoding before we can quote it: for HDR tone mapping the filter runs on log luminance (Durand & Dorsey 2002, where a range $\sigma_r \approx 0.4$ in $\log_{10}$ is typical); other tasks may use linear values or a specified output encoding. There is no single required domain for every bilateral application.



◀ 1 the noisy signal | 2 a Gaussian blur | 3 the bilateral filter ▶ | ↓ settings from the previous stage

Figure 10.1.1, stage 3 (reproduced). *The same figure at its third stage: the bilateral against the Gaussian.* Both plateaus are as clean as the blur made them, and the step survives, because for a pixel beside the edge the range weight has already set every neighbor on the far side to nearly zero. σ_r is what counts as too different.

As code, the brute-force bilateral is a double loop with a per-pixel normalization, and one subtlety distinguishes it from convolution: you may precompute the spatial weight g_s (it is the same window everywhere), but the range weight g_r must be recomputed for every pair, because it depends on the value difference at that pixel. The range function can be tabulated for quantized differences; it is the resulting image-dependent kernel that cannot be reused as one convolution. Specify spatial truncation and boundary handling, include the center sample, and accumulate in a sufficiently wide type.

Algorithm 1

```
precompute g_s over the window          # spatial weights, same for every pixel
for each output pixel p:
    sum ← 0;  W ← 0                      # accumulator and normalizer
    for each neighbour q in the window around p:
        w ← g_s(‖p − q‖) · g_r(|I[p] − I[q]|)   # range term recomputed per pair
        sum ← sum + w · I[q]
        W   ← W + w
    I_bf[p] ← sum / W                    # per-pixel normalization
```

For a color image the range distance $|I_p - I_q|$ is the 3-D distance in RGB (or YUV); everything else is unchanged.

The filter also travels under other names in editing software. Photoshop's Surface Blur is a bilateral filter: its *radius* is the spatial width and its *threshold* is the range width, over a box spatial kernel rather than a Gaussian. The name is worth knowing: those two sliders are the two widths of this section, and a tool many readers already use is a direct way to feel what each one does.

Sidebar 2: debugging the bilateral

Two extreme settings of the range width σ_r give two predictable outcomes that bracket the whole filter, the fastest way to sanity-check an implementation (the recipe also lives in *Developing, testing & debugging*). Figure 10.1.7

- Crank σ_r very large. The range term flattens, $g_r \to 1$, and drops out of the product. The bilateral must collapse to a plain Gaussian, so run it side by side with your already-tested Gaussian blur; they should agree pixel for pixel.
- Shrink σ_r very small. Now a pixel averages only with near-identical neighbors, so edges stay untouched and the output is essentially the input. Run it on a synthetic half-plane or rectangle (one hard edge): the bilateral must leave that edge razor-sharp.

If either limit misbehaves, the bug is in the range term or the normalization; fix that before trusting any middle σ_r.

The 1-D picture carries straight to the 2-D image: the bilateral smooths flat texture and noise inside each region while holding the edges, so a portrait emerges with clean skin and intact eyelashes, and an HDR base layer stops cleanly at the silhouette instead of ramping across it (Figure 10.1.2).

Bilateral filtering smooths texture within regions but does not average across edges

Figure 10.1.2 (reproduced). *The bilateral filter on a real photograph. Left, the input: a white egret against busy foliage. Center, a plain Gaussian blur of the same radius: it smooths the foliage texture but also smears the bird's edges into the background. Right, the bilateral filter: texture and noise are smoothed away within each region while the strong edges, the bird's silhouette and the beak, stay crisp. The same behavior as the 1-D scanline, now on a 2-D image.*

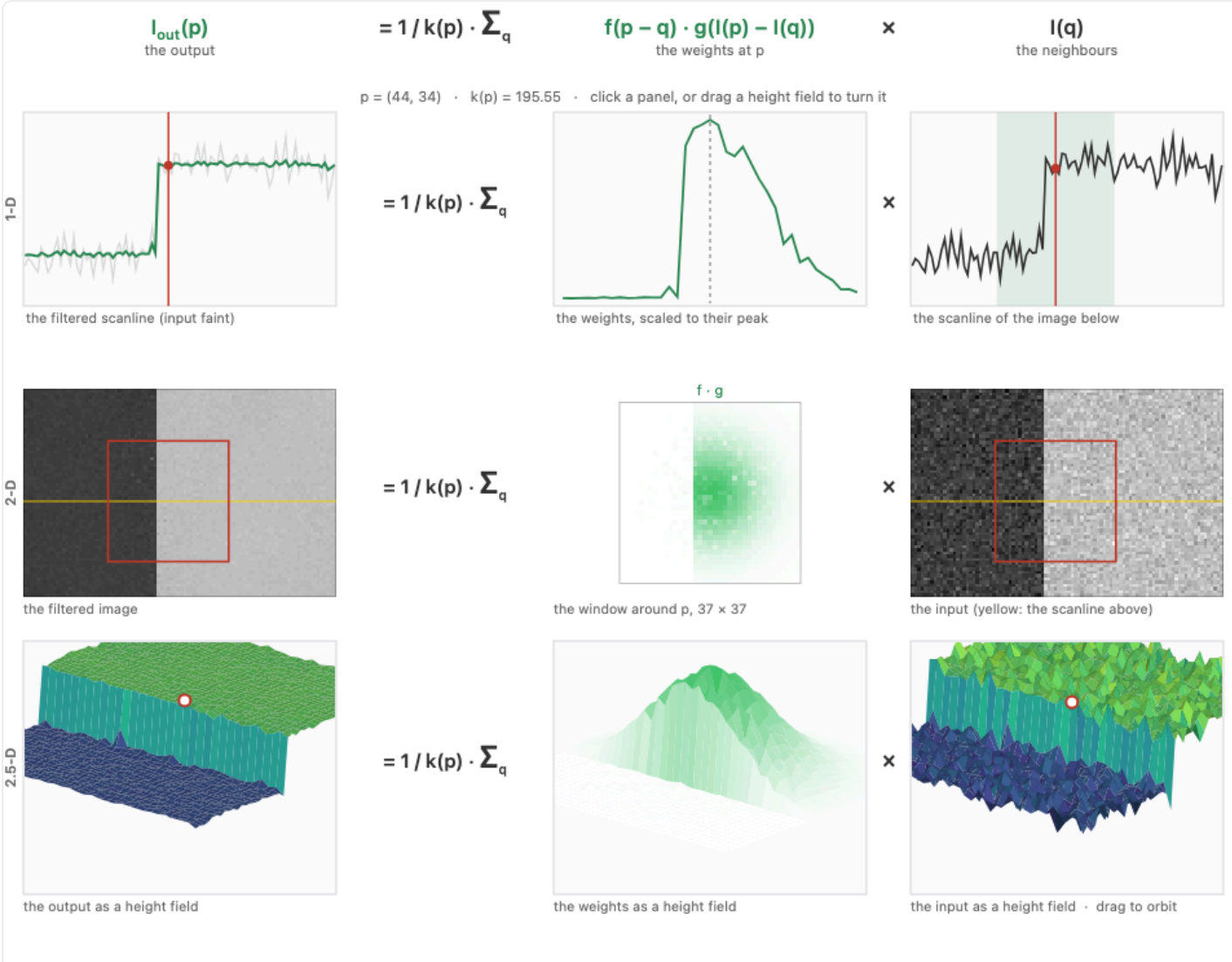


Figure 10.1.3 (reproduced). *One filter, three views, and the equation it comes from (interactive).* The same data is drawn as a 1-D scanline, as the 2-D image that scanline is a row of, and as a height field, where the picture is a surface and the filter's weights are a second surface sitting over it. Each row reads left to right in the order of the equation at the top: the output, the weights at the selected pixel, then the input. The weights can be split into the spatial Gaussian f and the range Gaussian g, which is where the height field earns its place: f is the same hump everywhere, g is a cliff that follows the image, and their product is the kernel actually applied, different at every pixel. Step the selected pixel across the edge, or let it sweep, and watch that kernel lose the far side. The filter can be switched to a plain Gaussian, which drops g, or to the unnormalized form that filters the difference from the center pixel and never divides. Interactive: choose a noisy step edge (its height and noise) or a photograph, set σ_s and σ_r, split the weights, switch the filter, and move the selected pixel by clicking, stepping or animating.

The range Gaussian g_r is more than a clever weight; it is a similarity, an affinity between two pixels: a number that says *how much these two pixels belong together*, computed from their value difference. The bilateral averages pixels in proportion to that affinity. Every later method in this chapter, and several beyond it, is a different answer to one question: *how should we measure whether two pixels belong together?*

BIG LESSON (L10.1) — EDGE-PRESERVING = AFFINITY

Use the color / intensity difference between two pixels as how much they *belong together*, an affinity, then average (or connect) pixels in proportion to it. Pixels with high affinity are treated as measurements of the same underlying thing and get averaged together; pixels with low affinity, across an edge, are kept apart. The bilateral filter's range weight g_r is the first full instance, but the *same* affinity drives the bilateral grid, the guided filter, non-local means (NL-means), colorization, and matting / segmentation that follow. Edge-preserving is affinity. *(This is L10.1* show*, registered in Part 13, Denoising where the bilateral first appeared for noise; this is its full edge-preserving treatment.)*

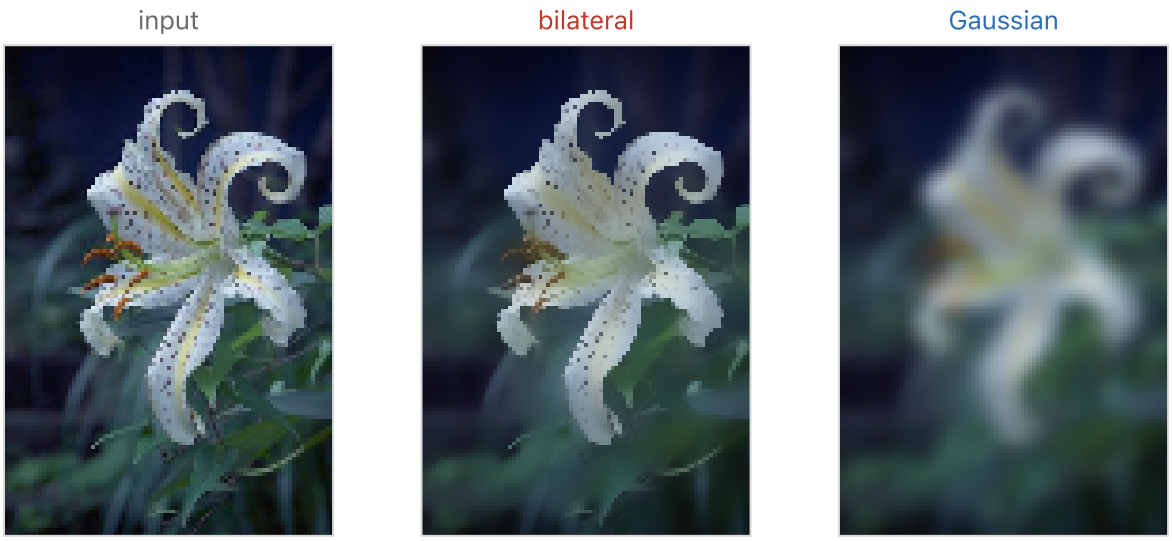


Figure 10.1.4 (reproduced). *The bilateral filter on a real image. Unlike a Gaussian blur, which smooths across edges and softens the whole image, the bilateral filter weights neighbors by both spatial distance and intensity similarity, so it smooths flat regions while largely preserving strong edges. Compare the input, the bilateral result, and a plain Gaussian of the same spatial extent; vary the range and spatial widths to see the trade-off, and note that a very large range width turns the bilateral back into that Gaussian. In this demo the color channels are filtered with weights computed from their gray intensity, so an edge between two colors of the same brightness is not protected; the range Gaussian reduces mixing across an edge rather than forbidding it, and the spatial kernel is cut off at three σ on a downsampled working copy of the image.*

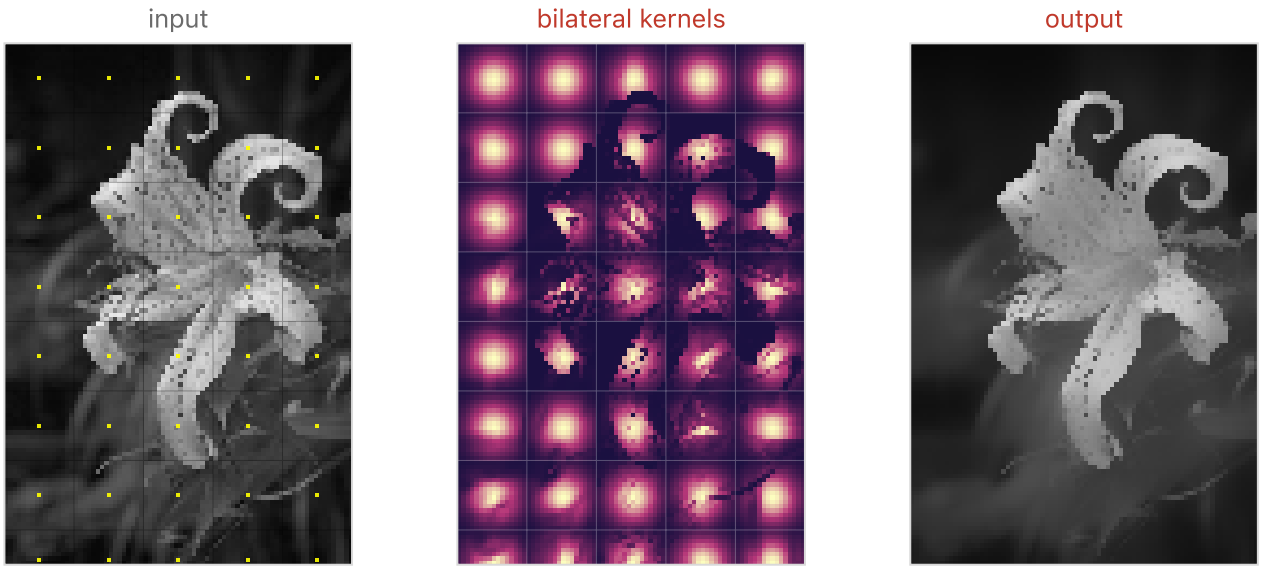
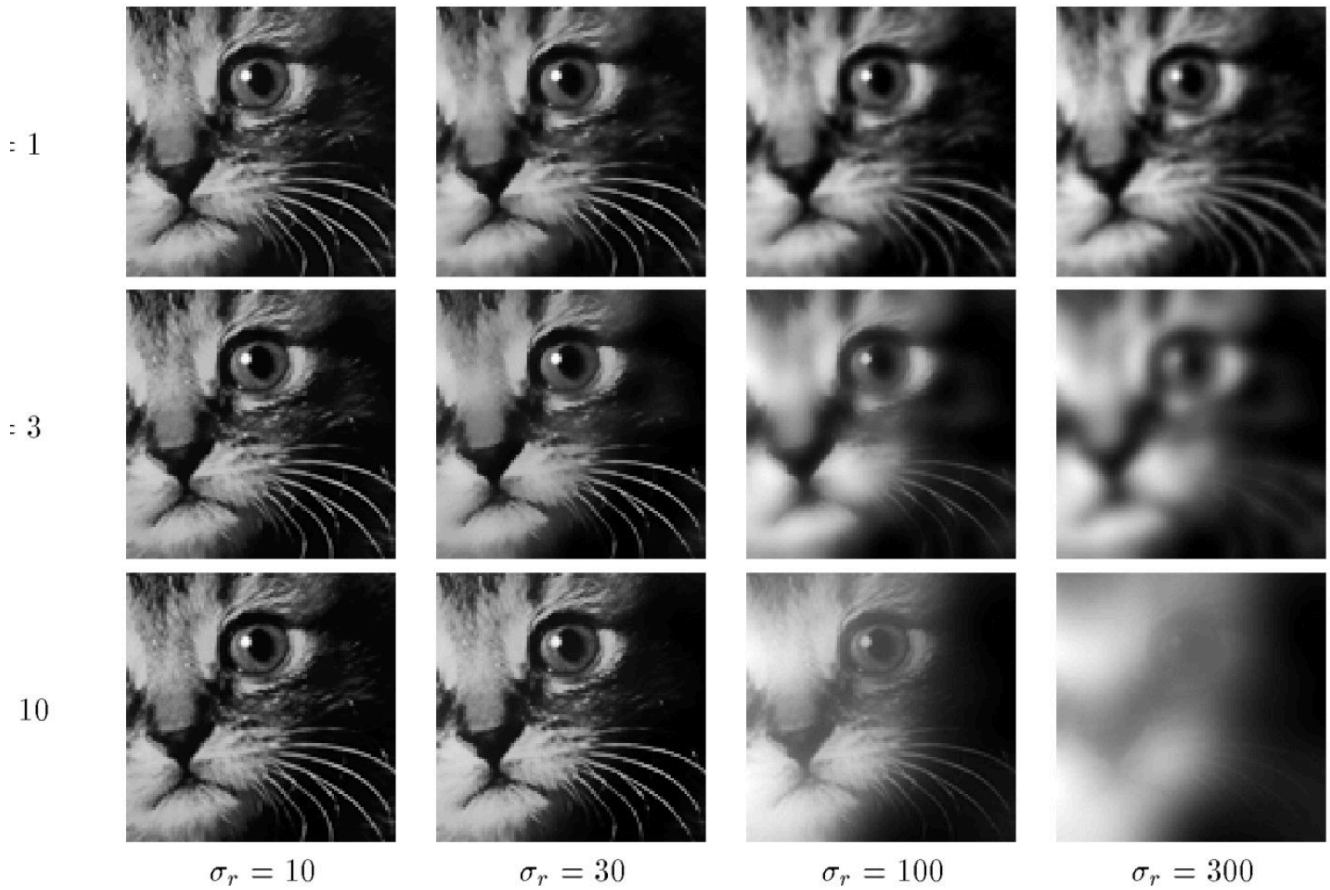


Figure 10.1.5 (reproduced). *The bilateral filter's kernel is different at every pixel; this figure makes that visible. The image is tiled into patches about $4\sigma_s$ wide (the filter itself cuts its support off at ±2σ_s); for each patch the middle panel paints the raw weight its center pixel gives every pixel of the patch, the spatial Gaussian times the range Gaussian $g_s \cdot g_r$ (bright = high weight). These weights are not normalized, so brightness shows relative preference, not each neighbor's share of the final average. In a flat patch the weight map is just the round spatial Gaussian, an ordinary blur kernel. Straddling an edge, the range term suppresses the far side and the kernel shrinks onto the center's own region. That space-varying reshaping is why the output (right) smooths texture yet averages very little across an edge: the affinity of L10.1* show *made into a picture. Pick the image and σ_s, σ_r, and switch the range term between gray and RGB, both computed on encoded pixel values.*



Reproduced from Tomasi & Manduchi (1998), with permission of C. Tomasi.

Figure 10.1.6 (reproduced). *The bilateral filter in the paper that named it. The original demonstration that combining a domain (spatial) Gaussian with a range (intensity-similarity) Gaussian smooths within regions while leaving strong edges intact: the affinity idea this whole chapter develops. After Tomasi & Manduchi 1998.*

Cross / joint bilateral

OPTIONAL
in the book ↗

So far the affinity has been read off the *same* image we are smoothing (the range weight uses I itself). But there is no reason it must. The cross (or joint) bilateral filter reads the range weight from a different image, a guide J, while still averaging the target I:

$$I_p^{\mathrm{cb}} = \frac{1}{W_p} \sum_q g_s(\|p-q\|)\, g_r(|J_p - J_q|)\, I_q, \qquad W_p = \sum_q g_s(\|p-q\|)\, g_r(|J_p - J_q|). \tag{7}$$

The averaging runs over the target I; the affinity comes from the guide J. Why want that? Because sometimes the image you wish to smooth is too noisy or too dark for its own edges to be trusted. If you read the affinity off I, you

are reading off the noise. If you hold a cleaner signal that shares the same scene geometry, borrow *its* edges instead.

The headline application is flash / no-flash photography (Petschnigg et al. 2004; Eisemann & Durand 2004). Shoot a dim scene twice: once without flash (the right mood and color, but dark and noisy) and once with flash (clean and well-exposed, but flat, harsh lighting). After registration, many object boundaries correspond. Motion, occlusion, flash shadows and specularities can introduce mismatched edges, so a practical pipeline needs confidence masks or fallback behavior. So smooth the noisy no-flash image using the flash image as the guide: the flash image's clean edges drive the range weight, telling the filter where it may and may not average, while the values being averaged come from the ambient no-flash shot. The result keeps the no-flash mood and kills the no-flash noise (Figure 10.1.8). We develop the full flash/no-flash pipeline later (forward-ref → Flash / no flash, computational illumination).

no flash, the image to filter, brightened × 21.5

flash, the guide

bilateral: range weights from the no-flash itself

cross bilateral: range weights from the flash

both average the same no-flash values: σs = 1.0% of the diagonal (13 px), σr = 0.25 on log10 intensity; zoom × 2, click a picture to move it

Figure 10.1.8 (reproduced). *The cross bilateral filter on a real flash / no-flash pair, Eisemann and Durand's own photographs Eisemann & Durand 2004. Top: the no-flash frame, the image to filter, shown brightened, and the flash frame, the guide. Bottom: the no-flash frame filtered with range weights read from itself (the ordinary bilateral) and read from the flash (the cross bilateral), with the same σ_s and σ_r. Around the candles the no-flash frame is broken into JPEG blocks and noise: the ordinary bilateral takes the blocks for edges and keeps them, while the cross bilateral smooths inside the flash's clean edges. It fails where the no-flash frame has an edge the flash lacks. The flames light only the no-flash frame, so they spread into a glow; and an intensity guide cannot see a color edge between two surfaces equally bright under the flash, such as a blue candle against pink icing. "Each channel" compares each color channel of the guide separately, as Petschnigg et al. 2004 do. On the toy, whose no-flash frame has real signal, the cross bilateral mostly loses what the ordinary one keeps, such as the white of the eye, which the flash shows as white as the fur. Click a picture to move the zoom.*

The full flash/no-flash story, including Petschnigg's detail transfer, Eisemann & Durand's intrinsic-relighting pipeline, flash-shadow removal, and their end-to-end results, belongs with the rest of computational illumination, where capturing the scene under two lights is the whole subject (Flash / no flash).

The cross bilateral is the first instance of a more general joint / guided idea (filter one image using structure taken from another) that culminates in the guided filter (later in the part).

Brute force

Brute-force translational alignment (SSD / NCC)

in the book ↗

For a tight burst the whole image slides as one rigid 2-D shift (dx, dy), and the simplest aligner imaginable just tries every shift in a search window and keeps the best-scoring one. A double loop over (dx, dy), with no cleverness at all, is useful here because it leaves little room for implementation errors. Everything then rides on the *score*: the function that says how well two frames agree at a given candidate shift (Figure 12.1.2).

The default score is the sum of squared differences (SSD),

$$\mathrm{SSD}(dx, dy) = \sum_{x,y} \big(I_1(x, y) - I_2(x + dx,\ y + dy)\big)^2, \tag{8}$$

SSD is small when the sampled intensities agree, which need not happen at a unique displacement. Repeated patterns create several minima; flat regions create weak curvature. Same exposure does not guarantee brightness constancy, and an HDR bracket can add clipping and motion. Mean-subtracted normalized cross-correlation handles a positive affine intensity change on the evaluated region:

$$\mathrm{NCC}(dx, dy) = \frac{\sum (I_1 - \bar{I}_1)(I_2 - \bar{I}_2)}{\sqrt{\sum (I_1 - \bar{I}_1)^2}\,\sqrt{\sum (I_2 - \bar{I}_2)^2}}, \tag{9}$$

NCC subtracts the mean and divides by the norms of the centered patches. It is invariant to positive gain and constant offset when neither patch is constant. Mask clipping and invalid pixels; do not divide by nearly zero variance. Nonlinear tone changes and spatially varying illumination require a better model than a single gain and offset.

Two practical notes. First, the cheapest unit test you will ever write: align a synthetic image to a *known* shift of itself and confirm the minimum lands exactly there, by construction. Catch the indexing, sign, and interpolation bugs on that toy, not on noisy real frames where a one-pixel error hides in the grain. Second, sub-pixel accuracy: the integer-grid minimum is rarely the true minimum, so fit a parabola to the SSD basin (or upsample the surface) and read off the vertex to get a fractional shift, realized by bilinear or bicubic interpolation when you resample. Those tiny fractional offsets are *noise* to be removed when you are denoising, but they are *signal* to be exploited when you are super-resolving: a burst whose frames are sub-pixel-shifted samples the scene more densely than any

one frame can, which is precisely what burst super-resolution mines (Wronski et al. 2019 Wronski et al. 2019; cross-ref Chapter 14.2, Multi-frame super-resolution).

Prokudin-Gorsky's sequential color-separation captures provide an early alignment example Prokudin-Gorskii. NCC can be a useful baseline for their digitized channels, but spectral color differences are not generally a single gain and offset. Gradients, suitable crops and rejection of damaged or moving regions may improve the estimate (Figure 12.1.4).

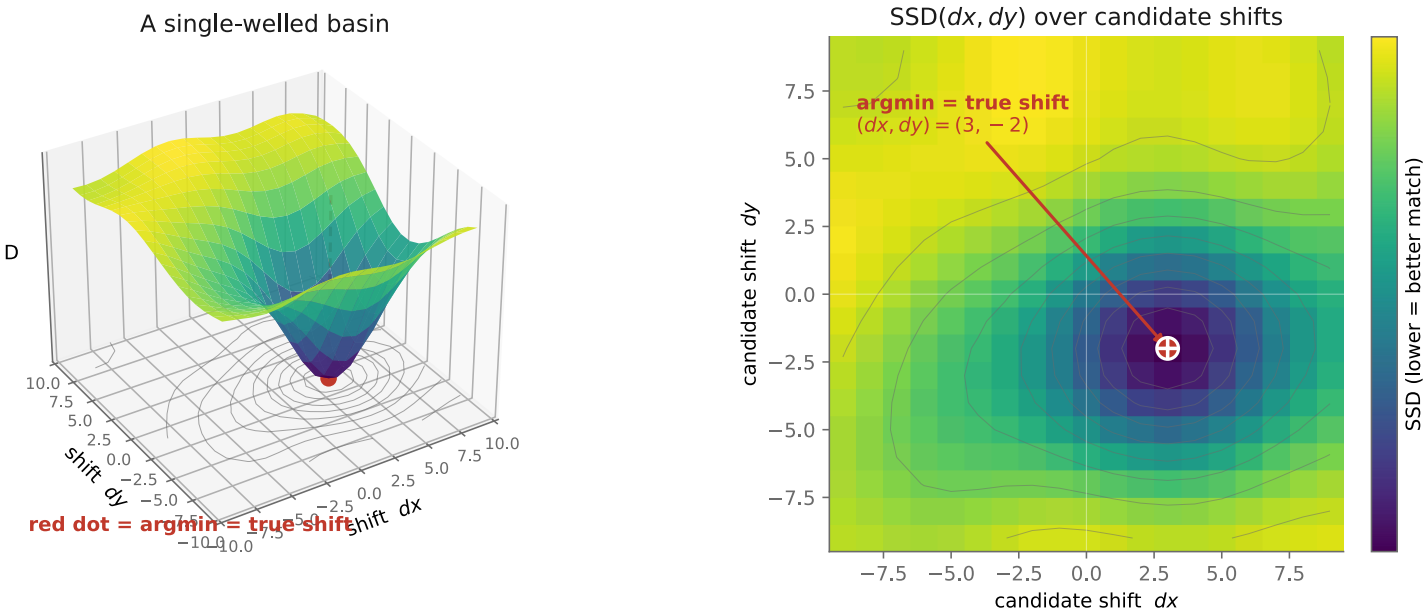
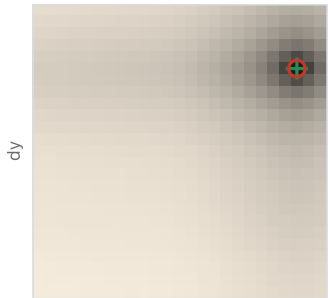


Figure 12.1.2 (reproduced). *The algorithm is the argmin of this score. For a pair of frames, the SSD score is evaluated over a window of candidate shifts (dx, dy) and plotted as a surface (or heatmap). It is a single-welled basin whose minimum sits exactly at the true shift d, where every edge has landed back on its partner. The brute-force aligner is literally an argmin over this surface; a parabola fit to the basin around the minimum recovers the sub-pixel shift.*

Both steps can be run on a real burst: search the window, read off the shifts, and average with and without them (Figure 12.1.3).

frame 6, as it came

SSD against frame 1, every shift



dx from −12 to +12

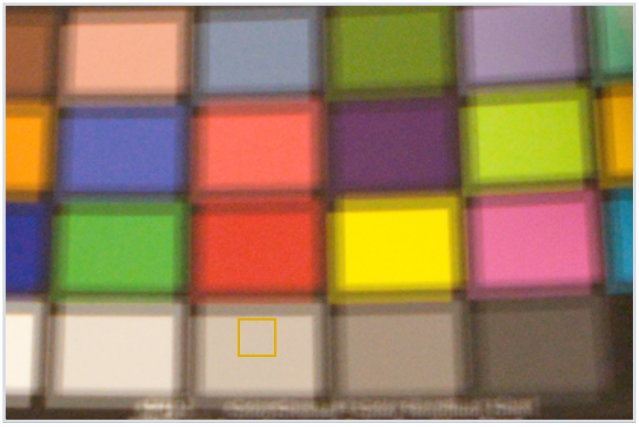
+ true shift

○ smallest SSD

frame	true shift	found
2	(0, 6)	(0, 6) ✓
3	(6, 4)	(6, 4) ✓
4	(4, 1)	(4, 1) ✓
5	(7, 1)	(7, 1) ✓
6	(10, -7)	(10, -7) ✓
7	(12, -8)	(12, -8) ✓
8	(12, -8)	(12, -8) ✓

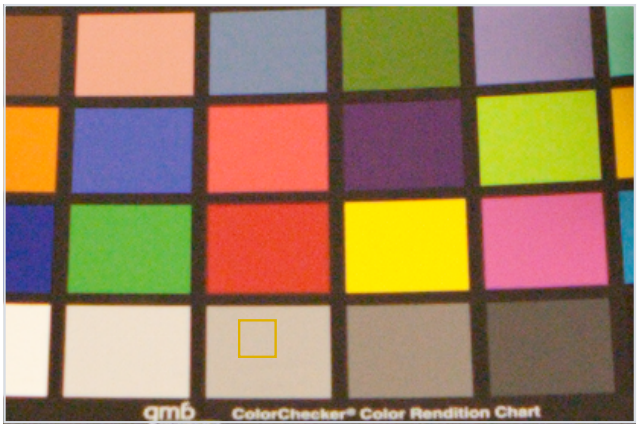
7 of 7 found

8 frames, averaged as they came



noise in the grey patch 1.48 · error against the truth 33.19

8 frames, aligned, then averaged



noise in the grey patch 0.95 · error against the truth 1.85

one frame: noise 2.81, so 8 aligned frames should reach 1.06 (1/√N, plus the truth's own noise). Codes out of 255; the truth is the mean of all 50 tripod frames.

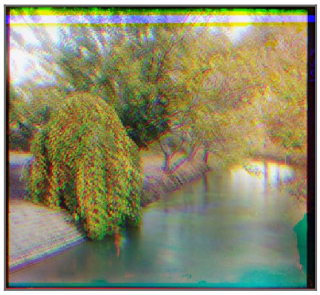
Figure 12.1.3 (reproduced). *Brute-force alignment of a hand-held burst (interactive). Eight frames of an ISO 3200 burst, each displaced along a hand-tremor path so the true shifts are known. For one frame, the sum of squared differences against the first frame is computed for every shift in the search window; its minimum is the estimated shift, and a table compares all seven estimates with the truth. Shrink the window below the tremor and the true shift falls outside it: the search returns the best wrong shift, on the window's edge. Below, the first N frames averaged as they came and after alignment, each compared with the mean of fifty tripod frames. Averaging lowers the noise either way, but without alignment it also blurs the picture, so only the aligned average approaches the tripod mean and gains the expected $1/\sqrt{N}$.*

Translational alignment in the wild: registering Prokudin-Gorsky's colour-separation plates

three plate negatives

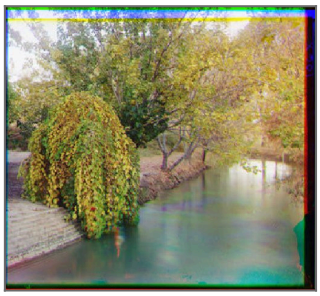
same scene through B, G, R filters

stacked as-is → colour fringes



plates shifted between exposures

channels aligned (min SSD/NCC) → clean



each channel shifted to its best match

Figure 12.1.4 (reproduced). *The translational aligner, a century early. Prokudin-Gorsky's three color-separation plates (blue, green, red), shot in sequence with a shifting camera. Top: the raw plates stacked as RGB, with strong colored fringes along every edge, because the channels are mutually mis-shifted. Bottom: each channel shift-searched against a reference by NCC (the plates differ in brightness, so not SSD) and re-stacked, so the fringes collapse into a clean, correctly registered color photograph. Same algorithm as the burst aligner, applied across channels instead of frames.*

Compare candidate shifts over a common valid support, or normalize a masked score by a sufficient valid-sample count. Otherwise larger shifts can win merely by discarding difficult pixels. A low residual and a high margin over competing shifts are complementary checks; neither detects all model mismatch.

The separated color exposures of Prokudin-Gorsky provide a useful application Prokudin-Gorskii. Shifted channels turn an otherwise sharp edge into red and cyan fringes when the plates are stacked. Search for a translation of each channel relative to a reference and inspect whether those fringes shrink. However, a red object can be bright in one filtered exposure and dark in another. Normalized correlation compensates for a common offset and scale, not for every spectral difference between objects. Comparing gradients or selecting regions with shared structure can help. Motion between exposures remains a separate problem that a single channel translation cannot fix.

Coarse-to-fine alignment on a pyramid

OPTIONAL
in the book ↗

Brute force has two failure modes that bite together. Large motion forces a large, and therefore expensive, search window; and a raw SSD surface is not always the clean single-welled basin of the idealized figure. Repetitive texture (a brick wall, a picket fence) and aliasing carve *local minima* into it, and a local refinement started far from the answer can fall into the wrong well. An exhaustive search finds the best score in its window, though that score may favor a false match. Going multiscale addresses both, though only the first reliably (Figure 12.1.6).

Build a Gaussian pyramid of both frames and align coarse-to-fine (the pyramid of Chapter 6.6, Pyramids and wavelets, here repurposed as a *search strategy*). At the tiny, heavily-blurred top of the pyramid, a *small* search in pixels covers a *large* real motion, because each coarse pixel stands for many fine ones, so large displacements become cheap. Better still, the low-frequency surface at the top may be smoother because blurring attenuates high-frequency detail, often making a rough estimate easier to obtain. Propagate that estimate down to the next level, scale it up, and refine it with a *tiny* local search; the high frequencies you add as you descend mostly refine the estimate, because the local search at each level is too small to jump to a distant minimum. That is also the risk: if the coarse estimate is wrong, the finer levels rarely recover.

Pyramids can smooth the cost and reduce a large displacement to a smaller coarse-grid shift. They do not guarantee one clean basin: periodic texture and lost small structures can remain ambiguous. Verify the refined result on the original images.

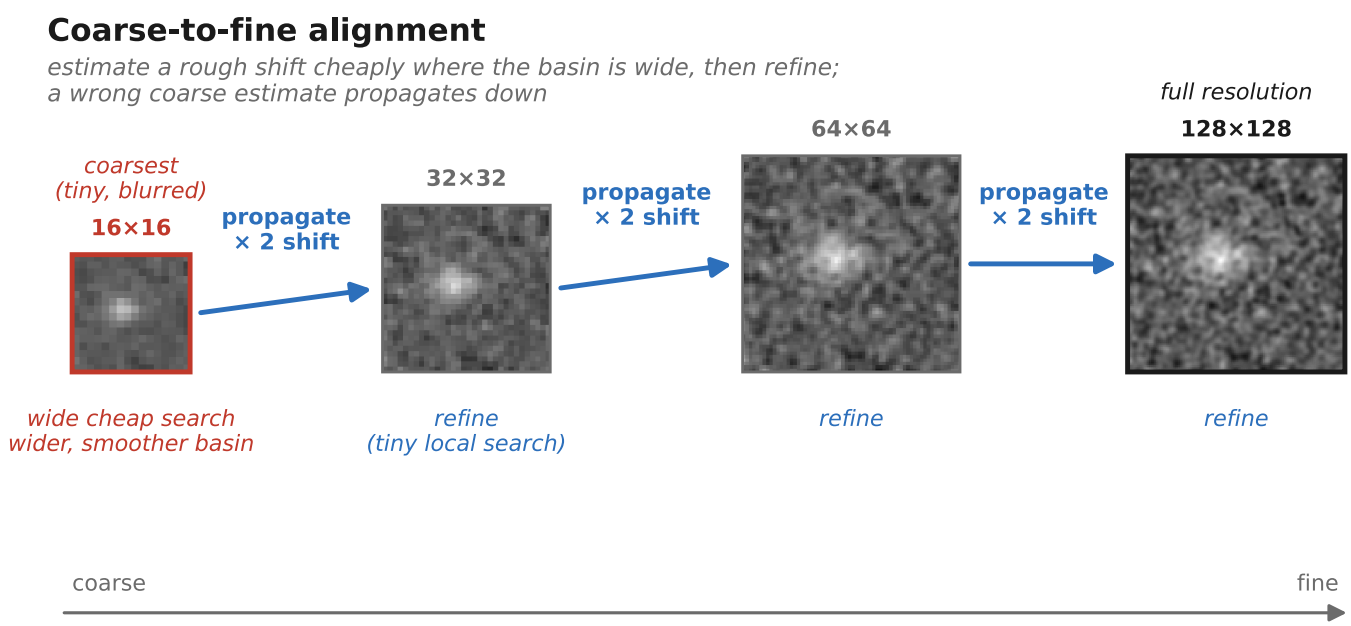


Figure 12.1.6 (reproduced). *A search strategy disguised as a pyramid. A Gaussian pyramid of both frames, aligned top-down. At the blurred top: a small, cheap search over a wide, single-welled low-frequency surface finds a rough shift that corresponds to a large real motion, and repetitive-texture local minima are fewer because blurring attenuated the fine texture that creates them. Descending: the coarse estimate is upsampled, scaled, and refined by a tiny local search at each finer level; high frequencies sharpen it but, with a small local search, rarely overturn it. Large motions are captured and many local minima are avoided in one structure, and the blur-then-decimate construction is exactly the L16 prefilter.*

Build each level by filtering before decimation. Otherwise the coarse image can contain alias patterns that disagree between frames, creating false evidence for motion. A pyramid is useful both as a representation and as a search strategy: solve large motions on a coarse grid, then recover precision from finer samples. HDR+ applies this idea to tiles before robust burst merging Hasinoff et al. 2016. Local refinement should still be checked against alternatives when the coarse choice is ambiguous.

Denoising by averaging

in the book ↗

Photograph a gray card several times without changing the camera settings. The value at one pixel fluctuates, but the average becomes more stable as the number of frames grows. Chapter 2.8, Noise, signal-to-noise ratio and dynamic range established the $1/\sqrt{N}$ law. Here we derive it for a frame stack and examine what happens when motion, outliers, or calibration errors violate its assumptions.

Fix the exposure, gain, and processing; keep the illumination stable; and align corresponding scene points before combining their measurements. Use linear values with the black level removed and preserve negative noise excursions. Under these conditions, independent zero-mean errors average out. They need not be Gaussian or symmetric: an unclipped Poisson count is unbiased even at low light.

> 💡 **BIG LESSON (L13.1) — CAPTURE THE FULL SET, DECIDE LATER**
>
> Recording several captures lets us choose or combine their measurements afterward. A filter wheel is the plain example: shoot the scene through a red, a green and a blue filter (or through a polarizer at several angles) and the choice of how to combine the bands can be made later, at the desk. Exposure brackets, focal stacks, and panoramas extend this idea to different capture settings.

> 💡 **BIG LESSON (L13.2) — MULTIPLE-EXPOSURE IMAGING BREAKS THE LIMITS OF A SINGLE CAPTURE**
>
> Several compatible measurements can reduce uncertainty or cover information missing from one frame. This is a different benefit from the previous lesson. Deferring a choice picks among things one exposure could have recorded; combining exposures can exceed what any single exposure could record, such as a noise level below one frame's, a dynamic range wider than the sensor's, or a depth of field deeper than the aperture allows. The reconstruction must account for differences in exposure, focus, viewpoint, or scene motion.

Why averaging works: the $1/\sqrt{N}$ derivation

in the book ↗

Let $X_i = \mu + n_i$ be the measurement of one scene point in frame i. The desired signal is μ, and the errors satisfy $\mathbb{E}[n_i] = 0$, $\mathrm{Var}(n_i) = \sigma^2$, and independence across frames. Equal variance is a per-pixel assumption across the stack; it does not require bright and dark pixels to have equal noise.

The estimate is $\bar{X} = N^{-1} \sum_{i=1}^{N} X_i$:

Algorithm 2

```
mean = zero_image()
for frame in aligned_linear_frames:
    mean += frame
mean /= number_of_frames
```

Its expected value is

$$\mathbb{E}[\bar{X}] = \frac{1}{N} \sum_i \mathbb{E}[X_i] = \mu. \tag{10}$$

Thus the estimator is unbiased: over repeated stacks, its average equals the true signal. An individual finite stack still contains error. Scaling a random variable by $1/N$ scales its variance by $1/N^2$, and independent variances add, giving

$$\mathrm{Var}(\bar{X}) = \frac{1}{N^2} \sum_i \mathrm{Var}(X_i) = \frac{\sigma^2}{N}, \qquad \sigma_{\bar{X}} = \frac{\sigma}{\sqrt{N}}. \tag{11}$$

Nothing is blurred by this, which is what separates it from every single-image method in the part: each pixel is averaged only with repeated measurements of itself, never with a different pixel. The cost is in the exponent. Halving the noise again takes four times as many frames, so there are sharp diminishing returns to spending photons this way (Figure 13.2.1).

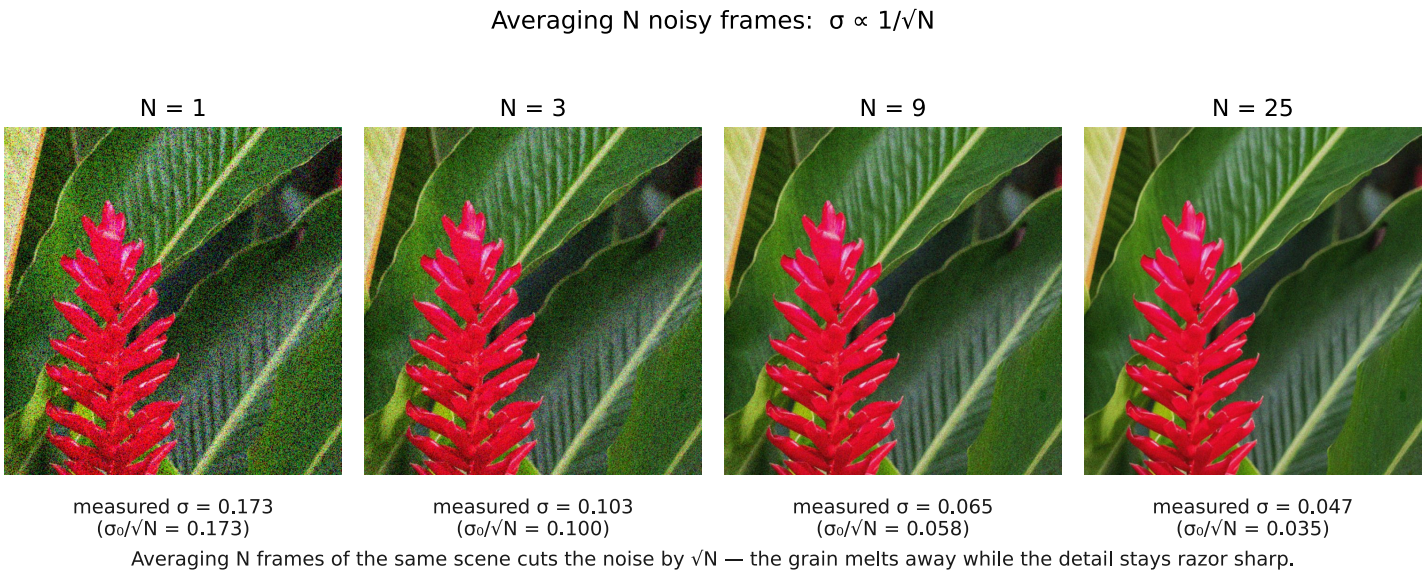


Figure 13.2.1 (reproduced). *Denoising by averaging frames. The same static scene averaged over 1, 3, 9, and 25 noisy shots, on a real color photo with realistic shot-plus-read noise. The grain drops steadily (variance falls as $1/N$, so the noise standard deviation falls as $1/\sqrt{N}$) while detail stays perfectly sharp, because each pixel is averaged only with repeated measurements of itself. Going from 1 to 25 frames cuts the noise by a factor of 5 (i.e. $\sqrt{25}$).*

The signal stays constant while the noise standard deviation falls. Figure 13.2.2 shows the resulting narrowing of a flat patch's histogram.

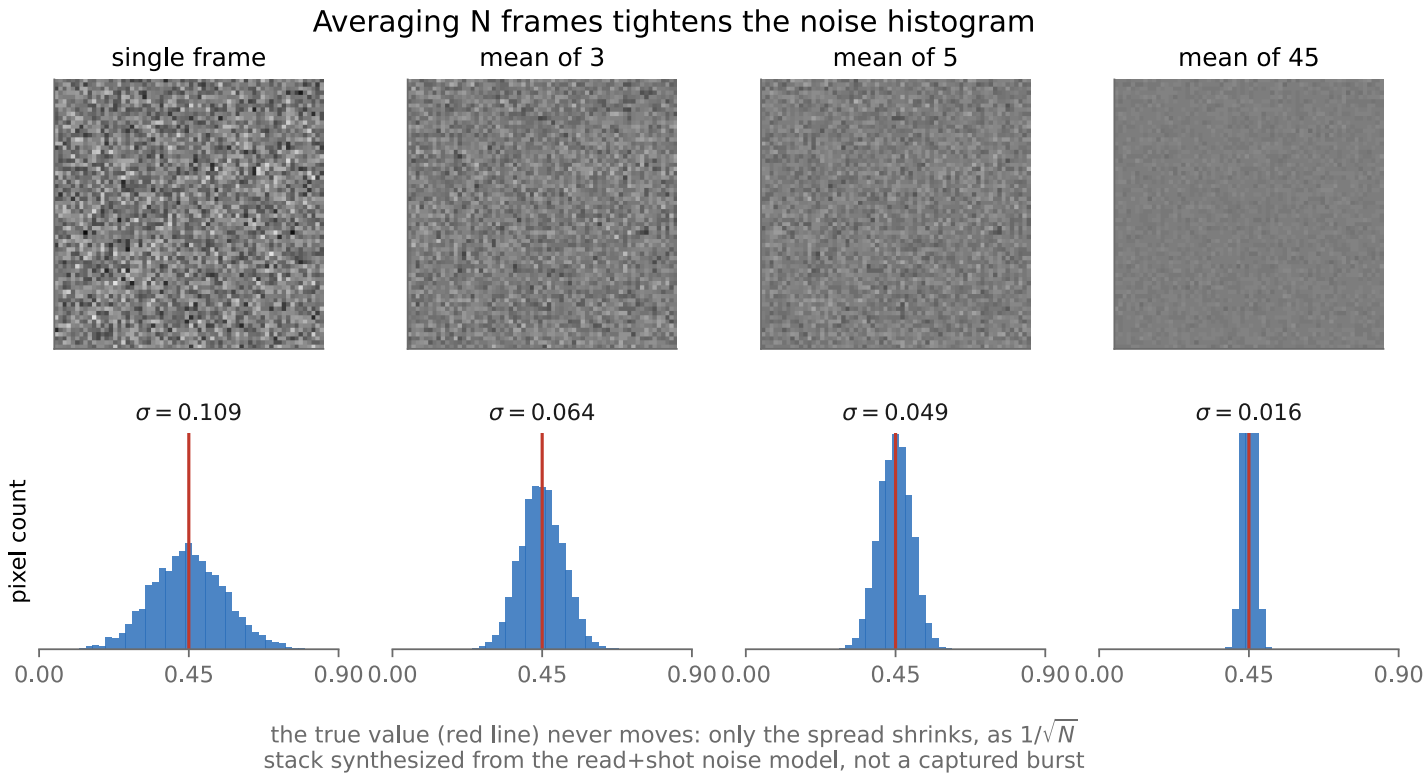


Figure 13.2.2 (reproduced). *Averaging tightens the histogram. One static gray patch, its noise drawn from a measured read-plus-shot model rather than captured: a single frame, then the mean of 3, of 5, and of 45 frames. Below each, the histogram of the patch's pixel values: flat and broad for one frame, visibly narrowing toward a spike as N grows; the dark regions clean up first. The signal (the patch's true value) never moves; only the spread shrinks.*

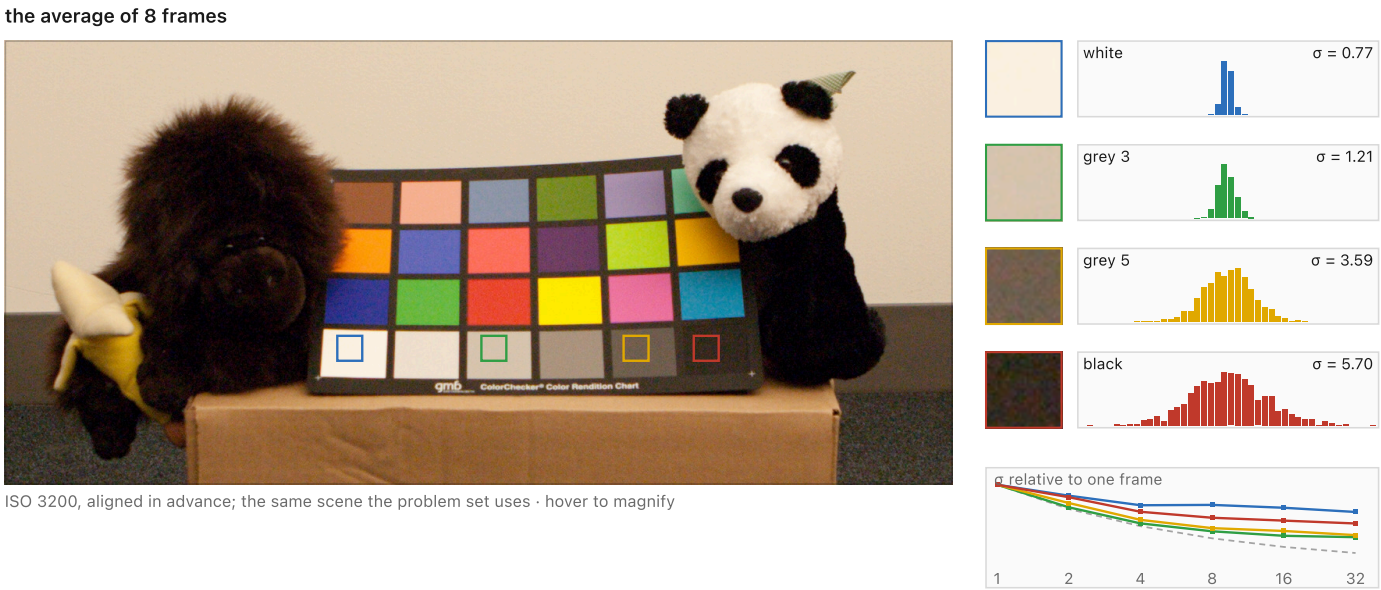
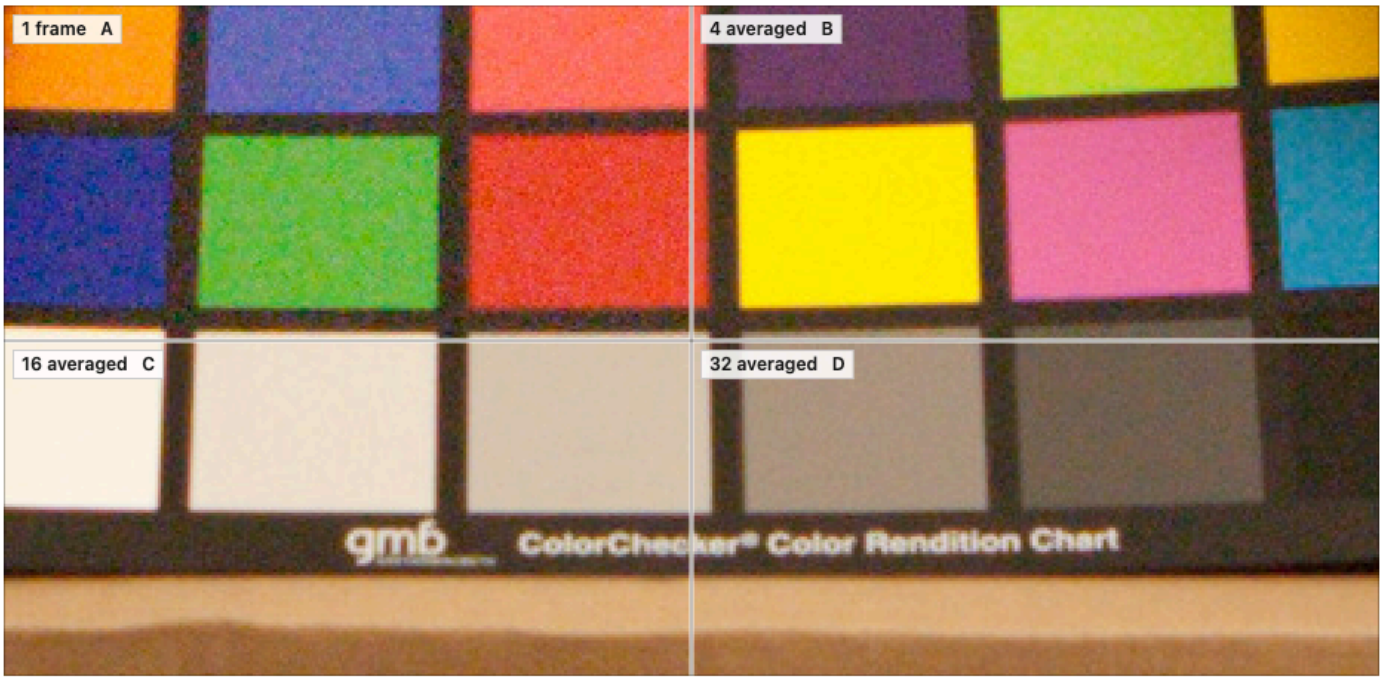


Figure 13.2.3 (reproduced). *The same claim on the course's own frames (interactive).* Fifty aligned exposures of a ColorChecker between two stuffed animals at ISO 3200, the set the problem set works from; the slider averages the first N of them, in powers of two. Beside the picture, a square inside four gray patches of the chart at its own resolution, the spatial histogram of that square, and its standard deviation. The histograms narrow as N grows, and the curve underneath plots that narrowing against the $1/\sqrt{N}$ line. The dark patches start widest, since the noise is signal-dependent. They also stop narrowing before the line does: what is left is fixed pattern, the same in every frame, which no amount of averaging removes and which Calibration frames: what averaging can't fix takes up.

4 panes of the same view



drag inside the picture to move the split; click to recentre the view

Figure 13.2.4 (reproduced). *Whether you can see it (interactive).* The same burst in the book's comparator: up to four versions at once, one frame against the average of 4, 16 and 32, divided at a point you drag, or swapped in place one at a time, which is the way to catch a small difference, since the eye reads a change better than it reads two things side by side. At the 1:1 zoom the grain is the picture's own, not a resampling of it; a click recentres, so the same detail can be carried from pane to pane.

BIG LESSON (THE $1/\sqrt{N}$ LAW)

Averaging four independent measurements of the same signal halves the noise standard deviation. Averaging 100 reduces it by a factor of ten. These statements require zero-mean errors and unchanged per-frame variance. Lowering the effective noise floor can extend the usable range, as described in L2.46 show, provided the captures have not clipped the signal.

SNR increases by $\sqrt{N}$, or $10\log_{10} N$ dB: about 3 dB per doubling of the frame count (Figure 13.2.5). This comparison adds capture time while holding each exposure fixed. It does not say that splitting one fixed total exposure into more frames improves SNR.

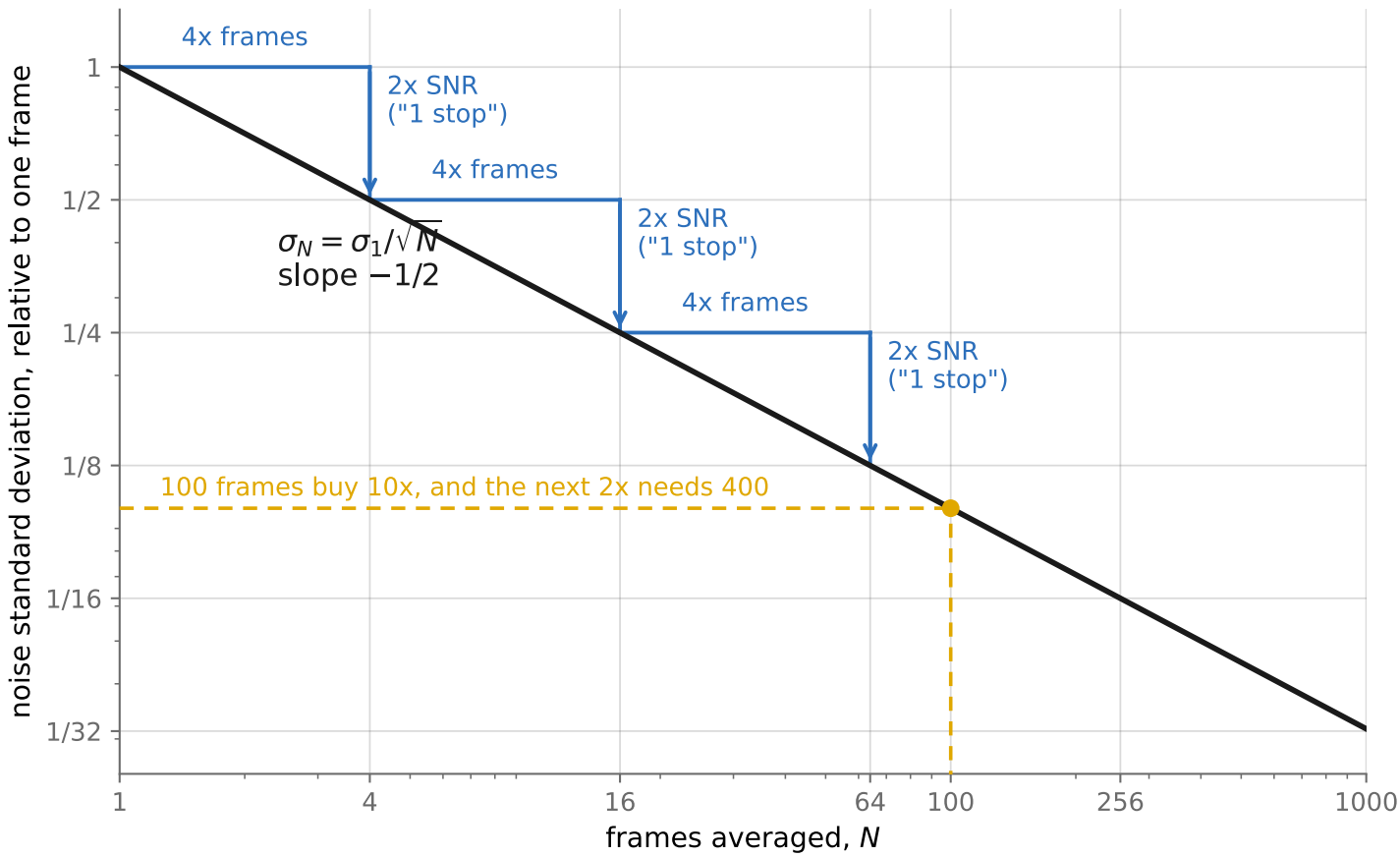


Figure 13.2.5 (reproduced). *The diminishing return, on log–log axes. Noise standard deviation versus number of frames N is a straight line of slope $-\frac{1}{2}$. Reference marks show that halving the noise standard deviation, the factor of two photographers loosely call a stop, costs $4\times$ the frames, and that 100 frames buy only a $10\times$ reduction. The straight line* is *the $1/\sqrt{N}$ law.*

The same stack estimates the per-frame variance:

$$\hat{\sigma}^2 = \frac{1}{N-1}\sum_i (X_i - \bar{X})^2, \qquad N > 1. \tag{12}$$

The divisor $N-1$ corrects for estimating the mean from the samples whose deviations we measure. For two fair coin flips coded as 0 and 1, the four outcomes give sums of squared deviations $0, \frac{1}{2}, \frac{1}{2}, 0$. Dividing by two gives an average estimate of $1/8$, half the true variance $1/4$; dividing by one gives $1/4$.

If errors are correlated, the omitted terms return:

$$\mathrm{Var}(\bar{X}) = \frac{1}{N^2}\left[\sum_i \mathrm{Var}(n_i) + 2\sum_{i<j}\mathrm{Cov}(n_i, n_j)\right]. \tag{13}$$

A common offset across frames does not diminish. A fixed calibration bias survives as an error in the mean even when the remaining random fluctuations obey the averaging law.

A four-frame example separates the two assumptions. If the same signal is measured four times with independent errors of standard deviation 8, their average has standard deviation 4. A fixed offset of 3 at that sensor position survives unchanged: averaging adds the offset four times and divides by four. A fluctuation shared by every frame likewise does not receive the independent-noise reduction. Thus a cleaner-looking average can still retain a structured bias. Calibration and averaging solve different parts of the measurement problem, as developed in Chapter 2.8, Noise, signal-to-noise ratio and dynamic range.

The square root also describes the diminishing return on capture time. Halving the noise standard deviation takes four times as many independent, equally exposed frames. Going from one frame to four and from four frames to sixteen buys the same factor of two, but the second improvement costs twelve extra frames. Motion, storage, and latency can make those extra frames more expensive than the noise reduction is worth.

When the plain mean is wrong: robust combination

OPTIONAL
in the book ↗

A satellite crossing one exposure adds a bright streak to the stack. The mean reduces its contribution by N, but leaves a visible trace. A per-pixel median is less sensitive to a small number of extreme values (Figure 13.2.6). It can still fail when the wrong scene or moving object supplies most of the samples at that position.

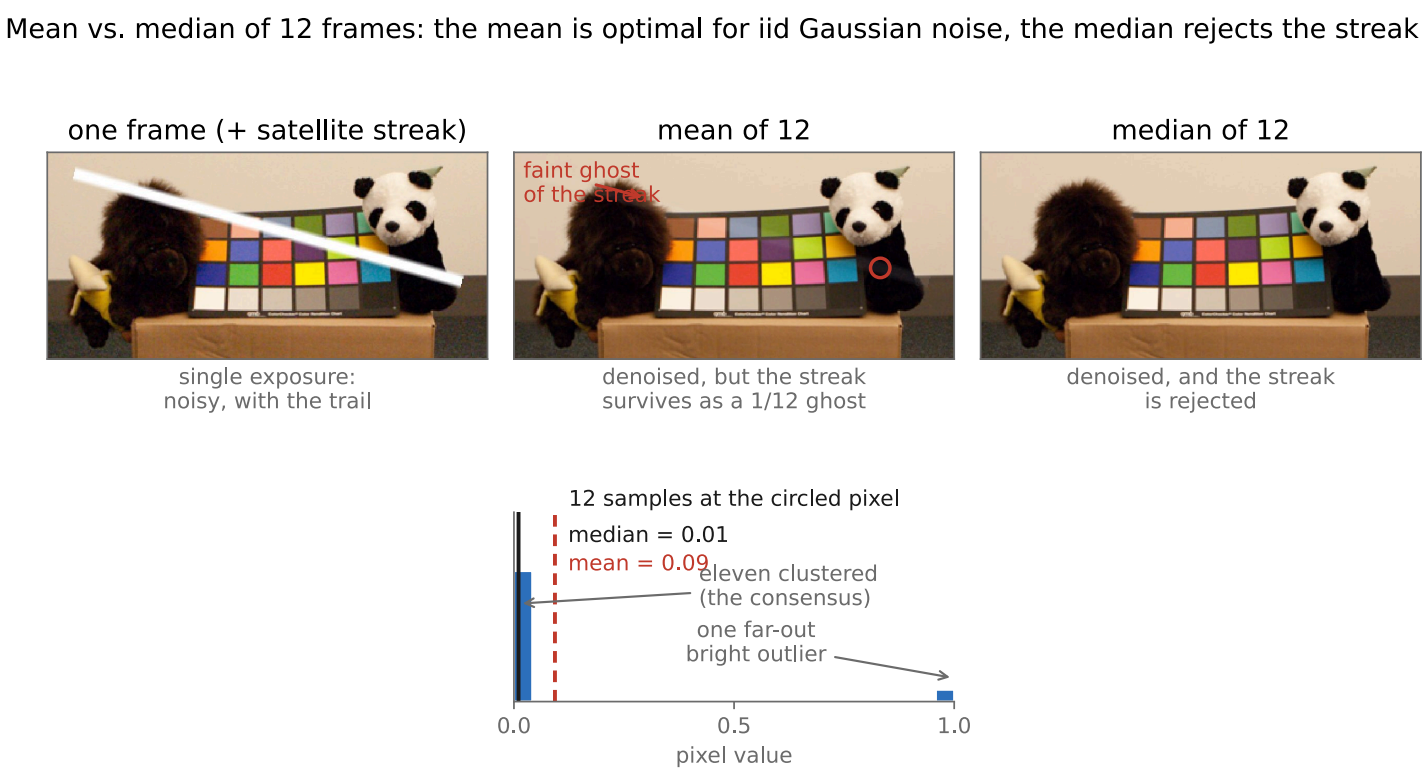


Figure 13.2.6 (reproduced). *Mean versus robust combine, on a stack with one satellite streak. The same scene stacked two ways: the plain per-pixel mean retains a faint ghost of the streak (one frame's outlier divided across N); the per-pixel median (or sigma-clip) rejects it entirely, leaving a clean result. The inset shows the per-pixel sample distribution at a streaked pixel: one far-out value the consensus ignores.*

For independent Gaussian samples of a constant signal, the large-sample variance of the median is approximately $\pi/2$ times that of the mean. Its standard deviation is about 1.253 times larger, corresponding to roughly 20% lower SNR. Robust rejection therefore has a cost when all samples follow the assumed Gaussian model.

Sigma-clipping rejects measurements whose deviations exceed a chosen multiple k of an estimated standard deviation. A robust initial center and scale prevent one large outlier from inflating its own rejection threshold. The following is an illustrative procedure for a sufficiently large stack, not a guarantee for a short burst:

Algorithm 3

```
for each output pixel p:
    values = valid_registered_samples(p)
    if values is empty:
        mark_output_invalid(p)
        continue
    center = median(values)
    scale = max(1.4826 * median(abs(values - center)), noise_floor(p))
    keep = abs(values - center) <= k * scale
    if too_few_samples(keep):
        out[p] = center
    else:
        out[p] = mean(values[keep])
```

The factor 1.4826 calibrates the median absolute deviation for Gaussian noise. Choose k and the minimum sample count using the noise model; rejection based on a short stack can discard valid measurements or retain an outlier. Winsorization, which caps extreme values, is distinct from discarding them. Moving boundaries and occlusion usually need alignment confidence and region-dependent weights in addition to a per-pixel test.

For independent unbiased measurements of unequal *known* variances v_i, the minimum-variance linear unbiased estimate uses

$$\hat{\mu} = \frac{\sum_i X_i/v_i}{\sum_i 1/v_i}, \qquad \mathrm{Var}(\hat{\mu}) = \frac{1}{\sum_i 1/v_i}. \tag{14}$$

Before combining different exposures, convert them to a common signal scale and transform their variances by the square of the same scale factor. Derive weights from calibrated parameters or a stable estimate, since weighting directly by a noisy measured value can bias the result. Noise-optimal HDR capture uses this relationship to select exposures and combine their useful ranges (Hasinoff et al. 2010).

This is the same neighborhood-selection question that appears in edge-aware filtering, now asked across frames. A sample can be nearby after alignment yet belong to a different surface because of motion or occlusion. Rejecting it prevents a ghost, but reduces the amount of independent evidence available there. HDR merging faces the same compromise, with the additional need to reject saturated measurements.

Handheld low light: the phone in your pocket

OPTIONAL
in the book ↗

A handheld burst replaces one motion-blurred long exposure by several shorter measurements. The merge aligns stable regions and reduces the contribution of measurements that disagree because of motion or occlusion. Near a moving subject, fewer usable samples mean less noise reduction. A single frame count therefore does not describe the quality everywhere in the result.

HDR+ combines burst capture with alignment and noise-aware merging (Hasinoff et al. 2016). Night Sight also estimates motion to choose exposure durations and frame counts, and uses low-light white balance and tone rendering (Liba et al. 2019). These later operations affect the appearance, so a clean-looking output should not be attributed to averaging alone. Chapter 23.3, Focal stacks and depth of field extension considers related capture-time tradeoffs involving aperture and focus (Hasinoff & Kutulakos 2009).

Night Sight makes the capture decision part of denoising Liba et al. 2019. A relatively still scene permits longer individual exposures, reducing how often read noise is incurred for a given light budget. Subject motion favors shorter exposures, which reduce blur within each frame but make each measurement noisier. Motion estimation therefore helps choose exposure durations and frame counts before the burst is combined. Alignment and robust merging then handle disagreement among frames. Low-light white balance and tone rendering address different questions afterward: a low-noise estimate can still have an unwanted cast or look unlike a night scene. The complete result depends on capture, combination, and rendering together.

Recovering a clean signal does not determine how bright a night photograph should look. A sufficiently long exposure can make a moonlit landscape resemble daylight. The Night Sight team drew on illusionistic painting to preserve the impression of night: strong contrast, dark surroundings, and shadows that approach black, even while revealing more color and detail than the eye could see. Their tone curve balances visibility against that impression of darkness. This is a rendering decision after reconstruction, and a useful connection to the discussion of pictorial cues and adaptation in the color and perception parts (Google's account of Night Sight).

Gaussian, median, and bilateral filtering

in the book ↗

A noisy photograph of a wall contains nearby pixels with similar clean values. Averaging them reduces noise. At the wall's boundary, however, the same operation mixes two surfaces and blurs the edge. Single-image local denoising must decide which neighbors can supply useful evidence about a pixel.

Everything in this chapter is a different answer to one question: which neighbors count as measurements of the same value? In a flat region every neighbor qualifies and averaging cleans up beautifully. At the boundary between a dark cake and a bright wall the neighbors on the far side are measurements of something else, and averaging them smears the two into a muddy band.

We compare a fixed Gaussian average, a median, and a bilateral average whose weights depend on the measured values. Their assumptions differ, and none guarantees preservation of every small feature. The derivation below uses independent zero-mean noise of constant variance in one channel; raw signal-dependent noise requires calibrated local variances or stabilization (Chapter 13.1, Noise models, transforms, and benchmarks).

Gaussian smoothing, and the assumption inside it

in the book ↗

Averaging a neighborhood assumes that every pixel in it is a measurement of the same underlying value, corrupted independently (the image is locally constant). Where that holds, the average is the right estimate and the noise cancels out. Where it fails, in particular at any boundary between two surfaces, the average mixes two different values and returns something that was never in the scene. A Gaussian kernel softens the assumption by trusting near neighbors more than far ones, but it still applies the same fixed weights at every pixel, whatever the image is doing there.

For analysis, consider a fixed clean image x. What varies is the measurement: the measured value is the truth plus an error, $y_q = x_q + n_q$, and the error is a fresh draw every time the shutter opens (Chapter 13.1, Noise models, transforms, and benchmarks).

For now, we assume two important properties of those draws. First, they average out, meaning the error at a pixel tends to zero as the number of shots increases. Second, they are independent between pixels: knowing that a photosite read high says nothing about its neighbor. That is true of shot noise and read noise, and false of fixed-pattern response and of noise that has already been through demosaicking or resampling. The variance σ^2 is the size of a typical error, squared; a refresher on these terms is in Refreshers#Probability and information theory.

Because noise is probabilistic, we analyze denoising through the thought experiment of shooting the same scene a large number of times with different noise instances each time. We then conceptually run the same filter on each frame and look at one output pixel across all the different results. We then characterize the performance of the method via statistical estimates such as the average squared error. This is related to the denoising by averaging technique studied in Chapter 13.2, Denoising by averaging but here the goal is to understand the expected error in one run via a probabilistic average error analysis, not to actually compute averages to reach the value of a pixel.

The average error can manifest in two main ways. The answers can be scattered, which means the filter still passes noise through. And their center can sit away from the truth, which means the filter has mixed in something that was never at this pixel. The first is the variance of the estimate, the second is its bias, and the error that matters, squared and averaged, is the bias squared plus the variance. A denoiser is an attempt to make that sum small.

A normalized Gaussian kernel supplies fixed nonnegative weights w_{pq} that decrease with distance from the output pixel p:

$$\hat{x}_p = \sum_q w_{pq} y_q, \qquad \sum_q w_{pq} = 1. \tag{15}$$

For independent errors of equal variance σ^2, the output variance and bias are

$$\mathrm{Var}(\hat{x}_p) = \sigma^2 \sum_q w_{pq}^2, \qquad \mathbb{E}[\hat{x}_p] - x_p = \sum_q w_{pq}(x_q - x_p). \tag{16}$$

The bias formula is important because it is the only place the noise-free image itself appears. The term $x_q - x_p$ captures the deviation from the assumption of a locally constant image. Let's look at variance and bias on two patches. On a flat patch every x_q equals x_p, so every term of the bias is zero, and with a box of nine equal weights the variance is $\sigma^2/9$: the noise standard deviation falls by three, and nothing is lost. Now slide that same window onto an edge where the far side is 100 units brighter, with half the weight landing across it. The variance is unchanged, still $\sigma^2/9$, while the bias is about 50 units, next to a noise level of perhaps two or three. The filter has not failed at its job; it has done exactly what it was told, on an assumption that stopped being true.

This is called the bias-variance tradeoff. Widening the kernel moves both terms at once, buying variance with bias, and no choice of width escapes the trade: at any setting, a fixed kernel that is wide enough to clean the flat regions is wide enough to damage the edges. That is why the chapters after this one change the weights rather than the width, why Chapter 13.12, Measuring a denoiser compares two denoisers at a matched operating point rather than at matched parameters, and what Chapter 13.13, The limits of denoising pushes to its bound.

Define the effective sample count $N_{\text{eff}} = 1/\sum_q w_{pq}^2$, which is the number of independent samples the weighted average is worth. Equal weights over N independent samples give $N_{\text{eff}} = N$ and recover the averaging law of Chapter 13.2, Denoising by averaging; a Gaussian gives rather fewer than its tap count, since the outer taps contribute little.

Choosing the width is therefore choosing what to sacrifice. Set σ_s from the size of the structure you are willing to lose: everything substantially smaller than the kernel is on its way out, and everything substantially larger survives with its edges softened. In practice the Gaussian is also the cheap choice, because it is separable and a two-dimensional blur costs two one-dimensional passes (Chapter 6.2, Convolution).

Figure 13.3.1 (reproduced). *Interactive. One noisy image, three filters, matched. Set a noise level, then adjust each filter, or let the panel search, until all three reach the same measured noise reduction in one flat reference patch, a much fairer comparison than equal parameter values. The match holds for that patch; elsewhere the three filters can still leave different amounts of noise. The panel then shows what each destroyed to get there: the Gaussian's smeared edge, the median's flattened texture and rounded corners, the bilateral's preserved edge and its staircase. A residual view (input minus output) is the sharpest diagnostic, since a residual that contains visible image structure is signal the filter removed.* Interactive: choose noise type and level, set each filter's parameters or let the panel match them, toggle residual and difference views, and move a crop selector across flat, edge and texture regions.

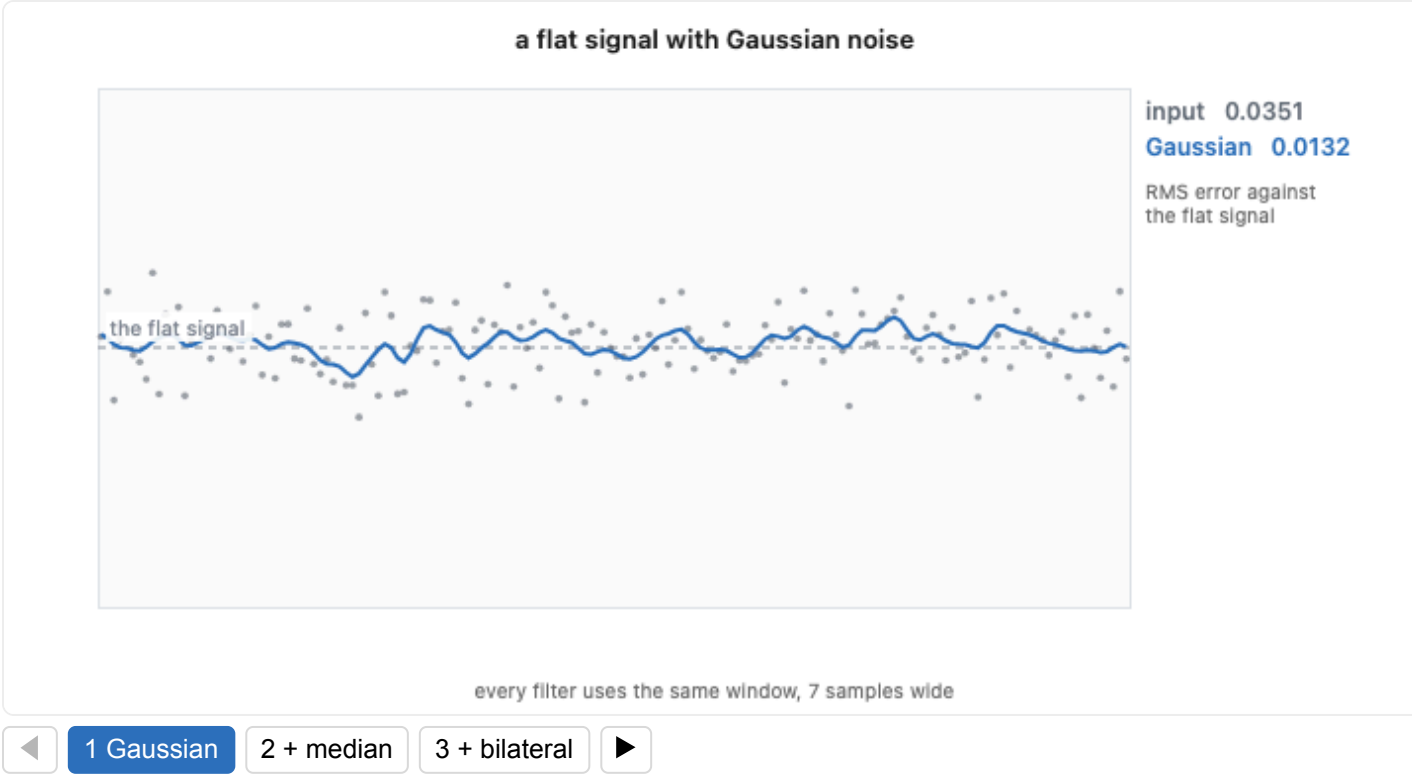
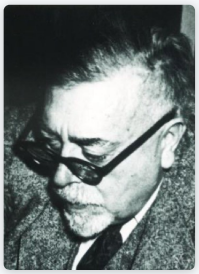


Figure 13.3.2 (reproduced). *The same filters on a scanline (interactive).* A flat signal with noise, and the Gaussian's output over it. In one dimension you can see what the filter did to every sample: with Gaussian noise it pulls the samples toward the level they are all measuring, and with impulse noise it spreads each stuck sample into a bump of the window's own width rather than removing it. Every filter in this figure uses the same window, which is what makes the comparison fair.

Start with the Gaussian alone in that panel, and set the other two aside until the sections that introduce them. Widen it and watch the two terms trade: the flat patch cleans up, the edge softens, and the residual view fills with the structure the filter took.

The frequency picture says the same thing and explains why blurring is worth doing at all. White noise has a flat power spectrum, so it carries as much energy at high frequencies as at low. Photographs do not: most of their energy sits at low frequencies, and the fine detail that occupies the high end is a small part of the total. A low-pass filter therefore discards a large share of the noise and a small share of the signal. That argument has an optimum, and the optimum is not a Gaussian.

Sidebar 3: the optimal *linear* filter, Wiener



For a stationary signal and independent additive stationary noise with known power spectra, the minimum-MSE convolution has frequency response $H = S_x/(S_x + S_n)$. It attenuates frequencies dominated by noise. Its optimality is conditional on those spectra and the linear, shift-invariant class; natural images have spatially varying structure. *Norbert Wiener (1894–1964), MIT mathematician and founder of cybernetics, derived this optimal filter from his World War II work on anti-aircraft*

fire control. Portrait: Konrad Jacobs, Oberwolfach, CC BY-SA 2.0 DE, via Wikimedia Commons.

The Wiener filter Wiener 1949 is what the frequency argument looks like when it is pushed to its conclusion: keep each frequency in proportion to how much of it is signal rather than noise. Its assumptions are the catch. It needs both spectra in advance, and it treats the image as one stationary process, when a photograph is a collection of regions with different statistics. A filter that could vary its behavior by region would do better, which is the move the rest of the chapter makes.

Sidebar 4: when the variance formula lies

The two expressions above are exact for fixed weights and independent errors. Three departures matter in practice.

- **Unequal variances.** With independent but differing σ_q, the output variance is $\sum_q w_{pq}^2 \sigma_q^2$. In raw, where the variance rides the signal, a neighborhood spanning bright and dark is a weighted average of unequal-quality measurements.
- **Correlated errors.** Any correlation needs the covariance terms, and demosaicking, resampling and lens shading all introduce it. Noise that has been through a pipeline is not the noise the formula assumes (Chapter 13.1, Noise models, transforms, and benchmarks).
- **Repeated taps at a boundary.** With a replicated border extension, several kernel taps can address the same sensor sample. Combine their weights before squaring them: repeated reads of one sample are perfectly correlated, not independent evidence. Counting taps instead overstates N_{eff} exactly where the estimate is already weakest.

None of this applies cleanly to weights computed from the noisy image itself, which is what the bilateral filter does later in this chapter.

The median, and the noise it is actually for

OPTIONAL
in the book ↗

The median selects the middle value of a sorted neighborhood. It is also a minimizer of

$$\hat{x}_p \in \arg\min_m \sum_{q \in \mathcal{N}(p)} |y_q - m|. \tag{17}$$

Changing an outlier's magnitude can move the mean arbitrarily far, but need not move the median. For example, the values $10, 10, 11, 11, 255$ have mean 59.4 and median 11. This makes the median useful for sparse impulse corruption (Figure 13.3.3).

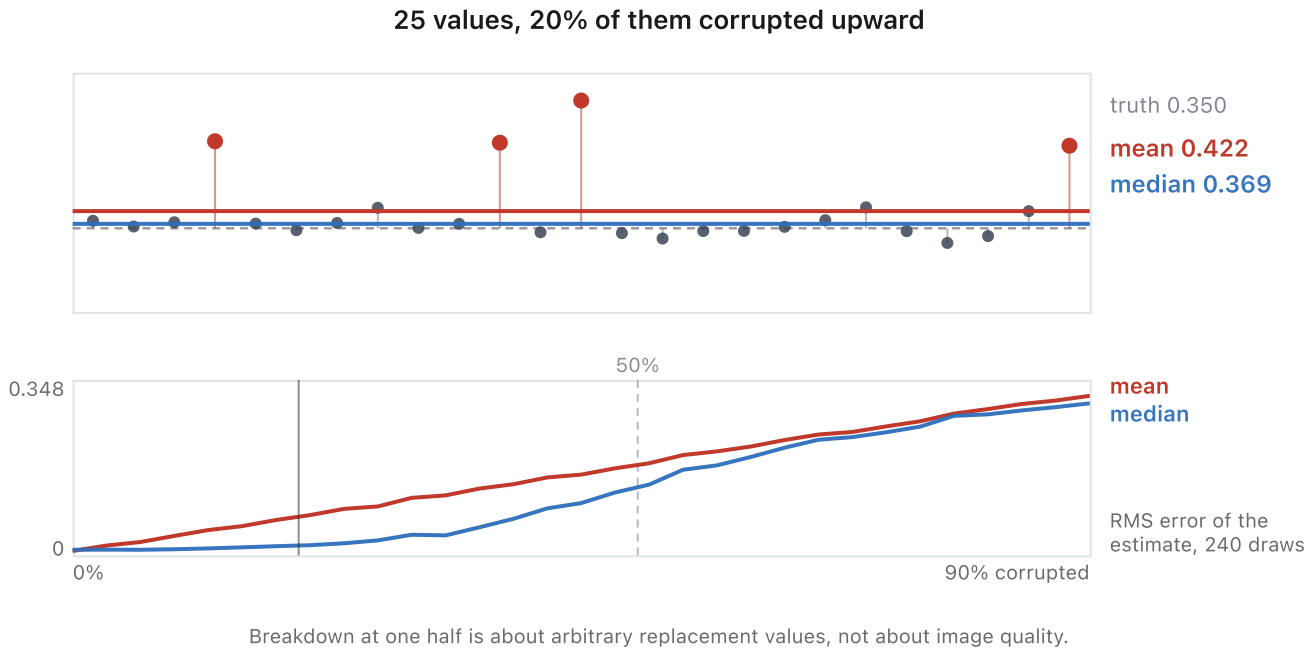
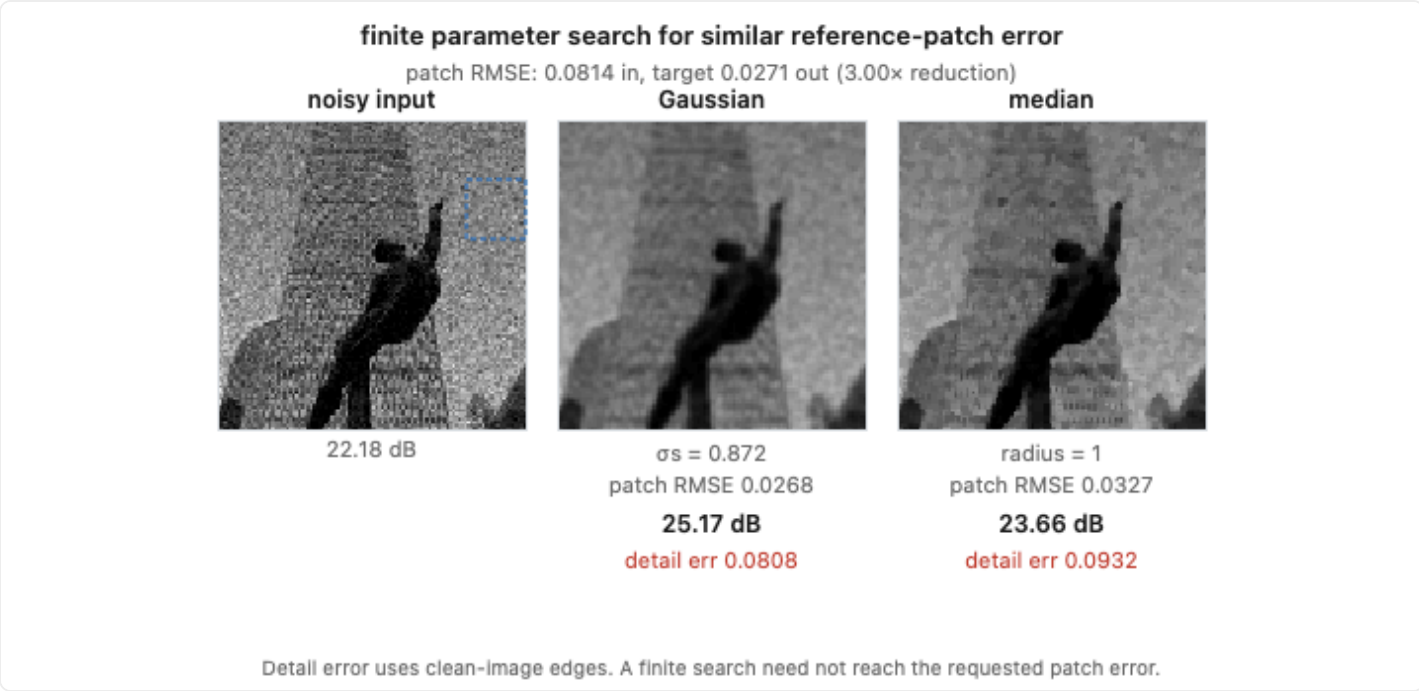
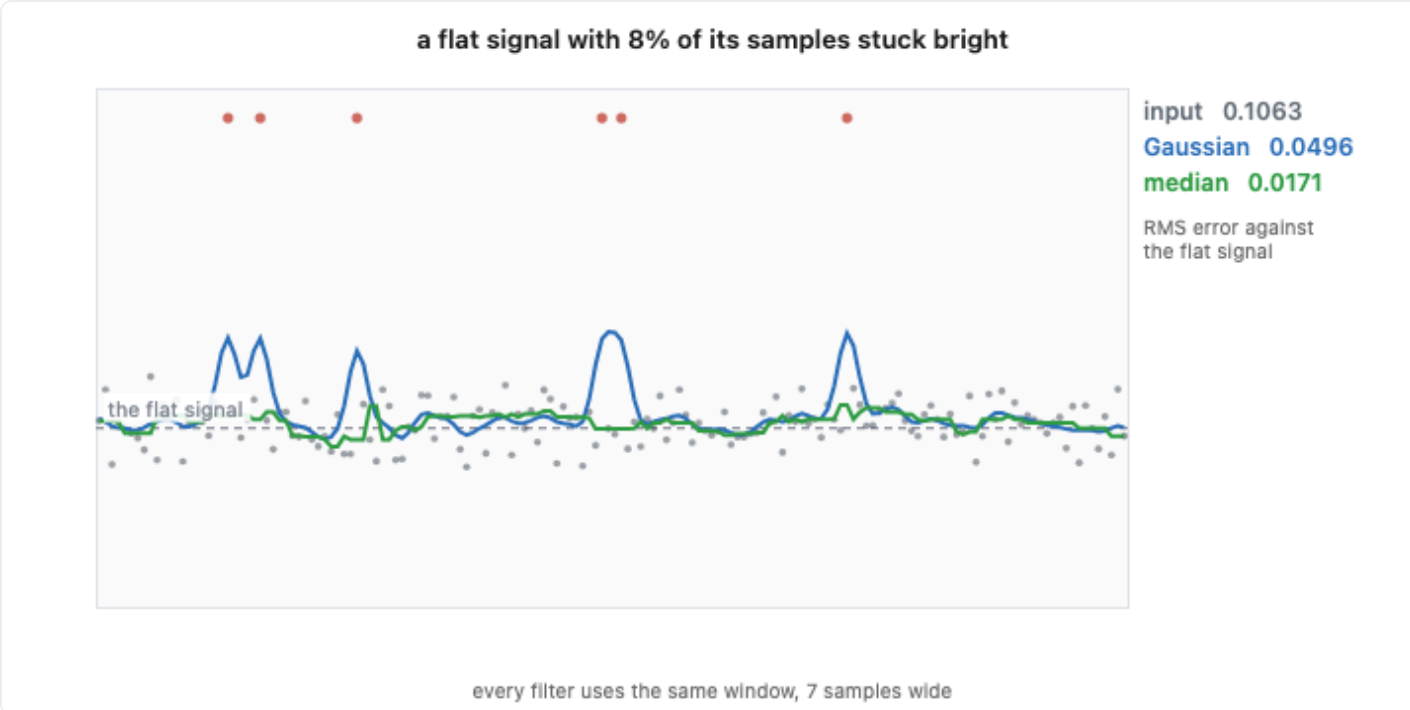


Figure 13.3.3 (reproduced). *One neighborhood, one estimate (interactive).* N values with a fraction of them corrupted, and the two estimates of the value they are all measurements of. The corruption is one-signed, a sample stuck bright rather than pushed either way, because symmetric outliers cancel in the mean and make it look robust for a reason that does not survive contact with a sensor. The mean leaves the true value as soon as anything is corrupted; the median does not move until the corrupted share passes one half. The curve underneath is the same experiment repeated at every fraction, so what you see is the behavior rather than one lucky draw. Interactive: set the number of values, the corrupted fraction and how far off they are, and watch both estimates and their error curves.



◀ 1 Gaussian | 2 + median | 3 + bilateral ▶ ⤓ settings from the previous stage

Figure 13.3.1, stage 2 (reproduced). *The same panel, with the median beside the Gaussian.* Both are searched to the same measured noise reduction in the flat patch, so what separates them is only what each one did to the rest of the picture: the Gaussian's detail error against the median's, and, in the other view, what each one removed.



◀ 1 Gaussian | 2 + median | 3 + bilateral ▶ ⤓ settings from the previous stage

Figure 13.3.2, stage 2 (reproduced). *The median on the same scanline.* Where the Gaussian turned each stuck sample into a bump, the median deletes it, as long as its window still holds a majority of clean samples: raise the impulse fraction until it does not and the median fails too. On Gaussian noise instead the order reverses slightly, the efficiency cost this section quotes.

For a constant signal with independent Gaussian errors, the sample median's large-sample standard deviation is about $\sqrt{\pi/2} = 1.253$ times the mean's. Robustness is useful when outliers are present, but has an efficiency cost when they are absent.

A median can preserve an ideal step when the correct side supplies a majority of the window. The same majority rule removes thin lines, isolated stars, and small details, and can round corners. It is nonlinear, so it has no single convolution kernel or frequency response. Repeated passes are not equivalent to one pass with a larger window. Iterating one median does, however, stop changing things: for a finite 1-D signal, repeated passes converge to a root signal that the filter leaves unchanged, made of constant runs and monotone stretches long enough to survive the window Gallagher & Wise 1981. Further passes then do nothing.

The median's majority rule explains both a strength and a failure. In a window containing mostly dark background and one extreme bright outlier, the middle sorted value remains dark. That suppresses a hot pixel, but it can also suppress a real isolated star or a thin bright line. At a broad ideal step, the side occupying most of the window determines the answer. The filter has no knowledge of which small structure the photographer intended to preserve. Inspect small real details as well as the removed outliers when choosing its window.

The bilateral filter: a neighborhood that looks at the pixels

in the book ↗

Put the two previous sections side by side and the bilateral filter is what falls out of their complaints.

The Gaussian failed by bias at a discontinuity. Its weights are decided by geometry alone, so at an edge it pours the far side into the answer, and widening it to clean the flat regions only makes that worse. The median answered the same edge by refusing to average at all: order the neighborhood, take the middle value, and a minority of unlike samples cannot move it. That buys robustness and pays for it twice. One order statistic throws away the noise reduction that averaging gives, so on a flat patch the median is the weaker estimator, and the majority rule that protects the edge deletes whatever the majority disagrees with.

The bilateral filter keeps the averaging and changes what decides the weights. Like the Gaussian it sums many neighbors, so noise still falls with the number of samples that genuinely contribute, which is the accuracy the median gave up. Like the median it declines to be dragged by values that do not belong. What differs is the reference. The median compares a neighbor against the rest of the neighborhood and keeps whatever wins the vote; the bilateral compares each neighbor against the center pixel's own measurement, so a neighbor earns its weight by agreeing with this pixel rather than with the crowd. The measurement at the pixel being filtered counts for more than all the others put together, which is what none of the fixed rules allows.

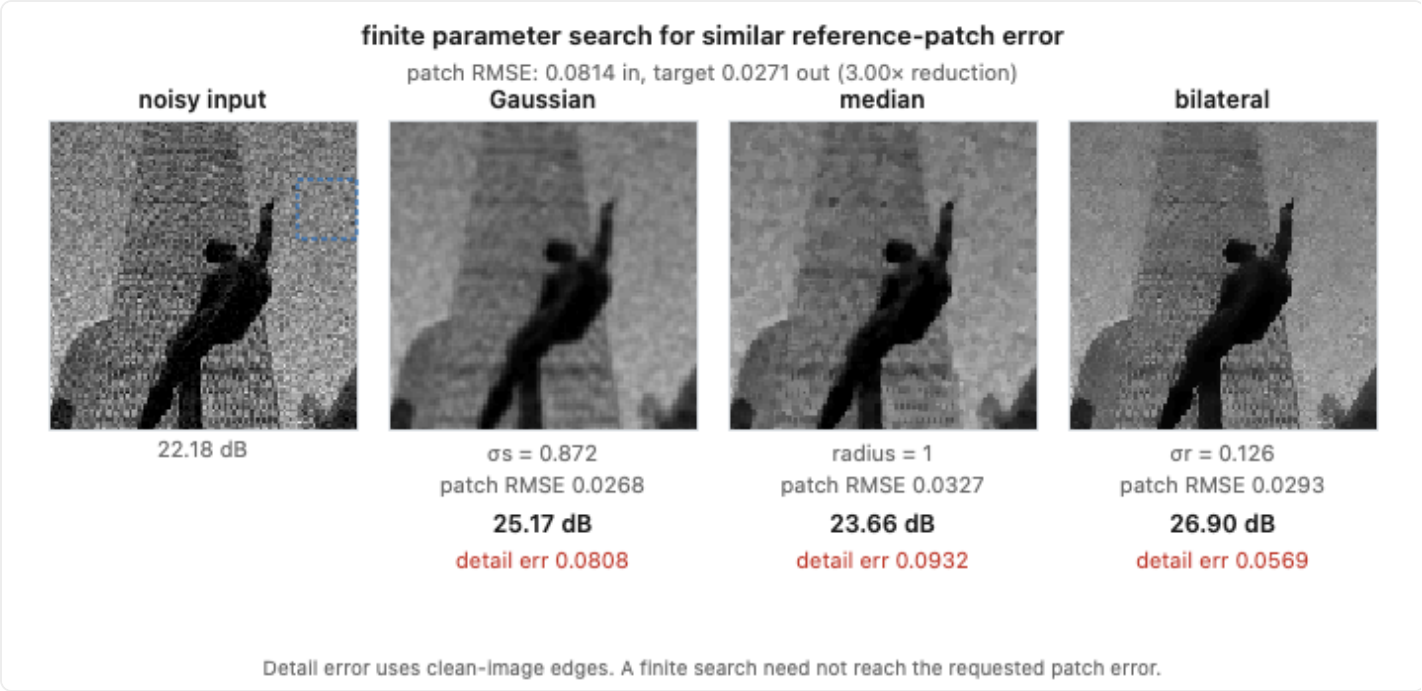
The difference is invisible on a straight edge, where half the window agrees and any sensible rule survives, and unmistakable at a corner. A pixel at the corner of a bright roof against dark sky sees roughly a quarter of its neighborhood belonging to its own surface and three quarters belonging to the other. Every rule that trusts the majority answers with the sky: the median rounds the corner off, which is exactly the failure the previous section described, and the Gaussian smears it. The bilateral asks whether a neighbor looks like this pixel, and a quarter of a window is plenty to average over. Thin structures are the same case stretched out, since a line one or two pixels wide is in the minority everywhere along its length.

Making the center the reference carries its own cost, which the formula below does nothing to hide. The center's value is itself a measurement, noise and all, so a badly corrupted pixel sets a badly chosen standard and holds on to its own error. Chapter 10.1, Bilateral filtering takes that up, including the repair of reading the weights from a pre-filtered copy.

The bilateral filter combines spatial distance with a difference in measured intensity (Tomasi & Manduchi 1998):

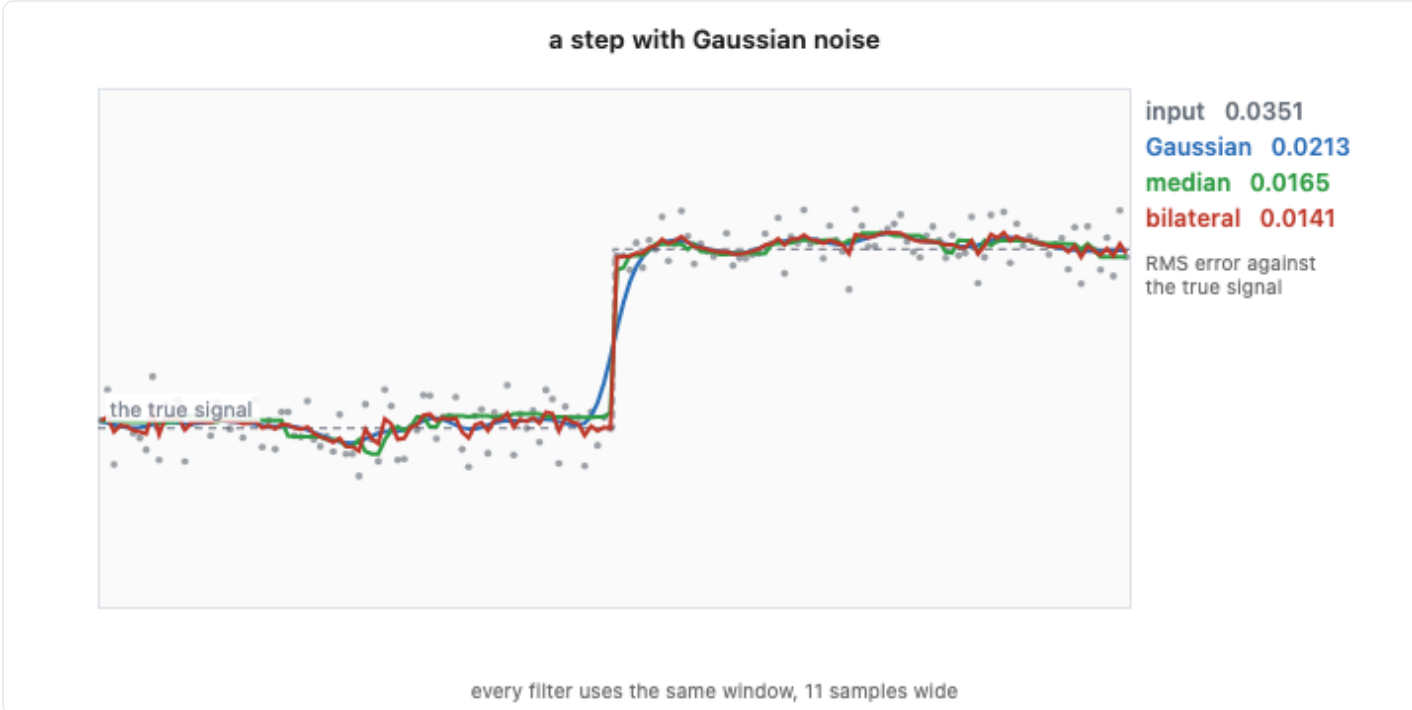
$$w_{pq} = \exp\left(-\frac{\|p-q\|^2}{2\sigma_s^2}\right)\exp\left(-\frac{(y_p-y_q)^2}{2\sigma_r^2}\right), \qquad \hat{x}_p = \frac{\sum_q w_{pq} y_q}{\sum_q w_{pq}}. \tag{18}$$

The spatial width σ_s is measured in pixels; the range width σ_r is measured in image-value units. An edge whose contrast is large compared with σ_r contributes little weight across its boundary. In the limit $\sigma_r \to \infty$, all range weights approach one and the filter becomes Gaussian smoothing. Merely making σ_r larger than the noise does not erase an edge with still larger contrast.



◀ 1 Gaussian | 2 + median | 3 + bilateral ▶ | ↓ settings from the previous stage

Figure 13.3.1, stage 3 (reproduced). *The panel complete, with the bilateral beside the other two.* All three are matched to the same measured noise reduction, so the comparison is honest: the bilateral pays the least detail error of the three here, and the view of what each removed shows why, since its residual carries the flat regions and leaves the edges alone.



◀ 1 Gaussian | 2 + median | 3 + bilateral | ▶ | ⤓ settings from the previous stage

Figure 13.3.2, stage 3 (reproduced). *And the bilateral.* On Gaussian noise it lands with the other two. On impulse noise it fails, and visibly: at a stuck sample the range weight rejects the clean neighbors, because they are the ones that differ from the center, so the filter averages the outlier with itself and keeps it. A filter that decides by value cannot fix a value that is wrong, which is why the range weight is read from a pre-cleaned copy when impulses are expected.

If σ_r is too small relative to the noise, valid neighbors appear different and get rejected. On a constant patch with independent noise, $y_p - y_q$ has standard deviation $\sqrt{2}\sigma$ for $p \neq q$. This supplies a scale for selecting the range width, although its best setting also depends on the detail to preserve.

Algorithm 4

```
for each output pixel p:
    total = 0
    weight_sum = 0
    for q in spatial_window(p):
        w = exp(-distance_squared(p,q)/(2*spatial_width²))
        w *= exp(-(y[p]-y[q])²/(2*range_width²))
        total += w * y[q]
        weight_sum += w
    out[p] = total / weight_sum
```

The spatial weights can be tabulated once for the chosen window, since they are the same at every pixel; the range weights and their sum have to be evaluated per output neighborhood, because they depend on that neighborhood's values. Keep the input separate from the output so that every weight uses the same image. The self-weight is positive and prevents a zero denominator. For color images, a shared weight computed from a color-vector distance

avoids choosing incompatible neighborhoods for different channels. Its metric must account for the chosen color space and noise scaling; Chapter 10.1, Bilateral filtering develops these choices and fast implementations.

The probe in Figure 13.3.4 shows how weights change at a boundary. Since noise affects the weights as well as the averaged values, the fixed-weight N_{eff} is a diagnostic of weight concentration, not an exact noise-reduction guarantee for the bilateral filter.

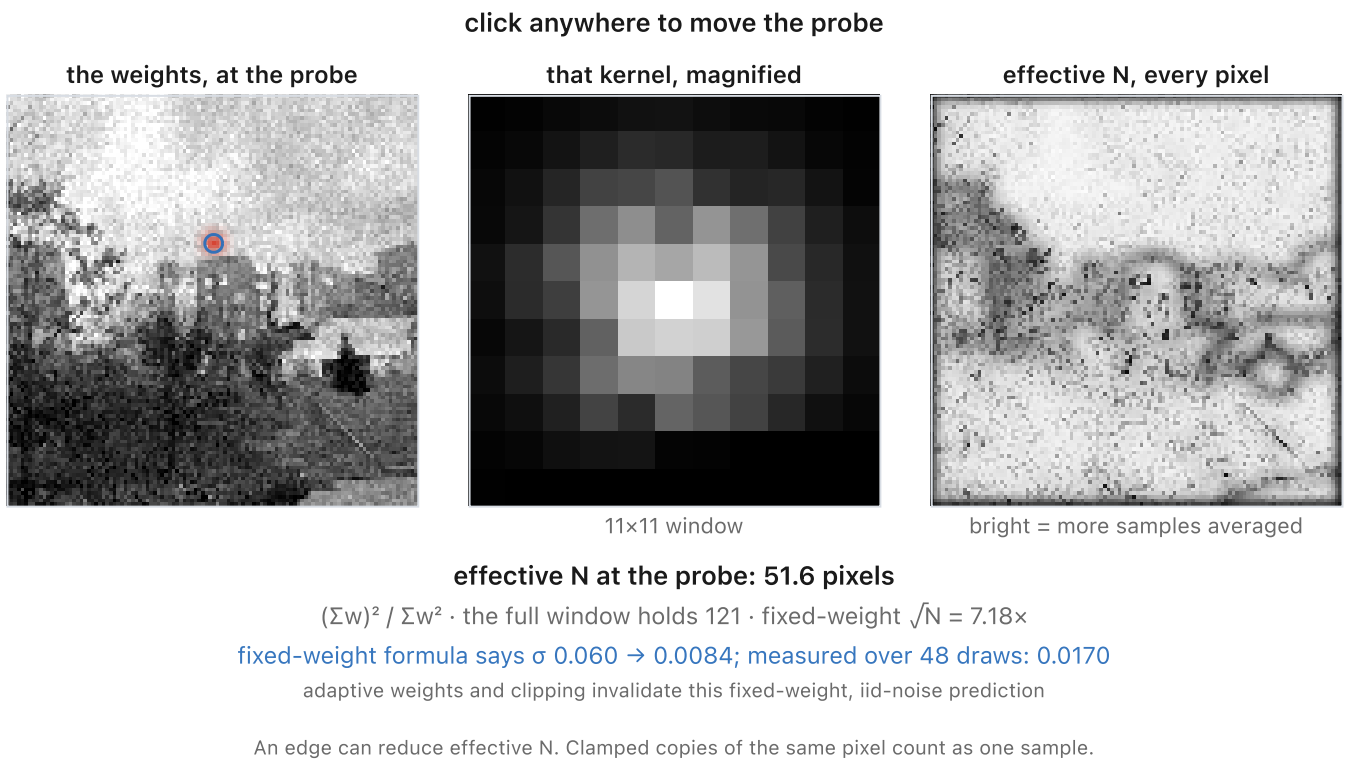


Figure 13.3.4 (reproduced). *Interactive. Click any pixel and the panel draws the weight map the bilateral filter is actually using at that pixel, over the image, as a heat overlay. On a flat region the map is the spatial Gaussian; on an edge it is a Gaussian cut in half; in texture it fragments. Sliders for the two sigmas reshape the map live, and a readout gives the effective number of averaged pixels, which is the N in the $1/\sqrt{N}$ law and therefore the noise reduction the filter can actually deliver at that pixel.* Interactive: click to place the probe, adjust σ_s and σ_r, and toggle a view that shows the effective N for every pixel at once, which maps exactly onto where the result stays noisy.

Strong or repeated filtering can flatten gradients into plateaus, retain isolated noisy pixels, or erase low-contrast texture. Similar edge-aware goals motivate anisotropic diffusion (Perona & Malik 1990), whose iterative update is a different operation.

Denoise color more than brightness

OPTIONAL
in the book ↗

Vision generally resolves luminance detail more finely than chromatic detail. Many denoisers therefore smooth chroma more strongly. This is a tolerance, not permission to erase arbitrary color structure: isoluminant colored edges and fine saturated patterns can be visible.

A luminance/chrominance representation provides separate controls for brightness grain and color blotches. Stronger chroma smoothing can clean a neutral wall while retaining luminance texture; across colored boundaries it can bleed colors. RGB methods can also learn or implement such coupling. Choose the representation and weights for the image, rather than treating one channel layout as necessary for good denoising.

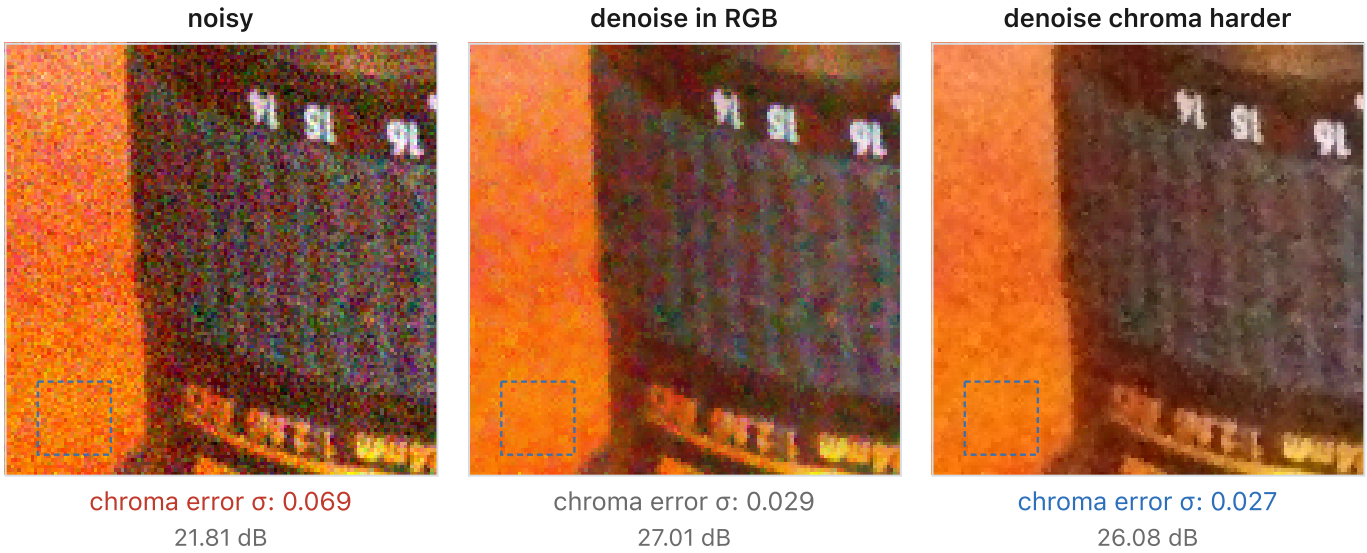


Figure 13.3.5 (reproduced). *Denoise color more than brightness. Interactive: set the input noise and the luma and chroma filter strengths and watch the comparison live. A noisy color crop (a muted, overcast coast: a sea-stack and rocks under a flat gray sky) is cleaned two ways with a per-channel bilateral filter. Working in RGB (center) applies the same filter to every channel; splitting into luma and chroma (right) lets the chroma channels be smoothed much harder while luma keeps a small spatial σ to protect the rock texture. Each bilateral has two knobs, a spatial σ in pixels and a range σ in normalized channel units, and its window is capped at a 7-pixel radius and clamped at the borders. The noise is seeded Gaussian noise added to encoded RGB and clipped to [0,1], with σ in 8-bit code values. The number under each panel is the standard deviation of the chroma error in a sky patch, measured against the clean crop after clipping; it ignores any mean color shift and says nothing about detail, so look at the rocks and at colored boundaries too. Heavier chroma filtering usually wins here because the eye resolves detail in brightness more than in color, the fact behind chroma subsampling, but it can also erase small colored detail or bleed color across an edge. The low-saturation scene makes that failure rare, not impossible.*

What each filter destroys

OPTIONAL
in the book ↗

Compare results on flat regions, edges, and texture. Matching residual noise in a constant patch gives one useful operating point, but does not determine a universal ranking. An impulse-removal test and a Gaussian-noise test reward different behavior. Also inspect the removed signal $y - \hat{x}$: recognizable structures suggest lost detail, while their absence does not prove the estimate is correct.

Gaussian smoothing blurs boundaries. The median suppresses minority features along with outliers. The bilateral filter depends on contrasts being distinguishable from noise. These are consequences of the estimators, not parameters that can always be tuned away.

A Gaussian blur removes noise but smears every edge; the bilateral filter only averages neighbours of a similar colour, so noise drops while the leaf veins and the flower's boundary stay sharp.

Figure 13.3.6 (reproduced). *Three denoisers on one noisy crop (a real color photo with realistic shot-plus-read noise). Left, the noisy input. Center, a Gaussian blur: the noise is gone but so are the edges; the whole image is smeared, because the filter averages across boundaries it cannot see. Right, the bilateral filter: the noise is gone* and *the edges are intact, because it only averages neighbors whose color is similar enough to belong to the same region. Same goal, two very different outcomes; the difference is whether the filter respects edges.*

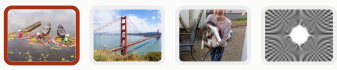
> 💡 **BIG LESSON (LOCAL DENOISING TRADES NOISE REDUCTION AGAINST SIGNAL MIXING)**
>
> A larger neighborhood supplies more samples but may mix different scene values. Edge-aware and robust rules reduce some mixing errors; they do not establish that every retained feature is signal or every removed feature is noise.

With every filter of this chapter on the table, the most useful exercise is to run them against the same noisy image and watch the trade-offs move. The lab below splits one image into four panes, the split following the mouse, each pane with its own filter, its own sliders and its own live score (Figure 13.3.7). Put the Gaussian beside the bilateral and slide across an edge; set a pane to the amplified error against the clean original to see where each filter cheats, or to the difference of two panes to compare them directly.

Algorithm lab — Denoising

nine ways to average without averaging across edges

Task Denoising



noise σ 15

1 2 4 lines

Compare | Algorithms | Pseudocode | Slides

A 29.8 dB · SSIM 0.885 · 30 ms

Gaussian blur

σ 1.2

B 32.4 dB · SSIM 0.927 · 0 ms

Bilateral

σ spatial 2

σ range 30

channels RGB together

C 28.9 dB · SSIM 0.828 · 0 ms

Pyramid coring

threshold 14

shrink soft (subtract T)

D 32.8 dB · SSIM 0.930 · 0 ms

Non-local means

strength h 12

patch 3x3

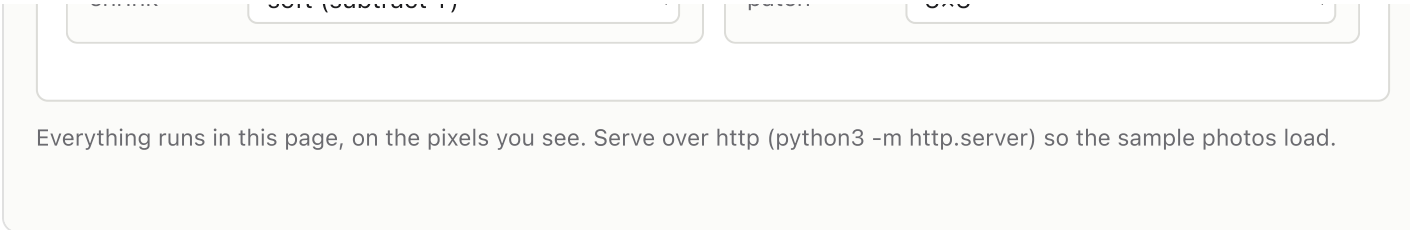


Figure 13.3.7 (reproduced). *Interactive: the denoising lab. One noisy image (seeded Gaussian noise on a clean original, with σ under your control) cleaned by up to four filters at once, chosen from nine classics: Gaussian, median, local Wiener, bilateral (including the luma/chroma split that denoises color harder than brightness), guided filter, anisotropic diffusion, total variation, pyramid coring, and non-local means, in a mouse-driven four-way split with per-pane parameter sliders and live PSNR and SSIM against the clean original. A pane can instead show the noisy input, the ground truth, the difference of two panes, or a pane's amplified error. The numbers confirm the chapter: a Gaussian can raise PSNR yet smear edges, while the affinity-based filters clean the flats and keep the edges.*

The limitation of a small window motivates the next methods. A faint texture may have too little evidence locally to distinguish it from noise. Multiscale methods look for related structure across scales; non-local methods search for repeated patches farther away; learned denoisers use regularities acquired from other images. Each broadens the evidence available to the estimate, while introducing its own assumptions about which structures should agree.

Non-local means and BM3D

Non-local means: similarity as the weight

OPTIONAL
in the book ↗

Non-local means (Buades et al. 2005) averages observed values with weights derived from patch similarity. One common version compensates for the expected noise contribution:

$$
\begin{aligned}
w(p,q) &= \exp\left[-\frac{\max(d^2(p,q)-2\sigma^2,0)}{h^2}\right], \\
Z(p) &= \sum_{q\in S_p} w(p,q), \qquad \hat{x}_p = \frac{1}{Z(p)}\sum_{q\in S_p} w(p,q)y_q.
\end{aligned}
\tag{19}
$$

Here S_p is a bounded search window and $h > 0$ has the same intensity units as σ. The normalization by m in the distance is essential: subtracting $2\sigma^2$ from an unnormalized sum would use the wrong noise scale. The subtraction and truncation are a practical weighting rule, not an unbiased estimator of similarity.

Algorithm 5

```
for p in image:
    w = empty_map()
    candidates = search_window(p) excluding p
    for q in candidates:
        d2 = mean((patch(y,p) - patch(y,q))**2)
        w[q] = exp(-max(d2 - 2*sigma**2, 0) / h**2)
    if candidates is empty:
        out[p] = y[p]
        continue
    w[p] = max(w[q] for q in candidates)
    if sum(w.values()) == 0:
        out[p] = y[p]
        continue
    out[p] = sum(w[q]*y[q] for q in candidates plus p) / sum(w.values())
```

This self-weight convention prevents an exact self-match from overwhelming much smaller neighboring weights. If the candidate set is empty or all weights underflow to zero, return y_p. Use a consistent boundary extension when extracting patches. For color images, distances can combine channels and the same weights can filter all channels, with the noise model adjusted for their variances and correlations.

Larger search windows cost more and offer more accidental matches as well as useful ones. Larger h accepts less similar patches and smooths more strongly. Patchwise variants estimate whole patches and aggregate their overlaps. These details distinguish implementations of Chapter 10.2, Non-local means that otherwise share the same equation.

The method also carries a bias. It averages a patch with the patches that resemble it, so it pulls the image slightly toward its own repeated structures, and details with no match elsewhere, a crack in one brick, a single hair, get fewer effective samples and either stay noisy or are smoothed toward their nearest look-alikes. This is the first appearance of a failure mode that every method borrowing evidence from elsewhere in the image shares: good at what repeats, weaker at what does not.

The direct search repeats much of its arithmetic. For a fixed displacement, form an image of squared differences between the input and its shifted copy. A box sum, implemented with a summed-area table Crow 1984 or separable running sums, then gives all patch distances for that displacement. This reduces the cost of comparing large patches. A bounded search window limits the number of displacements; a pilot denoising pass can help select candidates, provided the final average still uses the intended measurements. These accelerations change different parts of the computation and can be combined.

BM3D: group, transform, shrink, aggregate

OPTIONAL
in the book ↗

BM3D (Dabov et al. 2007) combines patch matching with Chapter 13.4, Pyramid and wavelet denoising. For each reference patch it selects similar patches, stacks them, applies a transform within each patch and along the stack,

shrinks the coefficients, and transforms back. The third dimension indexes patches; it is not a spatial axis in the photograph.

If the patches were identical, an orthonormal transform along the stack with a constant first basis vector would concentrate their shared signal in that component. Real matches differ, so other components also contain signal. Shrinkage can retain these differences when they are sufficiently strong. It can also erase a genuine detail present in only one patch.

The method has two passes. The first uses hard thresholding and aggregates the reconstructed patches into a basic estimate. The second uses that estimate to select groups and estimate coefficient power, but filters groups extracted from the original noisy image. In a simplified orthonormal-transform description, its empirical Wiener gains are

$$g_i = \frac{|c_i^{\text{basic}}|^2}{|c_i^{\text{basic}}|^2 + \sigma^2}, \qquad \hat{c}_i = g_i c_i^{\text{noisy}}. \tag{20}$$

A small pilot coefficient gives a small gain; a large one largely retains the noisy coefficient. With other transform normalizations, replace σ^2 by the appropriate coefficient noise variance. Group selection and overlapping samples make the actual statistics more complicated than this idealized expression.

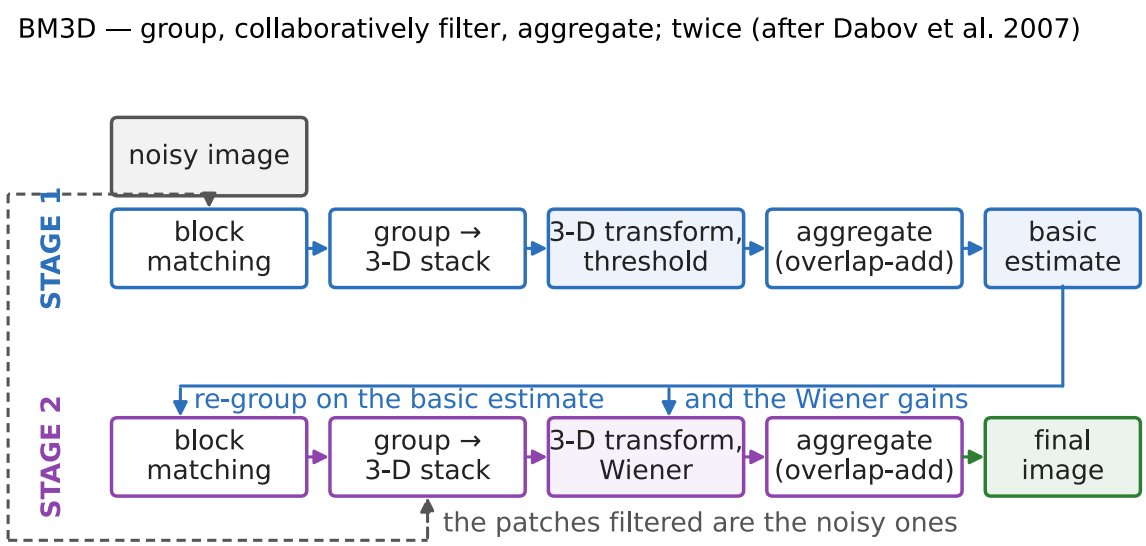


Figure 13.6.2 (reproduced). *The two-stage BM3D pipeline as a schematic: group similar blocks across the search window → 3-D collaboratively filter the stack → aggregate the filtered blocks back. Each stage runs twice: first a hard-threshold shrinkage to form a basic estimate, then a Wiener filter that uses that basic estimate as its pilot. This is the structure behind the interactive demo below Dabov et al. 2007.*

Algorithm 6

```
basic = aggregate(
    inverse_transform(hard_threshold(transform(group_from(y)), threshold))
    for each reference patch, with group matches found in y)

for each reference patch:
    positions = find_matches(basic)
    c_basic = transform(extract_group(basic, positions))
    c_noisy = transform(extract_group(y, positions))
    gains = abs(c_basic)**2 / (abs(c_basic)**2 + coefficient_noise_variance)
    accumulate(inverse_transform(gains * c_noisy), positions, group_weights)
out = accumulated_values / accumulated_weights
```

This outlines the data flow; a reproducible implementation must also specify transforms, matching thresholds, group sizes, spatial windows, and aggregation weights. Every reconstructed patch returns to its own original position. Overlapping estimates are combined with spatial and group reliability weights, then normalized. They share input pixels and are not independent new observations.

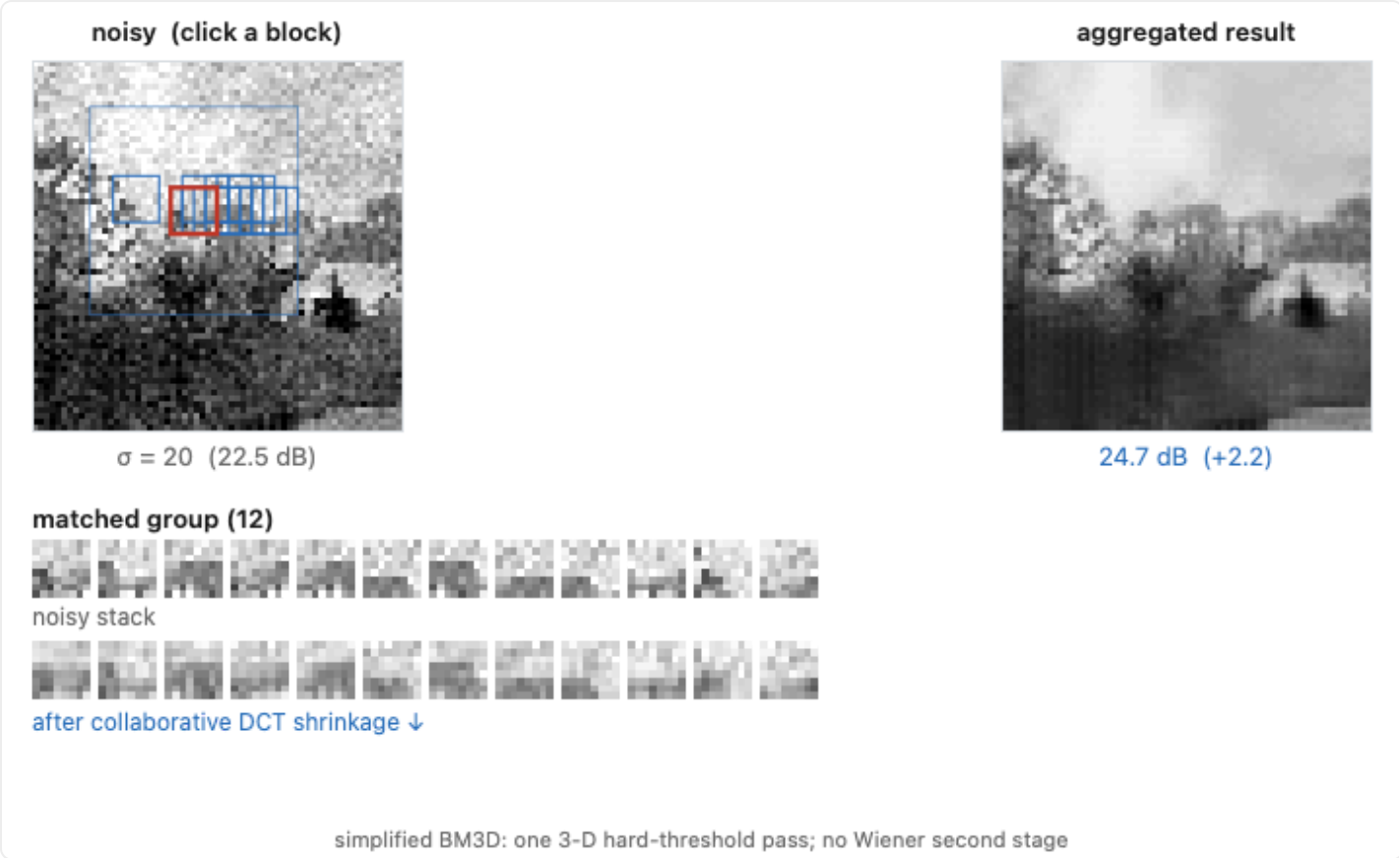


Figure 13.6.3 (reproduced). *Interactive. The three moves, made visible. Click a reference block in the noisy image and the demo finds its best matches across the search window and stacks them; the filmstrip shows that group before and after collaborative filtering: a 2-D DCT of each block and a 1-D DCT along the stack form one 3-D transform, and each of its coefficients survives only if it clears a hard threshold, which keeps what the group shares because shared content concentrates into a few large coefficients; the filtered blocks then go back to their positions and the overlaps are averaged. Move the reference into a repeating structure and the group fills with genuine matches, move it onto a unique object and the group is padded with poor ones, which is where the method's own bias becomes visible. The demonstration runs only BM3D's first, hard-thresholding stage and omits its second, Wiener-filtering stage.* Interactive: click to place the reference block, set the threshold and the group size, and watch the group, the shrunk coefficients and the aggregated result update together.

Suppose four matched patches were exactly equal except for independent noise. Along the stack dimension, their common component is constant, while their differences contain only noise. An orthonormal transform separates that constant direction from difference directions. Shrinking the latter suppresses disagreement while retaining the common patch. Real matches are only approximate: a small edge displacement or a texture difference can also occupy a difference direction. BM3D's grouping, shrinkage, and overlap aggregation must therefore balance agreement against genuine variation. This explains why grouping quality matters as much as the transform itself.

Learned denoisers

OPTIONAL
in the book ↗

Wavelet shrinkage assumes that useful image structure has a sparse representation. BM3D Dabov et al. 2007 searches for repetition within the observed photograph. A learned denoiser fits a mapping from noisy images to clean targets, using regularities across a training collection as well as spatial context within each input. Its behavior depends on the training images, corruption model, architecture, and loss.

For paired training with squared error, the ideal population solution is the posterior mean:

$$\arg\min_f \mathbb{E}\|f(y) - x\|_2^2 = \mathbb{E}[x \mid y]. \tag{21}$$

This characterizes an unrestricted optimum. Finite networks, finite data, and imperfect optimization only approximate it. A network can also be trained with another loss, in which case the optimal statistic changes. These distinctions matter when a trained denoiser is later interpreted as a score estimator in Chapter 13.11, Denoising as a universal prior.

Generative denoising with diffusion models

OPTIONAL
in the book ↗

In a very noisy photograph of a brick wall, several positions of a mortar line may be compatible with the measurement. Averaging those possibilities produces a softened edge. Choosing one produces a sharper edge whose position may be wrong. Generative denoising makes this choice explicit by producing plausible reconstructions conditioned on the noisy image.

The distinction matters where the observation leaves substantial uncertainty. It is not true that all posterior means look blurred or all posterior samples look realistic. Both depend on the scene, measurement, and learned distribution. A model that excludes the true structure can produce confident, consistent errors.

Denoising TLDR

OPTIONAL
in the book ↗

For ordinary photographic work, start with a pretrained denoiser designed for your input. The practical choices are what to capture, how much detail to retain, and how to judge the result.

- Preserve raw data when available. Match the denoiser to the camera data and processing domain it expects; a model for raw sensor noise and one for processed RGB solve different problems. In a custom pipeline, check black levels, image scaling, and noise units before tuning strength (Chapter 13.1, Noise models, transforms, and benchmarks).
- Use multiple frames when motion and capture time permit. Four aligned, independent exposures of unchanged duration halve noise standard deviation. Moving subjects and occlusions reduce that gain locally; splitting a

fixed total exposure adds read noise. Use a temporal method for video and inspect ghosting, flicker, and delay (Chapter 13.2, Denoising by averaging, Chapter 13.10, Video denoising).

- Set strength by the detail you need to preserve. Inspect hair, foliage, fabric, small text, and isolated highlights at full resolution and at the intended display size. Some remaining grain is preferable to erased texture. More plausible texture from a generative method does not establish that the camera recorded it. Keep the original and record whether the output uses generative reconstruction (Chapter 13.9, Generative denoising with diffusion models).
- Test on your own images. Compare identical inputs, inspect the removed residual, and use reference error and perceptual measures when a reliable reference exists. Include difficult scenes and measure latency and memory. A benchmark winner may behave differently on your camera (Chapter 13.12, Measuring a denoiser).
- For implementation, use BM3D Dabov et al. 2007 as a baseline requiring no learned weights. Reserve median filtering for sparse impulses or defects; ordinary spatial smoothing also removes detail. Train a new model when available models fail your domain. Noisy-target training can help, but requires the independence and alignment assumptions developed in Chapter 13.8, Denoising without clean data.

Recap: big lessons in this lecture

💡 **BIG LESSON (L2.41) — BRIGHT PIXELS ARE *NOISIER* (MORE ABSOLUTE NOISE) BUT HAVE *BETTER SNR***

Shot noise is Poisson, so its **variance grows linearly with the signal** ($\sigma^2 \propto N$) and its **standard deviation grows only as the square root** ($\sigma \propto \sqrt{N}$). Two consequences pull in opposite directions: the *absolute* noise is **larger in highlights** than in shadows (a 10,000-photon highlight has $\sigma = 100$; a 100-photon shadow has $\sigma = 10$), yet the **SNR** $= N/\sqrt{N} = \sqrt{N}$ **is higher in the highlights**: the signal outgrows the noise. So bright areas are noisier in absolute terms but **cleaner to the eye**; the shadows have the worst SNR (Figure 2.8.4, Figure 2.8.6). This is why **exposing to the right** and brighter scenes look smoother, and why we **gamma/log-encode** so equal *ratios* get equal steps.

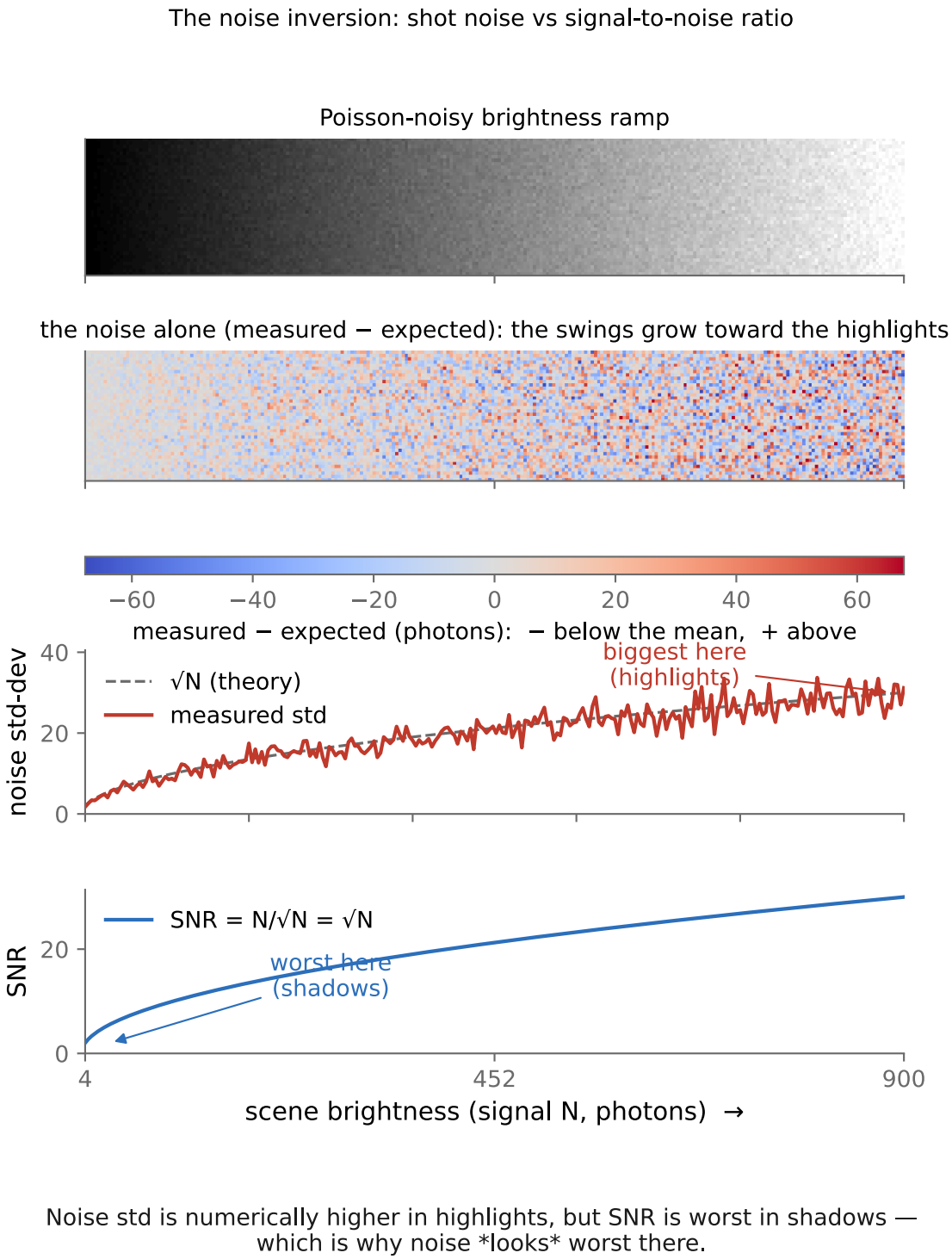


Figure 2.8.6 (reproduced). *Noise is numerically worse in highlights but looks worse in shadows. Three panels of the same brightness ramp, each labeled so the good direction is unambiguous. A standard-deviation map (absolute noise, higher* = worse*) is* brightest in the highlights*, since shot noise* $\sigma = \sqrt{N}$ *grows with brightness; an SNR map (* $= \sqrt{N}$*, higher* = better*) is* worst in the shadows*, since the ratio falls where photons are few; and a* ***noise-only*** *panel shows a single frame's residual, the measurement minus its expected value, on a diverging colormap (blue for negative excursions, red for positive, white at zero) so both signs read at once, the residual staying near zero-mean everywhere yet visibly growing in amplitude toward the highlights. Perception tracks the ratio, so shadows look noisiest, the reason for exposing-to-the-right and for gamma/log encoding. All three maps are computed in linear light and gamma-encoded only for display, so the noise magnitudes shown are faithful.*

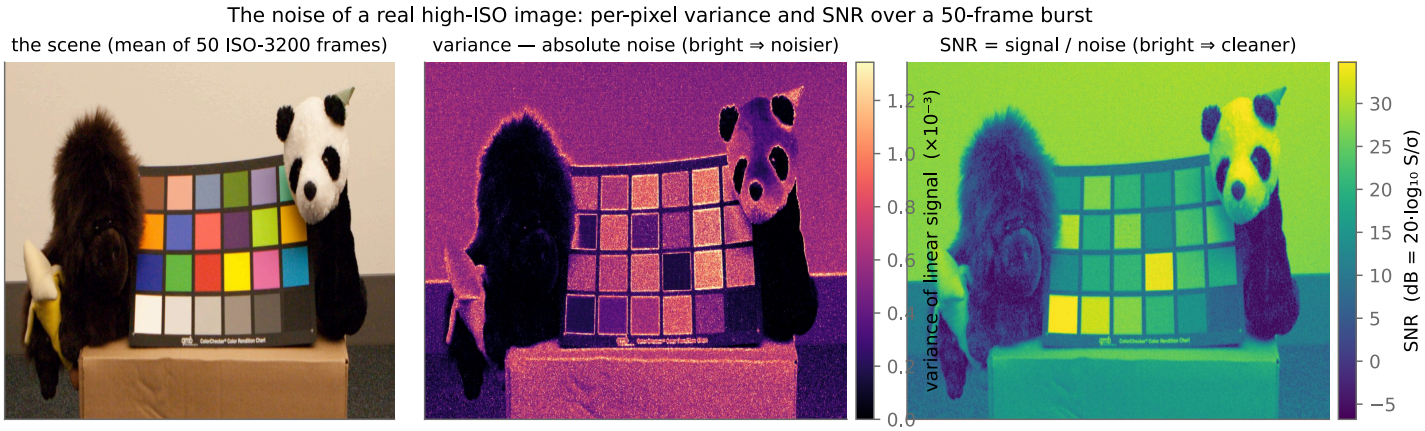


Figure 2.8.4 (reproduced). *The noise of a real high-ISO image, and its SNR. From a 50-frame ISO-3200 burst of one static scene: the scene itself (left, mean of the burst), the per-pixel variance across the burst (middle), and the per-pixel SNR = signal/noise (right). Computed on linearized values, so shot noise behaves physically. The variance map is brightest where the scene is bright (absolute noise grows with signal), while the SNR map is also brightest in the bright areas and worst in the shadows, because noise only grows as √signal so the ratio improves with light. Bright pixels are noisier in absolute terms yet cleaner in relative terms. This is the raw material the affine model below summarizes.*

💡 BIG LESSON (L10.1) — EDGE-PRESERVING = AFFINITY

Use the color / intensity difference between two pixels as how much they *belong together*, an affinity, then average (or connect) pixels in proportion to it. Pixels with high affinity are treated as measurements of the same underlying thing and get averaged together; pixels with low affinity, across an edge, are kept apart. The bilateral filter's range weight g_r is the first full instance, but the *same* affinity drives the bilateral grid, the guided filter, non-local means (NL-means), colorization, and matting / segmentation that follow. Edge-preserving is affinity. *(This is L10.1* show*, registered in Part 13, Denoising where the bilateral first appeared for noise; this is its full edge-preserving treatment.)*

💡 BIG LESSON — DEFINE THE SCALE OF AN UNNORMALIZED UPDATE

Removing the bilateral normalizer changes the filter. Specify a normalized spatial kernel or an explicit step size, then test constants, impulses and edges. A small total affinity makes a small correction; it does not establish that an isolated value is correct.

💡 BIG LESSON (L13.1) — CAPTURE THE FULL SET, DECIDE LATER

Recording several captures lets us choose or combine their measurements afterward. A filter wheel is the plain example: shoot the scene through a red, a green and a blue filter (or through a polarizer at several angles) and the choice of how to combine the bands can be made later, at the desk. Exposure brackets, focal stacks, and panoramas extend this idea to different capture settings.

💡 BIG LESSON (L13.2) — MULTIPLE-EXPOSURE IMAGING BREAKS THE LIMITS OF A SINGLE CAPTURE

Several compatible measurements can reduce uncertainty or cover information missing from one frame. This is a different benefit from the previous lesson. Deferring a choice picks among things one exposure could have recorded; combining exposures can exceed what any single exposure could record, such as a noise level below one frame's, a dynamic range wider than the sensor's, or a depth of field deeper than the aperture allows. The reconstruction must account for differences in exposure, focus, viewpoint, or scene motion.

💡 BIG LESSON (THE $1/\sqrt{N}$ LAW)

Averaging four independent measurements of the same signal halves the noise standard deviation. Averaging 100 reduces it by a factor of ten. These statements require zero-mean errors and unchanged per-frame variance. Lowering the effective noise floor can extend the usable range, as described in L2.46 show, provided the captures have not clipped the signal.

💡 BIG LESSON (LOCAL DENOISING TRADES NOISE REDUCTION AGAINST SIGNAL MIXING)

A larger neighborhood supplies more samples but may mix different scene values. Edge-aware and robust rules reduce some mixing errors; they do not establish that every retained feature is signal or every removed feature is noise.