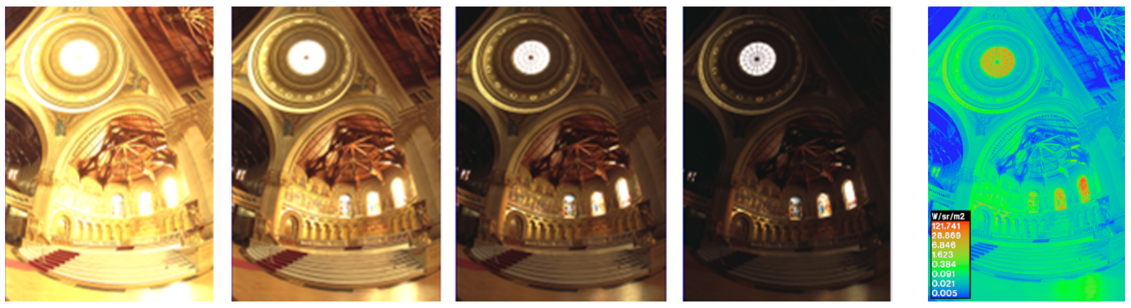
Chapter 10

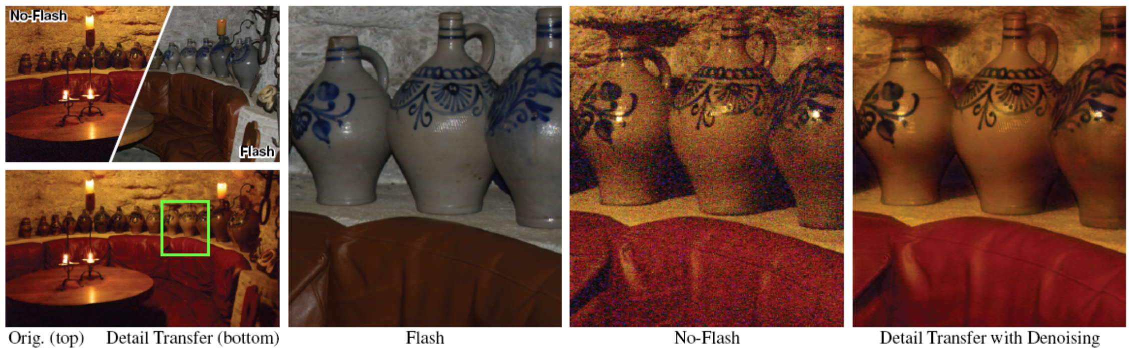
Computational photography

10.1 Photometric calibration . . . 486
10.1.1 Radiometric response function . . . 486
10.1.2 Noise level estimation . . . 488
10.1.3 Vignetting . . . 489
10.1.4 Optical blur (spatial response) estimation . . . 491
10.2 High dynamic range imaging . . . 494
10.2.1 Tone mapping . . . 499
10.2.2 *Application*: Flash photography . . . 506
10.3 Super-resolution, denoising, and blur removal . . . 508
10.3.1 Color image demosaicing . . . 515
10.3.2 Lens blur (bokeh) . . . 517
10.4 Image matting and compositing . . . 518
10.4.1 Blue screen matting . . . 519
10.4.2 Natural image matting . . . 521
10.4.3 Optimization-based matting . . . 524
10.4.4 Smoke, shadow, and flash matting . . . 527
10.4.5 Video matting . . . 528
10.5 Texture analysis and synthesis . . . 529
10.5.1 *Application*: Hole filling and inpainting . . . 531
10.5.2 *Application*: Non-photorealistic rendering . . . 532
10.5.3 Neural style transfer and semantic image synthesis . . . 534
10.6 Additional reading . . . 537
10.7 Exercises . . . 538

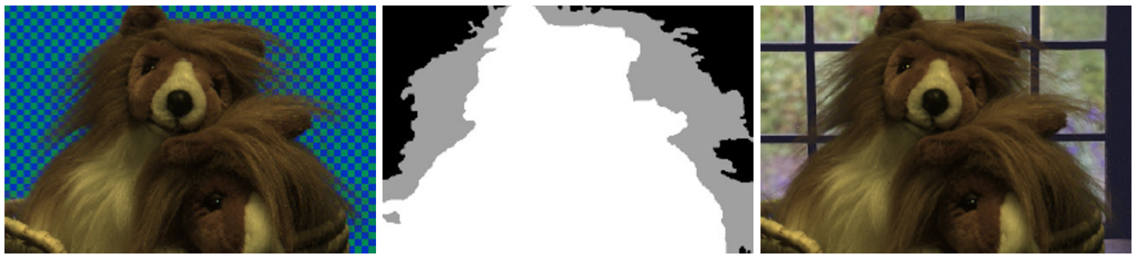
R. Szeliski, *Computer Vision*, Texts in Computer Science,
https://doi.org/10.1007/978-3-030-34372-9_10



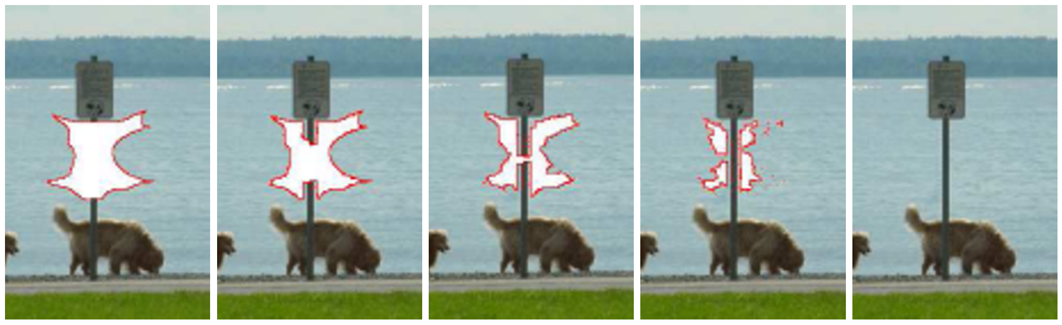
(a)


No-Flash
Flash
Orig. (top) Detail Transfer (bottom)
Flash
No-Flash
Detail Transfer with Denoising

(b)



(c)



(d)

Figure 10.1 Computational photography: (a) merging multiple exposures to create high dynamic range images (Debevec and Malik 1997) © 1997 ACM; (b) merging flash and non-flash photographs; (Petschnigg, Agrawala *et al.* 2004) © 2004 ACM; (c) image matting and compositing; (Chuang, Curless *et al.* 2001) © 2001 IEEE; (d) hole filling with inpainting (Criminisi, Pérez, and Toyama 2004) © 2004 IEEE.

Of all the advances in computer vision in the last decade, computational photography has arguably had the most widespread commercial impact. In 2010, the seminal *Frankencamera* paper by Adams, Talvala *et al.* (2010) had just been released, as had one of the first widely used in-camera panoramic image stitching apps.[1] Fast forward to 2020, and every smartphone now has built-in panoramic stitching, high dynamic range (HDR) exposure merging, and multi-image denoising and super-resolution (Hasinoff, Sharlet *et al.* 2016; Wronski, Garcia-Dorado *et al.* 2019; Liba, Murthy *et al.* 2019), and the newest phones are also simulating shallow depth of field (bokeh) with multiple lenses or dual pixels (Barron, Adams *et al.* 2015; Wadhwa, Garg *et al.* 2018; Garg, Wadhwa *et al.* 2019; Zhang, Wadhwa *et al.* 2020).

In Section 8.2, we described how to stitch multiple images into wide field of view panoramas, allowing us to create photographs that could not be captured with a regular camera. This is just one instance of *computational photography*, where image analysis and processing algorithms are applied to one or more photographs to create images that go beyond the capabilities of traditional imaging systems.

In this chapter, we cover a number of additional computational photography algorithms. We begin with a review of photometric image calibration (Section 10.1), i.e., the measurement of camera and lens responses, which is a prerequisite for many of the algorithms we describe later. We then discuss *high dynamic range imaging* (Section 10.2), which captures the full range of brightness in a scene through the use of multiple exposures (Figure 10.1a). We also discuss *tone mapping operators*, which map wide-gamut images back into regular display devices such as screens and printers, as well as algorithms that merge flash and regular images to obtain better exposures (Figure 10.1b).

Next, we discuss how the resolution and visual quality of images can be improved either by merging multiple photographs together or using sophisticated image priors or deep networks (Section 10.3). This includes algorithms for extracting full-color images from the patterned Bayer mosaics present in most cameras.

In Section 10.4, we discuss algorithms for cutting pieces of images from one photograph and pasting them into others (Figure 10.1c). In Section 10.5, we describe how to generate novel textures from real-world samples for applications such as filling holes in images (Figure 10.1d). We close with a brief overview of *non-photorealistic rendering* (Section 10.5.2), which can turn regular photographs into artistic renderings that resemble traditional drawings and paintings, and a discussion of neural network approaches to style transfer and semantic image synthesis (Section 10.5.3.

One topic that we do not cover extensively in this book is novel computational sensors, optics, and cameras. A nice survey can be found in an article by Nayar (2006), the book by Raskar and Tumblin (2010), and research papers such as Levin, Fergus *et al.* (2007). Some related discussion can also be found in Sections 10.2 and 14.3.

A good general-audience introduction to computational photography can be found in the article by Hayes (2008) as well as survey papers by Nayar (2006), Cohen and Szeliski (2006), Levoy (2006), and Debevec (2006).[2] Raskar and Tumblin (2010) give extensive coverage of topics in this area, with particular emphasis on computational cameras and sensors. The sub-field of high dynamic range imaging has its own book discussing research in this area (Reinhard, Heidrich *et al.* 2010), as well as a wonderful book aimed more at professional photographers (Freeman 2008).[3] A good survey of image matting is provided by Wang and Cohen (2009).

There are also several courses on computational photography where the instructors have provided extensive online materials, e.g., Yannis Gkioulekas' class at Carnegie Mellon,[4] Alyosha Efros'

[1] https://en.wikipedia.org/wiki/Photosynth#Mobile_apps

[2] See also the two special issue journals edited by Bimber (2006) and Durand and Szeliski (2007).

[3] Gulbins and Gulbins (2009) discuss related photographic techniques.

[4] CMU 15-463, http://graphics.cs.cmu.edu/courses/15-463

class at Berkeley,[5] Frédo Durand's Computation Photography course at MIT,[6] Marc Levoy's class at Stanford,[7] and a series of SIGGRAPH courses on Computational Photography.[8]

10.1 Photometric calibration

Before we can successfully merge multiple photographs, we need to characterize the functions that map incoming irradiance into pixel values and also the amount of noise present in each image. In this section, we examine three components of the imaging pipeline (Figure 10.2) that affect this mapping. For a more comprehensive, tunable model of modern digital camera processing pipelines, see the recent paper by Tseng, Yu *et al.* (2019).

The first is the *radiometric response function* (Mitsunaga and Nayar 1999), which maps photons arriving at the lens into digital values stored in the image file (Section 10.1.1). The second is *vignetting*, which darkens pixel values near the periphery of images, especially at large apertures (Section 10.1.3). The third is the *point spread function*, which characterizes the blur induced by the lens, anti-aliasing filters, and finite sensor areas (Section 10.1.4).[9] The material in this section builds on the image formation processes described in Sections 2.2.3 and 2.3.3, so if it has been a while since you looked at those sections, please go back and review them.

10.1.1 Radiometric response function

As we can see in Figure 10.2, a number of factors affect how the intensity of light arriving at the lens ends up being mapped into stored digital values. Let us ignore for now any non-uniform attenuation that may occur inside the lens, which we cover in Section 10.1.3.

The first factors to affect this mapping are the aperture and shutter speed (Section 2.3), which can be modeled as global multipliers on the incoming light, most conveniently measured in *exposure values* ($\log_2$ brightness ratios). Next, the analog to digital (A/D) converter on the sensing chip applies an electronic gain, usually controlled by the ISO setting on your camera. While in theory this gain is linear, as with any electronics non-linearities may be present (either unintentionally or by design). Ignoring, for now, photon noise, on-chip noise, amplifier noise, and quantization noise, which we discuss shortly, you can often assume that the mapping between incoming light and the values stored in a RAW camera file (if your camera supports this) is roughly linear.

If images are being stored in the more common JPEG format, the camera's image signal processor (ISP) next performs Bayer pattern demosaicing (Sections 2.3.2 and 10.3.1), which is a mostly linear (but often non-stationary) process. Some sharpening is also often applied at this stage. Next, the color values are multiplied by different constants (or sometimes a 3×3 color twist matrix) to perform color balancing, i.e., to move the white point closer to pure white. Finally, a standard gamma is applied to the intensities in each color channel and the colors are converted into YCbCr format before being transformed by a DCT, quantized, and then compressed into the JPEG format (Section 2.3.3). Figure 10.2 shows all of these steps in pictorial form.

Given the complexity of all of this processing, it is difficult to model the camera response function (Figure 10.3a), i.e., the mapping between incoming irradiance and digital RGB values, from

[5]Berkeley CS194-26/294-26, https://inst.eecs.berkeley.edu/~cs194-26/fa20

[6]MIT 6.815/6.865, https://stellar.mit.edu/S/course/6/sp15/6.815

[7]Stanford CS 448A, https://graphics.stanford.edu/courses/cs448a-10

[8]https://web.media.mit.edu/~raskar/photo.

[9]Additional photometric camera and lens effects include sensor glare, blooming, and chromatic aberration, which can also be thought of as a spectrally varying form of geometric aberration (Section 2.2.3).

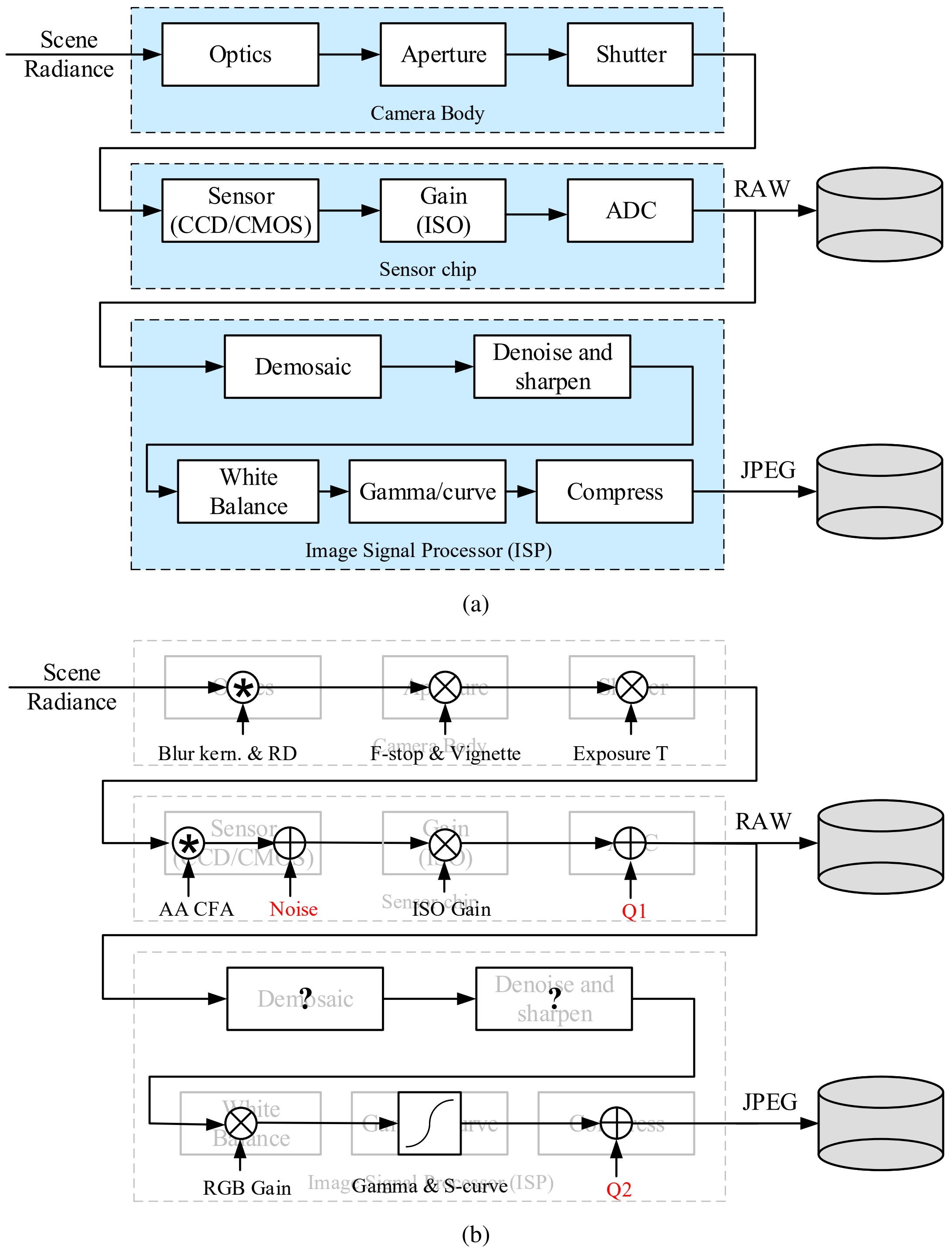
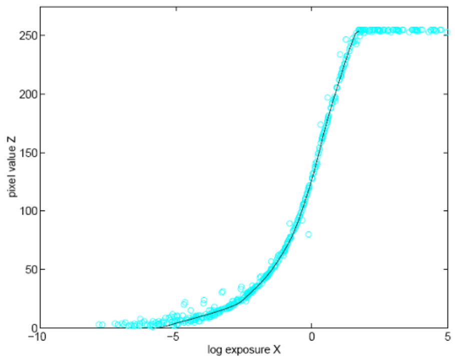
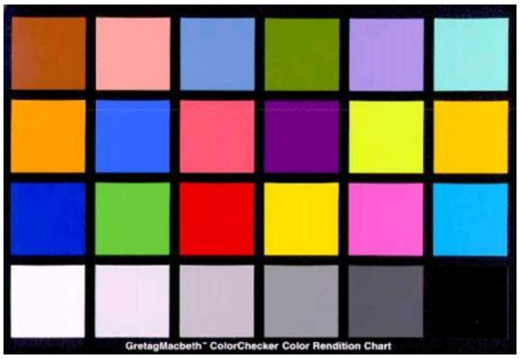


Figure 10.2 Image sensing pipeline: (a) block diagram showing the various sources of noise as well as the typical digital post-processing steps; (b) equivalent signal transforms, including convolution, gain, and noise injection. The abbreviations are: RD = radial distortion, AA = anti-aliasing filter, CFA = color filter array, Q1 and Q2 = quantization noise.



(a)



(b)

Figure 10.3 Radiometric response calibration: (a) typical camera response function, showing the mapping between incoming log irradiance (exposure) and output eight-bit pixel values, for one color channel (Debevec and Malik 1997) © 1997 ACM; (b) color checker chart.

first principles. A more practical approach is to calibrate the camera by measuring correspondences between incoming light and final values.

The most accurate, but most expensive, approach is to use an *integrating sphere*, which is a large (typically 1m diameter) sphere carefully painted on the inside with white matte paint. An accurately calibrated light at the top controls the amount of radiance inside the sphere (which is constant everywhere because of the sphere's radiometry) and a small opening at the side allows for a camera/lens combination to be mounted. By slowly varying the current going into the light, an accurate correspondence can be established between incoming radiance and measured pixel values. The vignetting and noise characteristics of the camera can also be simultaneously determined.

A more practical alternative is to use a calibration chart (Figure 10.3b) such as the Macbeth or Munsell ColorChecker Chart.[10] The biggest problem with this approach is to ensure uniform lighting. One approach is to use a large dark room with a high-quality light source far away from (and perpendicular to) the chart. Another is to place the chart outdoors away from any shadows. (The results will differ under these two conditions, because the color of the illuminant will be different.)

The easiest approach is probably to take multiple exposures of the same scene while the camera is on a tripod and to recover the response function by simultaneously estimating the incoming irradiance at each pixel and the response curve (Mann and Picard 1995; Debevec and Malik 1997; Mitsunaga and Nayar 1999). This approach is discussed in more detail in Section 10.2 on high dynamic range imaging.

If all else fails, i.e., you just have one or more unrelated photos, you can use an International Color Consortium (ICC) profile for the camera (Fairchild 2013).[11] Even more simply, you can just assume that the response is linear if they are RAW files and that the images have a $\gamma = 2.2$ non-linearity (plus clipping) applied to each RGB channel if they are JPEG images.

10.1.2 Noise level estimation

In addition to knowing the camera response function, it is also often important to know the amount of noise being injected under a particular camera setting (e.g., ISO/gain level). The simplest characterization of noise is a single standard deviation, usually measured in gray levels, independent of

[10] https://www.xrite.com.
[11] See also the ICC *Information on Profiles*, https://www.color.org/info_profiles2.xalter.

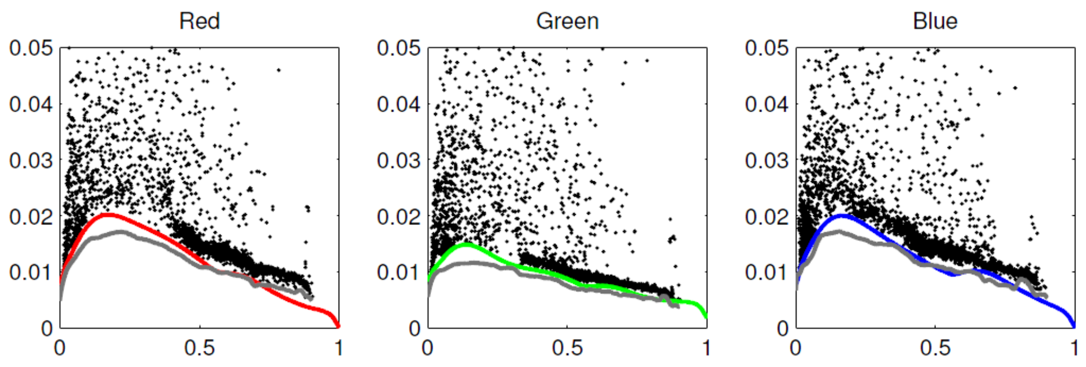


Figure 10.4 Noise level function estimates obtained from a single color photograph (Liu, Szeliski *et al.* 2008) © 2008 IEEE. The colored curves are the estimated NLF fit as the probabilistic lower envelope of the measured deviations between the noisy piecewise-smooth images. The ground truth NLFs obtained by averaging 29 images are shown in gray.

pixel value. A more accurate model can be obtained by estimating the noise level as a function of pixel value (Figure 10.4), which is known as the *noise level function* (Liu, Szeliski *et al.* 2008).

As with the camera response function, the simplest way to estimate these quantities is in the lab, using either an integrating sphere or a calibration chart. The noise can be estimated either at each pixel independently, by taking repeated exposures and computing the temporal variance in the measurements (Healey and Kondepudy 1994), or over regions, by assuming that pixel values should all be the same within some region (e.g., inside a color checker square) and computing a spatial variance.

This approach can be generalized to photos where there are regions of constant or slowly varying intensity (Liu, Szeliski *et al.* 2008). First, segment the image into such regions and fit a constant or linear function inside each region. Next, measure the (spatial) standard deviation of the differences between the noisy input pixels and the smooth fitted function away from large gradients and region boundaries. Plot these as a function of output level for each color channel, as shown in Figure 10.4. Finally, fit a lower envelope to this distribution to ignore pixels or deviations that are outliers. A fully Bayesian approach to this problem that models the statistical distribution of each quantity is presented by Liu, Szeliski *et al.* (2008). A simpler approach, which should produce useful results in most cases, is to fit a low-dimensional function (e.g., positive valued B-spline) to the lower envelope (see Exercise 10.2).

Matsushita and Lin (2007b) present a technique for simultaneously estimating a camera's response and noise level functions based on skew (asymmetries) in level-dependent noise distributions. Their paper also contains extensive references to previous work in these areas.

10.1.3 Vignetting

A common problem with using wide-angle and wide-aperture lenses is that the image tends to darken in the corners (Figure 10.5a). This problem is generally known as *vignetting* and comes in several different forms, including natural, optical, and mechanical vignetting (Section 2.2.3) (Ray 2002). As with radiometric response function calibration, the most accurate way to calibrate vignetting is to use an integrating sphere or a picture of a uniformly colored and illuminated blank wall.

An alternative approach is to stitch a panoramic scene and to assume that the true radiance at

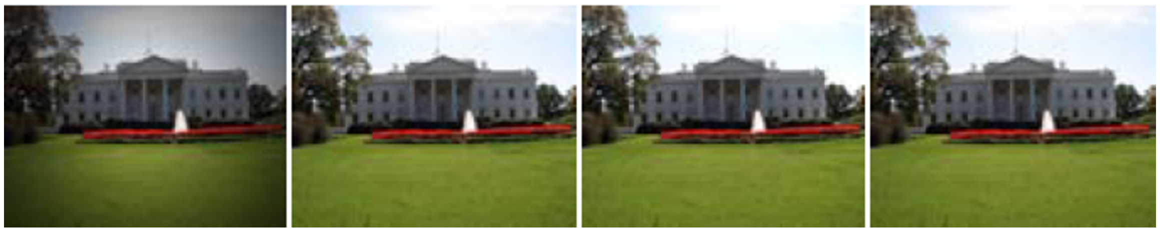


Figure 10.5 Single image vignetting correction (Zheng, Yu *et al.* 2008) © 2008 IEEE: (a) original image with strong visible vignetting; (b) vignetting compensation as described by Zheng, Zhou *et al.* (2006); (c–d) vignetting compensation as described by Zheng, Yu *et al.* (2008).

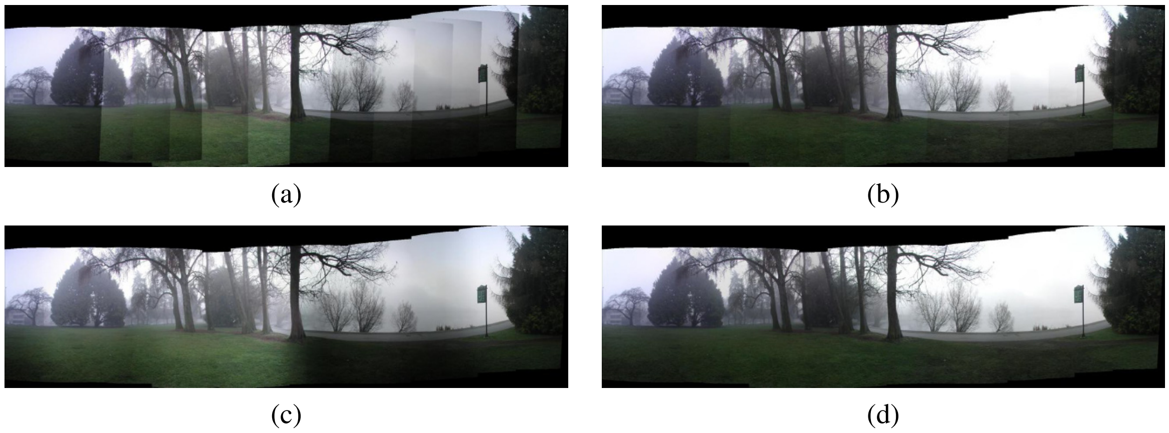


Figure 10.6 Simultaneous estimation of vignetting, exposure, and radiometric response (Goldman 2010) © 2011 IEEE: (a) original average of the input images; (b) after compensating for vignetting; (c) using gradient domain blending only (note the remaining mottled look); (d) after both vignetting compensation and blending.

each pixel comes from the central portion of each input image. This is easier to do if the radiometric response function is already known (e.g., by shooting in RAW mode) and if the exposure is kept constant. If the response function, image exposures, and vignetting function are unknown, they can still be recovered by optimizing a large least squares fitting problem (Litvinov and Schechner 2005; Goldman 2010). Figure 10.6 shows an example of simultaneously estimating the vignetting, exposure, and radiometric response function from a set of overlapping photographs (Goldman 2010). Note that unless vignetting is modeled and compensated, regular gradient-domain image blending (Section 8.4.4) will not create an attractive image.

If only a single image is available, vignetting can be estimated by looking for slow consistent intensity variations in the radial direction. The original algorithm proposed by Zheng, Lin, and Kang (2006) first pre-segmented the image into smoothly varying regions and then performed an analysis inside each region. Instead of pre-segmenting the image, Zheng, Yu *et al.* (2008) compute the radial gradients at all the pixels and use the asymmetry in this distribution (because gradients away from the center are, on average, slightly negative) to estimate the vignetting. Figure 10.5 shows the results of applying each of these algorithms to an image with a large amount of vignetting. Exercise 10.3 has you implement some of the above techniques.

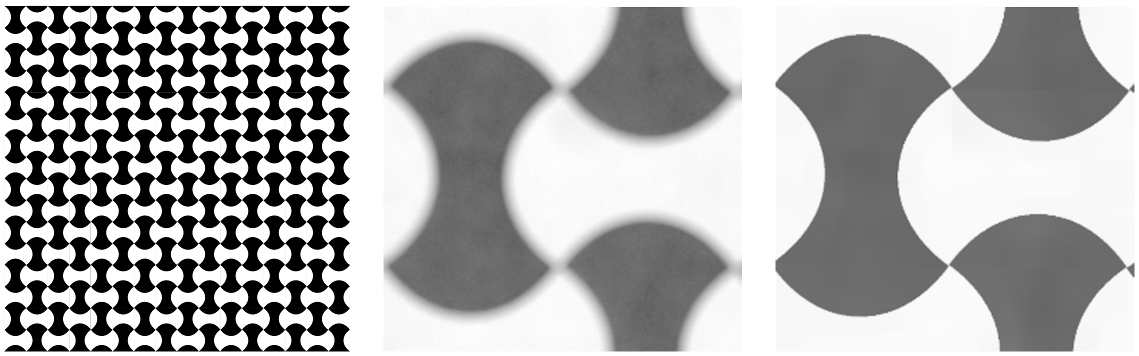


Figure 10.7 Calibration pattern with edges equally distributed at all orientations that can be used for PSF and radial distortion estimation (Joshi, Szeliski, and Kriegman 2008) © 2008 IEEE. A portion of an actual sensed image is shown in the middle and a close-up of the ideal pattern is on the right.

10.1.4 Optical blur (spatial response) estimation

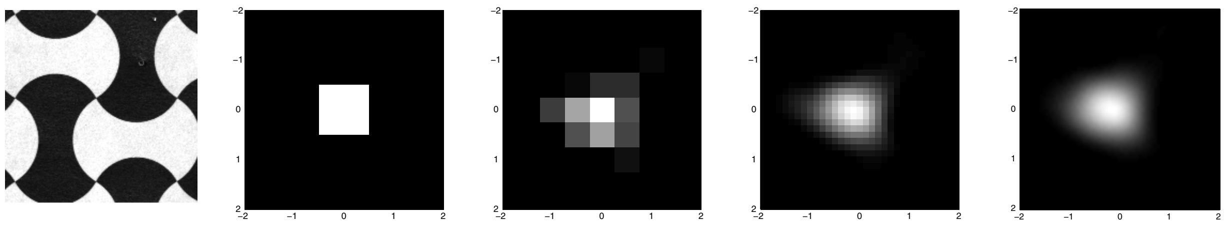
One final characteristic of imaging systems that you should calibrate is the spatial response function, which encodes the optical blur that gets convolved with the incoming image to produce the point-sampled image. The shape of the convolution kernel, which is also known as the *point spread function (PSF)* or *optical transfer function*, depends on several factors, including lens blur and radial distortion (Section 2.2.3), anti-aliasing filters in front of the sensor, and the shape and extent of each active pixel area (Section 2.3) (Figure 10.2). A good estimate of this function is required for applications such as multi-image super-resolution and deblurring (Section 10.3).

In theory, one could estimate the PSF by simply observing an infinitely small point light source everywhere in the image. Creating an array of samples by drilling through a dark plate and back-lighting with a very bright light source is difficult in practice.

A more practical approach is to observe an image composed of long straight lines or bars, as these can be fitted to arbitrary precision. Because the location of a horizontal or vertical edge can be *aliased* during acquisition, slightly slanted edges are preferred. The profile and locations of such edges can be estimated to sub-pixel precision, which makes it possible to estimate the PSF at sub-pixel resolutions (Reichenbach, Park, and Narayanswamy 1991; Burns and Williams 1999; Williams and Burns 2001; Goesele, Fuchs, and Seidel 2003). The thesis by Murphy (2005) contains a nice survey of all aspects of camera calibration, including the spatial frequency response (SFR), spatial uniformity, tone reproduction, color reproduction, noise, dynamic range, color channel registration, and depth of field. It also includes a description of a slant-edge calibration algorithm called `sfrmat2`.

The slant-edge technique can be used to recover a 1D projection of the 2D PSF, e.g., slightly vertical edges are used to recover the horizontal *line spread function* (LSF) (Williams 1999). The LSF is then often converted into the Fourier domain and its magnitude plotted as a one-dimensional *modulation transfer function* (MTF), which indicates which image frequencies are lost (blurred) and aliased during the acquisition process (Section 2.3.1). For most computational photography applications, it is preferable to directly estimate the full 2D PSF, as it can be hard to recover from its projections (Williams 1999).

Figure 10.7 shows a pattern containing edges at all orientations, which can be used to directly recover a two-dimensional PSF. First, corners in the pattern are located by extracting edges in the sensed image, linking them, and finding the intersections of the circular arcs. Next, the ideal pattern, whose analytic form is known, is warped (using a homography) to fit the central portion of the input


(a)

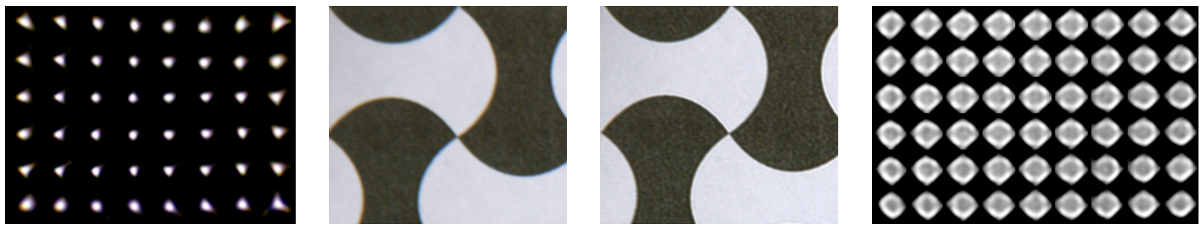

(b) (c)

Figure 10.8 Point spread function estimation using a calibration target (Joshi, Szeliski, and Kriegman 2008) © 2008 IEEE. (a) Sub-pixel PSFs at successively higher resolutions (note the interaction between the square sensing area and the circular lens blur). (b) The radial distortion and chromatic aberration can also be estimated and removed. (c) PSF for a misfocused (blurred) lens showing some diffraction and vignetting effects in the corners.

image and its intensities are adjusted to fit the ones in the sensed image. If desired, the pattern can be rendered at a higher resolution than the input image, which enables the estimation of the PSF to sub-pixel resolution (Figure 10.8a). Finally a large linear least squares system is solved to recover the unknown PSF kernel K,

$$K = \arg\min_K \|B - D(I * K)\|^2, \tag{10.1}$$

where B is the sensed (blurred) image, I is the predicted (sharp) image, and D is an optional downsampling operator that matches the resolution of the ideal and sensed images (Joshi, Szeliski, and Kriegman 2008). An alternative solution technique is to estimate 1D PSF profiles first and to then combine them using a Radon transform (Cho, Paris *et al.* 2011).

If the process of estimating the PSF is done locally in overlapping patches of the image, it can also be used to estimate the radial distortion and chromatic aberration induced by the lens (Figure 10.8b). Because the homography mapping the ideal target to the sensed image is estimated in the central (undistorted) part of the image, any (per-channel) shifts induced by the optics manifest themselves as a displacement in the PSF centers.[12] Compensating for these shifts eliminates both the achromatic radial distortion and the inter-channel shifts that result in visible chromatic aberration. The color-dependent blurring caused by chromatic aberration (Figure 2.21) can also be removed using the deblurring techniques discussed in Section 10.3. Figure 10.8b shows how the radial distortion and chromatic aberration manifest themselves as elongated and displaced PSFs, along with the result of removing these effects in a region of the calibration target.

The local 2D PSF estimation technique can also be used to estimate vignetting. Figure 10.8c shows how the mechanical vignetting manifests itself as clipping of the PSF in the corners of the

[12]This process confounds the distinction between geometric and photometric calibration. In principle, any geometric distortion could be modeled by spatially varying displaced PSFs. In practice, it is easier to fold any large shifts into the geometric correction component.

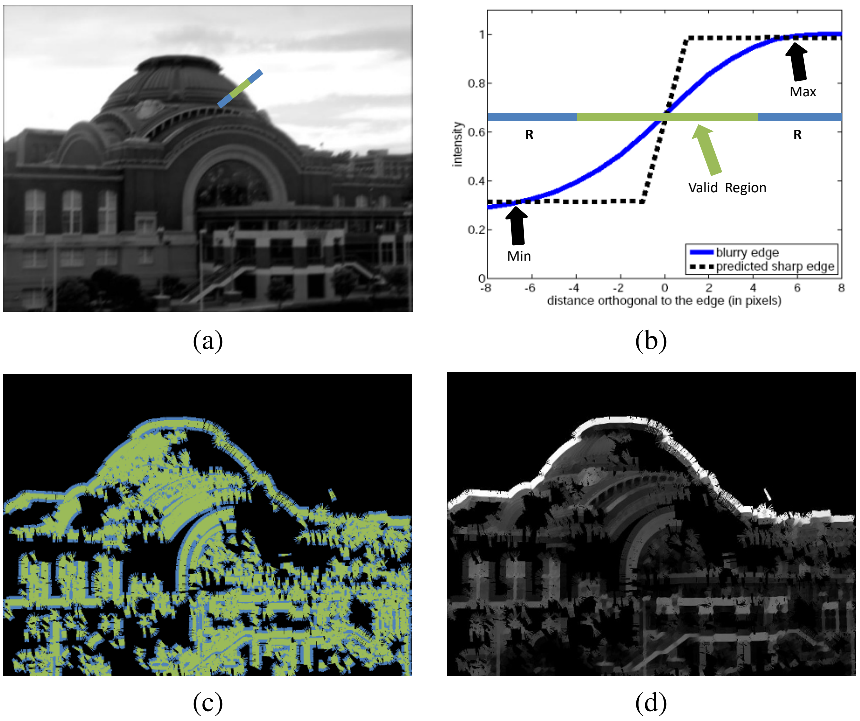


Figure 10.9 Estimating the PSF without using a calibration pattern (Joshi, Szeliski, and Kriegman 2008) © 2008 IEEE: (a) Input image with blue cross-section (profile) location, (b) Profile of sensed and predicted step edges, (c–d) Locations and values of the predicted colors near the edge locations.

image. For the overall dimming associated with vignetting to be properly captured, the modified intensities of the ideal pattern need to be extrapolated from the center, which is best done with a uniformly illuminated target.

When working with RAW Bayer-pattern images, the correct way to estimate the PSF is to only evaluate the least squares terms in (10.1) at sensed pixel values, while interpolating the ideal image to all values. For JPEG images, you should linearize your intensities first, e.g., remove the gamma and any other non-linearities in your estimated radiometric response function.

What if you have an image that was taken with an uncalibrated camera? Can you still recover the PSF an use it to correct the image? In fact, with a slight modification, the previous algorithms still work.

Instead of assuming a known calibration image, you can detect strong elongated edges and fit ideal step edges in such regions (Figure 10.9b), resulting in the sharp image shown in Figure 10.9d. For every pixel that is surrounded by a complete set of valid estimated neighbors (green pixels in Figure 10.9c), apply the least squares formula (10.1) to estimate the kernel K. The resulting locally estimated PSFs can be used to correct for chromatic aberration (because the relative displacements between per-channel PSFs can be computed), as shown by Joshi, Szeliski, and Kriegman (2008).

Exercise 10.4 provides some more detailed instructions for implementing and testing edge-based PSF estimation algorithms. An alternative approach, which does not require the explicit detection of edges but uses image statistics (gradient distributions) instead, is presented by Fergus, Singh *et al.* (2006).

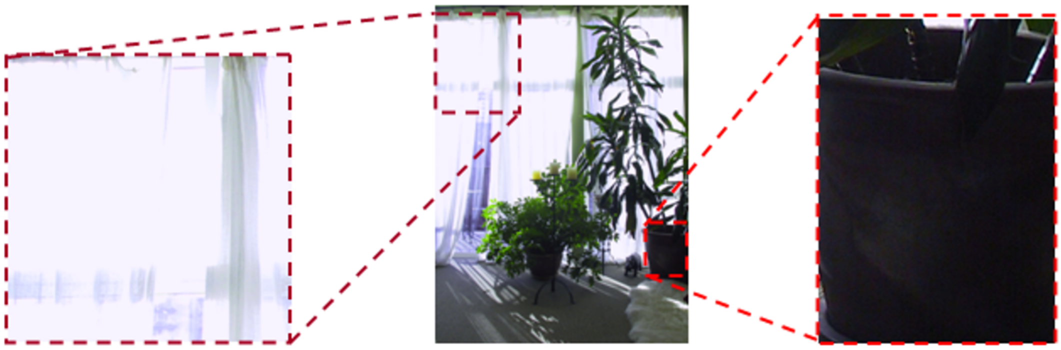


Figure 10.10 Sample indoor image where the areas outside the window are overexposed and inside the room are too dark.

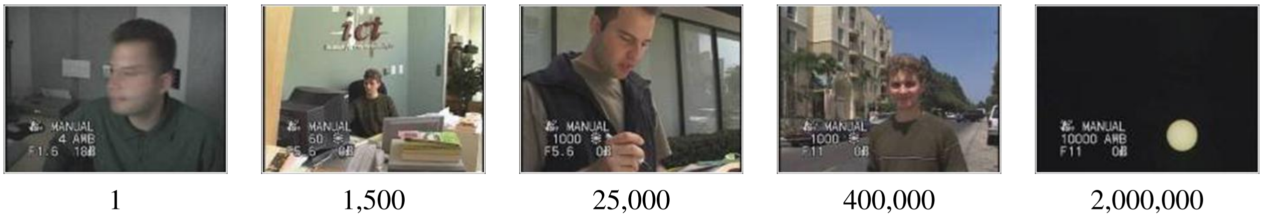


Figure 10.11 Relative brightness of different scenes, ranging from 1 inside a dark room lit by a monitor to 2,000,000 looking at the Sun. Photos courtesy of Paul Debevec.

10.2 High dynamic range imaging

As we mentioned earlier in this chapter, registered images taken at different exposures can be used to calibrate the radiometric response function of a camera. More importantly, they can help you create well-exposed photographs under challenging conditions, such as brightly lit scenes where any single exposure contains saturated (overexposed) and dark (underexposed) regions (Figure 10.10). This problem is quite common, because the natural world contains a range of radiance values that is far greater than can be captured with any photographic sensor or film (Figure 10.11). Taking a set of *bracketed exposures* (exposures taken by a camera in automatic exposure bracketing (AEB) mode to deliberately under- and over-expose the image) gives you the material from which to create a properly exposed photograph, as shown in Figure 10.12 (Freeman 2008; Gulbins and Gulbins 2009; Hasinoff, Durand, and Freeman 2010; Reinhard, Heidrich *et al.* 2010).

While it is possible to combine pixels from different exposures directly into a final composite (Burt and Kolczynski 1993; Mertens, Kautz, and Reeth 2007), this approach runs the risk of creating contrast reversals and halos. Instead, the more common approach is to proceed in three stages:

1. Estimate the radiometric response function from the aligned images.

2. Estimate a *radiance map* by selecting or blending pixels from different exposures.

3. Tone map the resulting high dynamic range (HDR) image back into a displayable gamut.

The idea behind estimating the radiometric response function is relatively straightforward (Mann and Picard 1995; Debevec and Malik 1997; Mitsunaga and Nayar 1999; Reinhard, Heidrich *et al.*

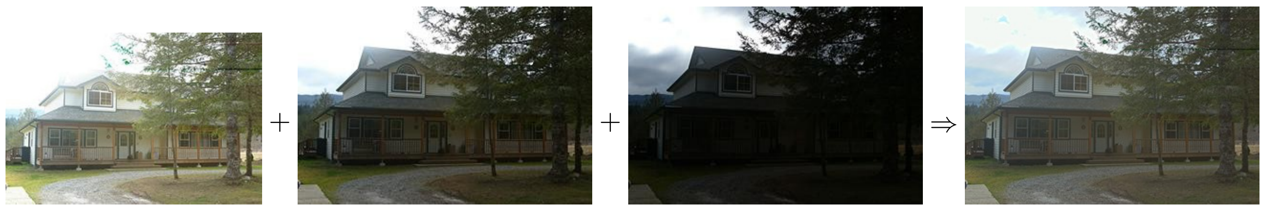


Figure 10.12 A bracketed set of shots (using the camera's automatic exposure bracketing (AEB) mode) and the resulting high dynamic range (HDR) composite.

2010). Suppose you take three sets of images at different exposures (shutter speeds), say at ± 2 exposure values.[13] If we were able to determine the irradiance (exposure) E_i at each pixel (2.102), we could plot it against the measured pixel value z_{ij} for each exposure time t_j, as shown in Figure 10.13.

Unfortunately, we do not know the irradiance values E_i, so these have to be estimated at the same time as the radiometric response function f, which can be written (Debevec and Malik 1997) as

$$z_{ij} = f(E_i\, t_j), \tag{10.2}$$

where t_j is the exposure time for the jth image. The inverse response curve f^{-1} is given by

$$f^{-1}(z_{ij}) = E_i\, t_j. \tag{10.3}$$

Taking logarithms of both sides (base 2 is convenient, as we can now measure quantities in EVs), we obtain

$$g(z_{ij}) = \log f^{-1}(z_{ij}) = \log E_i + \log t_j, \tag{10.4}$$

where $g = \log f^{-1}$ (which maps pixel values z_{ij} into log irradiance) is the curve we are estimating (Figure 10.13 turned on its side).

Debevec and Malik (1997) assume that the exposure times t_j are known. (Recall that these can be obtained from a camera's EXIF tags, but that they actually follow a power of 2 progression $\ldots$, $^1/_{128}$, $^1/_{64}$, $^1/_{32}$, $^1/_{16}$, $^1/_{8}$, $\ldots$ instead of the marked $\ldots$, $^1/_{125}$, $^1/_{60}$, $^1/_{30}$, $^1/_{15}$, $^1/_{8}$, $\ldots$ values—see Exercise 2.5.) The unknowns are therefore the per-pixel exposures E_i and the response values $g_k = g(k)$, where g can be discretized according to the 256 pixel values commonly observed in eight-bit images. (The response curves are calibrated separately for each color channel.)

In order to make the response curve smooth, Debevec and Malik (1997) add a second-order smoothness constraint

$$\lambda \sum_k g''(k)^2 = \lambda \sum [g(k-1) - 2g(k) + g(k+1)]^2, \tag{10.5}$$

which is similar to the one used in snakes (7.27). Because pixel values are more reliable in the middle of their range (and the g function becomes singular near saturation values), they also add a weighting (hat) function $w(k)$ that decays to zero at both ends of the pixel value range,

$$w(z) = \begin{cases} z - z_{\min} & z \le (z_{\min} + z_{\max})/2 \\ z_{\max} - z & z > (z_{\min} + z_{\max})/2. \end{cases} \tag{10.6}$$

[13] Changing the shutter speed is preferable to changing the aperture, as the latter can modify the vignetting and focus. Using ± 2 "f-stops" (technically, exposure values, or EVs, as f-stops refer to apertures) is usually the right compromise between capturing a good dynamic range and having properly exposed pixels everywhere.

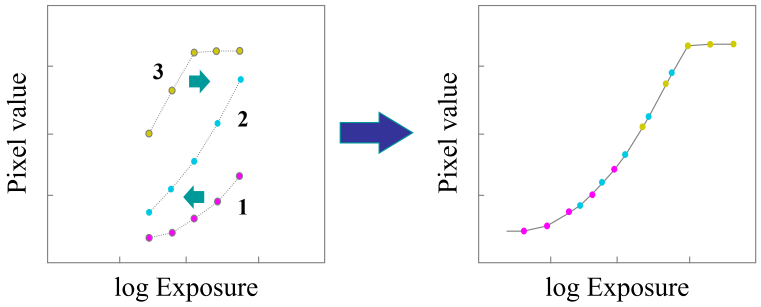


Figure 10.13 Radiometric calibration using multiple exposures (Debevec and Malik 1997). Corresponding pixel values are plotted as functions of log exposures (irradiance). The curves on the left are shifted to account for each pixel's unknown radiance until they all line up into a single smooth curve.

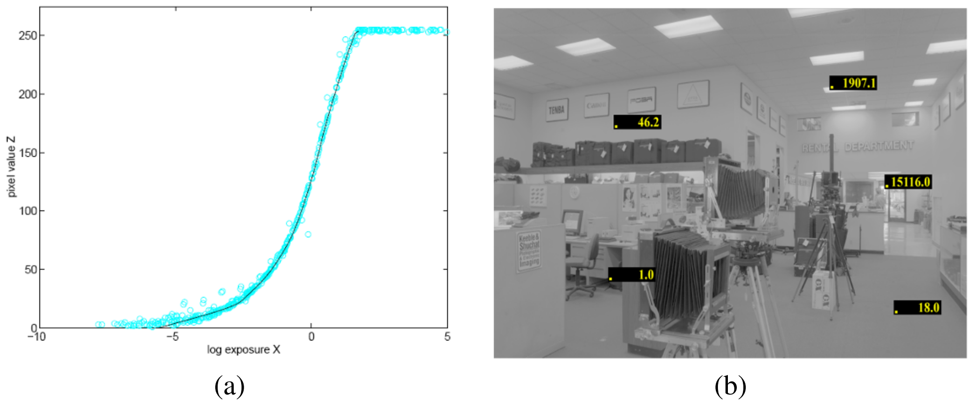


Figure 10.14 Recovered response function and radiance image for a real digital camera (DCS460) (Debevec and Malik 1997) © 1997 ACM.

Putting all of these terms together, they obtain a least squares problem in the unknowns $\{g_k\}$ and $\{E_i\}$,

$$E = \sum_i \sum_j w(z_{i,j})[g(z_{i,j}) - \log E_i - \log t_j]^2 + \lambda \sum_k w(k) g''(k)^2. \tag{10.7}$$

(To remove the overall shift ambiguity in the response curve and irradiance values, the middle of the response curve is set to 0.) Debevec and Malik (1997) show how this can be implemented in 21 lines of MATLAB code, which partially accounts for the popularity of their technique.

While Debevec and Malik (1997) assume that the exposure times t_j are known exactly, there is no reason why these additional variables cannot be thrown into the least squares problem, constraining their final estimated values to lie close to their nominal values $\hat{t}_j$ with an extra term $\eta \sum_j (t_j - \hat{t}_j)^2$.

Figure 10.14 shows the recovered radiometric response function for a digital camera along with select (relative) radiance values in the overall radiance map. Figure 10.15 shows the bracketed input images captured on color film and the corresponding radiance map. Note that while most research on high dynamic range imaging assumes that the radiometric (or camera) response function is independent of exposure, this is not actually the case. Rodríguez, Vazquez-Corral, and Bertalmío (2019) describe how to take this into account to get improved results.

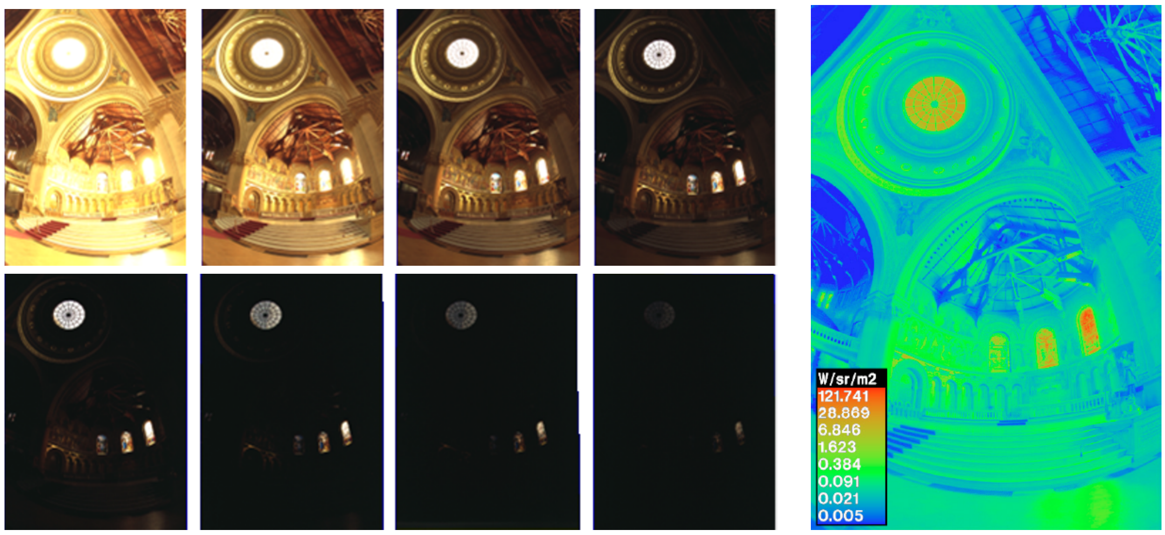


Figure 10.15 Bracketed set of exposures captured with a film camera and the resulting radiance image displayed in pseudocolor (Debevec and Malik 1997) © 1997 ACM.

While Debevec and Malik (1997) use a general second-order smooth curve g to parameterize their response curve, Mann and Picard (1995) use a three-parameter function

$$f(E) = \alpha + \beta E^{\gamma}, \tag{10.8}$$

while Mitsunaga and Nayar (1999) use a low-order ($N \leq 10$) polynomial for the inverse response function g. Pal, Szeliski *et al.* (2004) derive a Bayesian model that estimates an independent smooth response function for each image, which can better model the more sophisticated (and hence less predictable) automatic contrast and tone adjustment performed in today's digital cameras.

Once the response function has been estimated, the second step in creating high dynamic range photographs is to merge the input images into a composite *radiance map*. If the response function and images were known exactly, i.e., if they were noise free, you could use any non-saturated pixel value to estimate the corresponding radiance by mapping it through the inverse response curve $E = g(z)$.

Unfortunately, pixels are noisy, especially under low-light conditions when fewer photons arrive at the sensor. To compensate for this, Mann and Picard (1995) use the derivative of the response function as a weight in determining the final radiance estimate, because "flatter" regions of the curve tell us less about the incoming irradiance. Debevec and Malik (1997) use a hat function (10.6) which accentuates mid-tone pixels while avoiding saturated values. Mitsunaga and Nayar (1999) show that to maximize the signal-to-noise ratio (SNR), the weighting function must emphasize both higher pixel values and larger gradients in the transfer function, i.e.,

$$w(z) = g(z)/g'(z), \tag{10.9}$$

where the weights w are used to form the final irradiance estimate

$$\log E_i = \frac{\sum_j w(z_{ij})[g(z_{ij}) - \log t_j]}{\sum_j w(z_{ij})}. \tag{10.10}$$

Exercise 10.1 has you implement one of the radiometric response function calibration techniques and then use it to create radiance maps.

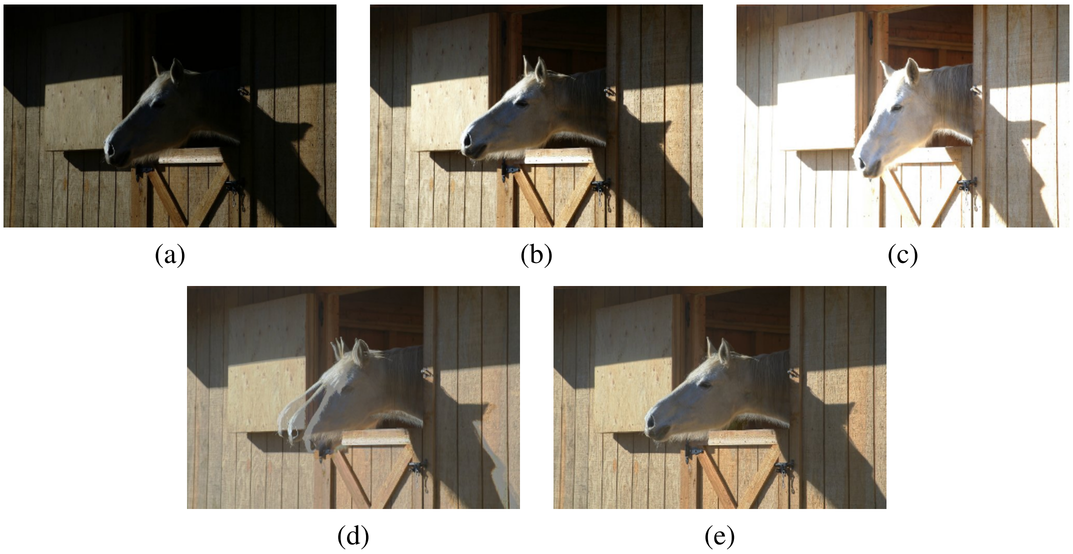


Figure 10.16 Merging multiple exposures to create a high dynamic range composite (Kang, Uyttendaele *et al.* 2003): (a–c) three different exposures; (d) merging the exposures using classic algorithms (note the ghosting due to the horse's head movement); (e) merging the exposures with motion compensation.

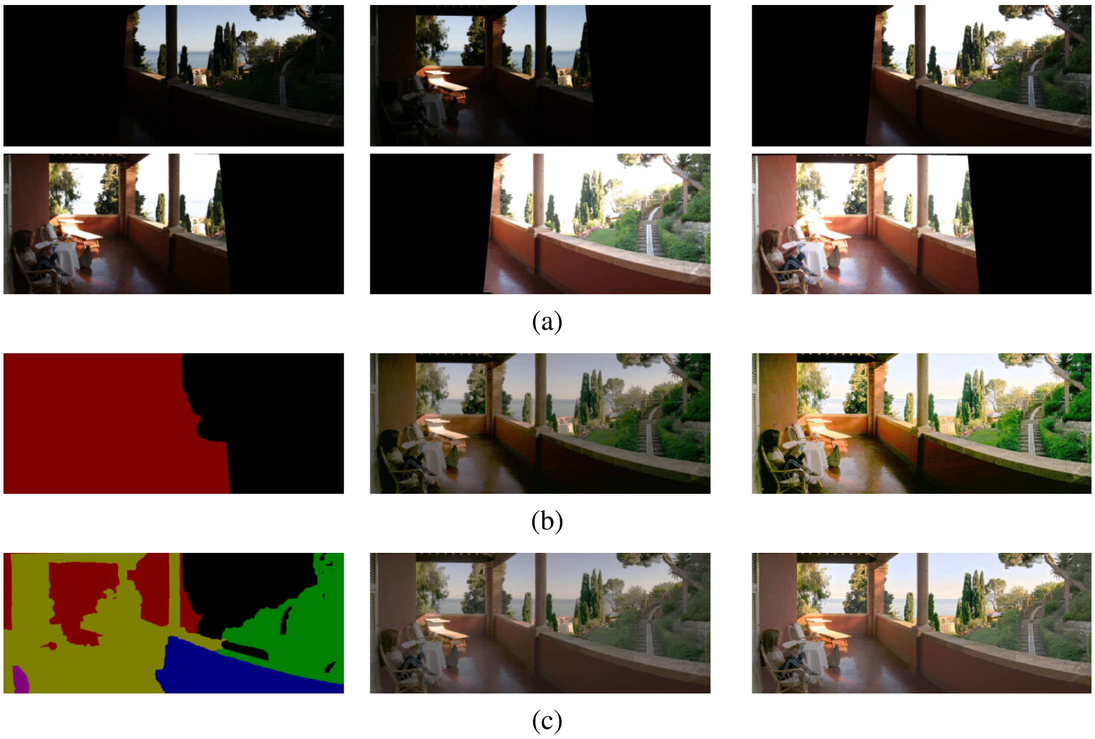


Figure 10.17 HDR merging with large amounts of motion (Eden, Uyttendaele, and Szeliski 2006) © 2006 IEEE: (a) registered bracketed input images; (b) results after the first pass of image selection: reference labels, image, and tone-mapped image; (c) results after the second pass of image selection: final labels, compressed HDR image, and tone-mapped image

Under real-world conditions, casually acquired images may not be perfectly registered and may contain moving objects. Ward (2003) uses a global (parametric) transform to align the input images, while Kang, Uyttendaele *et al.* (2003) present an algorithm that combines global registration with local motion estimation (optical flow) to accurately align the images before blending their radiance estimates (Figure 10.16). Because the images may have widely different exposures, care must be taken when estimating the motions, which must themselves be checked for consistency to avoid the creation of ghosts and object fragments.

Even this approach, however, may not work when the camera is simultaneously undergoing large panning motions and exposure changes, which is a common occurrence in casually acquired panoramas. Under such conditions, different parts of the image may be seen at one or more exposures. Devising a method to blend all of these different sources while avoiding sharp transitions and dealing with scene motion is a challenging problem. One approach is to first find a consensus mosaic and to then selectively compute radiances in under- and over-exposed regions (Eden, Uyttendaele, and Szeliski 2006), as shown in Figure 10.17. Additional techniques for constructing and displaying high dynamic range video are discussed in Myszkowski, Mantiuk, and Krawczyk (2008), Tocci, Kiser *et al.* (2011), Sen, Kalantari *et al.* (2012), Dufaux, Le Callet *et al.* (2016), Banterle, Artusi *et al.* (2017), and Kalantari and Ramamoorthi (2017). Another approach is to use deep learning techniques to infer the high dynamic range radiance image from a single low dynamic range image (Liu, Lai *et al.* 2020b).

Some cameras, such as the Sony α550 and Pentax K-7, have started integrating multiple exposure merging and tone mapping directly into the camera body. In the future, the need to compute high dynamic range images from multiple exposures may be eliminated by advances in camera sensor technology (Yang, El Gamal *et al.* 1999; Nayar and Mitsunaga 2000; Nayar and Branzoi 2003; Kang, Uyttendaele *et al.* 2003; Narasimhan and Nayar 2005; Tumblin, Agrawal, and Raskar 2005). However, the need to blend such images and to tone map them to lower-gamut displays is likely to remain.

HDR image formats. Before we discuss techniques for mapping HDR images back to a displayable gamut, we should discuss the commonly used formats for storing HDR images.

If storage space is not an issue, storing each of the R, G, and B values as a 32-bit IEEE float is the best solution. The commonly used Portable PixMap (.ppm) format, which supports both uncompressed ASCII and raw binary encodings of values, can be extended to a Portable FloatMap (.pfm) format by modifying the header. TIFF also supports full floating point values.

A more compact representation is the Radiance format (.pic, .hdr) (Ward 1994), which uses a single common exponent and per-channel mantissas. An intermediate encoding, OpenEXR from ILM,[14] uses 16-bit floats for each channel, which is a format supported natively on most modern GPUs. Ward (2004) describes these and other data formats such as LogLuv (Larson 1998) in more detail, as do the books by Freeman (2008) and Reinhard, Heidrich *et al.* (2010). An even more recent HDR image format is the JPEG XR standard.

10.2.1 Tone mapping

Once a radiance map has been computed, it is usually necessary to display it on a lower gamut (i.e., eight-bit) screen or printer. A variety of *tone mapping* techniques has been developed for this purpose, which involve either computing spatially varying transfer functions or reducing image gradients to fit the available dynamic range (Reinhard, Heidrich *et al.* 2010).

[14] https://www.openexr.net.

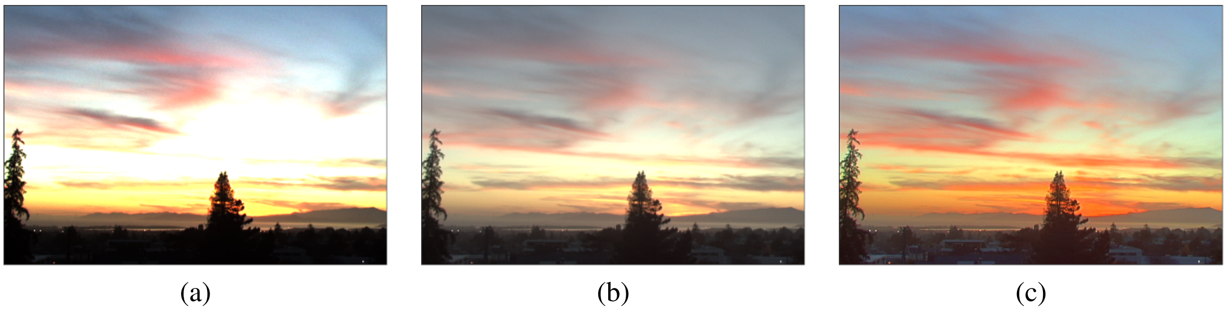


Figure 10.18 Global tone mapping: (a) input HDR image, linearly mapped; (b) gamma applied to each color channel independently; (c) gamma applied to intensity (colors are less washed out). Original HDR image courtesy of Paul Debevec, https://www.pauldebevec.com/Research/HDR. Processed images courtesy of Frédo Durand, MIT 6.815/6.865 course on Computational Photography.

The simplest way to compress a high dynamic range radiance image into a low dynamic range gamut is to use a global transfer curve (Larson, Rushmeier, and Piatko 1997). Figure 10.18 shows one such example, where a gamma curve is used to map an HDR image back into a displayable gamut. If gamma is applied separately to each channel (Figure 10.18b), the colors become muted (less saturated), as higher-valued color channels contribute less (proportionately) to the final color. Extracting the luminance channel from the color image using (2.104), applying the global mapping to the luminance channel, and then reconstituting the color image using (10.19) works better (Figure 10.18c).

Unfortunately, when the image has a really wide range of exposures, this global approach still fails to preserve details in regions with widely varying exposures. What is needed, instead, is something akin to the dodging and burning performed by photographers in the darkroom. Mathematically, this is similar to dividing each pixel by the *average* brightness in a region around that pixel.

Figure 10.19 shows how this process works. As before, the image is split into its luminance and chrominance channels. The log luminance image

$$H(x,y) = \log L(x,y) \tag{10.11}$$

is then low-pass filtered to produce a *base layer*

$$H_{\rm L}(x,y) = B(x,y) * H(x,y), \tag{10.12}$$

and a high-pass *detail layer*

$$H_{\rm H}(x,y) = H(x,y) - H_{\rm L}(x,y). \tag{10.13}$$

The base layer is then contrast reduced by scaling to the desired log-luminance range,

$$H'_{\rm H}(x,y) = s\, H_{\rm H}(x,y) \tag{10.14}$$

and added to the detail layer to produce the new log-luminance image

$$I(x,y) = H'_{\rm H}(x,y) + H_{\rm L}(x,y), \tag{10.15}$$

which can then be exponentiated to produce the tone-mapped (compressed) luminance image. Note that this process is equivalent to dividing each luminance value by (a monotonic mapping of) the average log-luminance value in a region around that pixel.

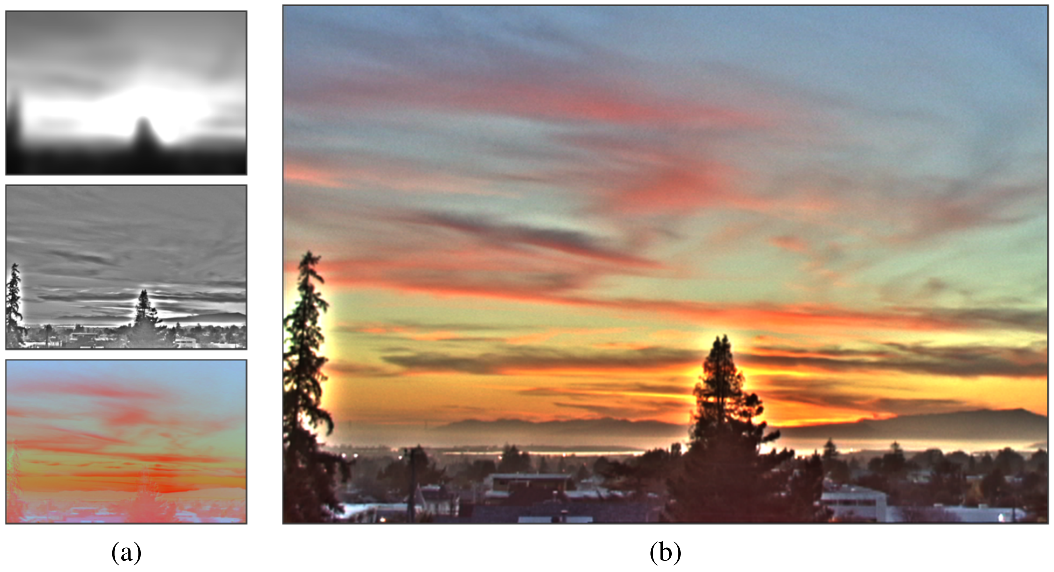


Figure 10.19 Local tone mapping using linear filters: (a) low-pass and high-pass filtered log luminance images and color (chrominance) image; (b) resulting tone-mapped image (after attenuating the low-pass log luminance image) shows visible halos around the trees. Processed images courtesy of Frédo Durand, MIT 6.815/6.865 course on Computational Photography.

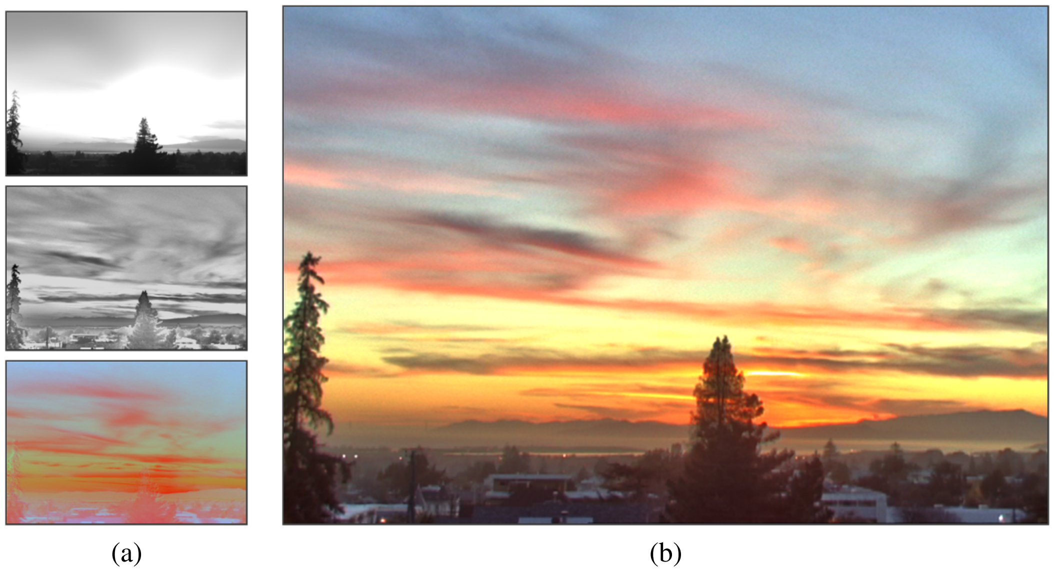


Figure 10.20 Local tone mapping using a bilateral filter (Durand and Dorsey 2002): (a) low-pass and high-pass bilateral filtered log luminance images and color (chrominance) image; (b) resulting tone-mapped image (after attenuating the low-pass log luminance image) shows no halos. Processed images courtesy of Frédo Durand, MIT 6.815/6.865 course on Computational Photography.

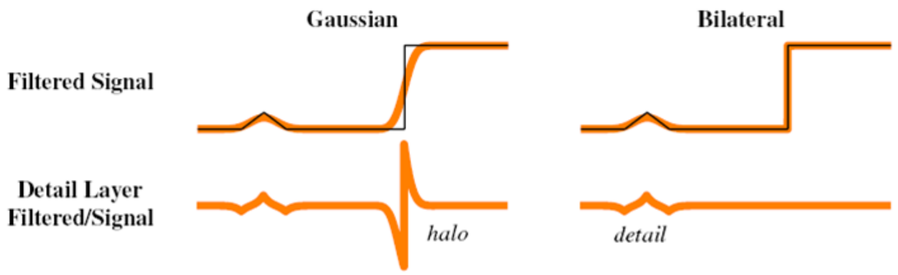


Figure 10.21 Gaussian vs. bilateral filtering (Petschnigg, Agrawala *et al.* 2004) © 2004 ACM: A Gaussian low-pass filter blurs across all edges and therefore creates strong peaks and valleys in the detail image that cause halos. The bilateral filter does not smooth across strong edges and thereby reduces halos while still capturing detail.

Figure 10.19 shows the low-pass and high-pass log luminance image and the resulting tone-mapped color image. Note how the detail layer has visible *halos* around the high-contrast edges, which are visible in the final tone-mapped image. This is because linear filtering, which is not edge preserving, produces halos in the detail layer (Figure 10.21).

The solution to this problem is to use an edge-preserving filter to create the base layer. Durand and Dorsey (2002) study a number of such edge-preserving filters, including anisotropic and robust anisotropic diffusion, and select bilateral filtering (Section 3.3.1) as their edge-preserving filter. (The paper by Farbman, Fattal *et al.* (2008) argues in favor of using a weighted least squares (WLF) filter as an alternative to the bilateral filter and Paris, Kornprobst *et al.* (2008) reviews bilateral filtering and its applications in computer vision and computational photography.) Figure 10.20 shows how replacing the linear low-pass filter with a bilateral filter produces tone-mapped images with no visible halos. Figure 10.22 summarizes the complete information flow in this process, starting with the decomposition into log luminance and chrominance images, bilateral filtering, contrast reduction, and re-composition into the final output image.

An alternative to compressing the base layer is to compress its *derivatives*, i.e., the gradient of the log-luminance image (Fattal, Lischinski, and Werman 2002). Figure 10.23 illustrates this process. The log-luminance image is differentiated to obtain a gradient image

$$H'(x, y) = \nabla H(x, y). \tag{10.16}$$

This gradient image is then attenuated by a spatially varying attenuation function $\Phi(x, y)$,

$$G(x, y) = H'(x, y)\,\Phi(x, y). \tag{10.17}$$

The attenuation function $I(x, y)$ is designed to attenuate large-scale brightness changes (Figure 10.24a) and is designed to take into account gradients at different spatial scales (Fattal, Lischinski, and Werman 2002).

After attenuation, the resulting gradient field is re-integrated by solving a first-order variational (least squares) problem,

$$\min \int\int \|\nabla I(x, y) - G(x, y)\|^2 dx\, dy \tag{10.18}$$

to obtain the compressed log-luminance image $I(x, y)$. This least squares problem is the same that was used for Poisson blending (Section 8.4.4) and was first introduced in our study of regularization (Section 4.2, 4.24). It can efficiently be solved using techniques such as multigrid and hierarchical basis preconditioning (Fattal, Lischinski, and Werman 2002; Szeliski 2006b; Farbman, Fattal *et al.*

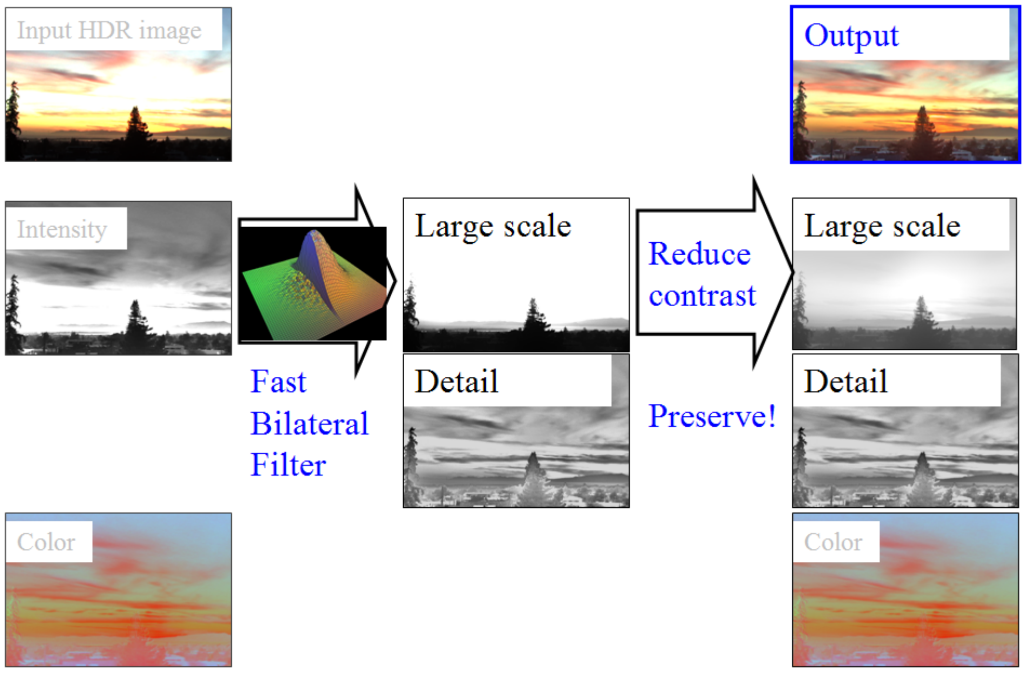


Figure 10.22 Local tone mapping using a bilateral filter (Durand and Dorsey 2002): summary of algorithm workflow. Images courtesy of Frédo Durand, MIT 6.815/6.865 course on Computational Photography.

2008; Krishnan and Szeliski 2011; Krishnan, Fattal, and Szeliski 2013). Once the new luminance image has been computed, it is combined with the original color image using

$$C_{\text{out}} = \left(\frac{C_{\text{in}}}{L_{\text{in}}}\right)^s L_{\text{out}}, \qquad (10.19)$$

where $C = (R, G, B)$ and L_{in} and L_{out} are the original and compressed luminance images. The exponent s controls the saturation of the colors and is typically in the range $s \in [0.4, 0.6]$ (Fattal, Lischinski, and Werman 2002). Figure 10.24b shows the final tone-mapped color image, which shows no visible halos despite the extremely large variation in input radiance values.

Yet another alternative to these two approaches is to perform the local dodging and burning using a locally scale-selective operator (Reinhard, Stark *et al.* 2002). Figure 10.25 shows how such a scale selection operator can determine a radius (scale) that only includes similar color values within the inner circle while avoiding much brighter values in the surrounding circle. In practice, a difference of Gaussians normalized by the inner Gaussian response is evaluated over a range of scales, and the largest scale whose metric is below a threshold is selected (Reinhard, Stark *et al.* 2002).

Another recently developed approach to tone mapping based on multi-resolution decomposition is the Local Laplacian Filter (Paris, Hasinoff, and Kautz 2011), which we introduced in Section 3.5.3. Coefficients in a Laplacian pyramid are constructed from locally contrast-adjusted patches, which enables the technique to not only tone map HDR images, but also to enhance local details and do style transfer (Aubry, Paris *et al.* 2014).

What all of these techniques have in common is that they adaptively attenuate or brighten different regions of the image so that they can be displayed in a limited gamut without loss of contrast. Lischinski, Farbman *et al.* (2006) introduce an *interactive* technique that performs this operation by interpolating a set of sparse user-drawn adjustments (strokes and associated exposure value cor-

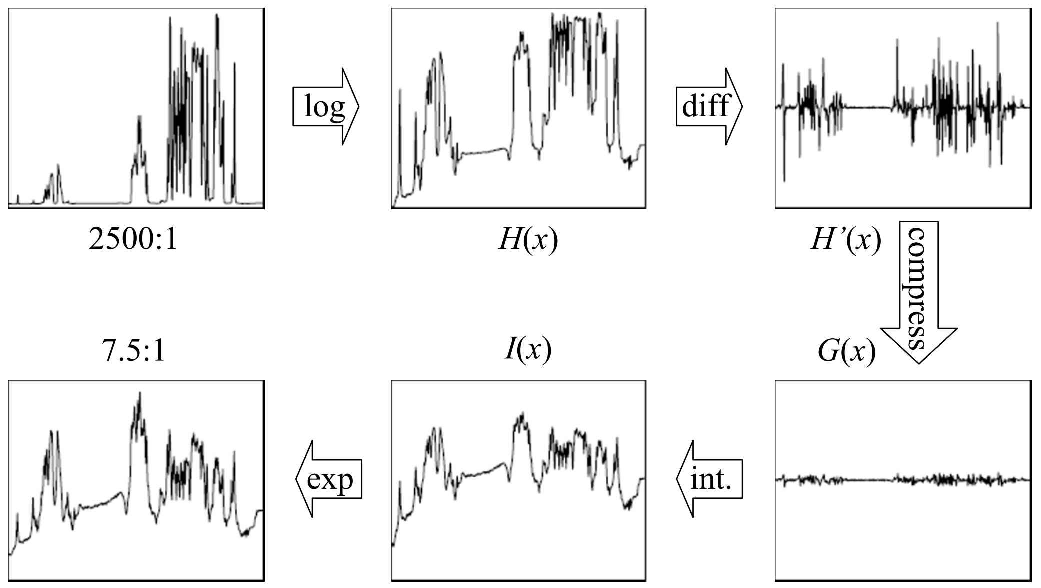


Figure 10.23 Gradient domain tone mapping (Fattal, Lischinski, and Werman 2002) © 2002 ACM. The original image with a dynamic range of 2415:1 is first converted into the log domain, $H(x)$, and its gradients are computed, $H'(x)$. These are attenuated (compressed) based on local contrast, $G(x)$, and integrated to produce the new logarithmic exposure image $I(x)$, which is exponentiated to produce the final intensity image, whose dynamic range is 7.5:1.

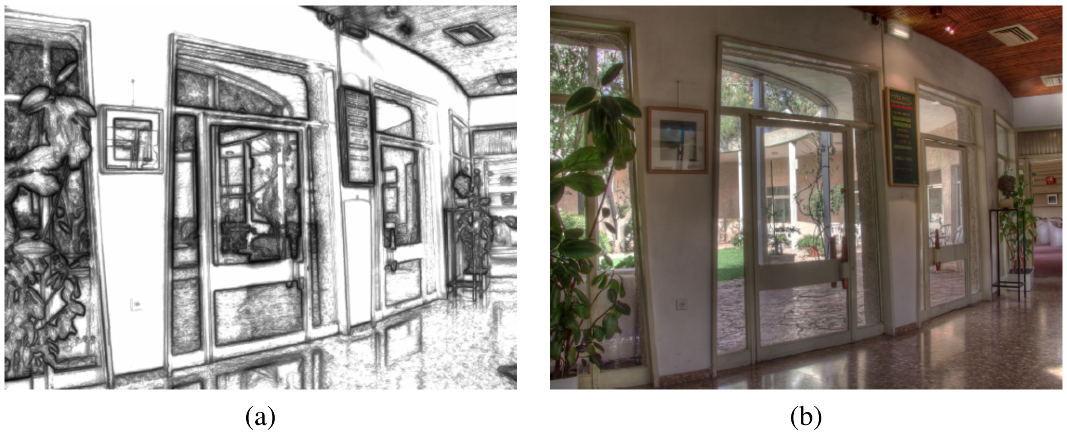


Figure 10.24 Gradient domain tone mapping (Fattal, Lischinski, and Werman 2002) © 2002 ACM: (a) attenuation map, with darker values corresponding to more attenuation; (b) final tone-mapped image.

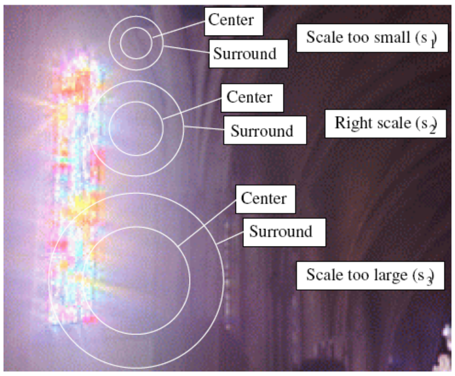


Figure 10.25 Scale selection for tone mapping (Reinhard, Stark *et al.* 2002) © 2002 ACM.

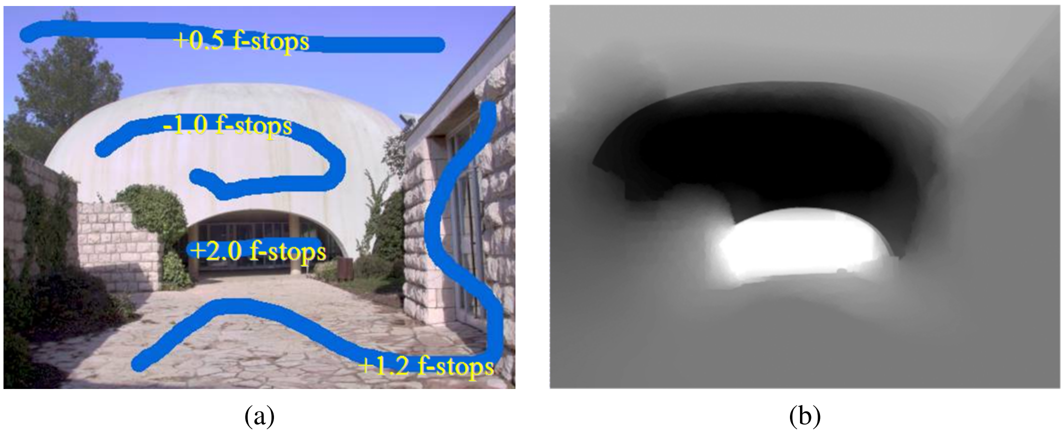


Figure 10.26 Interactive local tone mapping (Lischinski, Farbman *et al.* 2006) © 2006 ACM: (a) user-drawn strokes with associated exposure values $g(x,y)$; (b) corresponding piecewise-smooth exposure adjustment map $f(x,y)$.

rections) to a piecewise-continuous exposure correction map (Figure 10.26). The interpolation is performed by minimizing a locally weighted least squares (WLS) variational problem,

$$\min \int\int w_{\rm d}(x,y)\|f(x,y)-g(x,y)\|^2 dx\,dy + \lambda \int\int w_{\rm s}(x,y)\|\nabla f(x,y)\|^2 dx\,dy, \qquad (10.20)$$

where $g(x,y)$ and $f(x,y)$ are the input and output log exposure (attenuation) maps (Figure 10.26). The data weighting term $w_{\rm d}(x,y)$ is 1 at stroke locations and 0 elsewhere. The smoothness weighting term $w_{\rm s}(x,y)$ is inversely proportional to the log-luminance gradient,

$$w_{\rm s} = \frac{1}{\|\nabla H\|^\alpha + \epsilon} \qquad (10.21)$$

and hence encourages the $f(x,y)$ map to be smoother in low-gradient areas than along high-gradient discontinuities.[15] The same approach can also be used for fully automated tone mapping by setting

[15]In practice, the x and y discrete derivatives are weighted separately (Lischinski, Farbman *et al.* 2006). Their default parameter settings are $\lambda = 0.2$, $\alpha = 1$, and $\epsilon = 0.0001$.

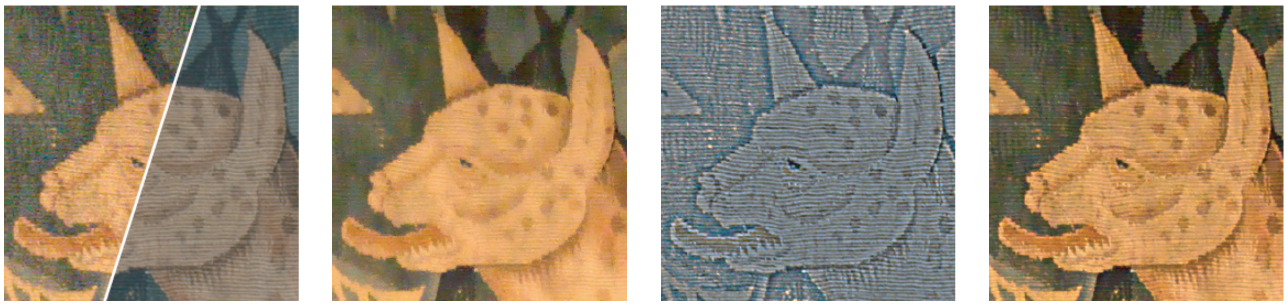


Figure 10.27 Detail transfer in flash/no-flash photography (Petschnigg, Agrawala *et al.* 2004) © 2004 ACM: (a) details of input ambient A and flash F images; (b) joint bilaterally filtered no-flash image A^{NR}; (c) detail layer F^{Detail} computed from the flash image F; (d) final merged image A^{Final}.

target exposure values at each pixel and allowing the weighted least squares to convert these into piecewise smooth adjustment maps.

The weighted least squares algorithm, which was originally developed for image colorization applications (Levin, Lischinski, and Weiss 2004), has since been applied to general edge-preserving smoothing in applications such as contrast enhancement (Bae, Paris, and Durand 2006) and tone mapping (Farbman, Fattal *et al.* 2008) where the bilateral filtering was previously used. It can also be used to perform HDR merging and tone mapping simultaneously (Raman and Chaudhuri 2007, 2009).

Given the wide range of locally adaptive tone mapping algorithms that have been developed, which ones should be used in practice? Freeman (2008) provides a great discussion of commercially available algorithms, their artifacts, and the parameters that can be used to control them. He also has a wealth of tips for HDR photography and workflow. I highly recommend his book for anyone contemplating additional research (or personal photography) in this area.

10.2.2 *Application*: Flash photography

While high dynamic range imaging combines images of a scene taken at different exposures, it is also possible to combine flash and non-flash images to achieve better exposure and color balance and to reduce noise (Eisemann and Durand 2004; Petschnigg, Agrawala *et al.* 2004).

The problem with flash images is that the color is often unnatural (it fails to capture the ambient illumination), there may be strong shadows or specularities, and there is a radial falloff in brightness away from the camera (Figures 10.1b and 10.27a). Non-flash photos taken under low light conditions often suffer from excessive noise (because of the high ISO gains and low photon counts) and blur (due to longer exposures). Is there some way to combine a non-flash photo taken just before the flash goes off with the flash photo to produce an image with good color values, sharpness, and low noise? In fact, the discontinued FujiFilm FinePix F40fd camera takes a pair of flash and no flash images in quick succession; however, it only lets you decide to keep one of them.

Petschnigg, Agrawala *et al.* (2004) approach this problem by first filtering the no-flash (ambient) image A with a variant of the bilateral filter called the *joint bilateral filter*[16] in which the range kernel (3.36)

$$r(i,j,k,l) = \exp\left(-\frac{\|f(i,j) - f(k,l)\|^2}{2\sigma_r^2}\right) \tag{10.22}$$

[16]Eisemann and Durand (2004) call this the *cross bilateral filter*.

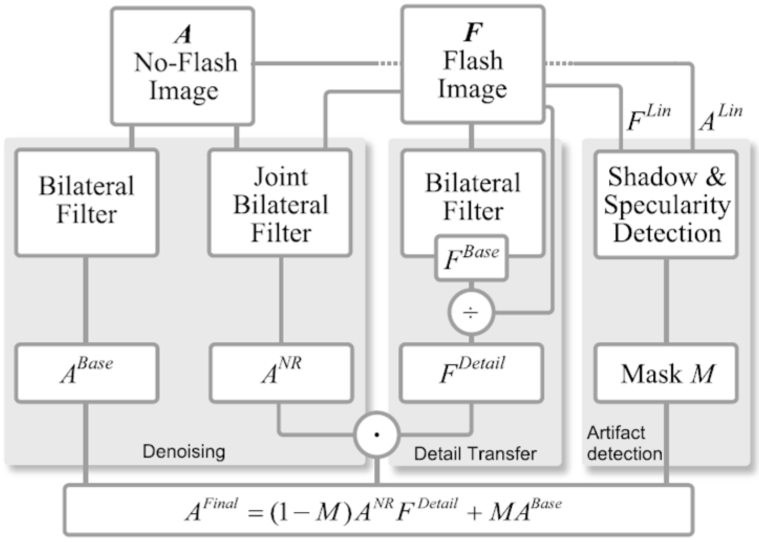


Figure 10.28 Flash/no-flash photography algorithm (Petschnigg, Agrawala *et al.* 2004) © 2004 ACM. The ambient (no-flash) image A is filtered with a regular bilateral filter to produce A^{Base}, which is used in shadow and specularity regions, and a joint bilaterally filtered noise reduced image A^{NR}. The flash image F is bilaterally filtered to produce a base image F^{Base} and a detail (ratio) image F^{Detail}, which is used to modulate the denoised ambient image. The shadow/specularity mask M is computed by comparing linearized versions of the flash and no-flash images.

is evaluated on the flash image F instead of the ambient image A, as the flash image is less noisy and hence has more reliable edges (Figure 10.27b). Because the contents of the flash image can be unreliable inside and at the boundaries of shadows and specularities, these are detected and a regular bilaterally filtered image A^{Base} is used instead (Figure 10.28).

The second stage of their algorithm computes a flash detail image

$$F^{Detail} = \frac{F + \epsilon}{F^{\text{Base}} + \epsilon}, \tag{10.23}$$

where F^{Base} is a bilaterally filtered version of the flash image F and $\epsilon = 0.02$. This detail image (Figure 10.27c) encodes details that may have been filtered away from the noise-reduced no-flash image A^{NR}, as well as additional details created by the flash camera, which often add crispness. The detail image is used to modulate the noise-reduced ambient image A^{NR} to produce the final results

$$A^{Final} = (1 - M)A^{NR}F^{Detail} + MA^{Base} \tag{10.24}$$

shown in Figures 10.1b and 10.27d.

Eisemann and Durand (2004) present an alternative algorithm that shares some of the same basic concepts. Both papers are well worth reading and contrasting (Exercise 10.6).

Flash images can also be used for a variety of additional applications such as extracting more reliable foreground mattes of objects (Raskar, Tan *et al.* 2004; Sun, Li *et al.* 2006). Given a large enough training set, it is also possible to decompose single flash images into their ambient and flash illumination components, which can be used to adjust their appearance (Aksoy, Kim *et al.* 2018). Flash photography is just one instance of the more general topic of *active illumination*, which is discussed in more detail by Raskar and Tumblin (2010) and Ikeuchi, Matsushita *et al.* (2020).

10.3 Super-resolution, denoising, and blur removal

While high dynamic range imaging enables us to obtain an image with a larger dynamic range than a single regular image, super-resolution enables us to create images with higher *spatial* resolution and less noise than regular camera images (Chaudhuri 2001; Park, Park, and Kang 2003; Capel and Zisserman 2003; Capel 2004; van Ouwerkerk 2006; Anwar, Khan, and Barnes 2020). Most commonly, super-resolution refers to the process of aligning and combining several input images to produce such high-resolution composites (Irani and Peleg 1991; Cheeseman, Kanefsky *et al.* 1993; Pickup, Capel *et al.* 2009; Wronski, Garcia-Dorado *et al.* 2019). However, some techniques can super-resolve a single image (Freeman, Jones, and Pasztor 2002; Baker and Kanade 2002; Fattal 2007; Anwar, Khan, and Barnes 2020) and are hence closely related to techniques for removing blur (Sections 3.4.1 and 3.4.2). Anwar, Khan, and Barnes (2020) provide a comprehensive review of single image super-resolution techniques with a particular focus on recent deep learning-based approaches.

A traditional way to formulate the super-resolution problem is to write down the stochastic image formation equations and image priors and to then use Bayesian inference to recover the super-resolved (original) sharp image. We can do this by generalizing the image formation equations used for image deblurring (Section 3.4.1), which we also used for blur kernel (PSF) estimation (Section 10.1.4). In this case, we have several observed images $\{o_k(\mathbf{x})\}$, as well as an image warping function $\hat{\mathbf{h}}_k(\mathbf{x})$ for each observed image (Figure 3.46). Combining all of these elements, we get the (noisy) observation equations[17]

$$o_k(\mathbf{x}) = D\{b(\mathbf{x}) * s(\hat{\mathbf{h}}_k(\mathbf{x}))\} + n_k(\mathbf{x}), \tag{10.25}$$

where D is the downsampling operator, which operates *after* the super-resolved (sharp) warped image $s(\hat{\mathbf{h}}_k(\mathbf{x}))$ has been convolved with the blur kernel $b(\mathbf{x})$. The above image formation equations lead to the following least squares problem,

$$\sum_k \|o_k(\mathbf{x}) - D\{b_k(\mathbf{x}) * s(\hat{\mathbf{h}}_k(\mathbf{x}))\}\|^2. \tag{10.26}$$

In most super-resolution algorithms, the alignment (warping) $\hat{\mathbf{h}}_k$ is estimated using one of the input frames as the *reference frame*; either feature-based (Section 8.1.3) or direct (image-based) (Section 9.2) parametric alignment techniques can be used. (A few algorithms, such as those described by Schultz and Stevenson (1996), Capel (2004), and Wronski, Garcia-Dorado *et al.* (2019) use dense (per-pixel flow) estimates.) A better approach is to re-compute the alignment by directly minimizing (10.26) once an initial estimate of $s(\mathbf{x})$ has been computed (Hardie, Barnard, and Armstrong 1997) or to *marginalize* out the motion parameters altogether (Pickup, Capel *et al.* 2007).

The point spread function (blur kernel) b_k is either inferred from knowledge of the image formation process (e.g., the amount of motion or defocus blur and the camera sensor optics) or calibrated from a test image or the observed images $\{o_k\}$ using one of the techniques described in Section 10.1.4. The problem of simultaneously inferring the blur kernel and the sharp image is known as *blind image deconvolution* (Kundur and Hatzinakos 1996; Levin 2006; Levin, Weiss *et al.* 2011; Campisi and Egiazarian 2017).[18]

[17] It is also possible to add an unknown bias–gain term to each observation (Capel 2004), as was done for motion estimation in (9.8).

[18] Notice that there is a chicken-and-egg problem if both the blur kernel and the super-resolved image are unknown. This can be "broken" either using structural assumptions about the sharp image, e.g., the presence of edges (Joshi, Szeliski, and Kriegman 2008) or prior models for the image, such as edge sparsity (Fergus, Singh *et al.* 2006).

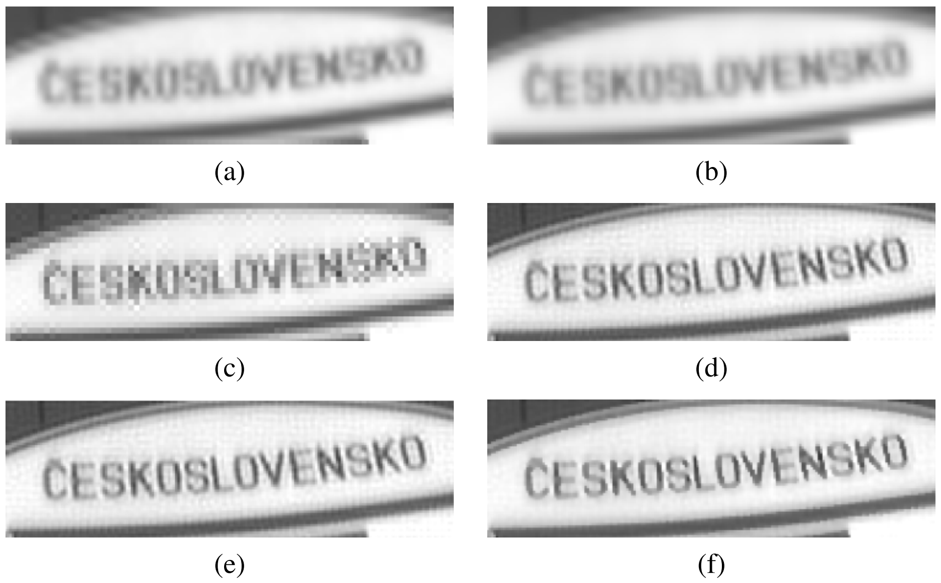


Figure 10.29 Super-resolution results using a variety of image priors (Capel 2001): (a) Low-res ROI (bicubic $3\times$ zoom); (b) average image; (c) MLE @ $1.25\times$ pixel-zoom; (d) simple $\|x\|^2$ prior ($\lambda = 0.004$); (e) GMRF ($\lambda = 0.003$); (f) HMRF ($\lambda = 0.01$, $\alpha = 0.04$). 10 images are used as input and a $3\times$ super-resolved image is produced in each case, except for the MLE result in (c).

Given an estimate of $\hat{\mathbf{h}}_k$ and $b_k(\mathbf{x})$, (10.26) can be re-written using matrix/vector notation as a large sparse least squares problem in the unknown values of the super-resolved pixels $\mathbf{s}$,

$$\sum_k \|\mathbf{o}_k - \mathbf{D}\mathbf{B}_k\mathbf{W}_k\mathbf{s}\|^2. \tag{10.27}$$

(Recall from (3.75) that once the warping function $\hat{\mathbf{h}}_k$ is known, values of $s(\hat{\mathbf{h}}_k(\mathbf{x}))$ depend linearly on those in $s(\mathbf{x})$.) An efficient way to solve this least squares problem is to use preconditioned conjugate gradient descent (Capel 2004), although some earlier algorithms, such as the one developed by Irani and Peleg (1991), used regular gradient descent (also known as iterative back projection (IBP) in the computed tomography literature).

The above formulation assumes that warping can be expressed as a simple (sinc or bicubic) interpolated resampling of the super-resolved sharp image, followed by a stationary (spatially invariant) blurring (PSF) and area integration process. However, if the surface is severely foreshortened, we have to take into account the spatially varying filtering that occurs during the image warping (Section 3.6.1), before we can then model the PSF induced by the optics and camera sensor (Wang, Kang *et al.* 2001; Capel 2004).

How well does this least squares (MLE) approach to super-resolution work? In practice, this depends a lot on the amount of blur and aliasing in the camera optics, as well as the accuracy in the motion and PSF estimates (Baker and Kanade 2002; Jiang, Wong, and Bao 2003; Capel 2004). Less blurring and more aliasing means that there is more (aliased) high frequency information available to be recovered. However, because the least squares (maximum likelihood) formulation uses no image prior, a lot of high-frequency noise can be introduced into the solution (Figure 10.29c).

For this reason, classic super-resolution algorithms assume some form of image prior. The simplest of these is to place a penalty on the image derivatives similar to Equations (4.29) and

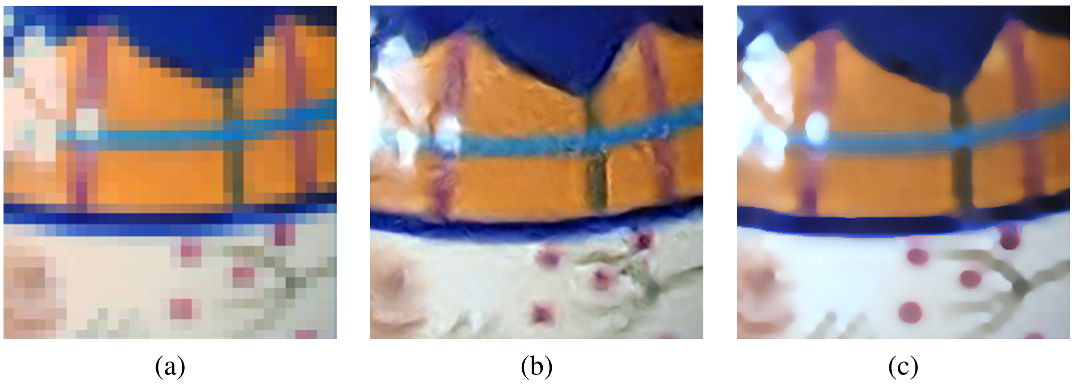


Figure 10.30 Example-based super-resolution: (a) original 32 × 32 low-resolution image; (b) example-based super-resolved 256 × 256 image (Freeman, Jones, and Pasztor 2002) © 2002 IEEE; (c) upsampling via imposed edge statistics (Fattal 2007) © 2007 ACM.

(4.42), e.g.,

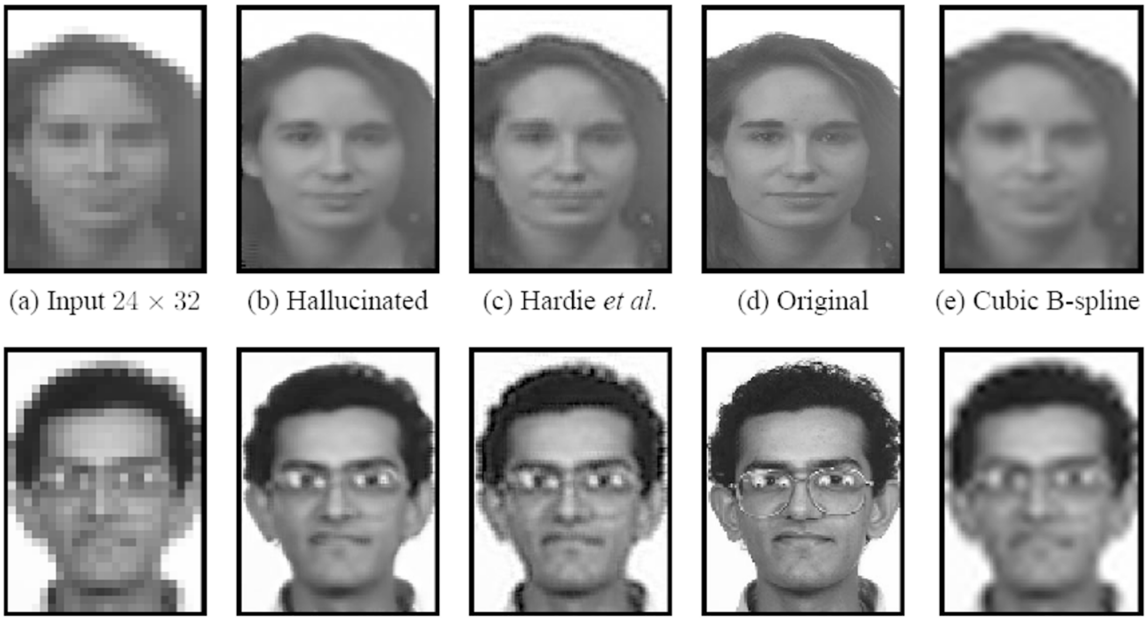
$$\sum_{(i,j)} \rho_p(s(i,j) - s(i+1,j)) + \rho_p(s(i,j) - s(i,j+1)). \tag{10.28}$$

As discussed in Section 4.3, when ρ_p is quadratic, this is a form of Tikhonov regularization (Section 4.2), and the overall problem is still linear least squares. The resulting prior image model is a Gaussian Markov random field (GMRF), which can be extended to other (e.g., diagonal) differences, as in Capel (2004) and Figure 10.29.

Unfortunately, GMRFs tend to produce solutions with visible ripples, which can also be interpreted as increased noise sensitivity in middle frequencies. A better image prior is a robust prior that encourages piecewise continuous solutions (Black and Rangarajan 1996), see Appendix B.3. Examples of such priors include the Huber potential (Schultz and Stevenson 1996; Capel and Zisserman 2003), which is a blend of a Gaussian with a longer-tailed Laplacian, and the even sparser (heavier-tailed) hyper-Laplacians used by Levin, Fergus *et al.* (2007) and Krishnan and Fergus (2009). It is also possible to learn the parameters for such priors using cross-validation (Capel 2004; Pickup 2007).

While sparse (robust) derivative priors can reduce rippling effects and increase edge sharpness, they cannot *hallucinate* higher-frequency texture or details. To do this, a training set of sample images can be used to find plausible mappings between low-frequency originals and the missing higher frequencies. Inspired by some of the example-based texture synthesis algorithms we discuss in Section 10.5, the *example-based super-resolution* algorithm developed by Freeman, Jones, and Pasztor (2002) uses training images to *learn* the mapping between local texture patches and missing higher-frequency details. To ensure that overlapping patches are similar in appearance, a Markov random field is used and optimized using either belief propagation (Freeman, Pasztor, and Carmichael 2000) or a raster-scan deterministic variant (Freeman, Jones, and Pasztor 2002). Figure 10.30 shows the results of hallucinating missing details using this approach and compares these results to a more recent algorithm by Fattal (2007). This latter algorithm learns to predict oriented gradient magnitudes in the finer resolution image based on a pixel's location relative to the nearest detected edge along with the corresponding edge statistics (magnitude and width). It is also possible to combine sparse (robust) derivative priors with example-based super-resolution, as shown by Tappen, Russell, and Freeman (2003).

An alternative (but closely related) form of hallucination is to *recognize* the parts of a training


(a) Input 24×32 (b) Hallucinated (c) Hardie *et al.* (d) Original (e) Cubic B-spline

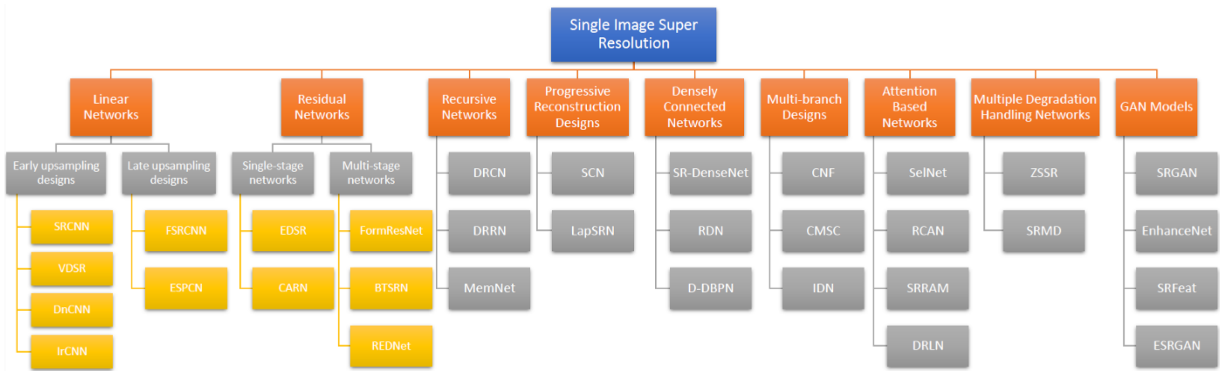
(f) Input 24×32 (g) Hallucinated (h) Hardie *et al.* (i) Original (j) Cubic B-spline

Figure 10.31 Recognition-based super-resolution (Baker and Kanade 2002) © 2002 IEEE. The *Hallucinated* column shows the results of the recognition-based algorithm compared to the regularization-based approach of Hardie, Barnard, and Armstrong (1997).

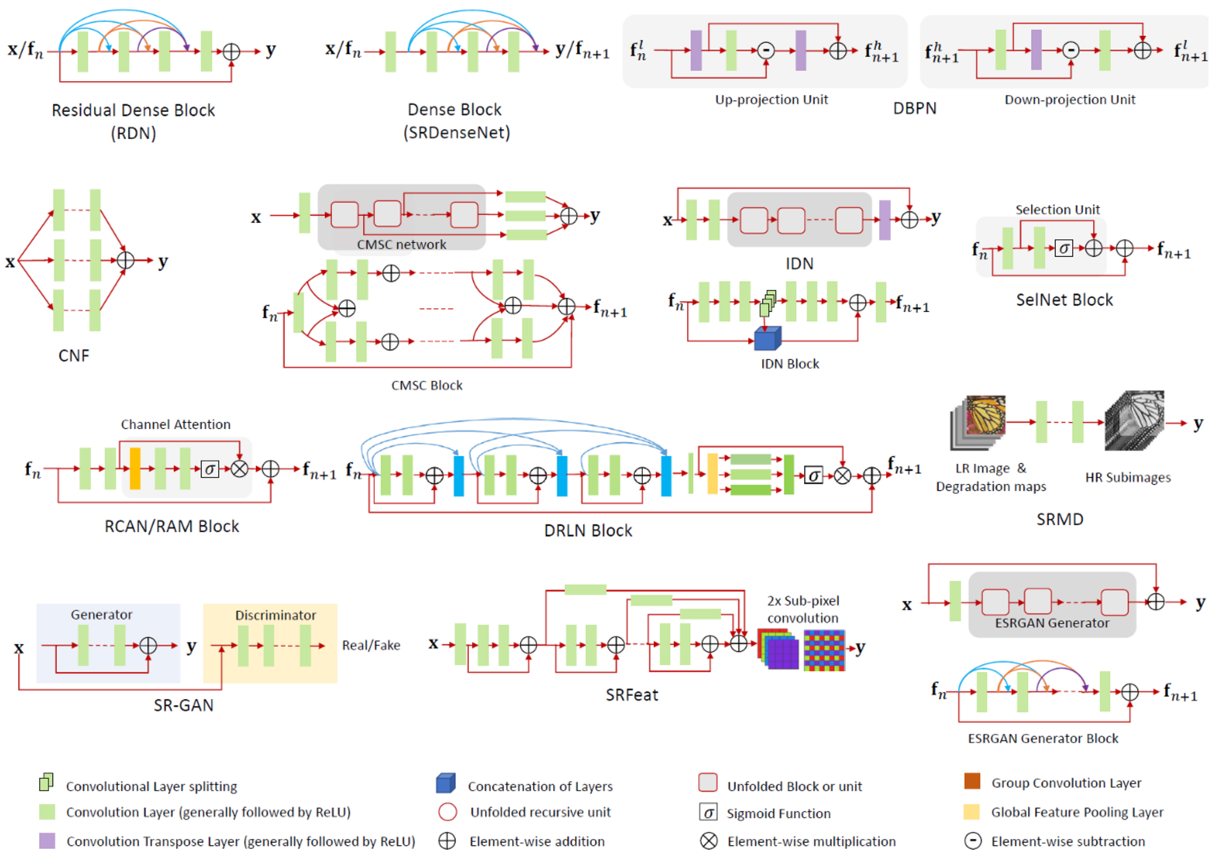
database of images to which a low-resolution pixel might correspond. In their work, Baker and Kanade (2002) use local derivative-of-Gaussian filter responses as features and then match *parent structure* vectors in a manner similar to De Bonet (1997).[19] The high-frequency gradient at each recognized training image location is then used as a constraint on the super-resolved image, along with the usual reconstruction (prediction) Equation (10.26). Figure 10.31 shows the result of hallucinating higher-resolution faces from lower-resolution inputs; Baker and Kanade (2002) also show examples of super-resolving known-font text. Exercise 10.7 gives more details on how to implement and test one or more of these super-resolution techniques.

The latest trend in super-resolution has been the use of deep neural networks to directly predict super-resolved images. This approach, which began with the seminal work of Dong, Loy *et al.* (2016), has generated dozens of different DNNs and architectures, including the Deep Learning Super Sampling hardware embedded in the latest NVIDIA graphics cards (Burnes 2020). The recent survey on single-image super-resolution by Anwar, Khan, and Barnes (2020) categorizes these algorithms into a taxonomy (Figure 10.32a), provides a pictorial summary network architectures (Figure 10.32b), and compares the super-resolution results both numerically and visually on noise-free known bicubic-kernel decimation image datasets. While the results shown in Figure 10.33 show dramatic differences between algorithms, it is not clear how well these algorithms generalize to real-world noisy input with unknown blur kernels. The RealSR real-world super-resolution dataset developed by (Cai, Zeng *et al.* 2019), shot using a zoom lens on a digital camera, provides a means to test (and train) algorithms on real imaging degradations. This dataset forms the basis

[19]For face super-resolution, where all the images are pre-aligned, only corresponding pixels in different images are examined.



(a)



(b)

Figure 10.32 Recent deep neural network algorithms for single image super-resolution (Anwar, Khan, and Barnes 2020) © 2020 ACM: (a) a taxonomy of the algorithms based on their general approach; (b) schematic architectures for a subset of the algorithms.

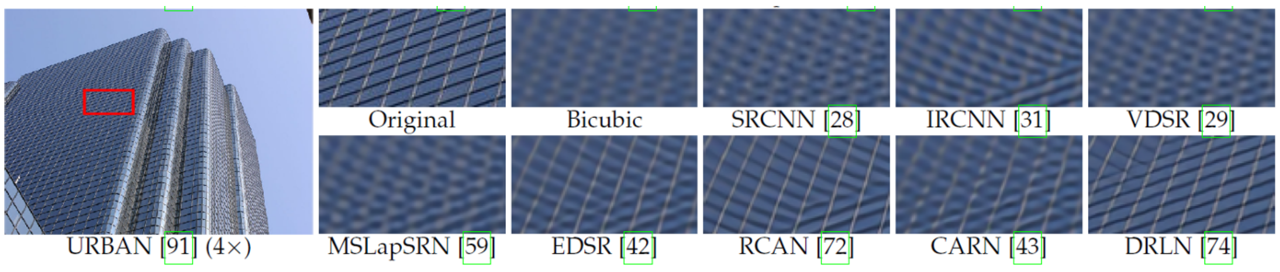


Figure 10.33 Visual comparison of some super-resolution algorithms (Anwar, Khan, and Barnes 2020) © 2020 ACM.

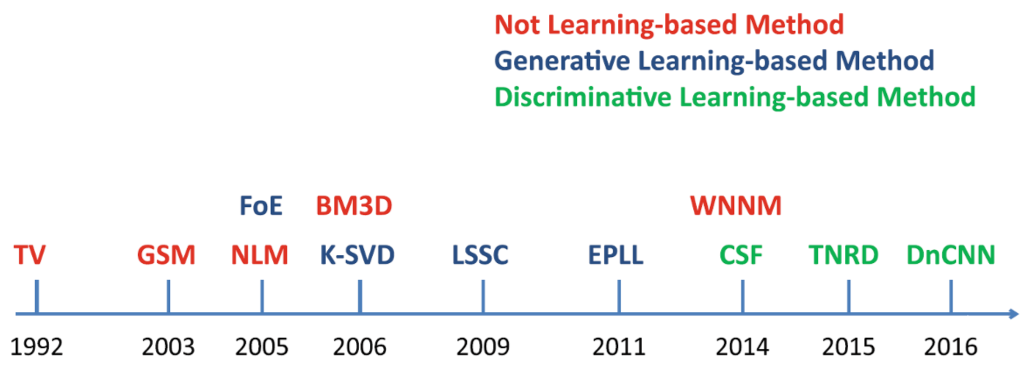

Not Learning-based Method
Generative Learning-based Method
Discriminative Learning-based Method
TV
1992
GSM
2003
FoE
NLM
2005
BM3D
K-SVD
2006
LSSC
2009
EPLL
2011
WNNM
CSF
2014
TNRD
2015
DnCNN
2016

Figure 10.34 Timeline of denoising algorithms from Gu and Timofte (2019) © 2019 Springer.

for the NTIRE challenges on real image super-resolution (Cai, Gu *et al.* 2019),[20] which provide empirical comparisons of recent deep network-based algorithms.

While single-image super-resolution is interesting, much more impressive (and practical) results can be obtained by building a multi-frame super-resolution algorithm directly into a smartphone camera, where the processing can be done jointly with the image demosaicing. We discuss recent work by Wronski, Garcia-Dorado *et al.* (2019) in Section 10.3.1 and Figure 10.38 on color image demosaicing. It is also possible to upsample videos temporally using frame interpolation (Section 9.4.1), spatially using video super-resolution (Liu and Sun 2013; Kappeler, Yoo *et al.* 2016; Shi, Caballero *et al.* 2016; Tao, Gao *et al.* 2017; Nah, Timofte *et al.* 2019; Isobe, Jia *et al.* 2020; Li, Tao *et al.* 2020), or simultaneously in both the spatial and temporal dimensions (Kang, Jo *et al.* 2020).

Single and multi-frame denoising

Image denoising is one of the classic problems in image processing and computer vision (Perona and Malik 1990b; Rudin, Osher, and Fatemi 1992; Buades, Coll, and Morel 2005b). Over the last four decades, hundreds of algorithms have been developed, and the field continues to be actively studied, with recent algorithms all being based on deep neural networks.

The latest benchmark for comparing image denoising algorithms, the NTIRE 2020 Challenge on Real Image Denoising (Abdelhamed, Afifi *et al.* 2020), is based on a smartphone image denoising dataset (SIDD) (Abdelhamed, Lin, and Brown 2018), where the noise-free ground truth images

[20] https://data.vision.ee.ethz.ch/cvl/ntire20/, https://data.vision.ee.ethz.ch/cvl/aim20/

were obtained by averaging sets of 150 noisy images. This provides much more realistic and varied real-world noise and image processing models than the synthetically noised images used in most previous benchmarks (with the exception of (Plötz and Roth 2017)).

A recent (brief) survey on image denoising by Gu and Timofte (2019) includes the following seminal denoising papers[21] (see Figure 10.34 for a timeline):

- total variation (TV) (Rudin, Osher, and Fatemi 1992; Chan, Osher, and Shen 2001; Chambolle 2004; Chan and Shen 2005),
- Gaussian scale mixtures (GSMs) (Lyu and Simoncelli 2009),
- Field of Experts (FoE) (Roth and Black 2009),
- non-local means (NLM) (Buades, Coll, and Morel 2005a,b),
- BM3D (Dabov, Foi *et al.* 2007),
- sparse overcomplete dictionaries (K-SVD) (Aharon, Elad, and Bruckstein 2006),
- expected patch log likelihood (EPLL) (Zoran and Weiss 2011),
- an MLP denoiser (Burger, Schuler, and Harmeling 2012),
- weighted nuclear norm minimization (WNNM) (Gu, Zhang *et al.* 2014),
- shrinkage fields (CSF) (Schmidt and Roth 2014),
- Trainable Nonlinear Reaction Diffusion (TNRD) (Chen and Pock 2016),
- a cross-channel noise model for color images (Nam, Hwang *et al.* 2016),
- a denoising residual CNN (DnCNN) (Zhang, Zuo *et al.* 2017), which is now considered the baseline for DNN denoising, and
- learning to see in the dark (Chen, Chen *et al.* 2018).

While these results show dramatic improvement over time, today's imaging sensors for the most part produce relatively clean images, except in low-light situations, where the ISO camera gain must be increased and the read and photon noise become comparable to the signal strength. In this regime, it is preferable, if possible, to take a rapid burst of images at low ISO (gain) and then combine these to obtain a denoised image (Hasinoff, Kutulakos *et al.* 2009; Hasinoff, Durand, and Freeman 2010; Liu, Yuan *et al.* 2014). This approach was generalized and applied to low-light photography in the HDR+ system of Hasinoff, Sharlet *et al.* (2016). More recent work along these lines, some of which combines low-light photography, demosaicing, and in some cases super-resolution, includes papers by Godard, Matzen, and Uyttendaele (2018), Chen, Chen *et al.* (2018), Mildenhall, Barron *et al.* (2018), Wronski, Garcia-Dorado *et al.* (2019), and (Rong, Demandolx *et al.* 2020). Liba, Murthy *et al.* (2019) describe the technology that underlies Google's *Night Sight* feature, which not only robustly aligns and merges different moving regions together under noisy conditions, but also introduces the concept of "motion metering" to determine the optimal number of frames and exposure times.

[21] I have added a few more papers from the ICCV tutorial by Brown (2019) and a few additional recommendations from Abdelrahman Abdelhamed.

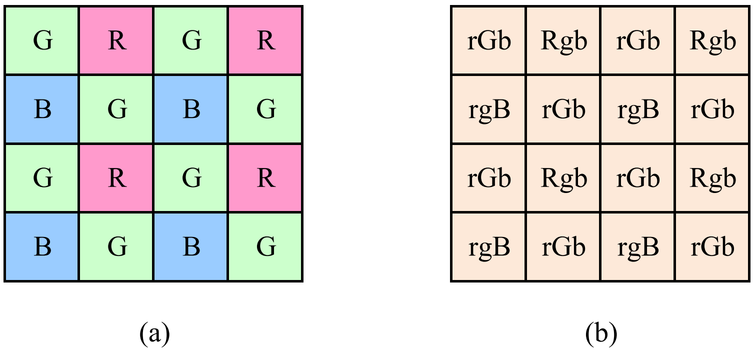


Figure 10.35 Bayer RGB pattern: (a) color filter array layout; (b) interpolated pixel values, with unknown (guessed) values shown as lower case.

Blur removal

Under favorable conditions, super-resolution and related upsampling techniques can increase the resolution of a well-photographed image or image collection. When the input images are blurry to start with, the best one can often hope for is to reduce the amount of blur. This problem is closely related to super-resolution, with the biggest differences being that the blur kernel b is usually much larger (and unknown) and the downsampling factor D is unity.

A large literature on image deblurring exists; some publications with nice literature reviews include those by Fergus, Singh *et al.* (2006), Yuan, Sun *et al.* (2008), and Joshi, Zitnick *et al.* (2009). It is also possible to reduce blur by combining sharp (but noisy) images with blurrier (but cleaner) images (Yuan, Sun *et al.* 2007), take lots of quick exposures (Hasinoff and Kutulakos 2011; Hasinoff, Kutulakos *et al.* 2009; Hasinoff, Durand, and Freeman 2010), or use *coded aperture* techniques to simultaneously estimate depth and reduce blur (Levin, Fergus *et al.* 2007; Zhou, Lin, and Nayar 2009). When available, data from on-board IMUs (inertial measurement units) can be used for blur kernel determination (Joshi, Kang *et al.* 2010). It is also possible to use information from dual-pixel sensors to aid the deblurring of misfocused images (Abuolaim and Brown 2020).

The past decade has seen the introductions of a large number of new learning-based deblurring algorithms (Sun, Cao *et al.* 2015; Schuler, Hirsch *et al.* 2016; Nah, Hyun Kim, and Mu Lee 2017; Kupyn, Budzan *et al.* 2018; Tao, Gao *et al.* 2018; Zhang, Dai *et al.* 2019; Kupyn, Martyniuk *et al.* 2019). There has also been some work on artificially re-introducing texture in deblurred images to better match the expected image statistics (Cho, Joshi *et al.* 2012), i.e., what is now commonly called *perceptual loss* (Section 5.3.4).

10.3.1 Color image demosaicing

A special case of super-resolution, which is used daily in most digital still cameras, is the process of *demosaicing* samples from a color filter array (CFA) into a full-color RGB image. Figure 10.35 shows the most commonly used CFA known as the *Bayer pattern*, which has twice as many green (G) sensors as red and blue sensors.

The process of going from the known CFA pixels values to the full RGB image is quite challenging. Unlike regular super-resolution, where small errors in guessing unknown values usually show up as blur or aliasing, demosaicing artifacts often produce spurious colors or high-frequency patterned *zippering*, which are quite visible to the eye (Figure 10.36b).

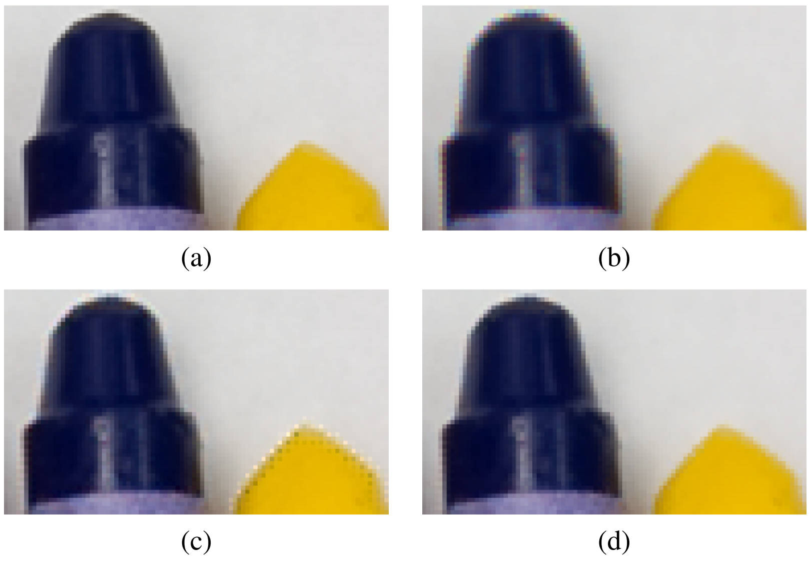


Figure 10.36 CFA demosaicing results (Bennett, Uyttendaele *et al.* 2006) © 2006 Springer: (a) original full-resolution image (a color subsampled version is used as the input to the algorithms); (b) bilinear interpolation results, showing color fringing near the tip of the blue crayon and zippering near its left (vertical) edge; (c) the high-quality linear interpolation results of Malvar, He, and Cutler (2004) (note the strong halo/checkerboard artifacts on the yellow crayon); (d) using the local two-color prior of Bennett, Uyttendaele *et al.* (2006).

Over the years, a variety of techniques have been developed for image demosaicing (Kimmel 1999). Longere, Delahunt *et al.* (2002), Tappen, Russell, and Freeman (2003), and Li, Gunturk, and Zhang (2008) provide surveys of the field as well as comparisons of previously developed techniques using perceptually motivated metrics. To reduce the zippering effect, most techniques use the edge or gradient information from the green channel, which is more reliable because it is sampled more densely, to infer plausible values for the red and blue channels, which are more sparsely sampled.

To reduce color fringing, some techniques perform a color space analysis, e.g., using median filtering on color opponent channels (Longere, Delahunt *et al.* 2002). The approach of Bennett, Uyttendaele *et al.* (2006) computes local two-color models from an initial demosaicing result, using a moving 5×5 window to find the two dominant colors (Figure 10.37).[22]

Once the local color model has been estimated at each pixel, a Bayesian approach is then used to encourage pixel values to lie along each color line and to cluster around the dominant color values, which reduces halos (Figure 10.36d). The Bayesian approach also supports the simultaneous application of demosaicing, denoising, and super-resolution, i.e., multiple CFA inputs can be merged into a higher-quality full-color image. More recent work that combines demosaicing and denoising includes papers by Chatterjee, Joshi *et al.* (2011) and Gharbi, Chaurasia *et al.* (2016). The NTIRE 2020 Challenge on Real Image Denoising (Abdelhamed, Afifi *et al.* 2020) includes a track on denoising RAW (i.e., color filter array) images. There's also an interesting paper by Jin, Facciolo, and Morel (2020) studying whether denoising should be applied before or after demosaicing.

As we mentioned before, burst photography (Cohen and Szeliski 2006; Hasinoff, Kutulakos *et al.* 2009; Hasinoff and Kutulakos 2011), i.e., the combination of rapidly acquired sequences of images, is becoming ubiquitous in smartphone cameras. A wonderful example of a recent system

[22] Previous work on locally linear color models (Klinker, Shafer, and Kanade 1990; Omer and Werman 2004) focuses on color and illumination variation within a single material, whereas Bennett, Uyttendaele *et al.* (2006) use the two-color model to describe variations across color (material) edges.

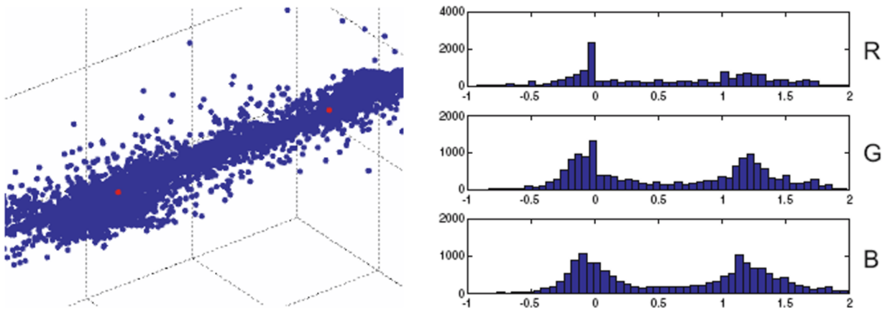


Figure 10.37 Two-color model computed from a collection of local 5 × 5 neighborhoods (Bennett, Uyttendaele *et al.* 2006) © 2006 Springer. After two-means clustering and reprojection along the line joining the two dominant colors (red dots), the majority of the pixels fall near the fitted line. The distribution along the line, projected along the RGB axes, is peaked at 0 and 1, the two dominant colors.

that performs joint demosaicing and multi-frame super-resolutions, based on locally adapted kernel functions (Figure 10.38), is the paper by Wronski, Garcia-Dorado *et al.* (2019), which underlies the *Super Res Zoom* feature in Google's Pixel smartphones.

10.3.2 Lens blur (bokeh)

The ability to create a shallow depth-of-field photograph using a large aperture (Section 2.2.3) has always been one of the advantages of large-format, e.g., single lens reflex (SLR), cameras. The desire to artificially simulate refocusable, shallow depth-of-field cameras was one of the driving impetuses behind computational photography (Levoy 2006) and led to the development of lightfield cameras (Ng, Levoy *et al.* 2005), which we discuss in Section 14.3.4. Although some commercial models, such as the Lytro, were produced, the ability to create such images with smartphone cameras has only recently become widespread.[23]

The Apple iPhone 7 Plus with its dual (wide/telephoto) lens was the first smartphone to introduce this feature, which they called the *Portrait mode*. Although the technical details behind this feature have never been published, the algorithm that estimates the depth image (which can be read out of the metadata in the portrait images) probably uses some combination of stereo matching and deep learning. A little later, Google released its own Portrait Mode, which uses the dual pixels, originally designed for focusing the camera optics, along with person segmentation to compute a depth map, as described in the paper by Wadhwa, Garg *et al.* (2018). Once the depth map has been estimated, a fast approximation to a back-to-front blurred *over* compositing operator is used to correctly blur the background without including foreground colors. More recently Garg, Wadhwa *et al.* (2019) have improved the quality of the depth estimation using a deep network, and also used two lenses (along with dual pixels) to produce even higher-quality depth maps (Zhang, Wadhwa *et al.* 2020).

One final word on *bokeh*, which is the term photographers use to describe the shape of the glints or highlights that appear in an image. This shape is determined by the configuration of the *aperture blades* that control how much light enters the lens (on larger-format cameras). Traditionally, these were made with straight metal leaves, which resulted in polygonal apertures, but they were then mostly replaced by curved leaves to produce a more circular shape. When using computational

[23]An earlier feature called Google Lens Blur, which required moving the camera in a pattern, https://ai.googleblog.com/2014/04/lens-blur-in-new-google-camera-app.html, was never widely used.

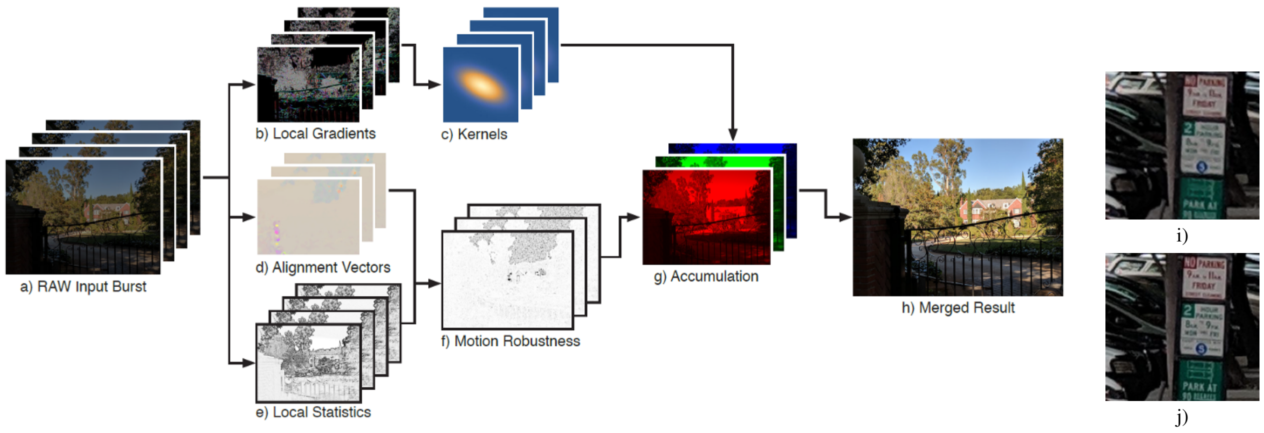


Figure 10.38 Hand-held multi-frame super-resolution (Wronski, Garcia-Dorado *et al.* 2019) © 2019 ACM. Processing pipeline, showing: (a) the captured burst of raw (Bayer CFA) images; (b) local gradients used to compute oriented kernels (c); (d) motion estimates, combined with local statistics (e) to compute blend weights (f). Results from (i) the previous method of Hasinoff, Sharlet *et al.* (2016) and (j) Wronski, Garcia-Dorado *et al.* (2019).

photography, we can use whatever shape is pleasing to the photographer, but preferably *not* a Gaussian blur, which does not correspond to any real aperture and produces indistinct highlights. The paper by Wadhwa, Garg *et al.* (2018) uses a circular bokeh for their depth-of-field effect and a more recent version performs the computations in the HDR (radiance) space to produce more accurate highlights.[24]

10.4 Image matting and compositing

Image matting and compositing is the process of cutting a foreground object out of one image and pasting it against a new background (Smith and Blinn 1996; Wang and Cohen 2009). It is commonly used in television and film production to composite a live actor in front of computer-generated imagery such as weather maps or 3D virtual characters and scenery (Wright 2006; Brinkmann 2008), and it has recently become a popular feature in video conferencing systems.

We have already seen a number of tools for interactively segmenting objects in an image, including snakes (Section 7.3.1), scissors (Section 7.3.1), and GrabCut segmentation (Section 4.3.2). While these techniques can generate reasonable pixel-accurate segmentations, they fail to capture the subtle interplay of foreground and background colors at *mixed pixels* along the boundary (Szeliski and Golland 1999) (Figure 10.39a).

To successfully copy a foreground object from one image to another without visible discretization artifacts, we need to *pull a matte*, i.e., to estimate a soft opacity channel α and the uncontaminated foreground colors F from the input composite image C. Recall from Section 3.1.3 (Figure 3.4) that the compositing equation (3.8) can be written as

$$C = (1 - \alpha)B + \alpha F. \tag{10.29}$$

This operator attenuates the influence of the background image B by a factor $(1 - \alpha)$ and then adds in the (partial) color values corresponding to the foreground element F.

[24]https://ai.googleblog.com/2019/12/improvements-to-portrait-mode-on-google.html

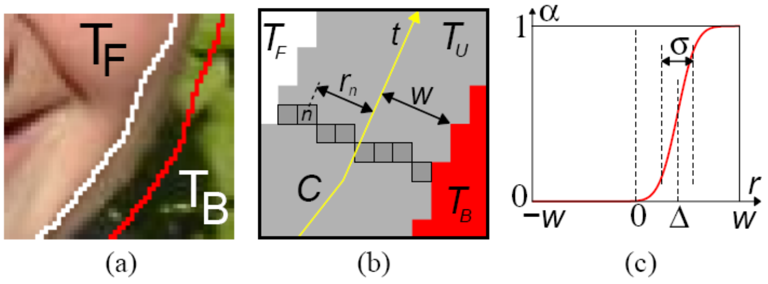


Figure 10.39 Softening a hard segmentation boundary (border matting) (Rother, Kolmogorov, and Blake 2004) © 2004 ACM: (a) the region surrounding a segmentation boundary where pixels of mixed foreground and background colors are visible; (b) pixel values along the boundary are used to compute a soft alpha matte; (c) at each point along the curve t, a displacement Δ and a width σ are estimated.

While the compositing operation is easy to implement, the reverse *matting* operation of estimating F, α, and B given an input image C is much more challenging (Figure 10.40). To see why, observe that while the composite pixel color C provides three measurements, the F, α, and B unknowns have a total of seven degrees of freedom. Devising techniques to estimate these unknowns despite the underconstrained nature of the problem is the essence of image matting.

In this section, we review a number of image matting techniques. We begin with *blue screen matting*, which assumes that the background is a constant known color, and discuss its variants, two-screen matting (when multiple backgrounds can be used) and difference matting (where the known background is arbitrary). We then discuss local variants of *natural image matting*, where both the foreground and background are unknown. In these applications, it is usual to first specify a *trimap*, i.e., a three-way labeling of the image into foreground, background, and unknown regions (Figure 10.40b). Next, we present some global optimization approaches to natural image matting. Finally, we discuss variants on the matting problem, including shadow matting, flash matting, and environment matting.

10.4.1 Blue screen matting

Blue screen matting involves filming an actor (or object) in front of a constant colored background. While originally bright blue was the preferred color, bright green is now more commonly used (Wright 2006; Brinkmann 2008). Smith and Blinn (1996) discuss a number of techniques for blue screen matting, which are mostly described in patents rather than in the open research literature. Early techniques used linear combinations of object color channels with user-tuned parameters to estimate the opacity α.

Chuang, Curless *et al.* (2001) describe a newer technique called Mishima's algorithm, which involves fitting two polyhedral surfaces (centered at the mean background color), separating the foreground and background color distributions, and then measuring the relative distance of a novel color to these surfaces to estimate α (Figure 10.41e). While this technique works well in many studio settings, it can still suffer from *blue spill*, where translucent pixels around the edges of an object acquire some of the background blue coloration.

Two-screen matting. In their paper, Smith and Blinn (1996) also introduce an algorithm called *triangulation matting* that uses more than one known background color to over-constrain the equations required to estimate the opacity α and foreground color F.

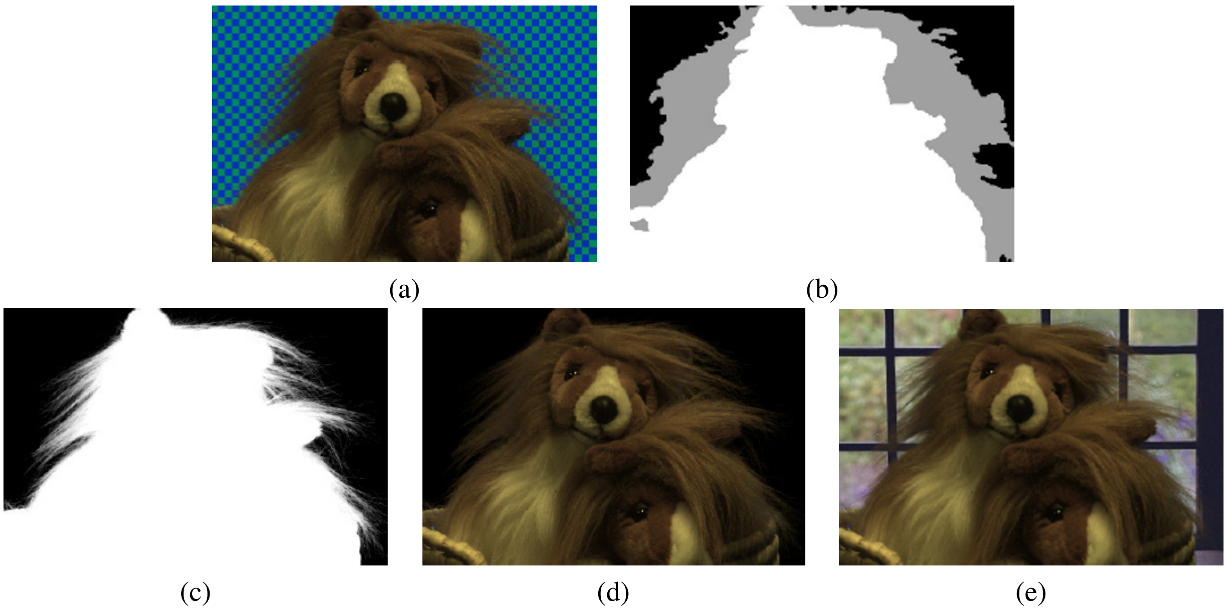


Figure 10.40 Natural image matting (Chuang, Curless *et al.* 2001) © 2001 IEEE: (a) input image with a "natural" (non-constant) background; (b) hand-drawn trimap—gray indicates unknown regions; (c) extracted alpha map; (d) extracted (premultiplied) foreground colors; (e) composite over a new background.

For example, consider in the compositing equation (10.29) setting the background color to black, i.e., $B = 0$. The resulting composite image C is therefore equal to αF. Replacing the background color with a different known non-zero value B now results in

$$C - \alpha F = (1 - \alpha)B, \tag{10.30}$$

which is an overconstrained set of (color) equations for estimating α. In practice, B should be chosen so as not to saturate C and, for best accuracy, several values of B should be used. It is also important that colors be linearized before processing, which is the case for *all* image matting algorithms. Papers that generate ground truth alpha mattes for evaluation purposes normally use these techniques to obtain accurate matte estimates (Chuang, Curless *et al.* 2001; Wang and Cohen 2007a; Levin, Acha, and Lischinski 2008; Rhemann, Rother *et al.* 2008, 2009).[25] Exercise 10.8 has you do this as well.

Difference matting. A related approach when the background is irregular but known is called *difference matting* (Wright 2006; Brinkmann 2008). It is most commonly used when the actor or object is filmed against a static background, e.g., for office video conferencing, person tracking applications (Toyama, Krumm *et al.* 1999), or to produce silhouettes for volumetric 3D reconstruction techniques (Section 12.7.3) (Szeliski 1993; Seitz and Dyer 1997; Seitz, Curless *et al.* 2006). It can also be used with a panning camera where the background is composited from frames where the foreground has been removed using a *garbage matte* (Section 10.4.5) (Chuang, Agarwala *et al.* 2002). Another application is the detection of visual continuity errors in films, i.e., differences in the background when a shot is re-taken at a later time (Pickup and Zisserman 2009).

In the case where the foreground and background motions can both be specified with parametric transforms, high-quality mattes can be extracted using a generalization of triangulation matting

[25] See the alpha matting evaluation website at http://alphamatting.com.

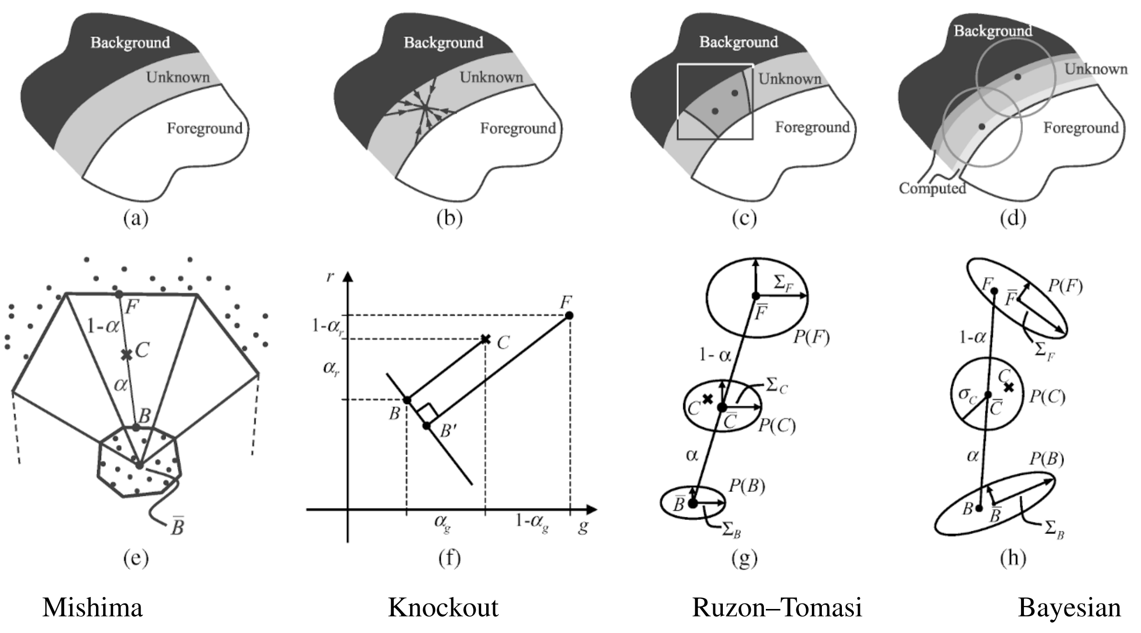


Figure 10.41 Image matting algorithms (Chuang, Curless *et al.* 2001) © 2001 IEEE. Mishima's algorithm models global foreground and background color distribution as polyhedral surfaces centered around the mean background (blue) color. Knockout uses a local color estimate of foreground and background for each pixel and computes α along each color axis. Ruzon and Tomasi's algorithm locally models foreground and background colors and variances. Chuang *et al.*'s Bayesian matting approach computes a MAP estimate of (fractional) foreground color and opacity given the local foreground and background distributions.

(Wexler, Fitzgibbon, and Zisserman 2002). When frames need to be processed independently, however, the results are often of poor quality (Figure 10.42). In such cases, using a pair of stereo cameras as input can dramatically improve the quality of the results (Criminisi, Cross *et al.* 2006; Yin, Criminisi *et al.* 2007).

10.4.2 Natural image matting

The most general version of image matting is when nothing is known about the background except, perhaps, for a rough segmentation of the scene into foreground, background, and unknown regions, which is known as the *trimap* (Figure 10.40b). Some techniques, however, relax this requirement and allow the user to just draw a few strokes or scribbles in the image: see Figures 10.45 and 10.46 (Wang and Cohen 2005; Wang, Agrawala, and Cohen 2007; Levin, Lischinski, and Weiss 2008; Rhemann, Rother *et al.* 2008; Rhemann, Rother, and Gelautz 2008). Fully automated single image matting results have also been reported (Levin, Acha, and Lischinski 2008; Singaraju, Rother, and Rhemann 2009). The survey paper by Wang and Cohen (2009) has detailed descriptions and comparisons of all of these techniques, a selection of which are described briefly below, while the website http://alphamatting.com has up-to-date lists and numerical comparisons of the most recent algorithms.

A relatively simple algorithm for performing natural image matting is Knockout, as described by Chuang, Curless *et al.* (2001) and illustrated in Figure 10.41f. In this algorithm, the nearest known foreground and background pixels (in image space) are determined and then blended with neighboring known pixels to produce a per-pixel foreground F and background B color estimate.

The background color is then adjusted so that the measured color C lies on the line between F and B. Finally, opacity α is estimated on a per-channel basis, and the three estimates are combined based on per-channel color differences. (This is an approximation to the least squares solution for α.) Figure 10.42 shows that Knockout has problems when the background consists of more than one dominant local color.

More accurate matting results can be obtained if we treat the foreground and background colors as distributions sampled over some region (Figure 10.41g–h). Ruzon and Tomasi (2000) model local color distributions as mixtures of (uncorrelated) Gaussians and compute these models in strips. They then find the pairing of mixture components F and B that best describes the observed color C, compute the α as the relative distance between these means, and adjust the estimates of F and B so that they are collinear with C.

Chuang, Curless *et al.* (2001) and Hillman, Hannah, and Renshaw (2001) use full 3×3 color covariance matrices to model mixtures of correlated Gaussians, and compute estimates independently for each pixel. Matte extraction proceeds in strips starting from known color values growing into the unknown regions, so that recently computed F and B colors can be used in later stages.

To estimate the most likely value of an unknown pixel's opacity and (unmixed) foreground and background colors, Chuang *et al.* use a fully Bayesian formulation that maximizes

$$P(F, B, \alpha | C) = P(C | F, B, \alpha) P(F) P(B) P(\alpha) / P(C). \tag{10.31}$$

This is equivalent to minimizing the negative log likelihood

$$L(F, B, \alpha | C) = L(C | F, B, \alpha) + L(F) + L(B) + L(\alpha) \tag{10.32}$$

(dropping the $L(C)$ term because it is constant).

Let us examine each of these terms in turn. The first, $L(C|F, B, \alpha)$, is the likelihood that pixel color C was observed given values for the unknowns (F, B, α). If we assume Gaussian noise in our observation with variance σ_C^2, this negative log likelihood (data term) is

$$L(C) = {}^1\!/_2 \|C - [\alpha F + (1 - \alpha) B]\|^2 / \sigma_C^2, \tag{10.33}$$

as illustrated in Figure 10.41h.

The second term, $L(F)$, corresponds to the likelihood that a particular foreground color F comes from the Gaussian mixture model. After partitioning the sample foreground colors into clusters, a weighted mean $\overline{F}$ and covariance Σ_F are computed, where the weights are proportional to a given foreground pixel's opacity and distance from the unknown pixel.[26] The negative log likelihood for each cluster is thus given by

$$L(F) = (F - \overline{F})^T \Sigma_F^{-1} (F - \overline{F}). \tag{10.34}$$

A similar method is used to estimate unknown background color distributions. If the background is already known, i.e., for blue screen or difference matting applications, its measured color value and variance are used instead.

An alternative to modeling the foreground and background color distributions as mixtures of Gaussians is to keep around the original color samples and to compute the most likely pairings that explain the observed color C (Wang and Cohen 2005, 2007a). These techniques are described in more detail in (Wang and Cohen 2009).

[26]Note that in this whole chapter, we mostly use upper-case italics to denote images or pixel values, even when they are color vectors. The covariance Σ_F is a 3×3 matrix for each foreground cluster.

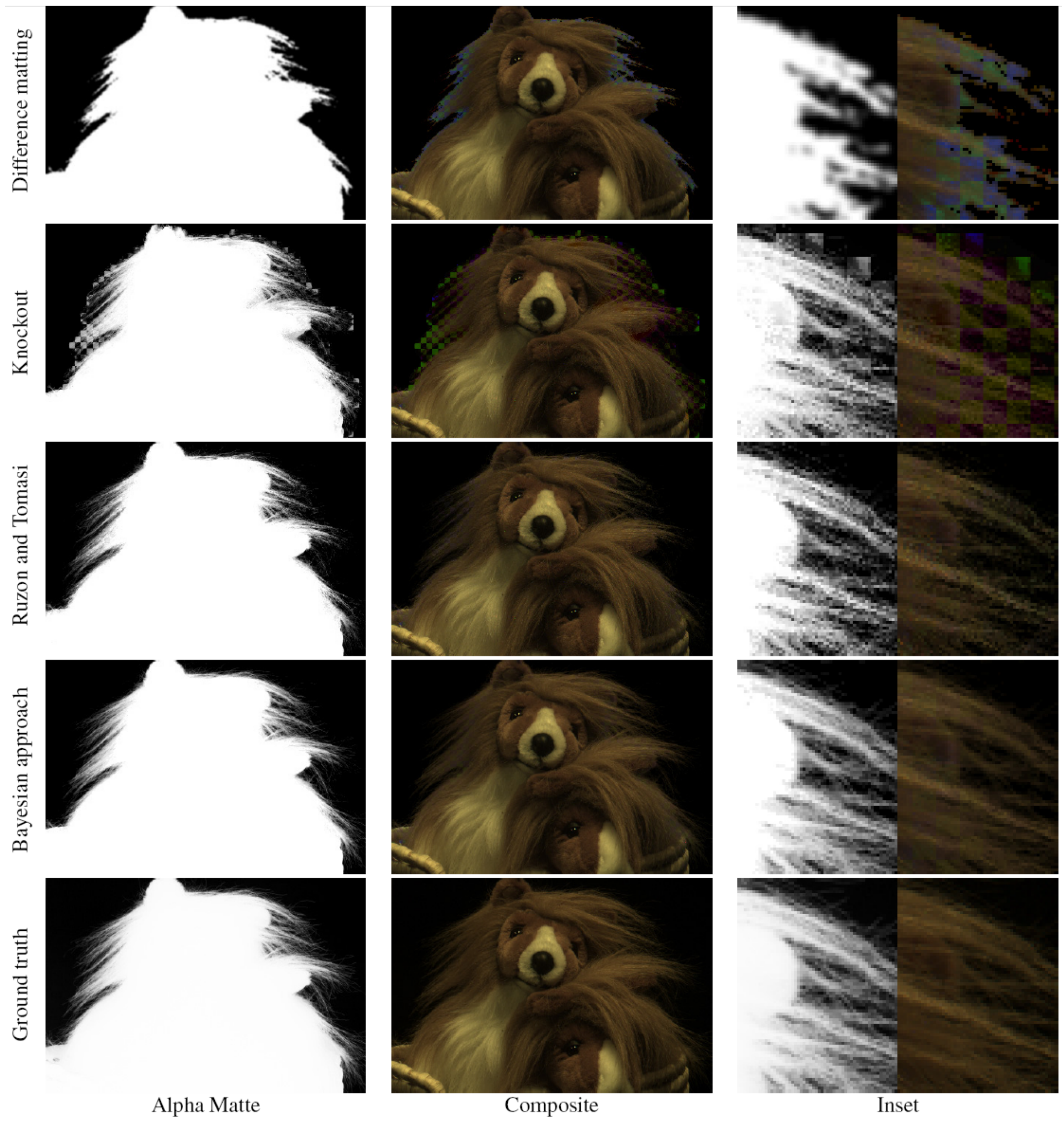


Figure 10.42 Natural image matting results (Chuang, Curless *et al.* 2001) © 2001 IEEE. Difference matting and Knockout both perform poorly on this kind of background, while the newer natural image matting techniques perform well. Chuang *et al.*'s results are slightly smoother and closer to the ground truth.

In their Bayesian matting paper, Chuang, Curless *et al.* (2001) assume a constant (non-informative) distribution for $L(\alpha)$. Follow-on papers assume this distribution to be more peaked around 0 and 1, or sometimes use Markov random fields (MRFs) to define a global correlated prior on $P(\alpha)$ (Wang and Cohen 2009).

To compute the most likely estimates for (F, B, α), the Bayesian matting algorithm alternates between computing (F, B) and α, as each of these problems is quadratic and hence can be solved as a small linear system. When several color clusters are estimated, the most likely pairing of foreground and background color clusters is used.

Bayesian image matting produces results that improve on the original natural image matting algorithm by Ruzon and Tomasi (2000), as can be seen in Figure 10.42. However, compared to later techniques (Wang and Cohen 2009), its performance is not as good for complex backgrounds or inaccurate trimaps (Figure 10.44).

10.4.3 Optimization-based matting

An alternative to estimating each pixel's opacity and foreground color independently is to use global optimization to compute a matte that takes into account correlations between neighboring α values. Two examples of this are border matting in the GrabCut interactive segmentation system (Rother, Kolmogorov, and Blake 2004) and Poisson Matting (Sun, Jia *et al.* 2004).

Border matting first dilates the region around the binary segmentation produced by GrabCut (Section 4.3.2) and then solves for a sub-pixel boundary location Δ and a blur width σ for every point along the boundary (Figure 10.39). Smoothness in these parameters along the boundary is enforced using regularization and the optimization is performed using dynamic programming. While this technique can obtain good results for smooth boundaries, such as a person's face, it has difficulty with fine details, such as hair.

Poisson matting (Sun, Jia *et al.* 2004) assumes a known foreground and background color for each pixel in the trimap (as with Bayesian matting). However, instead of independently estimating each α value, it assumes that the gradient of the alpha matte and the gradient of the color image are related by

$$\nabla \alpha = \frac{F - B}{\|F - B\|^2} \cdot \nabla C, \tag{10.35}$$

which can be derived by taking gradients of both sides of (10.29) and assuming that the foreground and background vary slowly. The per-pixel gradient estimates are then integrated into a continuous $\alpha(\mathbf{x})$ field using the regularization (least squares) technique first described in Section 4.2 (4.24) and subsequently used in Poisson blending (Section 8.4.4, Equation (8.75)) and gradient-based dynamic range compression mapping (Section 10.2.1, Equation (10.18)). This technique works well when good foreground and background color estimates are available and these colors vary slowly.

Instead of computing per-pixel foreground and background colors, Levin, Lischinski, and Weiss (2008) assume only that these color distributions can locally be well approximated as mixtures of two colors, which is known as the *color line model* (Figure 10.43a–c). Under this assumption, a closed-form estimate for α at each pixel i in a (say, 3×3) window W_k is given by

$$\alpha_i = \mathbf{a}_k \cdot (\mathbf{C}_i - \mathbf{B}_0) = \mathbf{a}_k \cdot \mathbf{C} + b_k, \tag{10.36}$$

where $\mathbf{C}_i$ is the pixel color treated as a three-vector, $\mathbf{B}_0$ is any pixel along the background color line, and $\mathbf{a}_k$ is the vector joining the two closest points on the foreground and background color lines, as shown in Figure 10.43c. (Note that the geometric derivation shown in this figure is an alternative

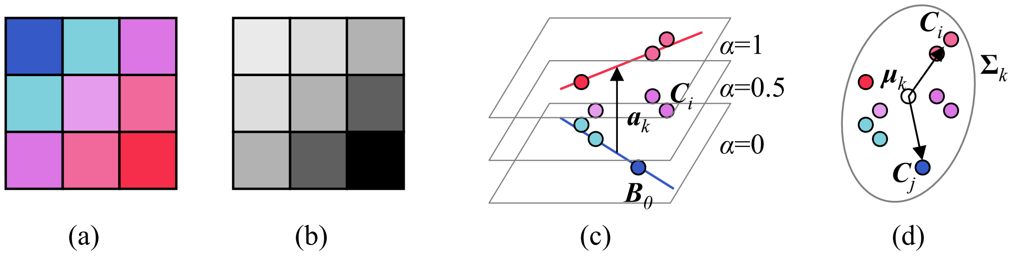


Figure 10.43 Color line matting (Levin, Lischinski, and Weiss 2008): (a) local 3 × 3 patch of colors; (b) potential assignment of α values; (c) foreground and background color lines, the vector $\mathbf{a}_k$ joining their closest points of intersection, and the family of parallel planes of constant α values, $\alpha_i = \mathbf{a}_k \cdot (\mathbf{C}_i - \mathbf{B}_0)$; (d) a scatter plot of sample colors and the deviations from the mean μ_k for two sample colors $\mathbf{C}_i$ and $\mathbf{C}_j$.

to the algebraic derivation presented by Levin, Lischinski, and Weiss (2008).) Minimizing the deviations of the alpha values α_i from their respective color line models (10.36) over all overlapping windows W_k in the image gives rise to the cost

$$E_\alpha = \sum_k \left(\sum_{i \in W_k} (\alpha_i - \mathbf{a}_k \cdot \mathbf{C}_i - b_k)^2 + \epsilon \|\mathbf{a}_k\| \right), \tag{10.37}$$

where the ϵ term is used to regularize the value of $\mathbf{a}_k$ in the case where the two color distributions overlap (i.e., in constant α regions).

Because this formula is quadratic in the unknowns $\{(\mathbf{a}_k, b_k)\}$, they can be eliminated inside each window W_k, leading to a final energy

$$E_\alpha = \alpha^T \mathbf{L} \alpha, \tag{10.38}$$

where the entries in the $\mathbf{L}$ matrix are given by

$$L_{ij} = \sum_{k: i \in W_k \wedge j \in W_k} \left(\delta_{ij} - \frac{1}{M} \left(1 + (\mathbf{C}_i - \mu_k)^T \hat{\mathbf{\Sigma}}_k^{-1} (\mathbf{C}_j - \mu_k) \right) \right), \tag{10.39}$$

where $M = |W_k|$ is the number of pixels in each (overlapping) window, μ_k is the mean color of the pixels in window W_k, and $\hat{\mathbf{\Sigma}}_k$ is the 3 × 3 covariance of the pixel colors plus $\epsilon/M\mathbf{I}$.

Figure 10.43d shows the intuition behind the entries in this affinity matrix, which is called the *matting Laplacian*. Note how when two pixels $\mathbf{C}_i$ and $\mathbf{C}_j$ in W_k point in opposite directions away from the mean μ_k, their weighted dot product is close to -1, and so their affinity becomes close to 0. Pixels close to each other in color space (and hence with similar expected α values) will have affinities close to $-2/M$.

Minimizing the quadratic energy (10.38) constrained by the known values of $\alpha = \{0, 1\}$ at scribbles only requires the solution of a sparse set of linear equations, which is why the authors call their technique a *closed-form solution* to natural image matting. Once α has been computed, the foreground and background colors are estimated using a least squares minimization of the compositing equation (10.29) regularized with a spatially varying first-order smoothness,

$$E = \sum_i \|C_i - [\alpha + F_i + (1 - \alpha_i)B_i]\|^2 + \lambda |\nabla \alpha_i| (\|\nabla F_i\|^2 + \|\nabla B_i\|^2), \tag{10.40}$$

where the $|\nabla \alpha_i|$ weight is applied separately for the x and y components of the F and B derivatives (Levin, Lischinski, and Weiss 2008).

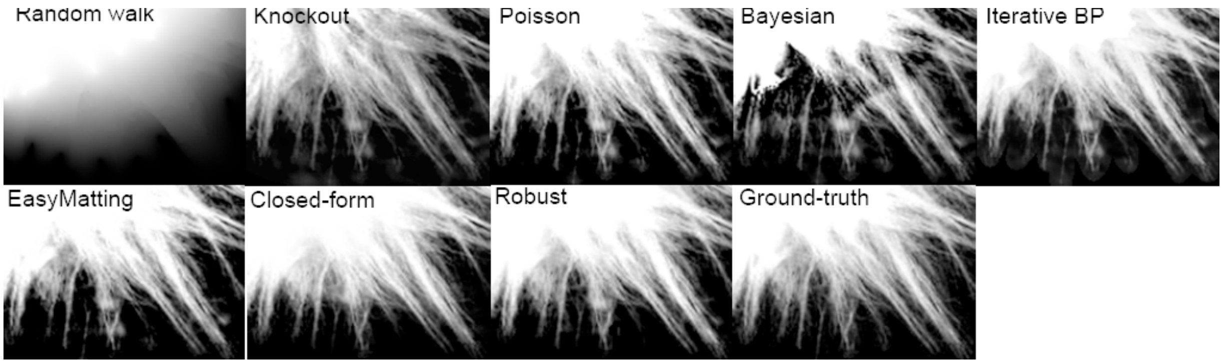


Figure 10.44 Comparative matting results for a medium accuracy trimap. Wang and Cohen (2009) describe the individual techniques being compared.

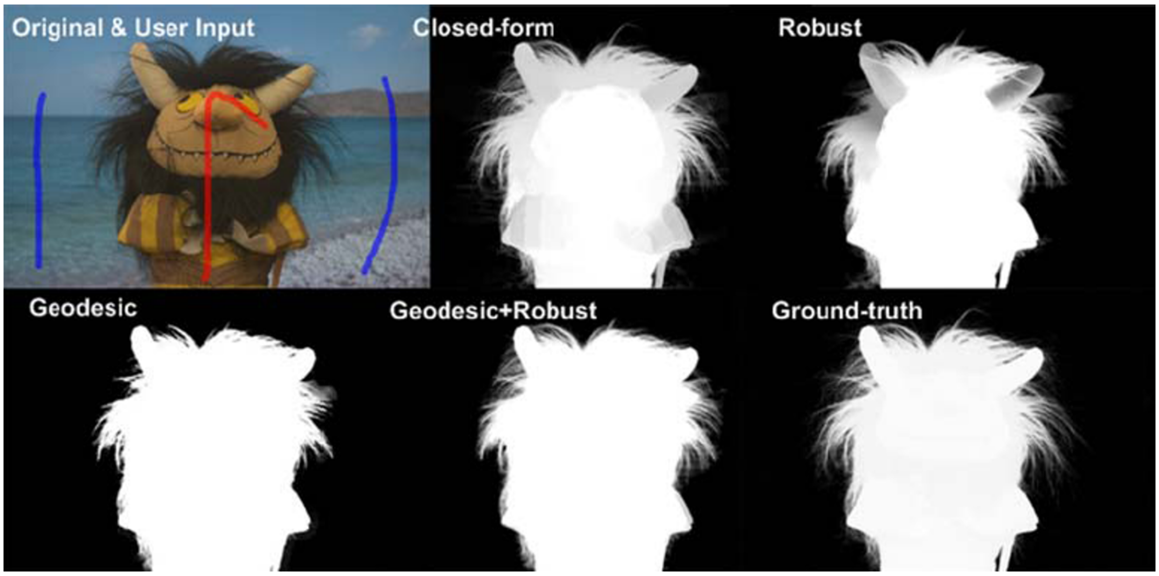


Figure 10.45 Comparative matting results with scribble-based inputs. Wang and Cohen (2009) describe the individual techniques being compared.

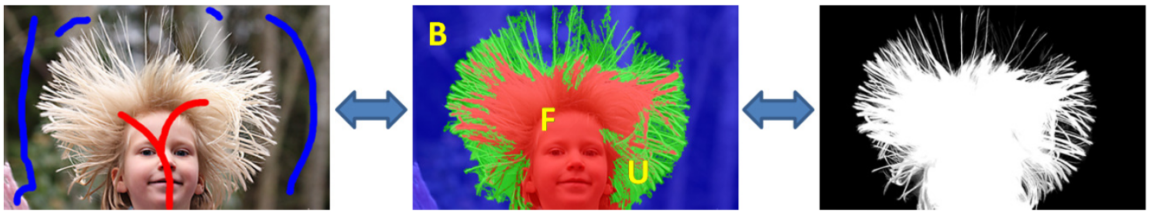


Figure 10.46 Stroke-based segmentation result (Rhemann, Rother *et al.* 2008) © 2008 IEEE.

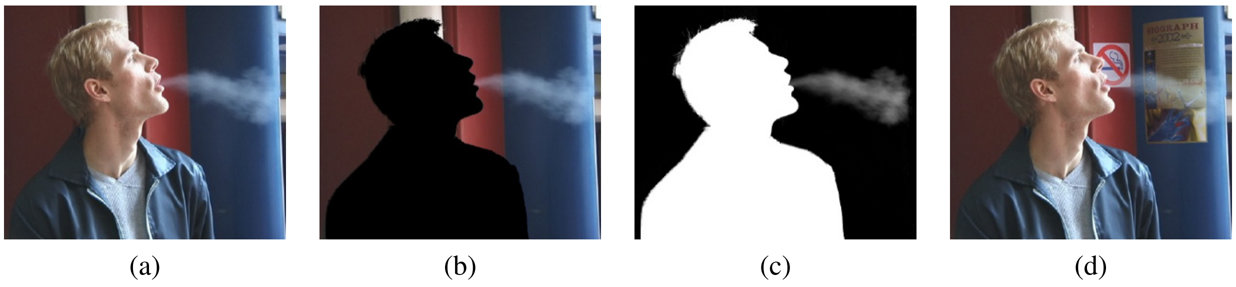


Figure 10.47 Smoke matting (Chuang, Agarwala *et al.* 2002) © 2002 ACM: (a) input video frame; (b) after removing the foreground object; (c) estimated alpha matte; (d) insertion of new objects into the background.

Laplacian (closed-form) matting is just one of many optimization-based techniques surveyed and compared by Wang and Cohen (2009). Some of these techniques use alternative formulations for the affinities or smoothness terms on the α matte, alternative estimation techniques such as belief propagation, or alternative representations (e.g., local histograms) for modeling local foreground and background color distributions (Wang and Cohen 2005, 2007a,b). Some of these techniques also provide real-time results as the user draws a contour line or sparse set of scribbles (Wang, Agrawala, and Cohen 2007; Rhemann, Rother *et al.* 2008) or even pre-segment the image into a small number of mattes that the user can select with simple clicks (Levin, Acha, and Lischinski 2008).

Figure 10.44 shows the results of running a number of the surveyed algorithms on a region of toy animal fur where a trimap has been specified, while Figure 10.45 shows results for techniques that can produce mattes with only a few scribbles as input. Figure 10.46 shows a result for an even more recent algorithm (Rhemann, Rother *et al.* 2008) that claims to outperform all of the techniques surveyed by Wang and Cohen (2009).

The latest results on natural image matting can be found on the http://alphamatting.com website created by Rhemann, Rother *et al.* (2009). It currently lists over 60 different algorithms, with most of the more recent algorithms using deep neural networks. The Deep Image Matting paper by Xu, Price *et al.* (2017) provides a larger database of 49,300 training images and 1,000 test images constructed by overlaying manually created color foreground mattes over a variety of backgrounds.[27]

Pasting. Once a matte has been pulled from an image, it is usually composited directly over the new background, unless the seams between the cutout and background regions are to be hidden, in which case Poisson blending (Pérez, Gangnet, and Blake 2003) can be used (Section 8.4.4).

In the latter case, it is helpful if the matte boundary passes through regions that either have little texture or look similar in the old and new images. Papers by Jia, Sun *et al.* (2006) and Wang and Cohen (2007b) explain how to do this.

10.4.4 Smoke, shadow, and flash matting

In addition to matting out solid objects with fractional boundaries, it is also possible to matte out translucent media such as smoke (Chuang, Agarwala *et al.* 2002). Starting with a video sequence, each pixel is modeled as a linear combination of its (unknown) background color and a constant foreground (smoke) color that is common to all pixels. Voting in color space is used to estimate this

[27]https://sites.google.com/view/deepimagematting

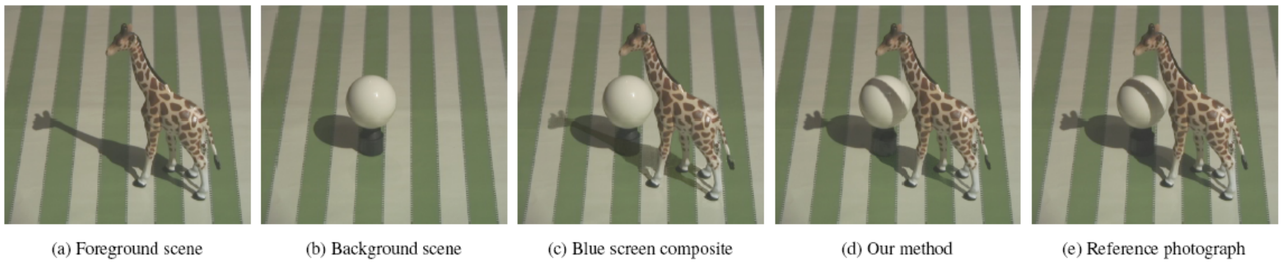


Figure 10.48 Shadow matting (Chuang, Goldman *et al.* 2003) © 2003 ACM. Instead of simply darkening the new scene with the shadow (c), shadow matting correctly dims the lit scene with the new shadow and drapes the shadow over 3D geometry (d).

foreground color and the distance along each color line is used to estimate the per-pixel temporally varying alpha (Figure 10.47).

Extracting and re-inserting shadows is also possible using a related technique (Chuang, Goldman *et al.* 2003; Wang, Curless, and Seitz 2020). Here, instead of assuming a constant foreground color, each pixel is assumed to vary between its fully lit and fully shadowed colors, which can be estimated by taking (robust) minimum and maximum values over time as a shadow passes over the scene (Exercise 10.9). The resulting fractional *shadow matte* can be used to re-project the shadow into a new scene. If the destination scene has a non-planar geometry, it can be scanned by waving a straight stick shadow across the scene. The new shadow matte can then be warped with the computed deformation field to have it drape correctly over the new scene (Figure 10.48). Shadows can also be extracted from video streams by extending video object segmentation algorithms (Section 9.4.3) to include shadows and other effects such as smoke (Lu, Cole *et al.* 2021). An example of useful shadow manipulation in photographs is the removal or softening of harsh shadows in people's portraits (Sun, Barron *et al.* 2019; Zhou, Hadap *et al.* 2019; Zhang, Barron *et al.* 2020), which is available as the Portrait Light feature in Google Photos.[28]

The quality and reliability of matting algorithms can also be enhanced using more sophisticated acquisition systems. For example, taking a flash and non-flash image pair supports the reliable extraction of foreground mattes, which show up as regions of large illumination change between the two images (Sun, Li *et al.* 2006). Taking simultaneous video streams focused at different distances (McGuire, Matusik *et al.* 2005) or using multi-camera arrays (Joshi, Matusik, and Avidan 2006) are also good approaches to producing high-quality mattes. These techniques are described in more detail in (Wang and Cohen 2009).

Lastly, photographing a refractive object in front of a number of patterned backgrounds allows the object to be placed in novel 3D environments. These environment matting techniques (Zongker, Werner *et al.* 1999; Chuang, Zongker *et al.* 2000) are discussed in Section 14.4.

10.4.5 Video matting

While regular single-frame matting techniques such as blue or green screen matting (Smith and Blinn 1996; Wright 2006; Brinkmann 2008) can be applied to video sequences, the presence of moving objects can sometimes make the matting process easier, as portions of the background may get revealed in preceding or subsequent frames.

[28] https://blog.google/products/photos/new-helpful-editor

Chuang, Agarwala *et al.* (2002) describe a nice approach to this *video matting* problem, where foreground objects are first removed using a conservative *garbage matte* and the resulting *background plates* are aligned and composited to yield a high-quality background estimate. They also describe how trimaps drawn at sparse keyframes can be interpolated to in-between frames using bi-direction optical flow. Alternative approaches to video matting, such as rotoscoping, which involves drawing curves or strokes in video sequence keyframes (Agarwala, Hertzmann *et al.* 2004; Wang, Bhat *et al.* 2005), are discussed in the matting survey paper by Wang and Cohen (2009). There is also a newer dataset of carefully matted stop-motion animation videos created by Erofeev, Gitman *et al.* (2015).[29]

Since the original development of video matting techniques, improved algorithms have been developed for both interactive and fully automated *video object segmentation*, as discussed in Section 9.4.3. The paper by Sengupta, Jayaram *et al.* (2020) uses deep learning and adversarial loss, as well as a motion prior, to provide high-quality mattes from small-motion handheld videos where a *clean plate* of the background has also been captured. Wang, Curless, and Seitz (2020) describe a system where shadows and occlusions can be determined by observing people walking around a scene, enabling the insertion of new people at correct scales and lighting. In follow-up work Lin, Ryabtsev *et al.* (2021) describe a high-resolution real-time video matting system along with two new video and image matting datasets. Finally, Lu, Cole *et al.* (2021) describe how to extract shadows, reflections, and other effects associated with objects being tracked and segmented in videos.

10.5 Texture analysis and synthesis

While texture analysis and synthesis may not at first seem like computational photography techniques, they are, in fact, widely used to repair defects, such as small holes, in images or to create non-photorealistic painterly renderings from regular photographs.

The problem of texture synthesis can be formulated as follows: given a small sample of a "texture" (Figure 10.49a), generate a larger similar-looking image (Figure 10.49b). As you can imagine, for certain sample textures, this problem can be quite challenging.

Traditional approaches to texture analysis and synthesis try to match the spectrum of the source image while generating shaped noise. Matching the frequency characteristics, which is equivalent to matching spatial correlations, is in itself not sufficient. The distributions of the responses at different frequencies must also match. Heeger and Bergen (1995) develop an algorithm that alternates between matching the histograms of multi-scale (steerable pyramid) responses and matching the final image histogram. Portilla and Simoncelli (2000) improve on this technique by also matching pairwise statistics across scale and orientations. De Bonet (1997) uses a coarse-to-fine strategy to find locations in the source texture with a similar *parent structure*, i.e., similar multi-scale oriented filter responses, and then randomly chooses one of these matching locations as the current sample value. Gatys, Ecker, and Bethge (2015) also use a pyramidal fine-to-coarse-to-fine algorithm, but using deep networks trained for object recognition. At each level in the deep network, they gather correlation statistics between various features. During generation, they iteratively update the random image until these more perceptually motivated statistic (Zhang, Isola *et al.* 2018) are matched. We give more details on this and other neural approaches to texture synthesis, such as Shaham, Dekel, and Michaeli (2019), in Section 10.5.3 on neural style transfer.

Exemplar-based texture synthesis algorithms sequentially generate texture pixels by looking for neighborhoods in the source texture that are similar to the currently synthesized image (Efros and Leung 1999). Consider the (as yet) unknown pixel $\mathbf{p}$ in the partially constructed texture on the left

[29]https://videomatting.com

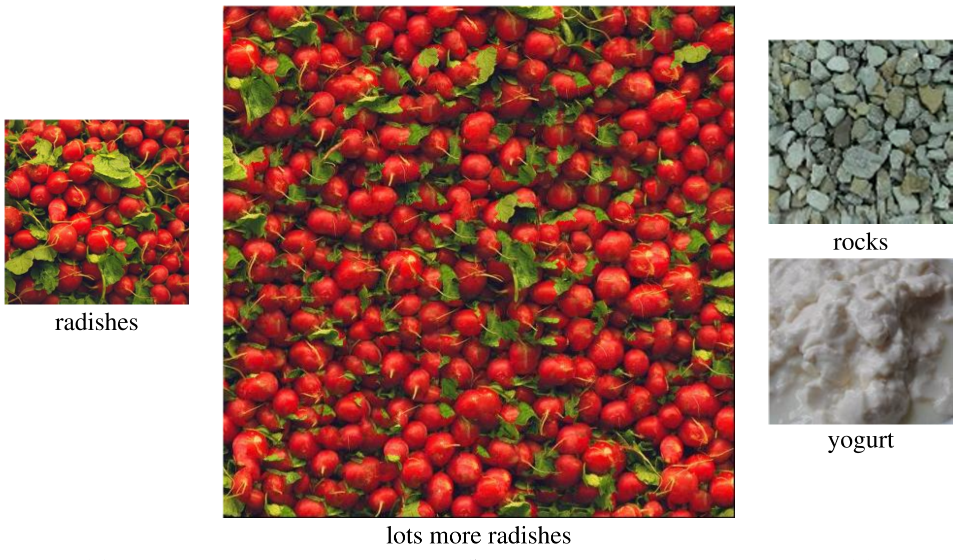


Figure 10.49 Texture synthesis: (a) given a small patch of texture, the task is to synthesize (b) a similar-looking larger patch; (c) other semi-structured textures that are challenging to synthesize. (Images courtesy of Alyosha Efros.)

side of Figure 10.50. As some of its neighboring pixels have been already been synthesized, we can look for similar partial neighborhoods in the sample texture image on the right and randomly select one of these as the new value of $\mathbf{p}$. This process can be repeated down the new image either in a raster fashion or by scanning around the periphery ("onion peeling") when filling holes, as discussed in (Section 10.5.1). In their actual implementation, Efros and Leung (1999) find the most similar neighborhood and then include all other neighborhoods within a $d = (1 + \epsilon)$ distance, with $\epsilon = 0.1$. They also optionally weight the random pixel selections by the similarity metric d.

To accelerate this process and improve its visual quality, Wei and Levoy (2000) extend this technique using a coarse-to-fine generation process, where coarser levels of the pyramid, which have already been synthesized, are also considered during the matching (De Bonet 1997). To accelerate the nearest neighbor finding, tree-structured vector quantization is used. A much faster version of such nearest neighbor search is the widely used randomized PatchMatch iterative update algorithm developed by Barnes, Shechtman *et al.* (2009).

Efros and Freeman (2001) propose an alternative acceleration and visual quality improvement technique. Instead of synthesizing a single pixel at a time, overlapping square blocks are selected using similarity with previously synthesized regions (Figure 10.51). Once the appropriate blocks have been selected, the seam between newly overlapping blocks is determined using dynamic programming. (Full graph cut seam selection is not required, because only one seam location per row is needed for a vertical boundary.) Because this process involves selecting small patches and them stitching them together, Efros and Freeman (2001) call their system *image quilting*. Komodakis and Tziritas (2007) present an MRF-based version of this block synthesis algorithm that uses a new, efficient version of loopy belief propagation they call "Priority-BP". Wei, Lefebvre *et al.* (2009) present a comprehensive survey of work in exemplar-based texture synthesis through 2009.

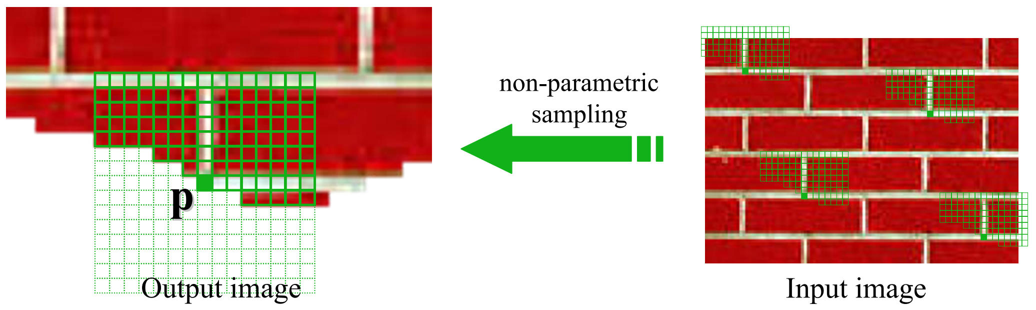


Figure 10.50 Texture synthesis using non-parametric sampling (Efros and Leung 1999). The value of the newest pixel **p** is randomly chosen from similar local (partial) patches in the source texture (input image). (Figure courtesy of Alyosha Efros.)

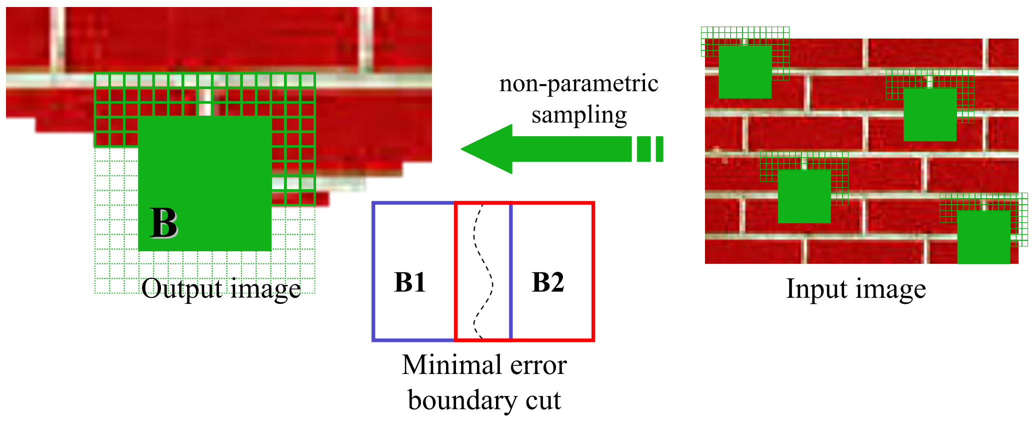
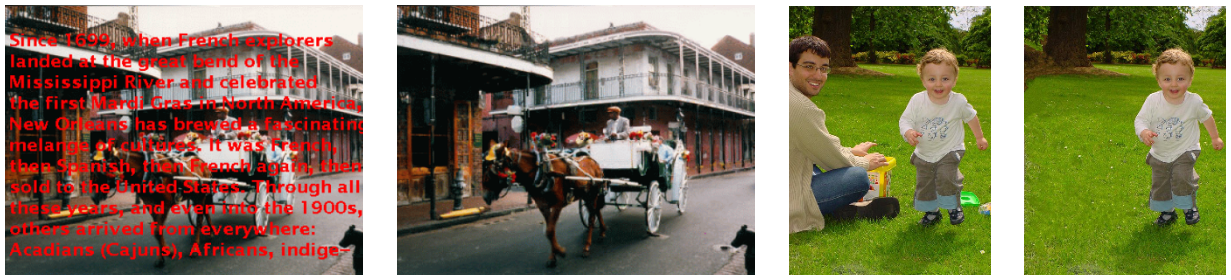


Figure 10.51 Texture synthesis by image quilting (Efros and Freeman 2001). Instead of generating a single pixel at a time, larger blocks are copied from the source texture. The transitions in the overlap regions between the selected blocks are then optimized using dynamic programming. (Figure courtesy of Alyosha Efros.)

10.5.1 *Application*: Hole filling and inpainting

Filling holes left behind when objects or defects are excised from photographs, which is known as *inpainting*, is one of the most common applications of texture synthesis. Such techniques are used not only to remove unwanted people or interlopers from photographs (King 1997) but also to fix small defects in old photos and movies (*scratch removal*) or to remove wires holding props or actors in mid-air during filming (*wire removal*). Bertalmio, Sapiro *et al.* (2000) solve the problem by propagating pixel values along isophote (constant-value) directions interleaved with some anisotropic diffusion steps (Figure 10.52a–b). Telea (2004) develops a faster technique that uses the fast marching method from level sets (Section 7.3.2). However, these techniques will not hallucinate texture in the missing regions. Bertalmio, Vese *et al.* (2003) augment their earlier technique by adding synthetic texture to the infilled regions.

The example-based (non-parametric) texture generation techniques discussed in the previous section can also be used by filling the holes from the outside in (the "onion-peel" ordering). However, this approach may fail to propagate strong oriented structures. Criminisi, Pérez, and Toyama (2004) use exemplar-based texture synthesis where the order of synthesis is determined by the strength of the gradient along the region boundary (Figures 10.1d and 10.52c–d). Sun, Yuan *et*


Since 1699, when French explorers landed at the great bend of the Mississippi River and celebrated the first Mardi Gras in North America, New Orleans has brewed a fascinating melange of cultures. It was French, then Spanish, then French again, then sold to the United States. Through all these years, and even into the 1900s, others arrived from everywhere: Acadians (Cajuns), Africans, indige-

(a) (b) (c) (d)

Figure 10.52 Image inpainting (hole filling): (a–b) propagation along isophote directions (Bertalmio, Sapiro *et al.* 2000) © 2000 ACM; (c–d) exemplar-based inpainting with confidence-based filling order (Criminisi, Pérez, and Toyama 2004).

al. (2004) present a related approach where the user draws interactive lines to indicate where structures should be preferentially propagated. Additional techniques related to these approaches include those developed by Drori, Cohen-Or, and Yeshurun (2003), Kwatra, Schödl *et al.* (2003), Kwatra, Essa *et al.* (2005), Wilczkowiak, Brostow *et al.* (2005), Komodakis and Tziritas (2007), and Wexler, Shechtman, and Irani (2007).

Most hole filling algorithms borrow small pieces of the original image to fill in the holes. When a large database of source images is available, e.g., when images are taken from a photo sharing site or the internet, it is sometimes possible to copy a single contiguous image region to fill the hole. Hays and Efros (2007) present such a technique, which uses image context and boundary compatibility to select the source image, which is then blended with the original (holey) image using graph cuts and Poisson blending. This technique is discussed in more detail in Section 6.4.4 and Figure 6.40.

As with other areas of image processing, deep neural networks are used in all of the latest techniques (Yang, Lu *et al.* 2017; Yu, Lin *et al.* 2018; Liu, Reda *et al.* 2018; Zeng, Fu *et al.* 2019; Yu, Lin *et al.* 2019; Chang, Liu *et al.* 2019; Nazeri, Ng *et al.* 2019; Ren, Yu *et al.* 2019; Shih, Su *et al.* 2020; Yi, Tang *et al.* 2020). Some of these papers have introduced interesting new extensions to neural network architectures, such as *partial convolutions* (Liu, Reda *et al.* 2018) and *partial convolutions* (Yu, Lin *et al.* 2019), the propagation of edge structures (Nazeri, Ng *et al.* 2019; Ren, Yu *et al.* 2019), multi-resolution attention and residuals (Yi, Tang *et al.* 2020), and iterative confidence feedback (Zeng, Lin *et al.* 2020). Inpainting has also been applied to video sequences (e.g., Gao, Saraf *et al.* 2020). Results on recent challenges on image inpainting can be found in the AIM 2020 Workshop and Challenges on this topic (Ntavelis, Romero *et al.* 2020a).

10.5.2 *Application*: Non-photorealistic rendering

Two more applications of the exemplar-based texture synthesis ideas are texture transfer (Efros and Freeman 2001) and image analogies (Hertzmann, Jacobs *et al.* 2001), which are both examples of non-photorealistic rendering (Gooch and Gooch 2001).

In addition to using a source texture image, texture transfer also takes a reference (or target) image, and tries to match certain characteristics of the target image with the newly synthesized image. For example, the new image being rendered in Figure 10.53c not only tries to satisfy the usual similarity constraints with the source texture in Figure 10.53b, but it also tries to match the luminance characteristics of the reference image. Efros and Freeman (2001) mention that blurred image intensities or local image orientation angles are alternative quantities that could be matched.

Hertzmann, Jacobs *et al.* (2001) formulate the following problem:

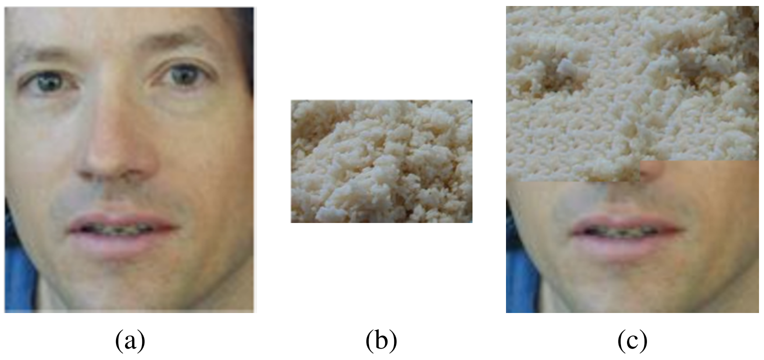


Figure 10.53 Texture transfer (Efros and Freeman 2001) © 2001 ACM: (a) reference (target) image; (b) source texture; (c) image (partially) rendered using the texture.

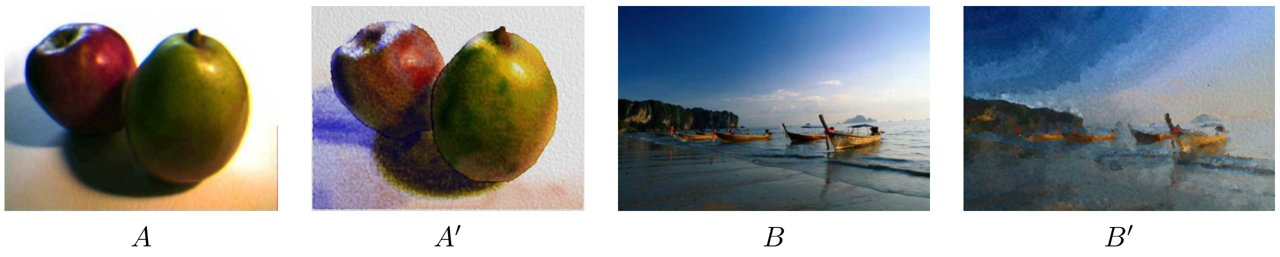


Figure 10.54 Image analogies (Hertzmann, Jacobs *et al.* 2001) © 2001 ACM. Given an example pair of a source image A and its rendered (filtered) version A', generate the rendered version B' from another unfiltered source image B.

> Given a pair of images A and A' (the unfiltered and filtered source images, respectively), along with some additional unfiltered target image B, synthesize a new filtered target image B' such that
> $$A : A' :: B : B'.$$

Instead of having the user program a certain non-photorealistic rendering effect, it is sufficient to supply the system with examples of before and after images, and let the system synthesize the novel image using exemplar-based synthesis, as shown in Figure 10.54.

The algorithm used to solve image analogies proceeds in a manner analogous to the texture synthesis algorithms of Efros and Leung (1999) and Wei and Levoy (2000). Once Gaussian pyramids have been computed for all of the source and reference images, the algorithm looks for neighborhoods in the source filtered pyramids generated from A' that are similar to the partially constructed neighborhood in B', while at the same time having similar multi-resolution appearances at corresponding locations in A and B. As with texture transfer, appearance characteristics can include not only (blurred) color or luminance values but also orientations.

This general framework allows image analogies to be applied to a variety of rendering tasks. In addition to exemplar-based non-photorealistic rendering, image analogies can be used for traditional texture synthesis, super-resolution, and texture transfer (using the same textured image for both A and A'). If only the filtered (rendered) image A' is available, as is the case with paintings, the missing reference image A can be hallucinated using a smart (edge preserving) blur operator. Finally, it is possible to train a system to perform *texture-by-numbers* by manually painting over a natural image with pseudocolors corresponding to pixels' semantic meanings, e.g., water, trees, and

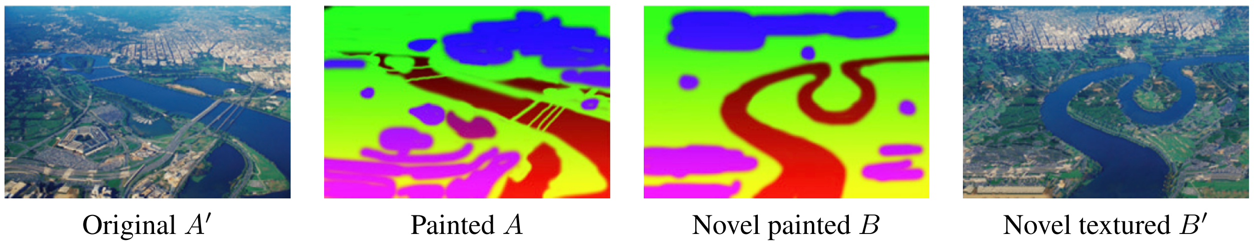


Figure 10.55 Texture-by-numbers (Hertzmann, Jacobs *et al.* 2001) © 2001 ACM. Given a textured image A' and a hand-labeled (painted) version A, synthesize a new image B' given just the painted version B.

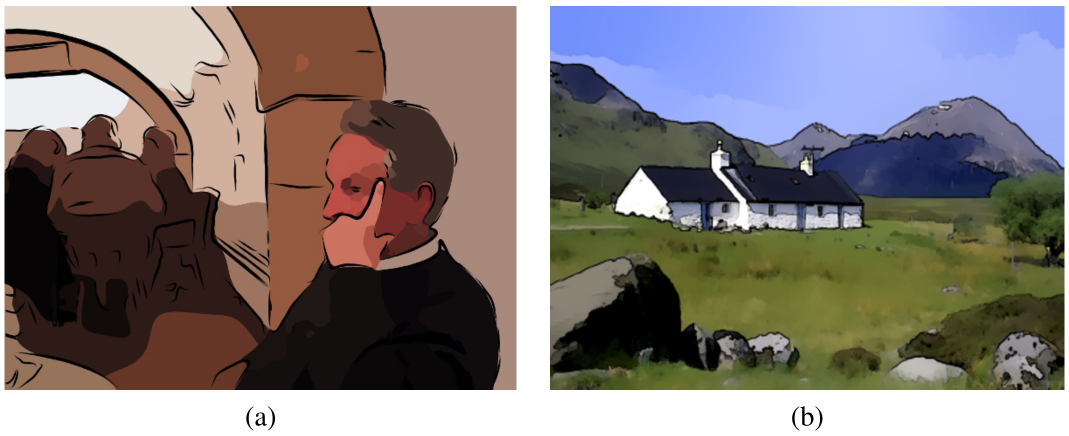


Figure 10.56 Non-photorealistic abstraction of photographs: (a) (DeCarlo and Santella 2002) © 2002 ACM and (b) (Farbman, Fattal *et al.* 2008) © 2008 ACM.

grass (Figure 10.55a–b). The resulting system can then convert a novel sketch into a fully rendered synthetic photograph (Figure 10.55c–d). In more recent work, Cheng, Vishwanathan, and Zhang (2008) add ideas from image quilting (Efros and Freeman 2001) and MRF inference (Komodakis, Tziritas, and Paragios 2008) to the basic image analogies algorithm, while Ramanarayanan and Bala (2007) recast this process as energy minimization, which means it can be viewed as a conditional random field (Section 4.3.1), and devise an efficient algorithm to find a good minimum.

More traditional filtering and feature detection techniques can also be used for non-photorealistic rendering.[30] For example, pen-and-ink illustration (Winkenbach and Salesin 1994) and painterly rendering techniques (Litwinowicz 1997) use local color, intensity, and orientation estimates as an input to their procedural rendering algorithms. Techniques for stylizing and simplifying photographs and video (DeCarlo and Santella 2002; Winnemöller, Olsen, and Gooch 2006; Farbman, Fattal *et al.* 2008), as in Figure 10.56, use combinations of edge-preserving blurring (Section 3.3.1) and edge detection and enhancement (Section 7.2.3).

10.5.3 Neural style transfer and semantic image synthesis

With the advent of deep learning, image-guided exemplar-based texture synthesis has mostly been replaced with statistics matching in deep networks (Gatys, Ecker, and Bethge 2015). Figure 10.57 illustrates the basic idea used in neural style transfer networks. In the original work of Gatys, Ecker,

[30]For a good selection of papers, see the Symposia on Non-Photorealistic Animation and Rendering (NPAR).

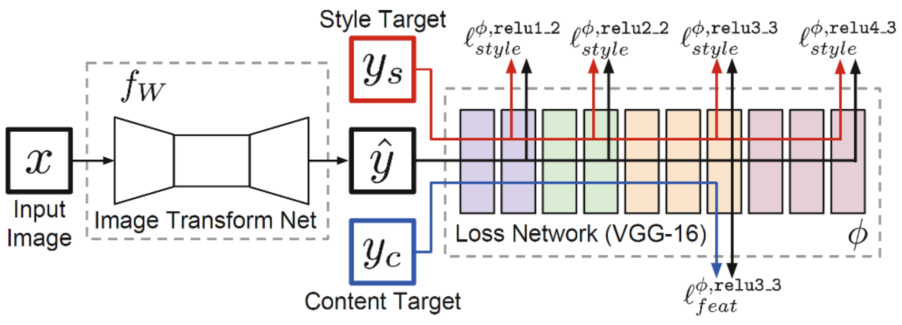

Style Target
y_s
f_W
x
Input Image
Image Transform Net
$\hat{y}$
y_c
Content Target
$\ell_{style}^{\phi,relu1\_2}$
$\ell_{style}^{\phi,relu2\_2}$
$\ell_{style}^{\phi,relu3\_3}$
$\ell_{style}^{\phi,relu4\_3}$
Loss Network (VGG-16)
ϕ
$\ell_{feat}^{\phi,relu3\_3}$

Figure 10.57 Network architecture for neural style transfer, which learns to transform images in one particular style (Johnson, Alahi, and Fei-Fei 2016) © 2016 Springer. During training, the content target image y_c is fed into the image transformation network as an input x, along with a style image y_s, and the network weights are updated so as to minimize the perceptual losses, i.e., the style reconstruction loss l_{style} and the feature reconstruction loss l_{feat}. The earlier network by Gatys, Ecker, and Bethge (2015) did not have an image transformation network, and instead used the losses to optimize the transformed image $\hat{y}$.

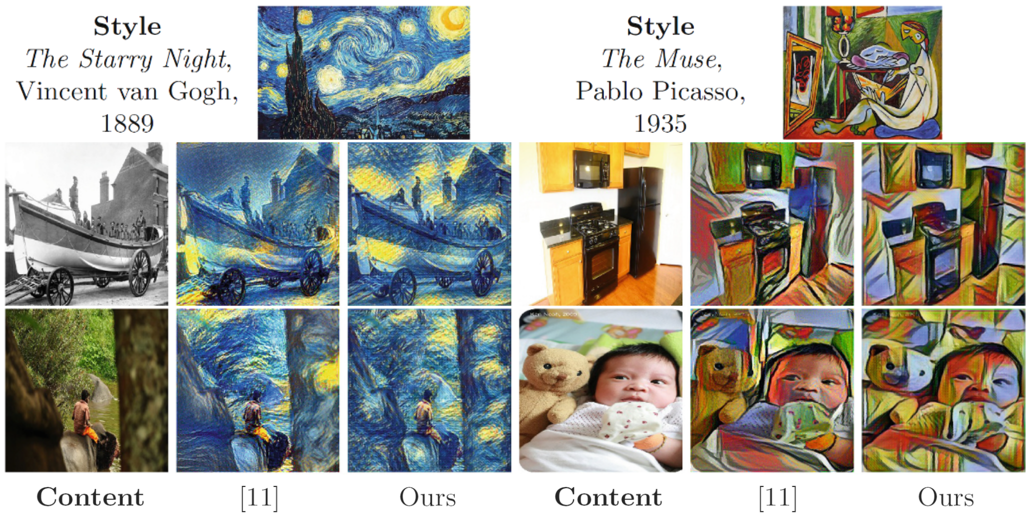
and Bethge (2016), a style image y_s and a content image y_c (see Figure 10.58 for examples) are input to a *loss network*, which compares features derived from the style and target images with those derived from the image $\hat{y}$ being synthesized. These losses are normally a combination of a *perceptual loss*. The gradients of these losses are used to adjust the generated image $\hat{y}$ in an iterative fashion, which makes this process quite slow.

To accelerate this, Johnson, Alahi, and Fei-Fei (2016) train a feedforward *image transformation network* with a fixed style image and many different content targets, adjusting the network weights so that the stylized image $\hat{y}$ resulting from a target y_c matches the desired statistics. When a new image x is presented to be stylized, it is simply run through the image transformation network. Figure 10.58a shows some comparisons between Gatys, Ecker, and Bethge (2016) and Johnson, Alahi, and Fei-Fei (2016).

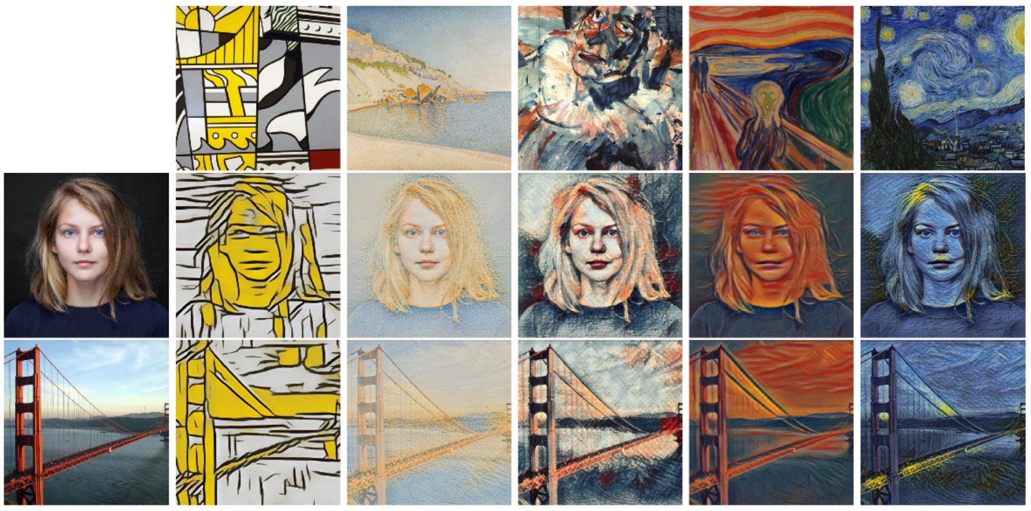
Perceptual loss has now become a standard component of image synthesis systems (Dosovitskiy and Brox 2016), often as an additional component to the generative adversarial loss (Section 5.5.4). They are also sometimes used as an alternative to older image quality metrics such as SSIM (Zhang, Isola *et al.* 2018; Talebi and Milanfar 2018; Tariq, Tursun *et al.* 2020; Czolbe, Krause *et al.* 2020).

The basic architecture in Johnson, Alahi, and Fei-Fei (2016) was extended by Ulyanov, Vedaldi, and Lempitsky (2017), who show that using *instance normalization* instead of batch normalization significantly improves the results. Dumoulin, Shlens, and Kudlur (2017) and Huang and Belongie (2017) further extended these ideas to train one network to mimic different styles, using *conditional instance normalization* and *adaptive instance normalization* to select among the pre-trained styles (or in-between blends), as shown in Figure 10.58b.

Neural style transfer continues to be an actively studied area, with related approaches working on more generalized *image-to-image translation* (Isola, Zhu *et al.* 2017) and *semantic photo synthesis* (Chen and Koltun 2017; Park, Liu *et al.* 2019; Bau, Strobelt *et al.* 2019; Ntavelis, Romero *et al.* 2020b) applications—see Tewari, Fried *et al.* (2020, Section 6.1) for a recent survey. Most of the newer architectures use generative adversarial networks (GANs) (Kotovenko, Sanakoyeu *et al.* 2019; Shaham, Dekel, and Michaeli 2019; Yang, Wang *et al.* 2019; Svoboda, Anoosheh *et al.* 2020; Wang, Li *et al.* 2020; Xia, Zhang *et al.* 2020; Härkönen, Hertzmann *et al.* 2020), which we discussed in Section 5.5.4. There's also a recent course on the more general topic of learning-based image synthesis (Zhu 2021).



(a)



(b)

Figure 10.58 Two examples of neural style transfer: (a) the pre-trained network of Johnson, Alahi, and Fei-Fei (2016) © 2016 Springer (labeled "Ours") vs. (Gatys, Ecker, and Bethge 2016) (labeled "[11]");, (b) a network that uses conditional instance normalization to mimic different styles (top row) applied to various content (left column) © (Dumoulin, Shlens, and Kudlur 2017).

10.6 Additional reading

Good overviews of the first decade of computational photography can be found in the book by Raskar and Tumblin (2010) and survey articles by Nayar (2006), Cohen and Szeliski (2006), Levoy (2006), Debevec (2006), and Hayes (2008), as well as two special journal issues edited by Bimber (2006) and Durand and Szeliski (2007). Notes from the courses on computational photography mentioned at the beginning of this chapter are another great source for more recent material and references.[31]

The sub-field of high dynamic range imaging has its own book discussing research in this area (Reinhard, Heidrich *et al.* 2010), as well as some books describing related photographic techniques (Freeman 2008; Gulbins and Gulbins 2009). Algorithms for calibrating the radiometric response function of a camera can be found in articles by Mann and Picard (1995), Debevec and Malik (1997), and Mitsunaga and Nayar (1999).

The subject of tone mapping is treated extensively in (Reinhard, Heidrich *et al.* 2010). Representative papers from the large volume of literature on this topic include (Tumblin and Rushmeier 1993; Larson, Rushmeier, and Piatko 1997; Pattanaik, Ferwerda *et al.* 1998; Tumblin and Turk 1999; Durand and Dorsey 2002; Fattal, Lischinski, and Werman 2002; Reinhard, Stark *et al.* 2002; Lischinski, Farbman *et al.* 2006; Farbman, Fattal *et al.* 2008; Paris, Hasinoff, and Kautz 2011; Aubry, Paris *et al.* 2014).

The literature on super-resolution is quite extensive (Chaudhuri 2001; Park, Park, and Kang 2003; Capel and Zisserman 2003; Capel 2004; van Ouwerkerk 2006). The term super-resolution usually describes techniques for aligning and merging multiple images to produce higher-resolution composites (Keren, Peleg, and Brada 1988; Irani and Peleg 1991; Cheeseman, Kanefsky *et al.* 1993; Mann and Picard 1994; Chiang and Boult 1996; Bascle, Blake, and Zisserman 1996; Capel and Zisserman 1998; Smelyanskiy, Cheeseman *et al.* 2000; Capel and Zisserman 2000; Pickup, Capel *et al.* 2009; Gulbins and Gulbins 2009; Hasinoff, Sharlet *et al.* 2016; Wronski, Garcia-Dorado *et al.* 2019). However, single-image super-resolution techniques have also been developed (Freeman, Jones, and Pasztor 2002; Baker and Kanade 2002; Fattal 2007; Dong, Loy *et al.* 2016; Cai, Gu *et al.* 2019; Anwar, Khan, and Barnes 2020). Such techniques are closely related to denoising (Zhang, Zuo *et al.* 2017; Brown 2019; Liba, Murthy *et al.* 2019; Gu and Timofte 2019), deblurring and blind image deconvolution (Campisi and Egiazarian 2017; Zhang, Dai *et al.* 2019; Kupyn, Martyniuk *et al.* 2019), and demosaicing (Chatterjee, Joshi *et al.* 2011; Gharbi, Chaurasia *et al.* 2016; Abdelhamed, Afifi *et al.* 2020).

A good survey on image matting is given by Wang and Cohen (2009). Representative papers, which include extensive comparisons with previous work, include (Chuang, Curless *et al.* 2001; Wang and Cohen 2007a; Levin, Acha, and Lischinski 2008; Rhemann, Rother *et al.* 2008, 2009; Xu, Price *et al.* 2017). You can find pointers to recent papers and results on the http://alphamatting.com website created by Rhemann, Rother *et al.* (2009). Matting ideas can also be applied to manipulate shadows (Chuang, Goldman *et al.* 2003; Sun, Barron *et al.* 2019; Zhou, Hadap *et al.* 2019; Zhang, Barron *et al.* 2020; Wang, Curless, and Seitz 2020) and videos (Chuang, Agarwala *et al.* 2002; Wang, Bhat *et al.* 2005; Erofeev, Gitman *et al.* 2015; Sengupta, Jayaram *et al.* 2020; Lin, Ryabtsev *et al.* 2021).

The literature on texture synthesis and hole filling includes traditional approaches to texture synthesis, which try to match image statistics between source and destination images (Heeger and

[31]CMU 15-463, http://graphics.cs.cmu.edu/courses/15-463/2008_fall, Berkeley CS194-26/294-26, https://inst.eecs.berkeley.edu/~cs194-26/fa20, MIT 6.815/6.865, https://stellar.mit.edu/S/course/6/sp08/6.815/materials.html, Stanford CS 448A, https://graphics.stanford.edu/courses/cs448a-08-spring, CMU 16-726, https://learning-image-synthesis.github.io, and SIGGRAPH courses, https://web.media.mit.edu/~raskar/photo.

Bergen 1995; De Bonet 1997; Portilla and Simoncelli 2000), as well as approaches that search for matching neighborhoods or patches inside the source sample (Efros and Leung 1999; Wei and Levoy 2000; Efros and Freeman 2001; Wei, Lefebvre *et al.* 2009) or use neural networks (Gatys, Ecker, and Bethge 2015; Shaham, Dekel, and Michaeli 2019). In a similar vein, traditional approaches to hole filling involve the solution of local variational (smooth continuation) problems (Bertalmio, Sapiro *et al.* 2000; Bertalmio, Vese *et al.* 2003; Telea 2004). The next wave of techniques use data-driven texture synthesis approaches (Drori, Cohen-Or, and Yeshurun 2003; Kwatra, Schödl *et al.* 2003; Criminisi, Pérez, and Toyama 2004; Sun, Yuan *et al.* 2004; Kwatra, Essa *et al.* 2005; Wilczkowiak, Brostow *et al.* 2005; Komodakis and Tziritas 2007; Wexler, Shechtman, and Irani 2007). The most recent algorithms for image and video inpainting use deep neural networks (Yang, Lu *et al.* 2017; Yu, Lin *et al.* 2018; Liu, Reda *et al.* 2018; Shih, Su *et al.* 2020; Yi, Tang *et al.* 2020; Gao, Saraf *et al.* 2020; Ntavelis, Romero *et al.* 2020a). In addition to generating isolated patches of texture or inpainting missing region, related techniques can also be used to transfer the style of an image or painting to another one (Efros and Freeman 2001; Hertzmann, Jacobs *et al.* 2001; Gatys, Ecker, and Bethge 2016; Johnson, Alahi, and Fei-Fei 2016; Dumoulin, Shlens, and Kudlur 2017; Huang and Belongie 2017; Shaham, Dekel, and Michaeli 2019).

10.7 Exercises

Ex 10.1: Radiometric calibration. Implement one of the multi-exposure radiometric calibration algorithms described in Section 10.2 (Debevec and Malik 1997; Mitsunaga and Nayar 1999; Reinhard, Heidrich *et al.* 2010). This calibration will be useful in a number of different applications, such as stitching images or stereo matching with different exposures and shape from shading.

1. Take a series of bracketed images with your camera on a tripod. If your camera has an automatic exposure bracketing (AEB) modes, taking three images may be sufficient to calibrate most of your camera's dynamic range, especially if your scene has a lot of bright and dark regions. (Shooting outdoors or through a window on a sunny day is best.)

2. If your images are not taken on a tripod, first perform a global alignment.

3. Estimate the radiometric response function using one of the techniques cited above.

4. Estimate the high dynamic range radiance image by selecting or blending pixels from different exposures (Debevec and Malik 1997; Mitsunaga and Nayar 1999; Eden, Uyttendaele, and Szeliski 2006).

5. Repeat your calibration experiments under different conditions, e.g., indoors under incandescent light, to get a sense for the range of color balancing effects that your camera imposes.

6. If your camera supports RAW and JPEG mode, calibrate both sets of images simultaneously and to each other (the radiance at each pixel will correspond). See if you can come up with a model for what your camera does, e.g., whether it treats color balance as a diagonal or full 3×3 matrix multiply, whether it uses non-linearities in addition to gamma, whether it sharpens the image while "developing" the JPEG image, etc.

7. Develop an interactive viewer to change the exposure of an image based on the average exposure of a region around the mouse. (One variant is to show the adjusted image inside a window around the mouse. Another is to adjust the complete image based on the mouse position.)

8. Implement a tone mapping operator (Exercise 10.5) and use this to map your radiance image to a displayable gamut.

Ex 10.2: Noise level function. Determine your camera's noise level function using either multiple shots or by analyzing smooth regions.

1. Set up your camera on a tripod looking at a calibration target or a static scene with a good variation in input levels and colors. (Check your camera's histogram to ensure that all values are being sampled.)

2. Take repeated images of the same scene (ideally with a remote shutter release) and average them to compute the variance at each pixel. Discarding pixels near high gradients (which are affected by camera motion), plot for each color channel the standard deviation at each pixel as a function of its output value.

3. Fit a lower envelope to these measurements and use this as your noise level function. How much variation do you see in the noise as a function of input level? How much of this is significant, i.e., away from flat regions in your camera response function where you do not want to be sampling anyway?

4. (Optional) Using the same images, develop a technique that segments the image into near-constant regions (Liu, Szeliski *et al.* 2008). (This is easier if you are photographing a calibration chart.) Compute the deviations for each region from a *single* image and use them to estimate the NLF. How does this compare to the multi-image technique, and how stable are your estimates from image to image?

Ex 10.3: Vignetting. Estimate the amount of vignetting in some of your lenses using one of the following three techniques (or devise one of your choosing):

1. Take an image of a large uniform intensity region (well-illuminated wall or blue sky—but be careful of brightness gradients) and fit a radial polynomial curve to estimate the vignetting.

2. Construct a center-weighted panorama and compare these pixel values to the input image values to estimate the vignetting function. Weight pixels in slowly varying regions more highly, as small misalignments will give large errors at high gradients. Optionally estimate the radiometric response function as well (Litvinov and Schechner 2005; Goldman 2010).

3. Analyze the radial gradients (especially in low-gradient regions) and fit the robust means of these gradients to the derivative of the vignetting function, as described by Zheng, Yu *et al.* (2008).

For the parametric form of your vignetting function, you can either use a simple radial function, e.g.,

$$f(r) = 1 + \alpha_1 r + \alpha_2 r^2 + \cdots \tag{10.41}$$

or one of the specialized equations developed by Kang and Weiss (2000) and Zheng, Lin, and Kang (2006).

In all of these cases, be sure that you are using linearized intensity measurements, by using either RAW images or images linearized through a radiometric response function, or at least images where the gamma curve has been removed.

(Optional) What happens if you forget to undo the gamma before fitting a (multiplicative) vignetting function?

Ex 10.4: Optical blur (PSF) estimation. Compute the optical PSF either using a known target (Figure 10.7) or by detecting and fitting step edges (Section 10.1.4) (Joshi, Szeliski, and Kriegman 2008; Cho, Paris *et al.* 2011).

1. Detect strong edges to sub-pixel precision.
2. Fit a local profile to each oriented edge and fill these pixels into an ideal target image, either at image resolution or at a higher resolution (Figure 10.9c–d).
3. Use least squares (10.1) at valid pixels to estimate the PSF kernel K, either globally or in locally overlapping sub-regions of the image.
4. Visualize the recovered PSFs and use them to remove chromatic aberration or deblur the image.

Ex 10.5: Tone mapping. Implement one of the tone mapping algorithms discussed in Section 10.2.1 (Durand and Dorsey 2002; Fattal, Lischinski, and Werman 2002; Reinhard, Stark *et al.* 2002; Lischinski, Farbman *et al.* 2006) or any of the numerous additional algorithms discussed by Reinhard, Heidrich *et al.* (2010) and https://stellar.mit.edu/S/course/6/sp08/6.815/materials.html.

(Optional) Compare your algorithm to local histogram equalization (Section 3.1.4).

Ex 10.6: Flash enhancement. Develop an algorithm to combine flash and non-flash photographs to best effect. You can use ideas from Eisemann and Durand (2004) and Petschnigg, Agrawala *et al.* (2004) or anything else you think might work well.

Ex 10.7: Super-resolution. Implement one or more super-resolution algorithms and compare their performance.

1. Take a set of photographs of the same scene using a hand-held camera (to ensure that there is some jitter between the photographs).
2. Determine the PSF for the images you are trying to super-resolve using one of the techniques in Exercise 10.4.
3. Alternatively, simulate a collection of lower-resolution images by taking a high-quality photograph (avoid those with compression artifacts) and applying your own prefilter kernel and downsampling.
4. Estimate the relative motion between the images using a parametric translation and rotation motion estimation algorithm (Sections 8.1.3 or 9.2).
5. Implement a basic least squares super-resolution algorithm by minimizing the difference between the observed and downsampled images (10.26–10.27).
6. Add in a gradient image prior, either as another least squares term or as a robust term that can be minimized using iteratively reweighted least squares (Appendix A.3).
7. (Optional) Implement one of the example-based super-resolution techniques, where matching against a set of exemplar images is used either to infer higher-frequency information to be added to the reconstruction (Freeman, Jones, and Pasztor 2002) or higher-frequency gradients to be matched in the super-resolved image (Baker and Kanade 2002).
8. (Optional) Use local edge statistic information to improve the quality of the super-resolved image (Fattal 2007).

9. (Optional) Try some of the newest DNN-based super-resolution algorithms.

Ex 10.8: Image matting. Develop an algorithm for pulling a foreground matte from natural images, as described in Section 10.4.

1. Make sure that the images you are taking are linearized (Exercise 10.1 and Section 10.1) and that your camera exposure is fixed (full manual mode), at least when taking multiple shots of the same scene.

2. To acquire ground truth data, place your object in front of a computer monitor and display a variety of solid background colors as well as some natural imagery.

3. Remove your object and re-display the same images to acquire known background colors.

4. Use triangulation matting (Smith and Blinn 1996) to estimate the ground truth opacities α and pre-multiplied foreground colors αF for your objects.

5. Implement one or more of the natural image matting algorithms described in Section 10.4 and compare your results to the ground truth values you computed. Alternatively, use the matting test images published on http://alphamatting.com.

6. (Optional) Run your algorithms on other images taken with the same calibrated camera (or other images you find interesting).

Ex 10.9: Smoke and shadow matting. Extract smoke or shadow mattes from one scene and insert them into another (Chuang, Agarwala *et al.* 2002; Chuang, Goldman *et al.* 2003).

1. Take a still or video sequence of images with and without some intermittent smoke and shadows. (Remember to linearize your images before proceeding with any computations.)

2. For each pixel, fit a line to the observed color values.

3. If performing smoke matting, robustly compute the intersection of these lines to obtain the smoke color estimate. Then, estimate the background color as the other extremum (unless you have already taken a smoke-free background image).

   If performing shadow matting, compute robust shadow (minimum) and lit (maximum) values for each pixel.

4. Extract the smoke or shadow mattes from each frame as the fraction between these two values (background and smoke or shadowed and lit).

5. Scan a new (destination) scene or modify the original background with an image editor.

6. Re-insert the smoke or shadow matte, along with any other foreground objects you may have extracted.

7. (Optional) Using a series of cast stick shadows, estimate the deformation field for the destination scene to correctly warp (drape) the shadows across the new geometry. (This is related to the shadow scanning technique developed by Bouguet and Perona (1999) and implemented in Exercise 13.2.)

8. (Optional) Chuang, Goldman *et al.* (2003) only demonstrated their technique for planar source geometries. Can you extend their technique to capture shadows acquired from an irregular source geometry?

9. (Optional) Can you change the direction of the shadow, i.e., simulate the effect of changing the light source direction?

10. (Optional) Re-implement the facial shadow removal algorithm of Zhang, Barron *et al.* (2020) and try applying it to other domains.

Ex 10.10: Texture synthesis. Implement one of the texture synthesis or hole filling algorithms presented in Section 10.5. Here is one possible procedure:

1. Implement the basic Efros and Leung (1999) algorithm, i.e., starting from the outside (for hole filling) or in raster order (for texture synthesis), search for a similar neighborhood in the source texture image, and copy that pixel.

2. Add in the Wei and Levoy (2000) extension of generating the pixels in a coarse-to-fine fashion, i.e., generate a lower-resolution synthetic texture (or filled image), and use this as a guide for matching regions in the finer resolution version.

3. Add in the Criminisi, Pérez, and Toyama (2004) idea of prioritizing pixels to be filled by some function of the local structure (gradient or orientation strength).

4. Extend any of the above algorithms by selecting sub-blocks in the source texture and using optimization to determine the seam between the new block and the existing image that it overlaps (Efros and Freeman 2001).

5. (Optional) Implement one of the isophote (smooth continuation) inpainting algorithms (Bertalmio, Sapiro *et al.* 2000; Telea 2004).

6. (Optional) Add the ability to supply a target (reference) image (Efros and Freeman 2001) or to provide sample filtered or unfiltered (reference and rendered) images (Hertzmann, Jacobs *et al.* 2001), see Section 10.5.2.

7. (Optional) Try some of the newer DNN-based inpainting algorithms described at the end of Section 10.5.1.

Ex 10.11: Colorization. Implement the Levin, Lischinski, and Weiss (2004) colorization algorithm that is sketched out in Section 4.2.4 and Figure 4.10. If you prefer, you can implement this as a neural network (Zhang, Zhu *et al.* 2017). Find some historic monochrome photographs and some modern color ones. Write an interactive tool that lets you "pick" colors from a modern photo and paint over the old one. Tune the algorithm parameters to give you good results. Are you pleased with the results? Can you think of ways to make them look more "antique", e.g., with softer (less saturated and edgy) colors?

(Alternative) Implement or test out one of the newer "automatic colorization" algorithms such as Zhang, Isola, and Efros (2016) or (Vondrick, Shrivastava *et al.* 2018).

Ex 10.12: Style transfer. Try some of the non-photorealistic rendering or style transfer algorithms from Sections 10.5.2–10.5.3 on your own images. Can you come up with surprising results? How about failure cases?