CHAPTER 7

Application:Denoising Images by Optimization

You are given a noisy image and asked to produce the original. The detailed rules may vary somewhat in different applications. For example, you might have only a rough model of the noise process; you may have a detailed and accurate model of the noise process (though this is unusual); the noise might be deterministic, but depend on parameters you cannot know (this occurs with underwater images); you may need to denoise very few images (or even only one – astronomy applications); you may need to denoise any image, or any image of a particular class; and so on.

This chapter uses a master recipe for denoising. Write $\mathcal{N}$ for a noisy image, and think of denoising as finding a denoised image $\mathcal{D}$ that is (a) close to $\mathcal{N}$ and (b) more like a real image. Write

$$\begin{aligned} C(\mathcal{D}) &= [\text{distance from } \mathcal{D} \text{ to } \mathcal{N}] + [\text{unrealism cost for } \mathcal{D}] \\ &= [\text{data term}] + [\text{penalty term}] \end{aligned}$$

and choose a $\mathcal{D}$ that minimizes this cost function. Methods differ mainly by the penalty term, which has a significant effect on how hard the optimization problem is. This framework leads to very strong denoising methods, at the cost of solving what can be a nasty optimization problem.

Denoising is an important topic for at least two reasons. First, it is extremely useful to be able to improve images in various ways (Section **??**). Second, the way to denoise an image is to exploit information about what images are "like", so studying denoising strategies is a good way to build intuitions about images. For example, gaussian smoothing (Section 4.2.3) or median filtering (Section 4.2.4) to suppress noise works fairly well because most image pixels "look like" their neighbors. But for some noise models, other procedures are a better choice than gaussian smoothing. These procedures require building representations that expose what is important about being an image and depend on deep and interesting qualitative insights on images which will reappear.

7.1 CONSIDERATIONS

7.1.1 Denoising Color Images

Representing color is a broad subject, and Section 28.3 is a brief introduction. The particular color representation one uses can be important in denoising. Section 8.1 showed that gradients in R, G and B tend to appear in the same place. This is the result of two effects: RGB is not a particularly good color representation for many applications, because the representation is highly correlated; and isoluminant changes in color really are rare.

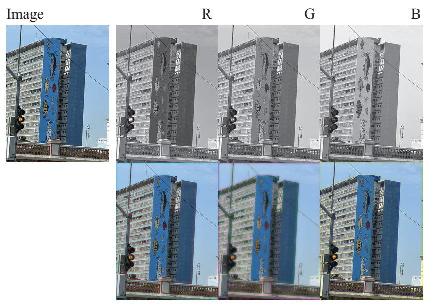

FIGURE 7.1: *RGB color components are heavily correlated, as you can see by looking at images where only one component has been smoothed.. The* **top row** *shows the R, G, and B components of the color image at the* **left**. *The* **bottom row** *shows color images obtained by smoothing one component, then recombining all three. Notice that smoothing any of the R, G, B components alone leads to odd color effects at edges (G is particularly bad).* Image credit: *Figure shows my photograph of a building in downtown Manaus.*

An important consequence is that denoising R, G, and B individually is a bad idea, because a prediction error in one component may result in strange color effects in the denoised images. This effect can become a serious issue for more sophisticated denoising algorithms. The effect is easily observed. Smoothing one of R, G, or B (but not the others) results in strange color effects (**exercises** , Figure 7.1).

Remarkably, one can choose a color space that largely decorrelates any image rather well. LAB is such a color space. It has three attractions: first, it splits images into an intensity component (L) and two color components (A) and (B), which largely do not depend on intensity; second, these components are largely decorrelated spatially; and third, short distances in the color space are rather a good representation of human perception of the size of color changes. Section 28.3.2 offers more detail on the construction of LAB. As Figure 7.1 shows, there is no particular advantage to choosing the 0, 1, 2 images over LAB.

Correlation between RGB components is an experimental fact about the world. Humans are much better at perceiving spatial detail in intensity than they are in color (this has to do with correlation between RGB components, as well as a variety of physiological details about how the human color system works). In an appropriately chosen color space, the color components can be heavily smoothed without affecting human perception of the image all that much (Figure 7.2). For each denoising method, I will apply the following strategy to deal with color images: decompose into LAB; use sophisticated denoising on L; and use a smoothed version of A and B.

7.1.2 Paired Evaluation

An important part of denoising is knowing how well a procedure actually works. For the moment, assume that you have access to a collection of test image pairs,

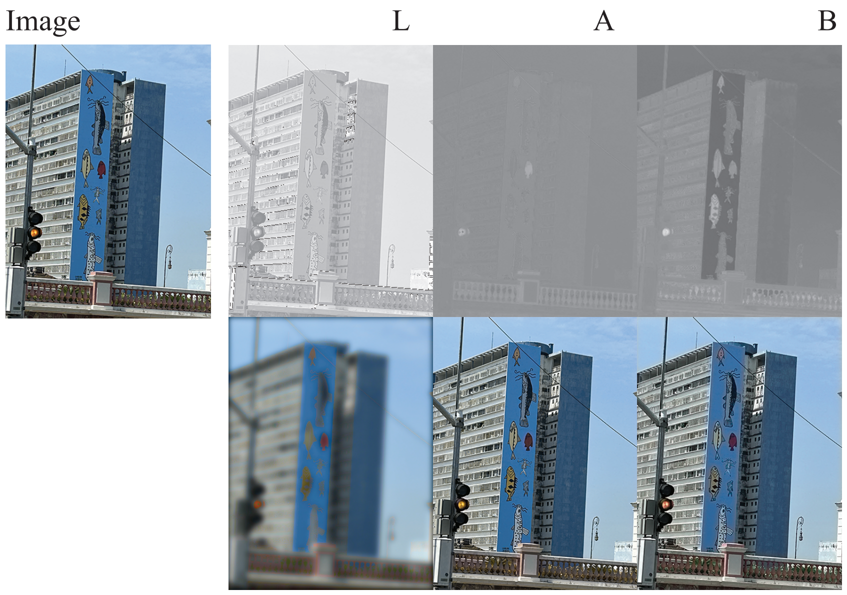

FIGURE 7.2: *Decorrelating the components of a color image before smoothing is important, but one does not need to do this on a per-image basis. The* **top row** *shows the L, A, and B components for this image on the* **right**. *Because these components can be negative, they have been scaled and shifted so that a zero value is mid gray, the largest value is bright and the smallest is dark (the same scale has been applied to each component so you can see relative sizes). The* **bottom row** *shows color images obtained by smoothing one component, then recombining all three. Smoothing L results in a blurry color image; smoothing A or B alone largely has no effect. This means one can use sophisticated methods on the L component and just smooth the A and B components.* Image credit: *Figure shows my photograph of a building in downtown Manaus.*

where you have a noisy version and the clean version.

One standard evaluation statistic is the mean *PSNR* or *peak signal-to-noise ratio*. For each pair $(\mathcal{N}, \mathcal{C})$ of noisy version - clean version, first denoise the noisy image to get $\mathcal{D}$. Now compute the PSNR for the pair $(\mathcal{D}, \mathcal{C})$, using

$$\text{psnr}(\mathcal{D}, \mathcal{C}) = 20 \log \frac{\max_{ij} \mathcal{C}_{ij}}{\sqrt{\sum_{ij} (\mathcal{D}_{ij} - \mathcal{C}_{ij})^2}}$$

and average that PSNR over pairs. The PSNR has some good properties: as $\mathcal{D}$ gets closer to $\mathcal{C}$, the PSNR gets larger; and $\text{psnr}(s\mathcal{D}, s\mathcal{C}) = \text{psnr}(\mathcal{D}, \mathcal{C})$ for $s > 0$ (so you can't change the PSNR by scaling the images). You need to know $\mathcal{C}$ to compute the PSNR, so you can only use PSNR to evaluate if you know the right answer. In some applications, versions of the original image that are uniformly slightly brighter or slightly darker might be acceptable, but the PSNR will penalize a method that can't estimate the brightness of the ground truth image. In these situations, one can use

$$\mathrm{psnr}(\mathcal{D}, \mathcal{C}) = 20 \log \frac{\underset{ij}{\max}\ \mathcal{C}_{ij}}{\underset{s}{\min}\ \sqrt{\sum_{ij} \left(s\mathcal{D}_{ij} - \mathcal{C}_{ij}\right)^2}}.$$

PSNR doesn't really take human perception into account, so that reconstructions with quite good PSNR might look bad to a user, and reconstructions with quite bad PSNR might look good. For example, imagine the reconstruction is the original image, but shifted by one pixel to the left (obtain the missing column by copying its neighbor). The PSNR might be quite bad, but the reconstruction would be good and would look good.

An ideal evaluation metric should not be seriously affected by shifts like this. A natural construction is to compare summary properties of windows of pixels rather than comparing pixels. This construction leads to the *SSIM* or *structural similarity index metric*. The clean image and the denoised image are broken into quite small overlapping windows; summary statistics for these windows are computed and compared, with a metric that is quite robust to changes in intensity; and the comparison is averaged over all windows. Implementations of SSIM appear in most API's.

Human observers have a variety of preferences that SSIM does not fully account for. For example, humans like sharp edges without ringing but can be relaxed about whether the edge is in the right place. As another example, humans are surprisingly good at perceiving lines, and dislike edges that are close to, but not on, a line. The *LPIPS* or *Learned Perceptual Image Patch Similarity* metric is an attempt to deal with this. The clean image and the denoised image are broken into overlapping windows; deep network features are computed for windows; a weighted difference is computed for these features; and the comparison is averaged over all windows. The features are learned using procedures quite like that of Chapters 15 and 16. The reference Implementation of LPIPS is at `https://github.com/richzhang/PerceptualSimilarity`, and many APIs offer LPIPS evaluation.

7.1.3 Blind Image Quality Evaluation

Sometimes it is difficult to find pairs of clean images and denoised versions. For example, you might not have clean versions of the noisy images; if you don't trust simulations of the noise model, you won't have pairs. Evaluation in this case involves telling whether a denoised image is "like an image", using an *image quality metric*. These metrics measure how much something that purports to be a natural image is "like an image." You should suspect that this is extremely hard to do accurately, because, if you could, then denoising might be straightforward – take something that purports to be a natural image, and make small adjustments until it really is a natural image.

Metrics tend to take the following form. Choose some patches in the image;

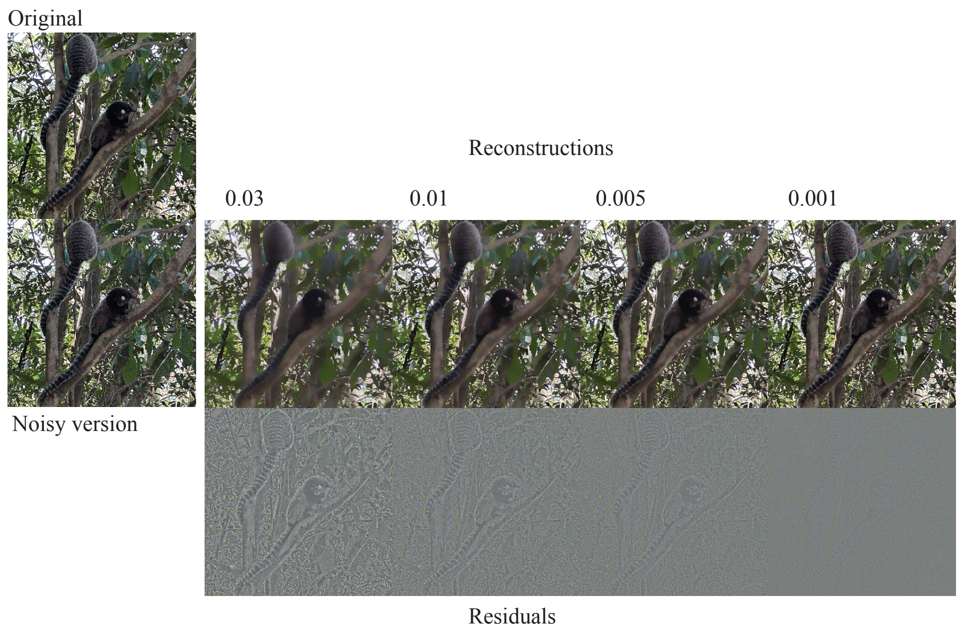

FIGURE 7.3: *A color image (***upper left***), with additive gaussian noise (***lower left***), denoised using weighted least squares (WLS; Section 7.2.1) at various settings of λ. The denoised images are in the* **top row***, with residual (noisy image - reconstructed image) in the* **bottom row***. The residual is shown on a scale where positive values are light, negative values are dark, and 0 is mid-gray. The noisy image was transformed to LAB; the L component was smoothed with WLS; and the A and B components were smoothed with a gaussian kernel. Notice the strong color noise in the residual. The image noise is independent in R, G and B – so colors can change sharply – and the denoiser is very effective at suppressing this effect. Notice how, as λ is increased, the image does not become blurry (compare Figure* **??***). Edges are largely preserved, but detail is lost with increasing λ.* Image credit: *Figure shows my photograph of marmosets in Sao Paulo.*

for these, compute some feature vectors; fit a probability model to these feature vectors; and compare the parameters of the fitted model to the parameters of a model fitted to a large number of natural images. Choosing some patches, rather than all, seems to improve the accuracy of the measure, likely because different images contain different numbers of "boring" patches – patches of, say, constant color – and this might bias the model. This recipe was established by the *NIQE* or *Natural Image Quality Evaluator* metric. Implementations of NIQE appear in most APIs.

7.2 DENOISING BY OPTIMIZATION

For this Chapter, the data term in the master recipe is

$$\sum_{ij} (\mathcal{D}_{ij} - \mathcal{N}_{ij})^2$$

(the ssd of Section 3.4.2). A good reconstruction could smooth the image over quite long scales in regions where $\mathcal{C}$ is constant. The reconstruction must preserve edges, so the smoothing would need to be over very short scales at edge points. Ideally, smoothing would be along an edge rather than across it. But $\mathcal{C}$ isn't known (otherwise there would be nothing to do). All this suggests that the penalty function needs to look at gradients in $\mathcal{D}$.

7.2.1 Weighted Least Squares

The *weighted least squares filter* (almost always WLS) seeks a $\mathcal{D}$ that is (a) close to $\mathcal{N}$ in squared error; (b) mostly has lower gradients than $\mathcal{N}$; and (c) has high gradients when $\mathcal{N}$ does.

Rearrange all images into vectors, so $\mathcal{N}$ is vector $\mathbf{n}$, and so on. This representation gives a convenient expression for the gradient. There are matrices $\mathcal{D}_x$ and $\mathcal{D}_y$ that compute the gradient from the image (rearrange the finite differences of Section 4.2.1, or the derivative of gaussians of Section 5.1.1, **exercises**). The gradient of $\lceil$ can be expressed as

$$\begin{pmatrix} \mathcal{D}_x \\ \mathcal{D}_y \end{pmatrix} \mathbf{d}$$

(where the vector of x-derivatives is stacked on the vector of y-derivatives).

Now write $\mathcal{A}_x(\mathbf{n})$, $\mathcal{A}_y(\mathbf{n})$ for diagonal matrices of weights obtained from the original image. Because these matrices are diagonal, think of them as producing pixel by pixel weights on the cost of a derivative in $\mathcal{D}$. So at a location where the value of $\mathcal{A}_y$ is small, $\mathcal{D}$ could have a large y-derivative, but at locations where the value is large, $\mathcal{D}$ must have a small y-derivative.

The new image $\mathbf{d}$ (a vector version of the denoised image $\mathcal{D}$) should then solve

$$\underset{\mathbf{u}}{\text{argmin}} \quad [\mathbf{u}-\mathbf{n}]^T[\mathbf{u}-\mathbf{n}] + \lambda \mathbf{u}^T \left[\mathcal{D}_x^T \mathcal{A}_x^T \mathcal{A}_x \mathcal{D}_x + \mathcal{D}_y^T \mathcal{A}_y^T \mathcal{A}_y \mathcal{D}_y\right] \mathbf{u}$$

where the first term pushes $\mathbf{d}$ to be like $\mathbf{n}$, the second term controls the derivatives of $\mathbf{d}$ and λ is some weight balancing the two terms. Write $\mathcal{L} = \left[\mathcal{D}_x^T \mathcal{A}_x^T \mathcal{A}_x \mathcal{D}_x + \mathcal{D}_y^T \mathcal{A}_y^T \mathcal{A}_y \mathcal{D}_y\right]$; then solving this problem is a matter of solving

$$\mathcal{F}(\lambda)\mathbf{d} = (\mathcal{I} + \lambda\mathcal{L})\mathbf{d} = \mathbf{n}$$

which can be done in a variety of ways (**exercises**). The key question here is the choice of $\mathcal{A}_x$ and $\mathcal{A}_y$. A large derivative in $\mathcal{D}$ should only occur when the derivative of $\mathcal{N}$ is large and reliable. Small derivatives $\mathcal{N}$ are untrustworthy, and should not appear in $\mathcal{D}$. So at pixels where $\mathbf{n}$ has a small x derivative, the relevant term in $\mathcal{A}_x$ should be big. Similarly, when $\mathbf{n}$ has a large x derivative, the relevant term in $\mathcal{A}_x$

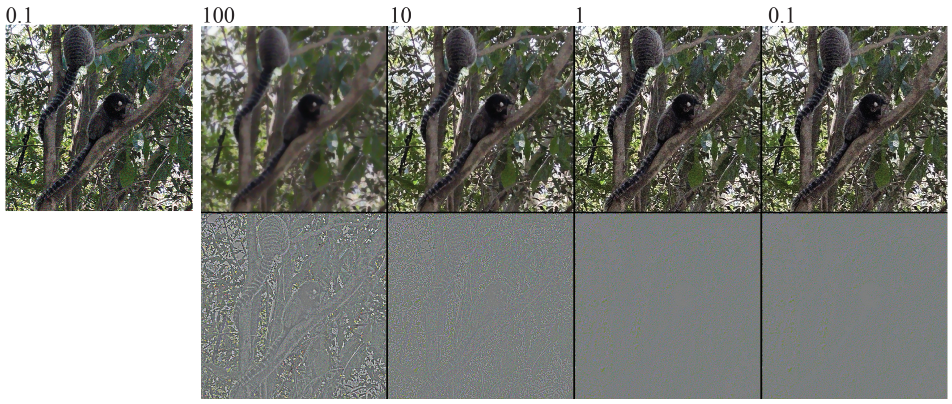

FIGURE 7.4: *The color image of Figure 7.11, with additive gaussian noise at $\sigma = 0.1$ denoised using total variation denoising (TVD; Section 7.2.2) at various settings of λ. The denoised images are in the* **top row***, with residual in the* **bottom row***. The residual is shown on a scale where positive values are light, negative values are dark, and 0 is mid-gray. The noisy image was transformed to LAB; the L component was smoothed with TVD; and the A and B components were smoothed with a gaussian kernel. Notice the strong color noise in the residual. The image noise is independent in R, G and B – so colors can change sharply – and the denoiser is very effective at suppressing this effect. Notice how, as λ is increased, the image does not become blurry (compare Figure* **??***). Edges are largely preserved, but detail is lost with increasing λ.* Image credit: *Figure shows my photograph of marmosets in Sao Paulo.*

should be small. A fair choice is to form $\mathbf{w} = \mathcal{D}_x\mathbf{n}$, and then the i, i'th element of $\mathcal{A}_x$ is

$$\frac{1}{|w_i|^{\alpha} + \epsilon}$$

where α is some power, typically between 1.2 and 2, and ϵ is a small positive constant to avoid division by zero; $\mathcal{A}_y$ follows.

There are several elements in this recipe that will recur. Notice that $\mathbf{d}$ is a *nonlinear* function of $\mathbf{n}$ (because elements of $\mathbf{n}$ are used in the construction of $\mathcal{L}$). Assume $\mathcal{L}$ is known; then $\mathbf{d}$ is linear in $\mathbf{n}$. You should think about this process as one in which $\mathbf{n}$ is used to choose a different linear operation to map $\mathbf{n}$ to the value of $\mathbf{d}$ at each pixel (**exercises**). In principle, one could insist that $\mathcal{L}$ be formed from the denoised image $\mathbf{u}$, but that would result in a very nasty optimization problem indeed.

Natural variants of this procedure involve estimating an "easy" denoised version $\hat{\mathbf{d}}$ of the original image, then using that to form $\mathcal{L}$. In the simplest such idea, $\hat{\mathbf{d}}$ is estimated with gaussian smoothing (but notice that doing this would be equivalent to using derivative of gaussians to form $\mathcal{D}_x$ and $\mathcal{D}_y$; **exercises**).

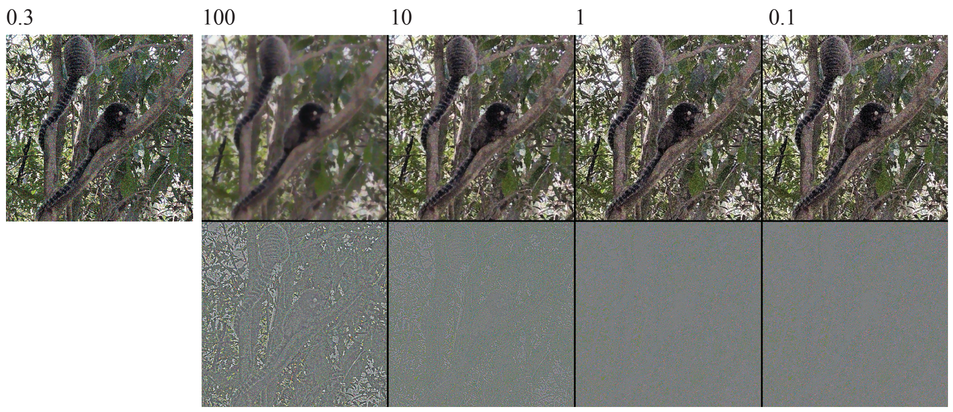

FIGURE 7.5: *The color image of Figure 7.11, with additive gaussian noise at $\sigma = 0.3$, denoised using total variation denoising (TVD; Section 7.2.2) at various settings of λ. Details in caption of Figure 7.4.*

7.2.2 Total Variation Denoising

One natural modification of the cost function for Section 7.2.1 would be to change the penalty term. One choice would be to try and obtain an image close to the noisy image but with many zeros in the gradient – so regions have constant brightness if possible, and a very slow ramp in intensity might be replaced with a constant value.

The *L2 norm*, defined by

$$\| \mathbf{v} \|_2 = \sqrt{\mathbf{v}^T \mathbf{v}}.$$

Weighted least squares penalized the squared L2 norm of the weighted gradient. Generally, a vector with small L2 norm can have many small, but non-zero, elements. This is because the square of a small number is very small, and the sum of many very small numbers is still small. The weights in weighted least squares tend to mitigate this, because small gradients have large penalty weights. **Warning:**It is quite common to refer to the *square* of the L2 norm as the L2 norm. I will try not to do this, because it's wrong, but you'll bump into this in the literature rather often.

An alternative is to penalize the *L1 norm* of the gradients. The L1 norm of a vector $\mathbf{v}$ is defined by

$$\| \mathbf{v} \|_1 = \sum_i | v_i |.$$

A vector with small L1 norm will tend to have zero elements. You can see this by comparing two cases. Write

$$C_2(\mathbf{u}) = \frac{1}{2} [\mathbf{u} - \mathbf{g}]^T [\mathbf{u} - \mathbf{g}] + \frac{\lambda}{2} \mathbf{u}^T \mathbf{u}$$

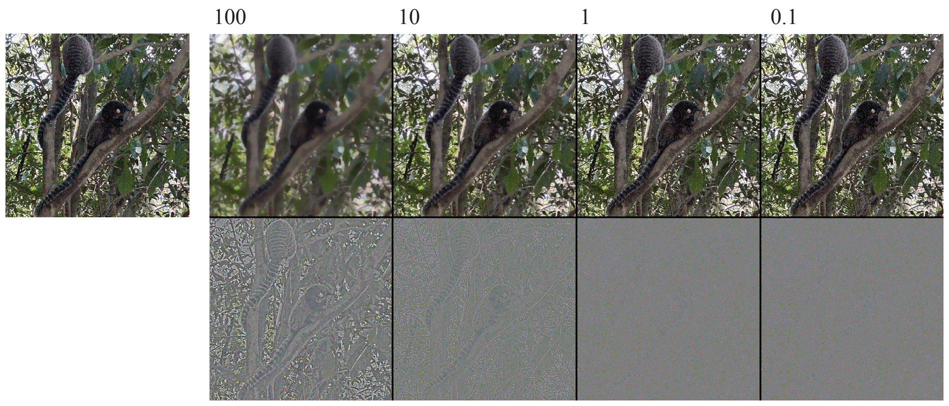

FIGURE 7.6: *The color image of Figure 7.11, with a variant of poisson noise where a randomly chosen pixel in a randomly chosen color channel is flipped, denoised using total variation denoising (TVD; Section 7.2.2) at various settings of λ. The denoised images are in the* **top row**, *with residual in the* **bottom row**. *The residual is shown on a scale where positive values are light, negative values are dark, and 0 is mid-gray. The noisy image was transformed to LAB; the L component was smoothed with TVD; and the A and B components were smoothed with a gaussian kernel. Notice the strong color noise in the residual. The image noise is independent in R, G and B – so colors can change sharply – and the denoiser is very effective at suppressing this effect. Notice how, as λ is increased, the image does not become blurry (compare Figure* **??***). Edges are largely preserved, but detail is lost with increasing λ.* Image credit: *Figure shows my photograph of marmosets in Sao Paulo.*

and notice that the $\mathbf{u}$ that minimizes $C_2(\mathbf{u})$ is

$$\frac{1}{1+\lambda}\mathbf{g}.$$

Now write

$$C_1(\mathbf{u}) = \frac{1}{2}\left[\mathbf{u}-\mathbf{g}\right]^T\left[\mathbf{u}-\mathbf{g}\right] + \lambda \| \mathbf{u} \|_1$$

and think about the $\mathbf{u}$ that minimizes $C_1(\mathbf{u})$. The penalty term isn't differentiable, which creates some inconvenience, but it is a sum over elements of $\mathbf{u}$. Now consider the i'th element of $\mathbf{u}$. If g_i is sufficiently large, then it is easy to show that

$$u_i = \frac{g_i}{1+\lambda}.$$

Now consider what happens when $g_i = \lambda$. If $u_i = 0$, then the cost will be $\lambda^2/2$, but if $u_i = \epsilon > 0$ where ϵ is small, the cost will be $(1/2)(\lambda^2 + \epsilon^2)$. This analysis implies correctly that if $-\lambda < g_i < \lambda$, $u_i = 0$. In turn, using an L1 norm as a penalty on the gradients tends to cause the reconstruction to have many zero gradients

In *total variation denoising*, the penalty is an L1 norm to the gradient. There are a variety of ways of doing this. In one approach, one seeks

$$\underset{u}{\text{argmin}} \ \frac{1}{2}\left[\mathbf{u}-\mathbf{g}\right]^T\left[\mathbf{u}-\mathbf{g}\right]+\lambda\left[\| \mathcal{D}_x\mathbf{u}\|_1+\|\mathcal{D}_y\mathbf{u}\|_1\right].$$

Note this cost function isn't differentiable, but it is convex. The optimization problem for this cost function is well understood, and is relatively easily managed (though beyond our scope). However, you should notice that the penalty encourages zeros in the x and y components of the gradient, which isn't necessarily the same as zero gradients. One could get a solution where the zeros in the x components are not aligned with the zeros of the y components, so the penalty is biased against some gradient directions but not others.

An alternative formulation requires a bit more notation. Write $d_{x,i}(\mathbf{u})$ for the i'th component of $\mathcal{D}_x\mathbf{u}$, and so on. Then solve

$$\underset{u}{\text{argmin}} \ \frac{1}{2}\left[\mathbf{u}-\mathbf{g}\right]^T\left[\mathbf{u}-\mathbf{g}\right]+\lambda\left[\sum_i\sqrt{d_{x,i}^2+d_{y,i}^2}\right]$$

which is also not differentiable. Solutions require rather more elaborate work than solutions for the previous formulation, and tend to be somewhat slower, but are not biased.

7.3 VARIANTS

7.3.1 Deblurring

Denoising takes something that isn't quite an image and finds an image that is very like it. Many phenomena can produce something that isn't quite an image. For example, take an image and blur it. The result isn't an image, but it is quite close to one. Recall from Section 41.2 that blurring is a linear operation. Write $\mathbf{t}$ for the true image in vector form, $\mathbf{d}$ for the deblurred estimate in vector form, $\mathbf{b}$ for the observed image in vector form, and $\mathcal{B}$ for the linear operator that blurs. Assume $\mathcal{B}$ is known, at least for the moment (**exercises**). Notice $\mathbf{b}$ is not *exactly* the blurred image. At the very least, there is some error from the numerical representation, and there might be some small noise present, too. Then

$$\mathbf{b}=\mathcal{B}\mathbf{t}+\xi$$

(where ξ is a vector of very small errors) and least-squares suggests choosing $\mathbf{d}$ that minimizes

$$(\mathcal{B}\mathbf{d}-\mathbf{b})^T(\mathcal{B}\mathbf{d}-\mathbf{b})$$

which would involve solving

$$\mathcal{B}^{\mathcal{T}}\mathcal{B}\mathbf{d}=\mathcal{B}^{\mathcal{T}}\mathbf{b}.$$

The least squares solution is not reliable, because $\mathcal{B}$ is a smoothing operator – say, convolve with a gaussian for concreteness. Then $\mathcal{B}^{\mathcal{T}}\mathcal{B}$ *must* have some small eigenvalues, because smoothing suppresses some patterns (which is the whole point). Note that $\mathcal{B}^{\mathcal{T}}\mathcal{B}$ must also have some eigenvalues fairly close to one, because

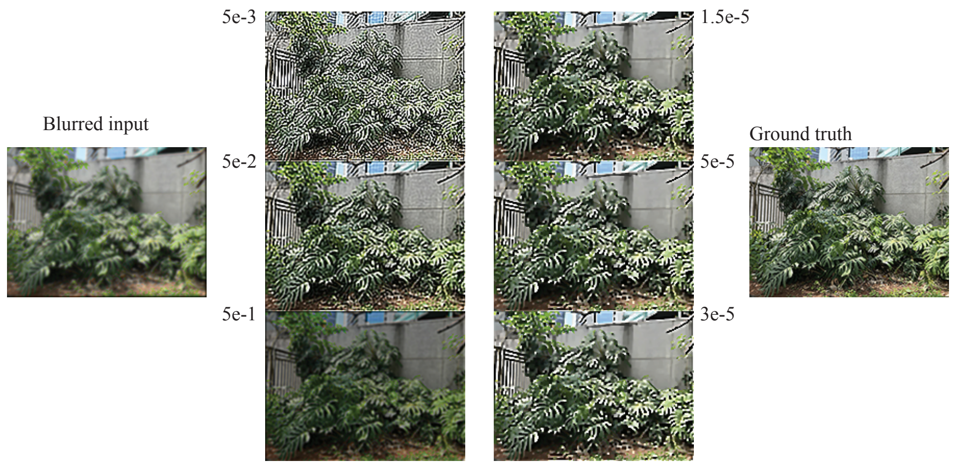

FIGURE 7.7: **Left** *shows an image blurred with a gaussian, $\sigma = 1$;* **center left** *shows regularized least squares reconstructions for different values of the regularization constant;* **center right** *shows deblurred images, using the WLS strategy of Section* **??***; and* **right** *shows the ground truth image. Notice that there must be many very small eigenvalues in $\mathcal{B}$, because when the regularization constant is small, the reconstruction is almost unrecognizable (the black and white "snakeskin" pattern is a set of very high frequency components that are easily smoothed out – which is why they correspond to small eigenvalues). Increasing the regularization constant helps control these patterns, but increasingly darkens the reconstruction. The WLS method requires an estimate of the gradient, which I took from the least-squares reconstruction. WLS helps control these components, but the best reconstruction certainly isn't perfect. Scoring whether a reconstruction is "like" an image is difficult.* Image credit: *Figure shows my photograph of a delicious monster in Sao Paulo.*

there are some patterns that change very little when smoothed. The patterns that are suppressed by smoothing must be exaggerated by $\left(\mathcal{B}^\mathcal{T}\mathcal{B}\right)^{-1}$, and may be very heavily exaggerated. This is a serious problem, because the reconstruction is

$$\begin{aligned}\left(\mathcal{B}^\mathcal{T}\mathcal{B}\right)^{-1}\mathcal{B}^\mathcal{T}\mathbf{b} &= \left(\mathcal{B}^\mathcal{T}\mathcal{B}\right)^{-1}\mathcal{B}^\mathcal{T}\left[\mathcal{B}\mathbf{t}+\xi\right] \\ &= \mathbf{t}+\left(\mathcal{B}^\mathcal{T}\mathcal{B}\right)^{-1}\mathcal{B}^\mathcal{T}\xi.\end{aligned}$$

Now even if ξ is very small, the term $\left(\mathcal{B}^\mathcal{T}\mathcal{B}\right)^{-1}\mathcal{B}^\mathcal{T}\xi$ is likely large, because $\left(\mathcal{B}^\mathcal{T}\mathcal{B}\right)^{-1}$ has some large eigenvalues, and $\mathcal{B}^\mathcal{T}\xi$ is likely to have some components in the direction of the corresponding eigenvectors.

There is a traditional procedure to handle very small eigenvalues in a matrix, known as *regularization.* One seeks a minimum of

$$C(\mathbf{u}) = (\mathcal{B}\mathbf{u}-\mathbf{b})^T(\mathcal{B}\mathbf{u}-\mathbf{b})+\lambda\mathbf{u}^T\mathbf{u}$$

by solving

$$(\mathcal{B}^\mathcal{T}\mathcal{B}+\lambda\mathcal{I})\mathbf{d} = \mathcal{B}^\mathcal{T}\mathbf{b}$$

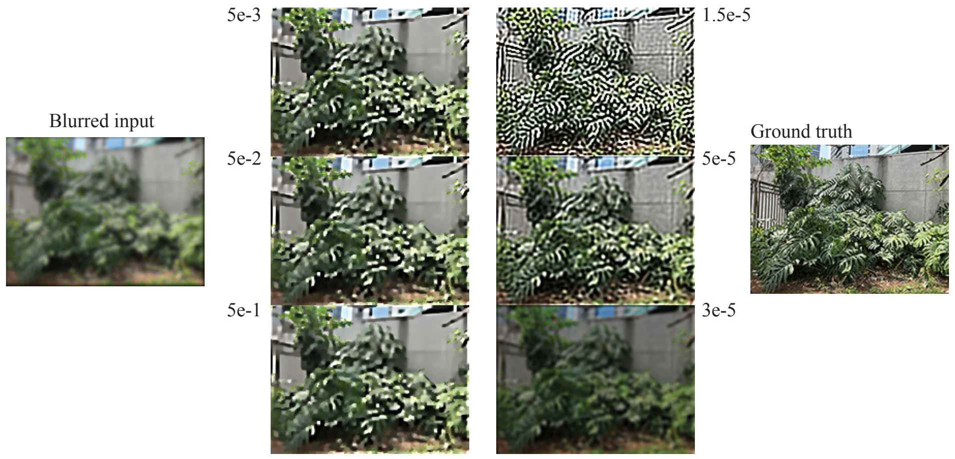

FIGURE 7.8: **Left** *shows an image blurred with a gaussian, $\sigma = 3$;* **center left** *shows regularized least squares reconstructions for different values of the regularization constant;* **center right** *shows deblurred images, using the WLS strategy of Section* **??***; and* **right** *shows the ground truth image (more details in the caption of Figure 7.7). This reconstruction problem is extremely difficult for these methods.* Image credit: *Figure shows my photograph of a delicious monster in Sao Paulo.*

for some choice of $\lambda > 0$, chosen to get good results. Notice that the map from blurry image $\mathbf{b}$ to deblurred image $\mathbf{d}$ is linear in $\mathbf{b}$, and should be shift invariant, too **exercises** .

You can see regularization either as penalizing solutions that are too big, or as constructing a matrix that is close to $(\mathcal{B}^\mathcal{T}\mathcal{B})^{-\infty}$ but does not have small eigenvalues. Alternatively, you can see regularization as a rather crude version of the cost function for Section 7.2.1, where the penalty term discourages images that are "too big", so:

$$\begin{aligned} C(\mathbf{u}) &= [\text{Term comparing } \mathcal{B}\mathbf{u} \text{ to } \mathbf{b}] + [\text{Term evaluating realism of } \mathbf{u}] \\ &= [\text{data term}] + [\text{penalty term}] \end{aligned}$$

meaning it is possible to apply the master recipe (Section 7.2). I will apply the methods of Sections 7.2.1 and 7.2.2 according to the master recipe, but other choices are possible.

Weighted least squares: The main question to deal with in using weighted least squares is how to form $\mathcal{A}_\S$ and $\mathcal{A}_\dagger$. One strategy is to form a regularized least squares estimate $\hat{\mathbf{u}}$, so

$$(\mathcal{B}^\mathcal{T}\mathcal{B} + \lambda\mathcal{I})\hat{\mathbf{u}} = \mathcal{B}^\mathcal{T}\mathbf{g}$$

then use this estimate to form $\mathcal{A}_\S$ and $\mathcal{A}_\dagger$. The problem then becomes large-scale linear algebra (**exercises**). Figures 7.7 and 7.8 show some results.

Total variation denoising: One natural strategy to use total variation denoising for upsampling and deblurring is to apply total variation denoising to

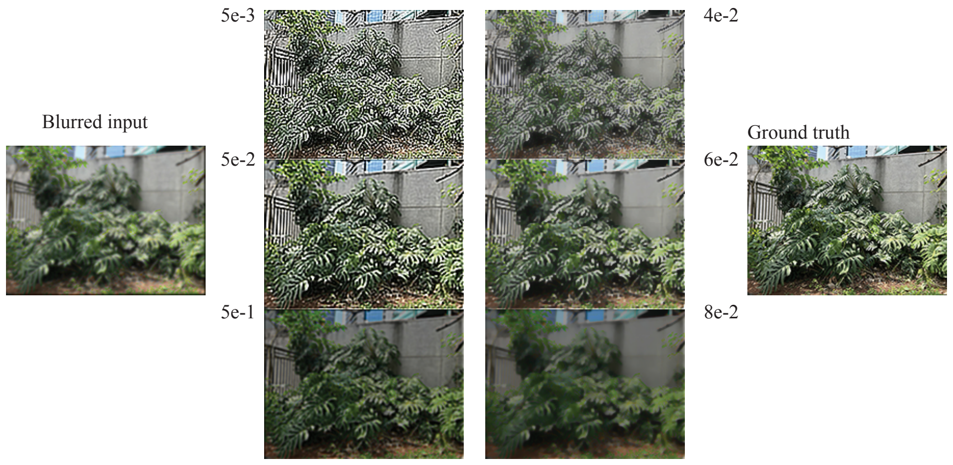

FIGURE 7.9: **Left** *shows an image blurred with a gaussian, $\sigma = 1$;* **center left** *shows regularized least squares reconstructions for different values of the regularization constant;* **center right** *shows deblurred images, using the TVD strategy of Section* **??***; and* **right** *shows the ground truth image (more details in the caption of Figure 7.7). TVD can control the problem high frequency components, with an appropriate choice of weight.* Image credit: *Figure shows my photograph of a delicious monster in Sao Paulo.*

a least squares prediction. Doing anything else requires substantial optimization tricks, sketched in the **exercises** . Figures 7.9 and 7.10 show some results.

Denoising by controlling high spatial frequency components is quite helpful, but deblurring exposes difficulties with the approach. Compare Figures 7.10 and **??** with Figures 7.9 and **??**, and notice how much harder it is to deblur a really blurry image than it is to deblur a slightly blurry image. This is because there is very much less image evidence of high spatial frequencies (**exercises**) and so the corresponding eigenvalues of $(\mathcal{B}^{\mathcal{T}}\mathcal{B})^{-\infty}$ are very large and very hard to control. Further, all the methods we have seen require engaging with inconvenient optimization problems. Chapters 41.2 and 41.2 describe much richer classes of image model that can be used for deblurring.

7.3.2 Trend and Detail Representations

The laplacian pyramid of Section 2.3.5 is an example of a *trend and detail* representation of the image at multiple scales. The coarsest layer ($\mathcal{L}_N$ in the notation of that section) contains a heavily smoothed version of the image (the trend), and each other layer contains detail at a particular scale (the layer index). The trend at the i'th layer (or scale) can be obtained by upsampling the trend at the $i+1$'th layer, then adding the detail from the i'th layer.

There is a large range of trend and detail representations available. You don't need to use a Gaussian filter, or down and upsampling by 2. For example, weighted least squares can produce a trend and detail representation. In the simplest version,

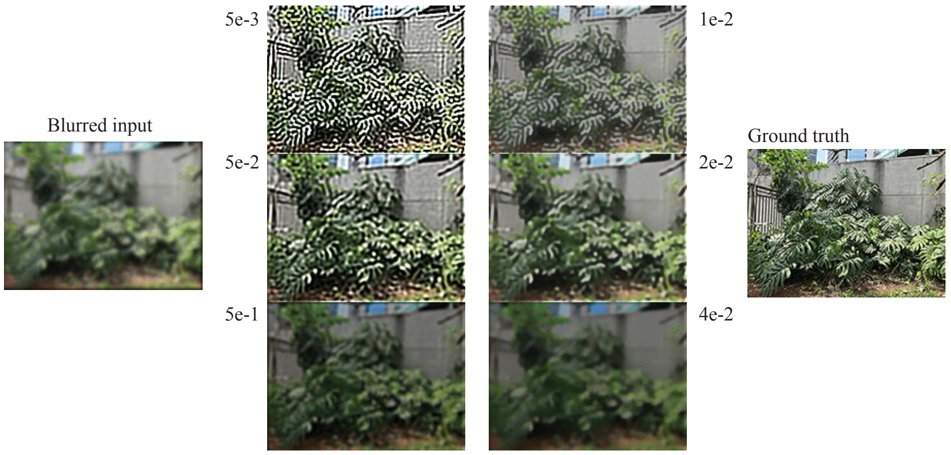

FIGURE 7.10: **Left** *shows an image blurred with a gaussian,* $\sigma = 3$*;* **center left** *shows regularized least squares reconstructions for different values of the regularization constant;* **center right** *shows deblurred images, using the TVD strategy of Section* **??***; and* **right** *shows the ground truth image (more details in the caption of Figure 7.7). TVD has a much harder time controlling these components than for Figure 7.9.* Image credit: *Figure shows my photograph of a delicious monster in Sao Paulo.*

apply WLS to an image $\mathcal{I}$ to get $W(\mathcal{I})$. This will be smoother than the original image away from gradients, so that $W(\mathcal{I})$ is trend and $\mathcal{I} - W(\mathcal{I})$ is detail. In turn, forming

$$W(\mathcal{I}) + \alpha(\mathcal{I} - W(\mathcal{I}))$$

for some positive weight α will yield emphasized detail if $\alpha > 1$, deemphasized detail if $\alpha < 1$, and the original image if $\alpha = 1$ (Figure **??**). Because weighted least squares is not *idempotent* (which means $W(W(\mathcal{I}))$ is not the same as $W(\mathcal{I})$), it can be used to produce a form of laplacian pyramid, too. Write P_i for the i'th layer,

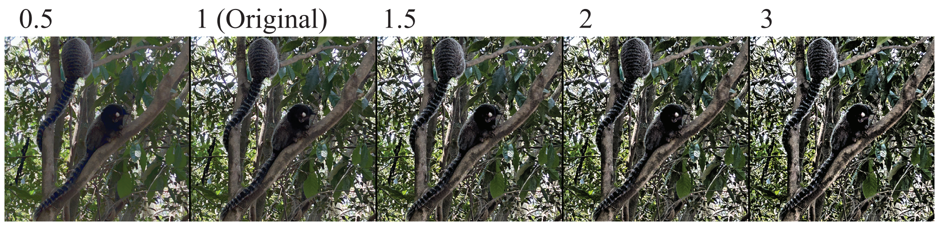

FIGURE 7.11: *For image* $\mathcal{I}$*, write* $W(\mathcal{I})$ *for the result of applying weighted least squares. Then these images are* $W(\mathcal{I}) + \lambda(\mathcal{I} - W(\mathcal{I}))$ *for different values of* λ*. This deemphasizes detail (***left***) for* $\lambda < 1$ *and emphasizes detail (***right***) for* $\lambda > 1$. Image credit: *Figure shows my photograph of marmosets in Sao Paulo.*

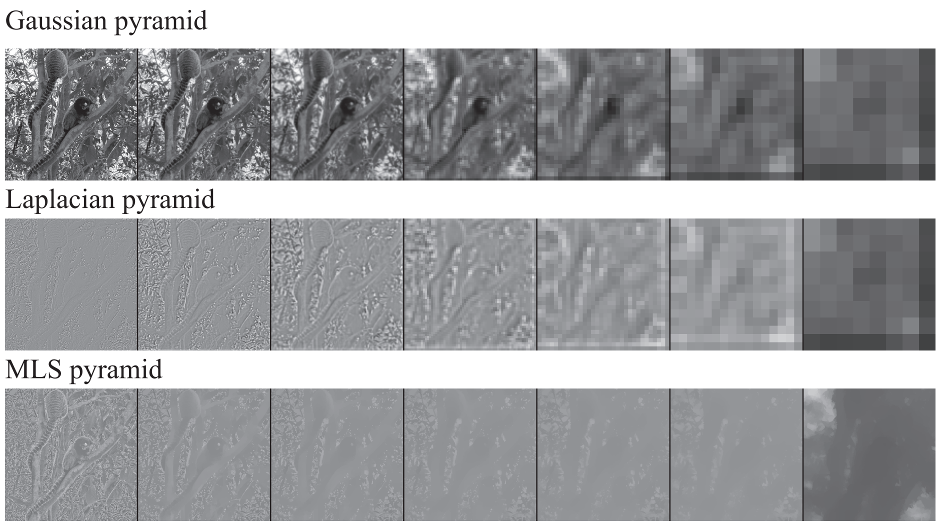

FIGURE 7.12: *A comparison of three types of pyramid.* **Top** *shows a gaussian pyramid, where each frame has been blown up to the same size (so the pixels in the coarsest layer are very large when printed). This is redundant, because each layer is quite like the previous layer.* **Center** *shows a laplacian pyramid. Notice how each layer contains rather distinct patterns at the scale associated with the layer. For example, the bars on the marmosets' tails appear in layer 2 and 3, but not others.* **Bottom** *shows a pyramid constructed using the weighted least squares method (but not downsampled). Layers again represent details of different sizes, but what appears in what layer is not the same as in the laplacian pyramid.* Image credit: *Figure shows my photograph of marmosets in Sao Paulo.*

and use

$$\begin{array}{rcl} W_0 & = & \mathcal{I} \\ W_1 & = & W(\mathcal{I}) \\ P_1 & = & W_0 - W_1 \\ \ldots & & \\ W_k & = & W(W_{k-1}) \\ P_k & = & W_{k-1} - W_k \\ \ldots & & \\ P_N & = W_N & \end{array}$$

This doesn't incorporate the downsampling in the laplacian pyramid, but that is an easy fix (**exercises**). Figure 7.12 compares different trend and detail representations.