Appendix A

Implementation

Much of the analysis in this dissertation treats images as continuous functions. There are, however, some specific issues that arise when applying these techniques to digital images. This section deals with the approaches that I have used to "discretize" the geometry-limited diffusion equations. It is not a detailed description of the code or algorithms that generated the examples that are presented. Instead, it gives descriptions of the decisions that are made in developing discrete versions of the various evolution equations. These descriptions are sufficiently detailed to be translated directly into computer algorithms.

A.1 Discrete derivatives

A.1.1 Finite differences

For this analysis assume that images are two-dimensional and are sampled on a regularly-spaces, rectilinear grid. Coordinates on that grid are denoted x and y, and $\vec{w}$ is used to denote a discrete x, y pair. A discrete image $f : [I] \mapsto \mathrm{I\!R}$ is a mapping from a discrete image domain $[I]$ to the set of reals. The spacing of the grid is h, and for a two-dimensional image $[I] \subset \{nh, mh \;\; \forall n, m \in \mathrm{I\!I}\}$, where $\mathrm{I\!I}$ is the set of integers. The notation "[ ]" is used to indicate the discrete version of a particular operator (or a discrete set in the case of $[I]$). Thus, $[\nabla]$, which is not yet defined, is the discrete gradient operator. For this analysis consider the discrete function f to be a *sampling* of some underlying continuous function, $\tilde{f} : I \mapsto \mathrm{I\!R}$, where $I \subset \mathrm{I\!R}^n$. Thus, $f(\vec{w}) = \tilde{f}(\vec{w}) \;\; \forall \vec{w} \in [I]$.

Derivatives of digital images are made with kernels or test functions, and these kernels has an associated size. The "smallest" kernel is the one associated with nearest neighbor differences. The method of *central differences* computes a first derivative of the underlying continuous function $\tilde{f}$ with an error that is of order h^2:

$$\overset{\circ}{\delta}_x f(x, y) = \frac{f(x+h, y) - f(x-h, y)}{2h}, \text{ and} \tag{A.1}$$

$$\overset{\circ}{\delta}_y f(x, y) = \frac{f(x, y+h) - f(x, y-h)}{2h}, \tag{A.2}$$

where $\overset{\circ}{\delta}$ indicates a central difference and the subscripts indicate the direction of the difference. Discrete derivatives are defined as follows:

$$\left[\frac{\partial}{\partial x}\right] f(x, y) = \overset{\circ}{\delta}_x f(x, y), \text{ and } \left[\frac{\partial}{\partial y}\right] f(x, y) = \overset{\circ}{\delta}_y f(x, y), \tag{A.3}$$

where $x, y \in [I]$.

In order to further compress the notation, let $\mathrm{d}x$ and $\mathrm{d}y$ denote vectors of length h associated with the directions of the x and y axis respectively. For a point in the image space $\vec{w} \in [I]$, the central differences are,

$$\overset{\circ}{\delta}_x f(\vec{w}) = \frac{f(\vec{w}+\mathrm{d}x) - f(\vec{w}-\mathrm{d}x)}{2|\mathrm{d}x|} \text{ and} \tag{A.4}$$

$$\overset{\circ}{\delta}_y f(\vec{w}) = \frac{f(\vec{w}+\mathrm{d}y) - f(\vec{w}-\mathrm{d}y)}{2|\mathrm{d}y|}. \tag{A.5}$$

The two diagonal directions are also interesting:

$$\mathrm{d}u = \mathrm{d}x + \mathrm{d}y \text{ and } \mathrm{d}v = \mathrm{d}x - \mathrm{d}y,$$

so that

$$\overset{\circ}{\delta}_u f(\vec{w}) = \frac{f(\vec{w}+\mathrm{d}u) - f(\vec{w}-\mathrm{d}x)}{2|\mathrm{d}u|} \tag{A.6}$$

$$= \frac{f(x+h, y+h) - f(x-h, y-h)}{2\sqrt{2}h}. \tag{A.7}$$

Central differences, $\overset{\circ}{\delta}$, introduce no geometric bias—they assign the values of derivatives (to within order h^2) at the "correct" location, which lies at the point $\vec{w}$, the center of the two samples. Beside central differences, offset differences are also useful. Let

$$\overset{+}{\delta}_i f(\vec{w}) = \frac{f(\vec{w}+\mathrm{d}i) - f(\vec{w})}{|\mathrm{d}i|} \text{ and} \tag{A.8}$$

$$\overset{-}{\delta}_i f(\vec{w}) = \frac{f(\vec{w}) - f(\vec{w}-\mathrm{d}i)}{|\mathrm{d}i|}, \quad i \in \{x, y, u, v\}.$$

Offset differences, $\overset{+}{\delta}$ and $\overset{-}{\delta}$, do not produce order h^2 approximations to $\tilde{f}$ at the grid point $\vec{w}$. They do, however, produce order h^2 approximations to the derivatives of $\tilde{f}$ in the "cracks" between grid points, which are $\vec{w} + \frac{1}{2}\mathrm{d}i$ and $\vec{w} - \frac{1}{2}\mathrm{d}i$ for $i \in \{x, y, u, v\}$. Repeated offset differences produce h^2 approximations to the second derivative at the grid point $\vec{w}$:

$$\left[\frac{\partial^2}{\partial x^2}\right] f(\vec{w}) = \frac{f(\vec{w}+\mathrm{d}x) - 2f(\vec{w}-\mathrm{d}x) + f(\vec{w}-\mathrm{d}x)}{|\mathrm{d}x|^2} = \overset{-}{\delta}_x \overset{+}{\delta}_x f(\vec{w}). \tag{A.9}$$

The discrete Laplacian in two dimensions is

$$[\Delta] = [\nabla \cdot \nabla] = \overset{-}{\delta}_x \overset{+}{\delta}_x + \overset{-}{\delta}_y \overset{+}{\delta}_y, \tag{A.10}$$

which generalizes easily to higher dimensions.

A.1.2 Gaussian kernels

The use of Gaussian "test" functions has been proposed as an alternative to nearest neighbor differences for making stable derivative measurements. There are several possible implementations.

The first is to compute the kernel over a finite window and then march this window over the entire image. Using the principles described in Sect. 2.5.1, one can analytically compute a full derivative-of-Gaussian kernel. This method is separable and is therefore relatively fast when reduced to two, one-dimensional convolutions. For a window that is $n \times n$ pixels, and $N \times N$ pixels in the image, the number of multiplications (after computing the window) is $2nN^2$. This approach suffers from the problem that the window is effectively a sampling of an analytical function. If this sampling is not fine enough (i.e., the kernel is small relative to the pixel size), then important information about the kernel is lost. In an extreme case this analytic function might have more zero-crossings than the window has samples, making an accurate analysis of zero-crossings virtually impossible. As a result, this method does not work well where the scale s is small relative to the order of the derivative.

The second strategy is to use the Fourier domain and the familiar derivative relationship

$$\frac{\partial}{\partial x} f(\vec{x}) = \mathcal{F}^{-1}[\imath\omega_x \mathcal{F}[f(\vec{w})]] \tag{A.11}$$

where $\mathcal{F}[\cdot]$ is Fourier transform (FFT in discrete domain) and $\vec{\omega} = \omega_x, \omega_y$ is a frequency which has components that correspond to the x and y directions in the image space. This derivative can be combined with Gaussian blurring:

$$\frac{\partial_s}{\partial x} f(\vec{x}) = \mathcal{F}^{-1}(\imath\omega_x \mathcal{F}(G(\vec{w}, s))\mathcal{F}(f(\vec{w}))).$$

The FFT is separable, and this method is efficient. It suffers, however, from the instability of computing derivatives in the frequency domain. At large scales this instability is offset by the low pass effects of the Gaussian. At small scales this approach will "ring" near edges, producing poor results.

A.1.3 Modified Bessel functions

Lindeberg (1991) deals with these issues in much greater detail and studies the practical and theoretical issues of using neighborhood operators in conjunction with finite differences when computing derivatives in the discrete domain. Rather than reproduce his results, I summarize. The appropriate blur kernels in the discrete domain are modified Bessel functions. These functions are the fundamental solutions to the discrete diffusion equation, which is given by

$$\frac{\overset{-}{\delta}_t}{|\mathrm{d}t|} f(\vec{w}) = \left(\overset{-}{\delta}_x \overset{+}{\delta}_x + \overset{-}{\delta}_y \overset{+}{\delta}_y\right) f(\vec{w}). \tag{A.12}$$

Equation (A.12) can be solved by the method of finite differences starting with the original image as the initial conditions $f(\vec{w}, 0)$. The recurrence relation is

$$f(\vec{w}, t + \mathrm{d}t) = f(\vec{w}, t) + \left(\overset{+}{\delta}_x \overset{-}{\delta}_x \ f(\vec{w}, t) + \overset{+}{\delta}_y \overset{-}{\delta}_y \ f(\vec{w}, t)\right) |\mathrm{d}t| \tag{A.13}$$

Solutions to this equation resemble the continuous case as the grid becomes finer ($h \rightarrow 0$). Alternatively, a Bessel function computed at a large scale resembles a Gaussian kernel. Derivatives of discrete images are taken using finite differences as described above.

In the examples in the dissertation, I use the Bessel function approach at small scales and use Gaussian kernels sampled within windows at larger scales. The Bessel functions are computed by solving Eq. (A.12) with $|\mathrm{d}t| = 0.25$ and using the relationship $2t = s^2$. When using diffusion as a means of implementing scale, I computed higher-order derivatives *after* diffusion using the *tightest fitting* differences that introduce no bias, i.e. centered on the grid point $\vec{w}$. For instance,

$$\begin{aligned} \left[\frac{\partial}{\partial x}\right] f(\vec{w}) &= \overset{\circ}{\delta}_x \ f(\vec{w}), \\ \left[\frac{\partial^2}{\partial x^2}\right] f(\vec{w}) &= \overset{-}{\delta}_x \overset{+}{\delta}_x \ f(\vec{w}), \end{aligned}$$

$$\begin{aligned}\left[\frac{\partial^3}{\partial x^3}\right] f(\vec{w}) &= \overset{\circ}{\delta}_x \overset{-}{\delta}_x \overset{+}{\delta}_x \; f(\vec{w}) \;\; , \text{ and} \\ \left[\frac{\partial^4}{\partial x^4}\right] f(\vec{w}) &= \overset{-}{\delta}_x \overset{+}{\delta}_x \overset{-}{\delta}_x \overset{+}{\delta}_x \; f(\vec{w}) \text{ for } \vec{w} \in [I].\end{aligned}$$

For scales larger than $2N$, where N is the total order of the derivative, I use sampled Gaussian kernels with a windows of size $6s$ (s is the scale or standard deviation of the Gaussian kernel), and I do an explicit convolution with this kernel. In all of these measurements the pixel spacing is the unit of measurement, so $h = 1$.

A.2 Anisotropic diffusion

The analyses of previous chapters concentrate on continuous representations of images under the assumption that the error in discrete approximations is limited only by the grid spacing. Solutions to (A.12) converge to the solutions of the continuous diffusion equation as $h \rightarrow 0$ and are stable for $|\mathrm{d}t| \leq 0.25$ (Petrovski 1967).

The challenge is to extend the algorithm of (A.12) to solve the geometry-limited diffusion equations. I address first the case of anisotropic diffusion and then generalize this analysis to cover the multi-valued case. As a discrete version of (2.9) consider

$$\begin{aligned}\frac{\overset{-}{\delta}_t \; f(\vec{w}, t)}{|\mathrm{d}t|} &= \overset{-}{\delta}_y \; g_{(y)}(|[\nabla]f|) \; \overset{+}{\delta}_y \; f(\vec{w}, t) \\ &\quad + \overset{-}{\delta}_x \; g_{(x)}(|[\nabla]f|) \; \overset{+}{\delta}_x \; f(\vec{w}, t),\end{aligned} \tag{A.14}$$

where the subscripts (in parenthesis) on the conductance terms indicate the derivatives with which they are associated. The difficulty is in determining the $g_{(\cdot)}(|[\nabla]f|)$ terms without introducing any geometric bias. Consider a point $\vec{w} = x, y$, and treat the equation in parts. According to the principle of diffusion, material flows from one area or element to another in proportion to the difference in concentrations in those areas. The term farthest right, $\overset{+}{\delta}_x \; f$, measures this difference for the point $\vec{w}$ and its $\mathrm{d}x$ neighbor. The conductance $g_{(x)}(\cdot)$ modulates the flow for this difference (if it is particularly high), and the term $\overset{-}{\delta}_x$ computes the net flow into the pixel at $\vec{w}$ from its $\mathrm{d}y$ and $-\mathrm{d}y$ neighbors. Thus, the term $g_{(x)}(\cdot)$ should not apply to the point $\vec{w}$ but to the difference between the point $\vec{w}$ and its $\mathrm{d}x$ neighbor. This difference is centered at the location $\vec{w} + \frac{1}{2}\mathrm{d}x$. Thus $g_{(x)}(|[\nabla]f|)$ should be computed using a convention for the gradient magnitude that is centered at $\vec{w} + \frac{1}{2}\mathrm{d}x$ as shown in Fig. A.1. The *gradient magnitude* requires two orthogonal derivatives centered at $\vec{w} + \frac{1}{2}\mathrm{d}x$. One derivative can be taken using $\overset{-}{\delta}_x$, but the other is not so easy—there are not enough pixels in the $\mathrm{d}y$ direction. This same problem arises in computing the term $g_{(y)}(\cdot)$ which must be centered at $\vec{w} + \frac{1}{2}\mathrm{d}y$.

Perona and Malik (1990) address this issue by computing conductance measurements using the absolute value of a directional derivative instead of the gradient magnitude. They compute flows to eight nearest neighbors, with the result

$$\begin{aligned}\frac{\overset{-}{\delta}_t \; f}{|\mathrm{d}t|} &= \overset{-}{\delta}_x \; g(|\; \overset{+}{\delta}_x \; f|) \; \overset{+}{\delta}_x \; f + \overset{-}{\delta}_y \; g(|\; \overset{+}{\delta}_y \; f|) \; \overset{+}{\delta}_y \; f \\ &\quad + \overset{-}{\delta}_u \; g(|\; \overset{+}{\delta}_v \; f|) \; \overset{+}{\delta}_u \; f + \overset{-}{\delta}_v \; g(|\; \overset{+}{\delta}_v \; f|) \; \overset{+}{\delta}_v \; f.\end{aligned} \tag{A.15}$$

Experiments with this algorithm show that it produces results which are not rotationally

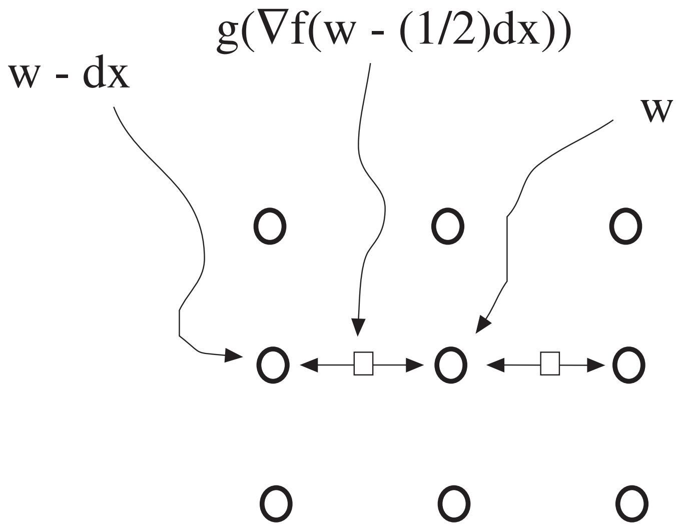

Figure A.1: The conductance applies at a point centered between two pixels, requiring a gradient magnitude that is centered at the point $\vec{w} + \frac{1}{2}\mathrm{d}x$.

symmetric—*it favors edges that are in the directions of the eight nearest neighbors.* This effect does not improve with finer grids but appears to become worse. As such, it defies the principle that underlies discrete approximations, which is that the accuracy of the approximation should improve with the decreasing grid spacing. The artifacts associated with this method result from the fact that a single directional derivative does not generally converge to the gradient magnitude in the limit $h \rightarrow 0$.

Another approach is to compute is to increase the size of the "footprint" of the diffusion operator using successive applications of central differences. The result is

$$\frac{\bar{\delta}_t}{|\mathrm{d}t|} f(\vec{w}) = \overset{\circ}{\delta}_x \, g(|[\nabla]f(\vec{w})|) \, \overset{\circ}{\delta}_x \, f(\vec{w}) + \overset{\circ}{\delta}_y \, g(|[\nabla]f(\vec{w})|) \, \overset{\circ}{\delta}_x \, f(\vec{w}) \tag{A.16}$$

where $|[\nabla]f| = \sqrt{(\overset{\circ}{\delta}_x \, f)^2 + (\overset{\circ}{\delta}_y \, f)^2}$. Both the conductance measurements and first-order derivatives are centered at pixel locations. This approach has the problem, however, of creating two independent grids (arranged like the light and dark squares on a checkerboard), and it produces unstable results. These two grids can be tied together by including diagonal terms:

$$\begin{aligned} \frac{\bar{\delta}_t \, f}{|\mathrm{d}t|} \;\; = \;\; & \overset{\circ}{\delta}_x \, g(|[\nabla]f(\vec{w})|) \, \overset{\circ}{\delta}_x \, f(\vec{w}) + \overset{\circ}{\delta}_y \, g(|[\nabla]f(\vec{w})|) \, \overset{\circ}{\delta}_x \, f(\vec{w}) \\ & + \overset{-}{\delta}_u \, g(|[\nabla]f(\vec{w} + \frac{1}{2}\mathrm{d}u)|) \, \overset{+}{\delta}_u \, f(\vec{w}) + \overset{-}{\delta}_v \, g(|[\nabla]f(\vec{w} + \frac{1}{2}\mathrm{d}u)|) \, \overset{+}{\delta}_v \, f(\vec{w}), \end{aligned} \tag{A.17}$$

where I have included the arguments of $f(\cdot)$ to indicate the placement of the derivatives. The gradient magnitude terms are computed using $|[\nabla]f(\vec{w})| = \sqrt{(\overset{\circ}{\delta}_i \, f(\vec{w}))^2 + (\overset{\circ}{\delta}_i \, f(\vec{w}))^2}$ for $i \in \{x, y\}$, and $|[\nabla]f(\vec{w} + \frac{1}{2}\mathrm{d}i)| = \sqrt{(\overset{+}{\delta}_i \, f(\vec{w}))^2 + (\overset{+}{\delta}_i \, f(\vec{w}))^2}$ for $i \in \{u, v\}$. This technique produces desirable results.

A faster method that has produced comparable results is to use the algorithm of (A.14) and compute the conductance measurements at $\vec{w} + \frac{1}{2}\mathrm{d}y$ using an averaging of pixels nearby. Compute x derivatives at the location $\vec{w} + \frac{1}{2}\mathrm{d}y$ by averaging the derivatives at both $\vec{w}$ and $\vec{w} + \mathrm{d}y$. Thus, at

$\vec{w} + \frac{1}{2}\mathrm{d}y$,

$$|[\nabla]f(\vec{w} + \frac{1}{2}\mathrm{d}y)| = \sqrt{(\frac{1}{2}\overset{\circ}{\delta}_x f(\vec{w}) + \frac{1}{2}\overset{\circ}{\delta}_x f(\vec{w} + \mathrm{d}y))^2 + (\overset{+}{\delta}_y f(\vec{w}))^2}. \tag{A.18}$$

And the anisotropic diffusion equation is

$$\frac{\bar{\delta}_t f}{|\mathrm{d}t|} = \bar{\delta}_x g(|[\nabla]f(\vec{w} + \frac{1}{2}\mathrm{d}x)|) \overset{+}{\delta}_x f(\vec{w}) + \bar{\delta}_y g(|[\nabla]f(\vec{w} + \frac{1}{2}\mathrm{d}y)|) \overset{+}{\delta}_y f(\vec{w}). \tag{A.19}$$

Experiments show that solutions to (A.14) and (A.19) are virtually identical and that with sufficiently small grid sizes they exhibit no noticeable artifacts.

A.3 Dissimilarity and multi-valued diffusion

The algorithm of A.19 is extended to include multi-valued diffusion. The multi-valued diffusion process requires computing first derivatives of all features $[\nabla]\vec{F}$ centered at the points $\vec{w} + \frac{1}{2}\mathrm{d}x$ and $\vec{w} + \frac{1}{2}\mathrm{d}y$ as well as the metric $\phi(\vec{F})$ at the same points. The metric is computed from the feature values, so the problem is to find feature values and all partial first derivatives at the points *in between* pixels.

I present two approaches with which I experimented. The first is to use the same averaging technique shown in Section A.2. Let a discrete derivative of $\vec{F}$ which is centered at point $\vec{w}$ be denoted by

$$\left[\frac{\partial}{\partial x}\right]\vec{F}(\vec{w}).$$

I use the following implementations for derivatives:

$$\left[\frac{\partial}{\partial y}\right]\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}x) = \frac{1}{2}(\overset{\circ}{\delta}_y \vec{F}(\vec{w}) + \overset{\circ}{\delta}_y \vec{F}(\vec{w} + \mathrm{d}x) \tag{A.20}$$

$$\left[\frac{\partial}{\partial x}\right]\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}x) = \overset{+}{\delta}_x \vec{F}(\vec{w}) \tag{A.21}$$

$$\left[\frac{\partial}{\partial x}\right]\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}y) = \frac{1}{2}(\overset{\circ}{\delta}_x \vec{F}(\vec{w}) + \overset{\circ}{\delta}_x \vec{F}(\vec{w} + \mathrm{d}y) \tag{A.22}$$

$$\left[\frac{\partial}{\partial y}\right]\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}y) = \overset{+}{\delta}_y \vec{F}(\vec{w}), \tag{A.23}$$

$$\tag{A.24}$$

and for the function values:

$$\begin{aligned}\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}y) &= \frac{1}{2}(\vec{F}(\vec{w}) + \vec{F}(\vec{w} + \mathrm{d}y)) \text{ and}\\ \vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}x) &= \frac{1}{2}(\vec{F}(\vec{w}) + \vec{F}(\vec{w} + \mathrm{d}x)).\end{aligned}$$

The combination of these definitions allows the computation of the derivative and the metric ϕ at locations halfway between grid points. The discrete multi-valued diffusion equation becomes

$$\begin{aligned}\frac{\bar{\delta}_t}{|\mathrm{d}t|}\vec{F}(\vec{w}) &= \bar{\delta}_x\, g\left(||\phi(\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}x))[\nabla]\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}x)||\right) \overset{+}{\delta}_x \vec{F} \\ &\quad + \bar{\delta}_y\, g\left(||\phi(\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}y))[\nabla]\vec{F}(\vec{w} + \frac{1}{2}\mathrm{d}y)||\right) \overset{+}{\delta}_y \vec{F}(\vec{w}).\end{aligned} \tag{A.25}$$

The second approach is to avoid computing features at in-between grid points by first computing a set of intermediate features as in Section 4.5.2. Let $\vec{G}(\vec{F})$ be a set of intermediate features that are computed *only at grid points*. The discrete multi-valued diffusion equation is

$$\begin{aligned} \frac{\bar{\delta}_t}{|\mathrm{d}t|}\vec{F}(\vec{w}) &= \bar{\delta}_x \, g\left(||[\nabla]\vec{G}(\vec{w}+\frac{1}{2}\mathrm{d}x)||\right) \overset{+}{\delta}_x \, \vec{F}(\vec{w}) \\ &+ \bar{\delta}_y \, g\left(||[\nabla]\vec{G}(\vec{w}+\frac{1}{2}\mathrm{d}y)||\right) \overset{+}{\delta}_y \, \vec{F}(\vec{w}). \end{aligned} \tag{A.26}$$

Assuming that one is able the express the dissimilarity in terms of such intermediate features, these two algorithms (A.25 and A.26), should approach the continuous case (and each other) as $h \rightarrow 0$. For a finite grid size, however, these two approaches can produce subtly different results. For the examples in this dissertation I use the second approach and compute intermediate features at the grid locations.

A.4 Noise

All of the "noisy" images that are used for the empirical analysis of Chapter 5 are produced using a pseudo-random noise generator that is available on the MasPar MP-1. The general strategy is to use images or "fields" of uncorrelated, uniformly-distributed random noise in order to produce the different kinds of noise that are used in the empirical studies.

A.4.1 Gaussian white noise

A number of noise types from Chapter 5 are computed from fields of uncorrelated Gaussian white noise, i.e. pixel intensities are sampled independently from Gaussian distributions. Samples from Gaussian distributions are obtained by computing a sum of samples from the uniform distribution. The algorithm is to sum M samples from a uniform probability density distribution

$$\Theta(x) = \begin{cases} 1 & \text{if } -\frac{1}{2} < x < \frac{1}{2} \\ 0 & \text{otherwise} \end{cases} \tag{A.27}$$

where $x \in \mathbb{R}$. This distribution has variance 1/12 and mean 0. If ρ_i is the ith sample from this uniform distribution, a sample from a normal distribution Λ with mean 0 and variance 1 is obtained by

$$\Lambda_k = \left(\frac{12}{N}\right)^{\frac{1}{2}} \sum_{i=1}^{M} \rho_i \tag{A.28}$$

Samples are computed in this fashion independently at every pixel location to produce an field or image of white noise.

A.4.2 Correlated Gaussian noise

Correlated Gaussian noise (CGN) is obtained by blurring white noise. For the examples in this dissertation, the approach is to compute a field of white noise g_σ and then blur with scale s using the discrete diffusion equation. Rescale the result in order to obtain the desired variance. This rescaling is achieved by using the result of Blom (1992) that

$$\sigma_s^2 = \frac{\sigma^2}{8\pi s^4} \tag{A.29}$$

for two-dimensional images, where σ^2 is the variance of the white noise, and s is the scale of the Gaussian kernel.

A.4.3 Patch noise

Additive "patch" noise is an attempt to produce correlated noise that includes sharp discontinuities (high frequency components). There are several possibilities for producing such images. One possibility is to use a stochastic process, such as the Ising or k-color processes (Ripley 1987), in order to produce distinct patches of the appropriate size, p. One could then color these patches with intensity values that are chosen from some distribution such as the Gaussian. The average size of these patches would be controlled through some parameter in the k-color process. In this dissertation I achieve a similar effect by applying anisotropic diffusion to fields of CGN. The degree of correlation in the CGN determines the size of the resulting patches.

The procedure for constructing a field of patchwise is as follows:

1. Construct a field of CGN, $g_{s,\sigma'}$, with $s = p/2$ and $\sigma' = 1.0$.
2. Let this field be the initial conditions for an anisotropic diffusion equation, with $k = 1.0$.
3. Choose the solution at time $t = 2s$.
4. Compute the pixelwise variance across the image, σ_I^2, and multiply the image by σ/σ_I, where σ^2 is the desired variance of the noise.

A.4.4 Boundary noise

The boundary noise is achieved by applying the morphological propagator described by van den Boomgaard (1992) with space-varying random coefficients. The differential equation applied to a continuous image $f(\vec{w})$ is

$$\frac{\partial f(\vec{w},t)}{\partial t} = g_{\sigma,s}(\vec{w},t)|\nabla f| \tag{A.30}$$

where $g_{\sigma,s}(\vec{w},t)$ is a field of time-varying CGN with variance σ^2 and scale s. A discrete form of this equation is achieved by using central differences for the gradient operator and experimenting to find time steps that do not cause instability:

$$\frac{\overset{+}{\delta} f(\vec{w},t)}{|\mathrm{d}t|} = g_{\sigma,s}(\vec{w},t)\left((\overset{0}{\delta}_x\ f(\vec{w},t))^2 + (\overset{0}{\delta}_y\ f(\vec{w},t))^2\right)^{\frac{1}{2}}.$$

Typical time steps are 0.1. Experiments show that this equation produces "wiggles" in the boundaries of objects in greyscale images. The scale s of the noise controls the "sharpness" of the wiggles (frequency), and the total amount of elapsed time controls the degree of perturbation (amplitude).

A.5 Highly parallel SIMD algorithm

Many of the experiments in this dissertation are computed using the MasPar MP-1, a highly parallel SIMD computer. The MasPar MP-1 is organized as an eight-connected, toroidal mesh with 64×128 processors. The diffusion equation solved by the method of finite differences scales very well on this type of architecture.

Pixels in images are allocated to processors using the "cut-and-stack" virtualization. Image dimensions are assumed to be whole multiples of the mesh size. The cut and stack layout is shown in Fig. A.2. Because most of the discrete operations involved nearest neighbor comparisons, the algo-

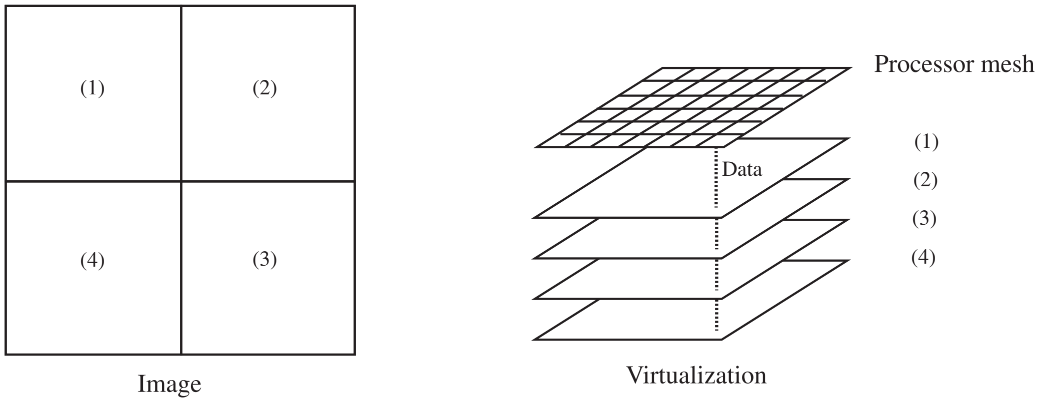

Figure A.2: The cut-and-stack virtualization for allocating image pixels to processors.

rithms discussed here run very efficiently on this architecture. The improvement over conventional workstations is very impressive, and typical run times are reduced by factor ranging from 20 to 200 for the same computation on a DECStation 5000.