Fast Adaptive Alternatives to Nonlinear Diffusion in Image Enhancement: Green's Function Approximators and Nonlocal Filters

Keywords: Anisotropic Diffusion, Nonlinear Adaptive Filtering, Image Enhancement, Real-Time Vision, Green's Functions.

Bruce Fischl†

Dept. Cognitive and Neural Systems
Boston University
677 Beacon Street
Boston, MA 02215
email: fischl@cns.bu.edu

Eric L. Schwartz†*

Dept. Cognitive and Neural Systems
Boston University
677 Beacon Street
Boston, MA 02215
email: eric@thing4.bu.edu
(617) 353-6179

* Author to whom reprint requests should be made.
† Supported in part by the office of naval research (ONR N00014-95-1-0409).

Abstract - *The goal of many early visual filtering processes is to remove noise while at the same time sharpening contrast. An historical succession of approaches to this problem, starting with the use of simple derivative and smoothing operators, and the subsequent realization of the relationship between scale-space and the isotropic diffusion equation, has recently resulted in the development of "geometry-driven" diffusion. Nonlinear and anisotropic diffusion methods, as well as image-driven nonlinear filtering, have provided improved performance relative to the older isotropic and linear diffusion techniques. These techniques, which either explicitly or implicitly make use of kernels whose shape and center are functions of local image structure, are too computationally expensive for use in real-time vision applications. In this paper, we review several recently developed methods which achieve results largely equivalent to those obtained from nonlinear diffusion. In the first technique, we use an adaptive (function approximation) approach to learn a kernel function which produces results typical of nonlinear anisotropic diffusion via spatial integration of the kernel across the image. Because of the analogy of this method with the linear Greens Function approach to PDE solution, we call this (nonlinear, space-variant) method a "Greens Function Approximator" (GFA). The second method involves the construction of a vector field of "offsets" at which to apply a (single-scale) filter, a process which is conceptually separated into two very different functions. The former function is a kind of generalized image skeletonization; the latter is conventional (but nonlocal) image filtering. The GFA method is about an order of magnitude faster than nonlinear diffusion on a serial machine. It can be fully parallelized, unlike the PDE approach, which has intrinsically serial components. The nonlocal filter is roughly an order of magnitude faster still, resulting in two orders of magnitude speed-up relative to direct PDE solution. An additional advantage of nonlocal filtering is that it allows hardware and software implementations to be applied with no modification, since the offset step reduces to an image pixel permutation, or look-up table operation, after the application of the filter. When combined with space-variant (e.g. log polar) architectures, which themselves provide between one and three orders of magnitude of speed-up relative to conventional image architectures, we are able to achieve image enhancements effects similar to those of nonlinear diffusion at frame rate (30Hz) on a single processor. Demonstration of these results on a portable active vision system will be provided.*

1. Introduction.

Many early vision systems employ some type of filtering in order to reduce noise and/or enhance contrast in regions which correspond to borders between different objects within an image. The logical extreme of this process is the creation of a piecewise constant image with step discontinuities at region boundaries. This goal is unattainable using linear filtering techniques, as noise reduction blurs the locations of boundaries between regions, sometimes to the point of fusing them.

In order to address this problem, Perona and Malik (Perona and Malik, 1987; Perona and Malik, 1990) introduced a nonlinear version of the diffusion equation previously used by Koenderink and Hummel (Koenderink, 1984; Hummel, 1986) for early visual processing. In this formulation, image intensity is treated as a conserved quantity and allowed to diffuse over time, with the amount of diffusion at a point being inversely related to the magnitude of the intensity gradient at that location. This process produces visually impressive results in terms of the creation of sharp boundaries separating uniform regions within an image, but is computationally expensive (see (ter Haar Romeny, 1994) or (Fischl and Schwartz, 1996a), for a more complete discussion of these issues).

Linear diffusion is identified with Gaussian filtering (Koenderink, 1984; Hummel, 1986) because the Gaussian is the Green's Function of the linear diffusion equation for an infinite domain. Thus, spatial integration with Gaussian kernels can be used to implement linear diffusion. The nonlinear anisotropic diffusion proposed by Perona and Malik (1987) has no known closed form solution analogous to the Green's Function solution of the linear equation, and therefore must be integrated numerically. This results in extremely high computational costs. In order to address these issues, we developed an adaptive technique which provides an approximate "Greens Function" kernel function for the nonlinear diffusion equation, thus transforming the partial differential equation (PDE) of diffusion into an integral equation (Fischl and Schwartz, 1996a). This nonlinear (and space-variant) analog of a Green's Function can then be used to construct a solution to the nonlinear diffusion PDE for a specific time via spatial integration. We achieved this by training a nonlinear function approximator to reproduce an approximate solution to the integral equation defined by the results of specific nonlinear diffusion methods, contingent on the initial data of a specific test image. This approach is explicitly multi-scale as the function approximation is based on Gaussian smoothed intensity derivatives at a variety of scales.

The advantage of the GFA method is that the training of the function approximator can be done off-line, as the adaptive process generalizes extremely well. The run-time application of filtering with the approximated "Greens Function" can then be performed in a single-time step. This allows the intrinsically serial diffusion PDE to be

replaced by a parallelizable spatial integration, yielding roughly an order of magnitude improvement in performance over nonlinear diffusion, even on a serial architecture.

An alternative branch of nonlinear diffusion research was developed by Nitzberg and Shiota (Nitzberg and Shiota, 1992). Instead of formulating the problem as a diffusion process, they constructed a nonlinear filter whose form and *location, relative to the pixel being filtered*, are functions of local image structure. The shape of their filter is narrow and elongated in the direction orthogonal to locally coherent gradients, and compact in regions of changing gradient direction in order to preserve corners and triple points.

The use of a nonlinear filter by itself addresses some of the issues of noise reduction. However, if the filter straddles an edge, it still results in blurring and hence either displacement or destruction of region boundaries. In order to alleviate this problem, Nitzberg and Shiota introduced an offset term which displaces kernel centers away from presumed edge locations, thus enhancing the contrast between adjacent regions without blurring their boundary. While their offset field works well in many situations, we have found that it does not perform adequately in images which contain edges at different scales unless it is applied iteratively, a computationally prohibitive process. Since their nonlinear filter implicitly combines the "offset" and "filtering" functions in a single (8 parameter) expression, it is difficult to design a filter that performs adequately for a variety of images. Moreover, the resulting application requires large and complex kernels and is therefore still extremely slow, as was the case for the original nonlinear diffusion approaches. The key idea of our research in this domain (Fischl and Schwartz, 1996b) is that by separating the functions of vector offset from image filtering, a much simpler and faster class of algorithms is provided. The offset vector approach we have developed has better performance than the original Nitzberg-Shiota method, and provides results comparable to nonlinear anisotropic diffusion methods, but with a speed-up of roughly two orders of magnitude.

The use of offset vectors obviates the necessity of working with multi-scale filters, or of determining the correct "scale" to be applied at different locations in the image. Computationally, this is a critical issue. Nitzberg and Shiota were able to obtain results which are comparable to anisotropic diffusion by iteratively applying relatively large filters to an image (e.g. 4 applications of an 11x11 filter). However, as is the case for nonlinear diffusion processes, the resulting procedure is far too time consuming for use in real-time vision systems. To alleviate this concern, we derived a method which provides anisotropic diffusion-quality image enhancement through a single application of a 3x3 filter coupled with an offset field. This is achievable because the offset vector field essentially introduces an adaptive scale parameter via the magnitude of the offset vectors.

From an implementation point of view, nonlocal filtering is attractive as it can be carried out as a postprocessing procedure. This allows us to apply the desired filter (e.g. 3x3 median filter) to the original image, and then use the offset vector field, in the form of a look-up table, to produce the final result by a simple pixel permutation. This technique permits conventional hardware (e.g. fast 3x3 filtering) and existing code to be applied, unchanged, to produce results which appear comparable to the much more computationally expensive nonlinear diffusion methods.

From a conceptual point of view, we provide a different interpretation of the role of "scale" in image processing. Although there is no explicit scale-space structure in this method, it nevertheless performs well on images with multi-scale features due to the adaptive scale in the form of the offset vector magnitude noted above. This is in contrast to the Nitzberg-Shiota technique, in which scale is addressed by modulating the shape of their filter, and is therefore essentially a vector-valued quantity (i.e. different directions have different scales).

The remainder of this paper is organized as follows. First, in section (2) we review the chain of research that led to the use of linear and subsequently nonlinear diffusion in image processing. In section (3) we briefly review the Greens Function Approximation, or GFA method (Fischl and Schwartz, 1996a) and follow that in section (4) with examples of its use. In sections (5)-(7) we summarize the basic idea of our offset image filtering, and contrast it with the specific algorithm proposed by Nitzberg and Shiota. The differences between the two approaches are illustrated by showing a class of images for which their method fails to perform adequately. In section (8) we compare the results of the offset filtering to nonlinear anisotropic diffusion. Finally, in section (9) we summarize the application of these results to space-variant (log-polar) image architectures, and suggest that the combination of the two techniques results in a procedure for obtaining image enhancement comparable to anisotropic diffusion in real-time (i.e. 30Hz).

2. Diffusion and Green's Functions.

Multi-scale image enhancement and representation is an important part of biological and machine early vision systems. The process of constructing this representation must be both rapid and insensitive to noise, while retaining image structure at all scales. Biological systems solve these problems in parallel through the use of retinal and cortical neurons sensitive to stimuli at a variety of spatial scales.

Attempts to address problems of this type in the machine vision community resulted in the scale-space formulation of Witken (Witken, 1983) in which an image is convolved with Gaussian kernels of various sizes. Koenderink (Koenderink, 1984) and Hummel (Hummel, 1986) have pointed out that the one-parameter family of images comprising scale-space can be equivalently viewed as snapshots of the time-evolution of the diffusion (or heat) equation

$$I_t = c\Delta I \tag{2.1}$$

Where I is the intensity image, c is a diffusion constant, It is the partial derivative of I with respect to time, and Δ is the Laplacian operator with respect to the spatial coordinates. The solution to equation (2.1) can be written in terms of the Green's Function of the system as

$$I(x, y, t) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} I(x', y', 0)\, G(x, x', y, y', t)\, dx'dy' \tag{2.2}$$

The identification of Gaussian filtering with diffusion provides a mathematical framework with which to analyze the scale-space formalism, but it does not address the issue of cross-scale comparison. While Koenderink restricted his analysis to the isotropic diffusion characterized by the linear heat equation, Perona and Malik (1987; 1990) suggested that a nonlinear anisotropic version of the heat equation could remedy some of the difficulties encountered in the use of linear scale-space. They proposed the following equation in which the conduction coefficient is not constant in space, but is rather a function of the magnitude of the intensity gradient of the image.

$$I_t = \nabla \cdot (c(|\nabla I|)\, \nabla I) \tag{2.3}$$

The Perona and Malik equation (2.3) is a nonlinear partial differential equation (PDE) which has no known closed-form solution, such as a Green's Function yields for the linear diffusion equation (2.1). Thus, in order to use the anisotropic diffusion equation for enhancing an image, equation (2.3) must be numerically integrated in time, a computationally intensive and inherently serial process. In the remainder of this paper we describe two methods for providing the same types of image enhancement as the anisotropic diffusion equation, without the need to numerically integrate a nonlinear PDE.

3. Green's Function Approximation (GFA) Filtering.

In this section we describe our earlier work in which we approximated the solution to a nonlinear diffusion process by transforming the numerical integration of the PDE describing diffusion into an adaptive filtering process (Fischl and Schwartz, 1996a). In this work we obtained an approximation of the nonlinear analog of a "Greens Function" for the original diffusion PDE. Our goal was to replace the PDE formulation, which is computationally expensive because it requires serial integration over time, with a single spatial integration or filtering step using the approximated "Greens Function". This enabled us to find an approximate solution of the diffusion problem in a single time-step by filtering the initial image data with the adaptively estimated "Greens Function".

Our strategy was to monitor the numerical integration of a nonlinear anisotropic diffusion equation, and to save the paths through which intensity valued diffused at each integration step. For a given image and evolution time, we therefore computed a set of space-variant kernels, called diffusion kernels, that *exactly* mirrored the integration of the nonlinear diffusion equation for that time. Visually examining the diffusion kernels obtained in this way yielded insight into the inefficiency of the diffusion approach. By studying the entire set of kernels with principal components analysis we found (perhaps not surprisingly!) that the kernels obtained over the full integration regime were accounted for by only a small number of different basic kernel shapes. Furthermore, the kernels occurred with specific offsets from the edges in the image. With minimal smoothing, approximately 90% of the variance in the kernels was accounted for by the first 5 or 10 principal components.

Visual examination of the diffusion kernels also made apparent where the inefficiency of the full diffusion equation solution was encountered. Although the resultant kernels appeared to be simple in gross outline, they actually

consisted of very detailed (high spatial frequency) structure, which can be seen in figure (3.1). Omitting this detailed structure via simplification through principal components analysis or lowpass filtering, made little difference to the final solution. Clearly, the PDE solution was “overfitting” the local image structure in a way which was computationally inefficient, since the details of the process were not meaningfully coupled to the final output!

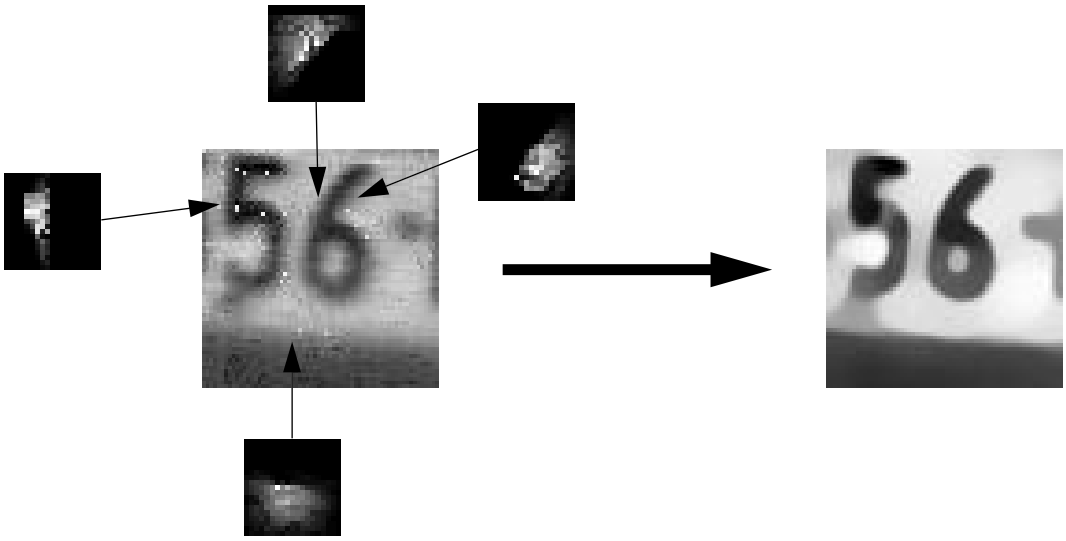

FIGURE 3.1. Typical diffusion kernels. Left: original image with diffusion kernels shown together with their location. Right: image after applying kernels to original.

Typically, the diffusion kernels had the appearance of delta functions located exactly on edge features in the image, which gradually transformed into rotationally asymmetric truncated Gaussians as one moved away from a strong edge. Figure (3.1) illustrates this effect. The large left hand image shows a region of a license plate before undergoing diffusion, while the right hand image shows the result of iterating the Perona and Malik type diffusion process proposed by El-Fallah and Ford (El-Fallah and Ford, 1994) for 100 time steps. The smaller images surrounding the initial license plate are the diffusion kernels at each of the four specified locations. As can be seen, the shape of each kernel has the same orientation as the dominant local edge direction. Further, the kernels from locations slightly offset from the edge are themselves offset so that they do not straddle the edge. It is clear from this observation that replacing the computationally expensive temporal integration of the diffusion equation with a simple filter, locating the filter centers at the correct offset from each pixel could greatly speed up the solution. This idea will be explored further in sections (5)-(7).

In the GFA work, we designed a procedure to adaptively learn the shape and position of the diffusion kernels directly, employing a nonlinear function approximator (a multi-layer perceptron trained with back-propagation) to estimate the principal component coefficients as a function of local image structure. A surprising result of this study was the discovery that a single, relatively small (64x64 pixels) image was sufficient for training a filter which achieved good results on a wide variety of images. The remainder of this section outlines the construction and estimation of the diffusion kernels used by the GFA.

3.1. Kernel Function Construction.

The advantage of the use of a Green's function in the solution of the linear heat equation is twofold. First, the Green's function generates a solution for the desired time without requiring the traversal of solutions at intervening times. Second, the integral form of the solution is amenable to parallel implementation. In order to obtain both of these advantages for the nonlinear heat equation (2.3) we take a similar approach and attempt to find the kernel function of the integral operator such that

$$I(x, y, t) = \iint I(x', y', 0)\, G_t(x, x', y, y', \nabla I)\, dx' dy' \tag{3.1}$$

The kernel function is subscripted with *t* to emphasize the fact that different kernel functions exist for different evolution times. Although equation (3.1) is ill-posed, we use the following simple idea to calculate the appropriate

kernel function. At each iteration of the numerical integration of the PDE (2.3), we track the diffusion paths of the intensity values. When carried out for n time steps, this allows the iterative computation of $G_{n\Delta t}$.

The function G_t of equation (3.1) consists of an array of two dimensional kernels, which we term diffusion kernels, one for each point in the image. We denote the kernel at image point (x,y) by $C_{x,y}$. The value of this kernel at image location $(x+i,y+j)$ is $C_{x,y}(i,j)$. The construction of the kernels is dependent on the form of the numerical integration of the diffusion equation. The algorithm we employ (see Fischl and Schwartz, 1996a) obtains the image at time $t+\Delta t$ via correlation of the image at time t with a set of space and time varying masks

$$I(x, y, t_0 + \Delta t) \approx \sum_{x'} \sum_{y'} K^{t_0}{}_{x,y}(x', y')\, I(x + x', y + y', t_0) \tag{3.2}$$

Where the mask weights are given by

$$K^{t_0}{}_{x,y} = \frac{\Delta t}{2} \begin{bmatrix} 0 & c^N(t_0) & 0 \\ c^W(t_0) & 2 - \left(\sum_{i \neq 0} c^i(t_0) \right) & c^E(t_0) \\ 0 & c^S(t_0) & 0 \end{bmatrix} \tag{3.3}$$

The superscripts 0,N,E,S,W refer to the central pixel and its four connected neighbors, with $\Delta t \leq 0.25$ for the algorithm to be numerically stable (Haberman, 1987). Thus at time t, the value of the mask associated with the image point (x,y) at location $(x+i,y+j)$ is given by $K^t_{x,y}(i,j)$. Since the four dimensional nature of these objects causes unnecessary notational clutter, we will proceed in one spatial dimension. The generalization to two dimensions is straightforward.

Before commencing with the details of the algorithm it is worth commenting on the different roles the masks and kernels play. The mask $K^t x(i)$ relates the image value at time t at position $x+i$ to the image value at time $t+1$ at position x.

$$I(x, t + 1) = \sum_i K^t{}_x(i)\, I(x + i, t) \tag{3.4}$$

Conversely, the kernel entry $C^t_x(j)$ prescribes the contribution of the initial image value at position $x+j$ to the image value at position x at time t (see figure (3.2). That is

$$I(x, t) = \sum_j C^t{}_x(j)\, I(x + j, 0) \tag{3.5}$$

Equation (3.5) is just a discrete statement of the Green's function property of the diffusion kernels we are seeking, analogous to equation (2.2), while equation (3.4) reiterates the role of the masks in the numerical integration of the PDE.

In order to construct the diffusion kernels $C^t_x(i)$ which parallelize the diffusion, we proceed inductively. For each point x in the image, we create a kernel C_x and initialize it with the Kronecker delta function

$$C^0{}_x(i) = \delta_i = \begin{cases} 1, & i = 0 \\ 0, & i \neq 0 \end{cases} \tag{3.6}$$

The application of this set of kernels to the initial image leaves it unchanged, and therefore equation (3.5) holds for $t=0$. We now seek a recursive update rule that will construct the diffusion kernel at time $t+1$ assuming we have

already built the kernels at time *t*. Combining equations (3.5) and (3.4) we find the relationship between the initial image and the image at time *t+1*.

$$I(x, t+1) = \sum_i K^t_x(i) \sum_j C^t_{x+i}(j)\, I(x+i+j, 0) \tag{3.7}$$

Since we are seeking $C^{t+1}_x(n)$, the kernel element at some arbitrary position *n*, we are only interested in the coefficients of (3.7) which multiply *I(x+n,0)*. Examining (3.7) for values of *j* such that *i+j=n*, we arrive at the recursive update law

$$C^{t+1}_x(n) = \sum_i K^t_x(i)\, C^t_{x+i}(n-i) \tag{3.8}$$

Equation (3.8) can be understood in the following way. The kernel value $C^t_{x+i}(n-i)$ represents the contribution of the initial intensity value at pixel *x+n* to the pixel at *x+i* at time *t.* This can be seen by noting that for *x'=x+i*, we have $C^t_{x+i}(n-i)=C^t_{x'}(n)$. $K^t_x(i)$ then gives the proportion of the total intensity at location *x* at time *t+1* which comes from position *x+i*. By summing over all *i*, we account for all possible paths the intensity value *I(x+n,0)* can diffuse through and arrive at position *x* at time *t+1*.

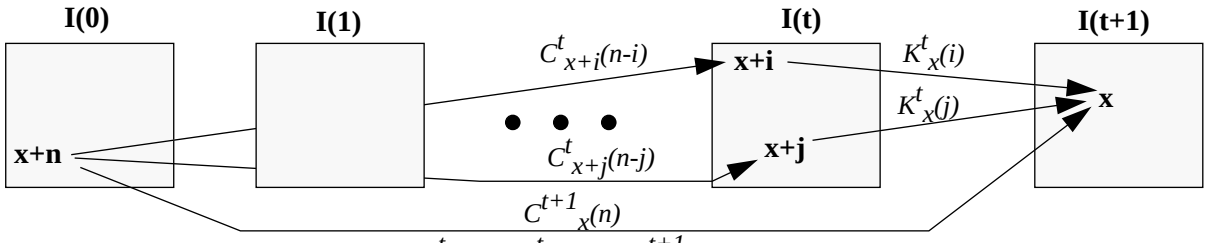

FIGURE 3.2. Illustration of the meaning of $C^t_{x+i}(j)$, $K^t_x(i)$ and $C^{t+1}_x(j)$ shown in two spatial dimensions. The composition of the two describe the contribution of an initial intensity value (left) to one at the next time step (right).

In practice, we must limit the support of the diffusion kernels *Cx*. This requires one additional modification to the algorithm. The value *Cx(j)* can be thought of as the percentage contribution of the initial image intensity at *x+j* to the final image value at *x*. Given that the *Cx* are an array of percentages, their sum must be unity. If the support of the C_x is the entire image, this occurs naturally. However, when a more limited support is imposed, each C_x must be explicitly normalized after every time step in order to preserve the conservation of image intensity.

3.2. Kernel estimation.

Given the kernel function construction algorithm described in section (3.1), we now seek a means of approximating an appropriate kernel function for an arbitrary image. That is, given a novel image, we want to estimate a set of kernels which yield an image that is perceptually similar to the one that would have been obtained by integrating the diffusion equation.

Even limiting the support of the kernels, the output of the kernel estimation process is relatively high dimensional. Since the amount of data required to fit an *n*-dimensional function increases exponentially with the dimensionality (Duda and Hart, 1973), we must compress the kernel representation considerably. To accomplish this, we perform a principal components analysis (PCA) of the smoothed kernels which yields a set of basis functions for kernel construction, as well as the coefficients of the expansion, which must be estimated for each specific image.

The estimation of the PCA coefficients is accomplished using a multi-layer perceptron as a function approximator trained with a modified backpropagation algorithm (Werbos, 1974). The function approximation task is simplified somewhat due to the uncorrelated nature of the principal components. This allows us to train separate networks for each component, obviating the need to learn nonlinear interactions between coefficients.

The features used as inputs to the function approximator must have some degree of noise tolerance while also representing the image structure in a neighborhood of each point. A simple scheme which satisfies these criteria is the use of a 3x3 window of Gaussian smoothed intensity gradient magnitude based on *x* and *y* image derivatives gener-

ated by a Sobel operator. The 3x3 neighborhood gives the network coarse information about both local orientation (if one exists), as well as edge offset.

4. Results of GFA Filtering.

We have applied the Green's function approximation or GFA algorithm to two different diffusion schemes. The filter was constructed by training the multilayer perceptron on coefficients derived from the single 64x64 pixel image shown in figure (3.1). The kernels were derived by iterating the Malik-Perona diffusion algorithm on the that image for 100 time steps. Note that all images were scaled so that their intensity values are between zero and one, all edge maps were generated using the same parameter set, and all diffusion generated images were created using the same number of iterations (100), and identical parameter values.

4.1. Gradient-based diffusion.

The first diffusion process we approximate is the Perona and Malik (1987, 1990) scheme. They choose to use the simple nonlinear diffusion equation (2.3) with a conductance coefficient *c()* which is a decreasing function of the image intensity gradient magnitude. They give a number of mild constraints on the form of the function. One choice of *c()* they discuss is

$$c(\nabla I) = e^{-\left(\frac{|\nabla I|}{k}\right)^2} \tag{4.1}$$

Figure (4.1) displays the results of applying the Malik-Perona equation as well as the Green's Function Approximation (GFA) filter to a novel real-world noisy image, corrupted with naturally occurring noise. Note the noise-

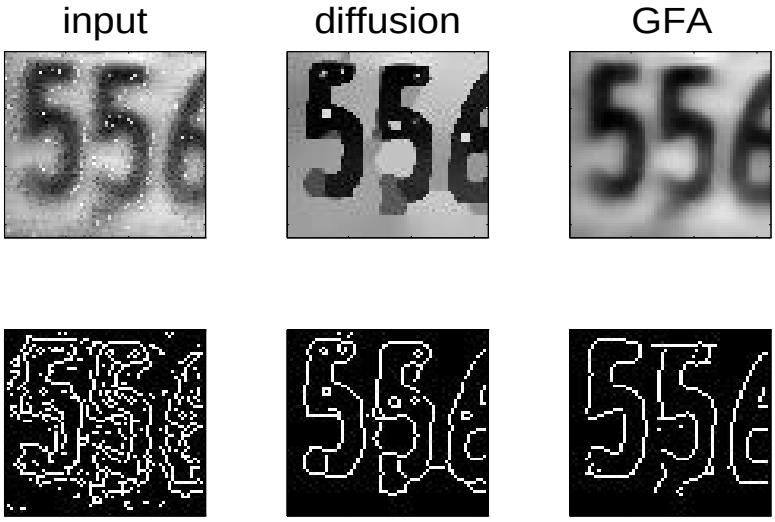

FIGURE 4.1. Perona-Malik diffusion (K=0.05) and the GFA filtered image. Top left: original image. Top center: Perona-Malik diffusion. Top right: diffusion approximation. Bottom row: associated edge maps generated by non-maximum suppression of gradient magnitude with thresholding.

enhancing property of the Malik-Perona process. Specifically, the speckle noise in all three digits has been retained and expanded into white blocks in each character. The GFA filtered image does not evidence this effect, as the interior of all three characters is uniform.

4.2. Mean curvature-based diffusion.

El-Fallah and Ford (1994) addressed the issue of the lack of noise-tolerance in the Malik and Perona equation. They proposed a conductance function of the form:

$$c(\nabla I) = \frac{1}{\sqrt{1 + k^2 |\nabla I|^2}} \tag{4.2}$$

Equation (4.2) is derived by embedding the image into R^3, and setting the right hand side of (2.3) equal to $2H$, or twice the mean curvature of the surface. Solving this system for a conductance function yields the *c()* of equation (4.2). Figure (4.1) depicts the mean curvature based diffusion as well as its approximation. As can be seen, the mean

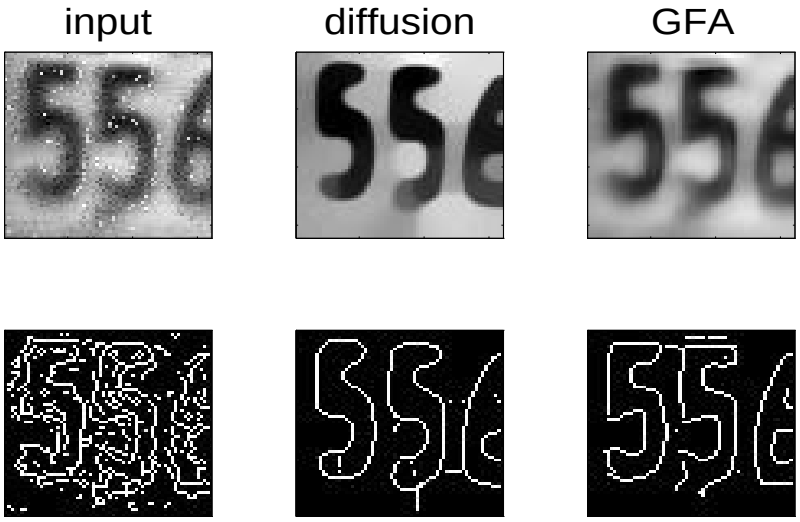

FIGURE 4.2. Mean curvature-based diffusion (*k*=50) and its approximation. Top left: original image. Top center: image after mean curvature diffusion. Top right: diffusion approximation. Bottom row: associated edge maps generated in the same way as figure (4.1).

curvature-based diffusion has good noise-suppression qualities. The character outlines are quite clear, with only minor noise edges remaining, although high curvature points such as corners in the characters are destroyed by the application of mean curvature diffusion, while they are retained by the GFA.

5. Image Filtering and Displacement Vector Fields.

The GFA approach outlined above is a significant improvement over anisotropic diffusion in terms of computational speed. However, it is still too computationally intensive for use in real-time vision systems. In order to address this problem, we made a close study of the diffusion kernels estimated by the GFA function approximator in an effort to simplify the process. Viewing the kernels elucidated some basic facts about the behavior of nonlinear diffusion processes. The reduced amount of diffusion in boundary regions causes edges in an image to become barriers to the spreading and averaging of intensities. This results in diffusion kernels which average intensity values from one side of an edge or another, but not both. Furthermore, the exact shape of the kernel did not seem to be as important as the fact that it was not permitted to straddle an edge, as evidenced by the minimal impact that lowpass filtering the diffusion kernels has on the result of applying them. From these insights, it became clear that it was possible to formulate a heuristic procedure for simplifying this process by filtering regions which were displaced from the boundary, as we will outline in this section.

The purpose of filtering an image is to exchange the intensity value at each pixel for some linear or nonlinear function of its near neighbors, with the intent of producing a value which is more representative of the region in which it lies. In image regions which correspond to the interior of an object this type of filtering produces desirable results. However, pixel values that lie on the border of two regions are not representative of either, but rather of some intermediate value. In this case, instead of calculating a new value for the border pixel using neighboring intensities, it is more effective to use a neighborhood which is offset from the edge, and thus more representative of the interior values. A useful metaphor for understanding this procedure is to imagine that the boundary “repels” the filter, pushing it into the interior of a region.

Offset filtering requires the generation of a vector field over the image domain which specifies an appropriate displacement at each point. Intuitively, the displacement direction should be either parallel or antiparallel to the dom-

inant local gradient direction, based on which interior region is judged to be ‘closer’. Nitzberg and Shiota (Nitzberg and Shiota, 1992) proposed a method based on gradient direction as well as magnitude which performs well in many cases. However, their technique fails for images which contain edges at a number of scales, as we will show below.

6. Offset Vector Field Computation.

Nitzberg and Shiota employed an adaptive filter, using local image structure to modulate the shape of the filter to average along an edge, but not across it. However, if the kernel is symmetric around the central pixel, some averaging of edge values must occur (see figure (6.1)). In order to alleviate this problem, Nitzberg and Shiota proposed an offset

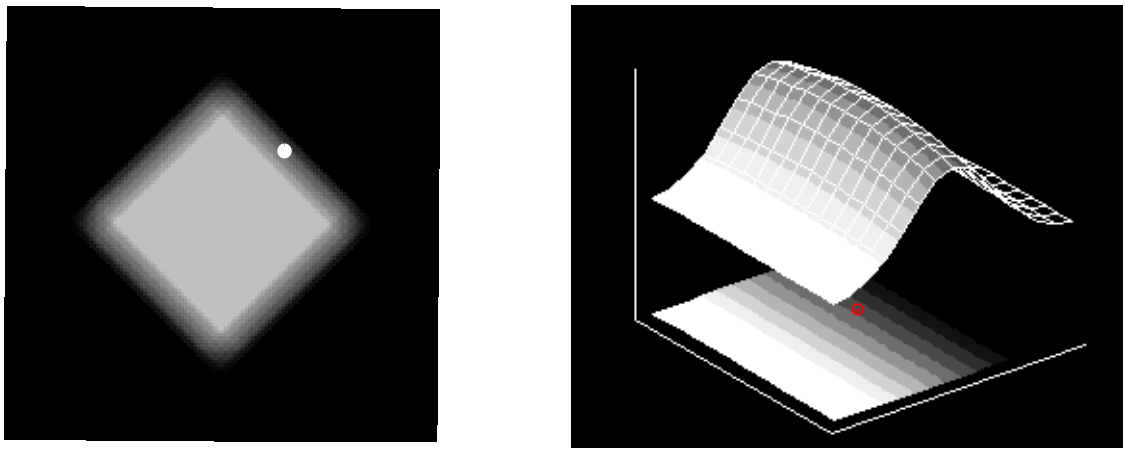

FIGURE 6.1. Example of the form of the Nitzberg-Shiota filter. Left: a blurred square. The dot represents the center of the filter. Right: the form that the filter takes, together with the local neighborhood used to construct it. By mapping the local intensity values onto the surface of the filter, it is clear that some averaging of edge values still occurs.

term which pushes the center of the kernel away from the point being filtered. The purpose of generating this type of offset vector field is to displace filters away from border areas. We have found that a reasonable means of accomplishing this is to displace in the direction normal to the boundary of a region, as this is the direction with no component along the edge. In order to compute this type of vector field, three issues must be addressed, each of which depends on an estimate of the position and orientation of the local edge, if one exists.

The first is the determination of the normal vector itself. Since the gradient is normal to the level sets of an image, using the gradient direction as an estimation of the normal is a reasonable approach. Once the normal vector has been computed, it must be assigned a sign. That is, a determination must be made as to whether the displacement should be in the direction of increasing or decreasing gradient. This choice reflects a decision as to which region the point in question has been assigned - the region at the “top” of the gradient, or the region at the “bottom”. A reasonable criterion for making this choice is to attempt to displace away from the midpoint of the presumed edge location. Finally, once the normal vector and its sign have been fixed, the magnitude of the displacement must be determined. The magnitude should be sufficient to displace the kernel entirely out of the border area, but small enough to avoid displacing it out of small regions representing fine-scale image structure.

The computation of an appropriate offset vector field $\boldsymbol{v(z)}$ can therefore be separated into the calculation of three separate quantities, each of which is computed from an initial estimate of image gradients generated by applying a 3x3 Sobel operator to a Gaussian smoothed (σ=2) image. These three quantities which correspond to the direction, sign, and length of the normal vector are termed *offset orientation* $\boldsymbol{O(z)}$*, offset direction* $\boldsymbol{d(z)}$*, and offset magnitude* $\boldsymbol{m(z)}$ respectively.[1]

Symbolically the offset calculation can be written in terms of these quantities as

1. Actually, these are all function of the intensity gradient as well as position $\boldsymbol{z}$. We suppress this functional dependence to prevent unnecessary notational clutter.

$$\boldsymbol{v}(\boldsymbol{z}) = m(\boldsymbol{z})\, d(\boldsymbol{z})\, \frac{\boldsymbol{O}(\boldsymbol{z})}{|\boldsymbol{O}(\boldsymbol{z})|}, \qquad \boldsymbol{z} = \begin{bmatrix} x & y \end{bmatrix}^T \tag{6.1}$$

Since the sign of the orientation vector $\boldsymbol{O}(\boldsymbol{z})$ has yet to be determined, we constrain its angle to be in the range $[0,\pi]$. Offset direction $d(\boldsymbol{z})$ is a binary value (1 or -1) corresponding to the choice of a sign for the normal vector, and determines whether the offset vector is in the orientation direction or the opposite one (i.e. orientation+π). Finally, offset magnitude $m(\boldsymbol{z})$ encodes the length of the offset vector. The orientation and direction are computed first via

$$\boldsymbol{O}(\boldsymbol{z}) = \int_W s(\nabla I(\boldsymbol{z}+\boldsymbol{z}'))\, \nabla I(\boldsymbol{z}+\boldsymbol{z}')\, d\boldsymbol{z}', \qquad s(\boldsymbol{z}) = \begin{cases} -1, & x < 0 \\ 1, & x \geq 0 \end{cases} \tag{6.2}$$

$$d(\boldsymbol{z}) = \operatorname{sgn}\left(\int_W |\boldsymbol{O}(\boldsymbol{z}) \cdot \nabla I(\boldsymbol{z}+\boldsymbol{z}')|\, (\boldsymbol{O}(\boldsymbol{z}) \cdot \boldsymbol{z}')\, d\boldsymbol{z}' \right) \tag{6.3}$$

W is a window around the point $\boldsymbol{z}=(x,y)^T$, typically 3x3, and $\boldsymbol{z}'=(x',y')^T$ is the vector from the central point $\boldsymbol{z}$ to each point in the window. The orientation is basically a vector sum of gradients in the window W, with the function s used to limit the orientation to the range $[0,\pi]$. The direction calculation is made up of two terms. The first discounts gradients which are not in the orientation direction, and are therefore either caused by noise or a corner; while the second simply causes edges in one direction to push the offset vector in the opposite direction.

Once the orientation and direction of the vector field have been determined, the magnitude is calculated via a one dimensional search in the offset direction for a zero crossing of the vector field in that direction (i.e. until the dot product of the offset at the central point with a point in the offset direction is less than or equal to zero). This indicates that the vector field has either vanished, signifying the interior of a region, or has changed orientation by at least 90^o, possibly indicating the presence of the far edge of the region. The dot product therefore provides a barrier which prevents the offset vector from extending across additional edges in the offset direction.

The offset calculation is thus a two step procedure. First, an initial offset field is computed using equations (6.2)-(6.3) via

$$\boldsymbol{v}_i(\boldsymbol{z}) = d(\boldsymbol{z})\, \boldsymbol{O}(\boldsymbol{z}) \tag{6.4}$$

Then, we search in the offset direction for the first point $\boldsymbol{z}'$ such that the dot product of the initial offset at $\boldsymbol{z}'$ with the initial offset at the central point $\boldsymbol{z}$ is non-positive.

$$m(\boldsymbol{z}) = min\ \alpha : \boldsymbol{v}_i\Big(\boldsymbol{z} + \alpha \boldsymbol{v}_i(\boldsymbol{z})\Big) \cdot \boldsymbol{v}_i(\boldsymbol{z}) \leq 0 \tag{6.5}$$

Finally, we form the final vector field using $m(\boldsymbol{z})$ as the magnitude of the initial vector field

$$\boldsymbol{v}(\boldsymbol{z}) = m(\boldsymbol{z})\, \frac{\boldsymbol{v}_i(\boldsymbol{z})}{|\boldsymbol{v}_i(\boldsymbol{z})|} \tag{6.6}$$

In contrast, Nitzberg and Shiota bundle the generation of their offset vector field into a single calculation given by

$$\boldsymbol{v}(\boldsymbol{z}) = \int_W \frac{1}{|\boldsymbol{z}'|} (\nabla I(\boldsymbol{z}+\boldsymbol{z}') \cdot \boldsymbol{z}')\, \nabla I(\boldsymbol{z}+\boldsymbol{z}')\, p(\boldsymbol{z}')\, d\boldsymbol{z}' \tag{6.7}$$

$$\phi(\boldsymbol{v}) = c\frac{\boldsymbol{v}}{\sqrt{\mu^2 + \|\boldsymbol{v}\|^2}} \tag{6.8}$$

Where p is a Gaussian, and ϕ is a vector valued compressive nonlinearity, used to limit the length of the displacement vectors, with c specifying the maximum length, and μ modulating the slope of the compression. The resulting offset vector field is smooth with a zero-crossing in magnitude around the estimated center of an edge, so that the vector field reverses direction as a point changes from "interior" to "exterior". However, the smoothness of their vector field is a serious drawback. It causes the vector field to have small magnitude throughout the interior of broad edges, resulting in offsets which are insufficient to displace a filter out of the edge region.

In order to see the efficacy of the search mechanism for adaptively determining the appropriate offset magnitude, vis-a-vis the Nitzberg-Shiota approach, consider figures (6.2)-(6.4). Figure (6.2) depicts an image which contains

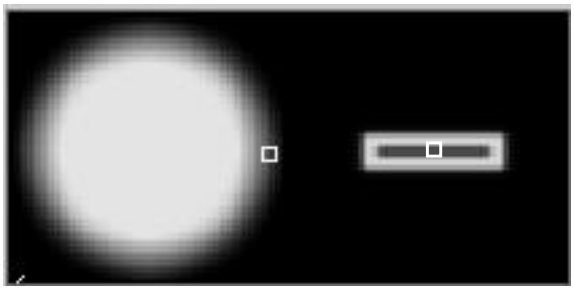

FIGURE 6.2. Test image which contains both large scale edges (the circle) as well as small-scale image structure (the rectangle). The boxes show the location of the insets in figures (6.3) and (6.4)

both large and small-scale image features in the form of a circle and a rectangle respectively. Figure (6.3) illustrates the form of the Nitzberg-Shiota vector field in these regions. In order to generate substantial displacements on the border of the circle (center), it is necessary to set the parameter c in equation (6.8) to such a large value, that the vectors in the vicinity of the rectangle are magnified to extend entirely out of the rectangle (left). Subsequent filtering using this offset field results in destruction of the rectangle while the circle is only mildly enhanced (right).

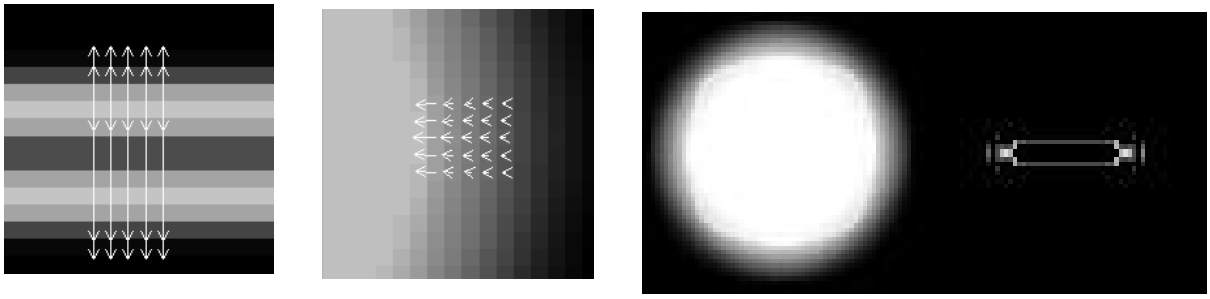

FIGURE 6.3. Nitzberg-Shiota offset field in the center of the rectangle (left) and on the right edge of the circle (center). Right: the result of a 3x3 offset median filter.

In contrast, the search mechanism we employ enables the offset magnitude to be determined adaptively by local image structure in the offset direction. This permits the vectors on the border of the circle to grow to the length necessary to extend entirely out of the border region (figure (6.4), center), while the offsets around the rectangle are constrained by the small scale image structure in that region (figure (6.4), left). The subsequent filtering results in well-defined boundaries for both the circle and the rectangle (figure (6.4), right).

It is worth noting that the adaptive offset magnitude embeds a different notion of scale into offset filtering than is usually used in contemporary applications of diffusion or scale-space architectures. The diffusion formalism grew out of linear filtering techniques such as those of Burt (Burt and Adelson, 1983), Witken (Witken, 1983) and Marr (Marr and Hildreth, 1980). In these approaches, the scale of a feature is defined by the size of the kernel required to detect it. In the anisotropic extension of the diffusion paradigm, scale is associated with integration time modulated by local gradient magnitude, and by extension with the distance across which intensity values diffuse to arrive at a given location. Regions of high gradient inhibit the amount of diffusion, and are thus associated with a smaller scale than

smoother image areas. In our approach, the relationship between scale and distance is made explicit via the magnitude of the displacement vector at a given location. Larger scale (i.e. more blurred) edges result in longer displacement vectors, but no change in filter size. Conversely, the presence of small scale image features constrains the length of the displacement vectors, preserving the features in question. The smoothing associated with diffusion can then be accomplished using any of a variety of standard fixed-size filters, which are applied nonlocally at the offset location.

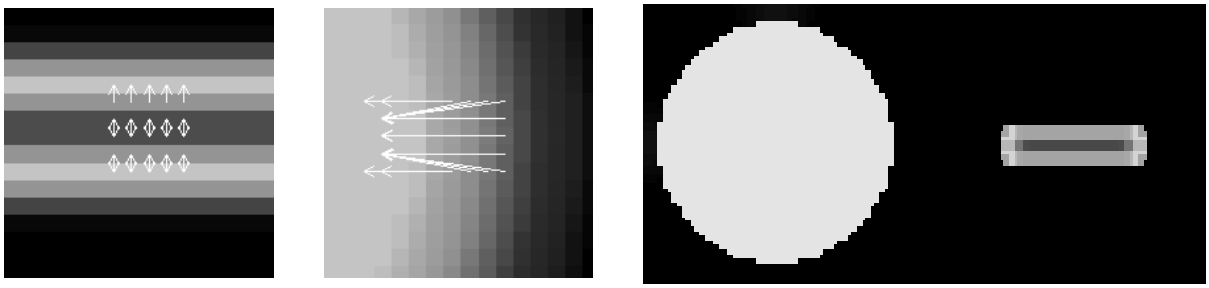

FIGURE 6.4. Adaptive length offset vector field computed via equations (6.1)-(6.6) in the center of the rectangle (left) and on the right edge of the circle (center). Right: the result of a 3x3 offset median filter.

7. Postprocessing with displacement vector fields.

The most straightforward implementation of offset filtering is to apply the displaced filter directly to the (offset) pixel neighborhood. However, filtering with a displacement vector field can be formulated in a different way which greatly simplifies both software development, and potential hardware implementation of this process. The offset filter process outlined in section (6) is identical to filtering an image without a displacement vector field, then using the displacement vectors to shuffle the positions of the image intensity values. In this way, the value at each point in the filtered image is replaced with the value at the location specified by the displacement vector field.

The transformation of offset filtering into a postprocessing procedure has a number of notable advantages. Most importantly, it allows efficient implementations of offset filtering using existing algorithms and fast hardware. The median filter is an excellent example of this process, since efficient implementations exist which make use of the overlap of neighboring windows to speed up the median computation (Huang et al., 1979; Danielsson, 1981). Straightforward use of displaced windows renders this method inapplicable. However, applying the displacement vectors after the application of a standard median filter enables the use of this type of optimization. From an implementation standpoint, the post-processing procedure obviates the need to modify each individual filter to employ a displacement field. Furthermore, postprocessing permits the offset computation to be carried out on the smoother filtered image. The post-processing approach is obviously advantageous with respect to both hardware development, as the pixel permutation procedure has a straightforward hardware implementation, as well as software implementations, because the filters can be developed independently from the estimation of the offset field.

8. Results of Nonlocal Filtering.

In this section we present a comparison of an offset median filter with the result of using anisotropic diffusion for image filtering. We use the median as it is a nonlinear filter with good noise suppression capabilities at relatively low computational cost, although other filters such as a simple Gaussian also work well with the offset field. This highlights an additional advantage of nonlocal filtering: the choice of filter can be made independently from the use of the offset vector field, perhaps on the basis of estimated image statistics. The Perona-Malik technique for nonlinear diffusion is not noise-tolerant (Whitaker and Pizer, 1991; El-Fallah and Ford, 1994), and is hence inappropriate for comparison purposes. For that reason, the images presented in this section are generated using the mean curvature based diffusion algorithm of El-Fallah and Ford (El-Fallah and Ford, 1994) which has good noise-suppression qualities.

The offset computation slows down the median filter by about a factor of six or seven, but is still approximately an order of magnitude faster than our earlier Greens Function approximation to nonlinear diffusion (Fischl and Schwartz, 1996a), which was itself roughly an order of magnitude faster than the nonlinear diffusion process. Running on a 50 MHz Sparc-10, a 3x3 median filter applied to a 256x256 pixel image requires approximately 0.75 seconds. Using the displacement vector field increases the time to roughly 5.5 seconds, while the 100 time steps used to integrate the anisotropic diffusion PDE necessitate about 450 seconds. In addition, the offset median frequently outperforms the anisotropic diffusion in terms of image quality. This is illustrated in figure (8.1). As can be seen in this

image, the using the identical diffusion regime as the one used in section (4.2) (100 iterations, k=50) results in both too much blurring of the eyes and nose, as well as the retention of speckle noise in a variety of locations. In contrast, the offset median filtered image shown on the right has retained small-scale image structure in the form of the eyes, nose and mouth, as well as providing substantial noise-reduction and enhancement of large-scale features such as outline of the face.

FIGURE 8.1. Comparison of offset median filtering with the El-Fallah and Ford mean-curvature based diffusion. Left: original image. Center: mean-curvature filtered image. Right: offset median filtered image.

9. Conclusion.

Linear filtering can be used to efficiently reduce noise in images at the cost of blurring and possibly fusing region boundaries. Nonlinear techniques are useful in this context, resulting in both contrast enhancement as well as noise reduction. The general goal of the various approaches that have been developed is to avoid "smoothing" across edge structure in the image, while smoothing along the edge structure. Anisotropic diffusion equation based methods achieve this by modifying the diffusion constant adaptively so that more diffusion occurs along, as opposed to across edges (Perona and Malik, 1987). Neural network approaches achieve similar goals by emulating this behavior with detailed networks of model neurons (Cohen and Grossberg, 1984; Grossberg and Mingolla, 1985). However, the computational cost of these algorithms prohibits their use in real-time or quasi real-time vision applications.

Historically, there has been a progression of ideas starting with simple, single scale local filters (e.g. Sobel operator), followed by the introduction of multiple scale filters (Marr and Hildreth, 1980), the Laplacian pyramid (Burt and Adelson, 1983), scale-space (Witken, 1983), the linear diffusion equation (Koenderink, 1984; Hummel, 1986), nonlinear diffusion (Perona and Malik, 1987), Greens Function approximation to nonlinear diffusion (Fischl and Schwartz, 1996a), and finally offset vector field nonlinear filtering (Fischl and Schwartz, 1996b). The GFA method eliminates the explicit diffusion mechanism from multi-scale image processing, while the offset vector nonlocal filter eliminates the need for an explicit multi-scale analysis entirely, replacing it with the notion of scale in the form of the magnitude of the offset vector field, coupled with a simple, single-scale filter.

In this paper we have reviewed two alternative approaches to the standard use of anisotropic diffusion for image enhancement. The first of these, called a Green's Function Approximator or GFA, adaptively generates a filter which can be used to obtain an approximate solution to the nonlinear diffusion equation for a specific time via spatial integration. The resulting filter is noise-tolerant, parameter free, and roughly an order of magnitude faster than the numerical integration of the anisotropic diffusion equation.

The second alternative is nonlocal filtering, which is a simplification and extension of the GFA technique, modifies the use of standard image filters such as the mean or median, to make use of displacement vector fields. The displacement vectors push kernels away from edge regions, preventing edge blurring and destruction, while achieving results which appear to be qualitatively similar to diffusion based approaches. Computationally, the nonlocal filtering provides an additional order of magnitude speed-up, thus giving about two orders of magnitude with respect to the conventional implementation of nonlinear diffusion.

We have found both the GFA and the offset filter methods to hold up well for a wide variety of images, and in all cases, to provide very significant improvements in speed of computation. When combined with space-variant vision representations, such as using a log-polar image architecture (e.g. Rojer and Schwartz, 1990) which has been shown to be a natural multi-scale structure for use in diffusion (Fischl et al., 1996), it is possible to achieve frame-rate

enhancement, providing 3-5 orders of magnitude of speedup over conventional anisotropic diffusion on space-invariant image architectures (two orders of magnitude in the diffusion stage, and one to three orders of magnitude from the space-variant pixel compression). In summary, we have outlined two new approaches to image filtering which achieve results that are comparable to nonlinear diffusion. The simplest and fastest result is provided by the offset vector field filter, which has the following implementational advantages:

- *Speed*: the offset vector filter is approximately two orders of magnitude faster than nonlinear diffusion, and roughly one order of magnitude faster than the Greens Function approximator (Fischl and Schwartz, 1996a) to nonlinear diffusion.
- *Hardware application*: By using the image permutation form of the offset vector filter, it is possible to use existing, or future, fast filter hardware, and a simple LUT or image permutation, to implement the nonlocal filtering.
- *Algorithm design*: By separating the process into a generalized skeletonization (i.e. determing the orientation, direction and magnitude of the offset vector field), and a simple single scale filter, the design of new versions of this class of algorithm is greatly simplified.

Finally, from a theoretical point of view, the following insights are provided by this work:

- Both learning approaches (to generate the GFA) and nonlocal (offset vector field) methods provide promising lines of investigation in image segmentation and enhancement.
- The desirable aspects of scale-space methods are retained without the need to explicitly introduce scale, which is represented in this method by the magnitude of the offset vector field.
- The desirable performance of nonlinear diffusion is retained without reference to any underlying diffusive, i.e. intrinsically serial, process.
- Nonlocal filter operators, implicit in the work of Nitzberg and Shiota, are explicitly developed in this paper.
- The combination of two very different aspects of image processing (i.e. generalized skeletonization as represented by the determination of the offset vector field orientation, direction and magnitude) with conventional image filtering, seem to offer a fertile area for future development.

10. Bibliography.

Burt, P. and Adelson, E. H. (1983). The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 9(4):532–540.

Cohen, M. A. and Grossberg, S. (1984). Neural dynamics of brightness perception: Features, boundaries, diffusion, and resonance. *Perception and Psychophysics*, 36:428–456.

Danielsson, P.-E. (1981). Getting the median faster. *Computer Graphics, and Image Processing*, 17:71–78.

Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. John Wiley and Sons, New York.

El-Fallah, A. I. and Ford, G. E. (1994). Nonlinear adaptive image filtering based on inhomogeneous diffusion and differential geometry. *SPIE Image and Video Processing II*, 2182:49–63.

Fischl, B., Cohen, M. A., and Schwartz, E. L. (1996). Real-time anisotropic diffusion using space-variant vision. Technical Report CAS/CNS-TR-95-026, Boston University, Department of Cognitive and Neural Systems, 677 Beacon St., Boston MA.

Fischl, B. and Schwartz, E. (1996a). Learning an integral equation approximation to nonlinear anisotropic diffusion in image processing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, in press.

Fischl, B. and Schwartz, E. L. (1996b). Adaptive nonlocal filtering: A fast alternative to anisotropic diffusion for image enhancement. Technical Report CAS/CNS-TR-96-029, Boston University, Department of Cognitive and Neural Systems, 677 Beacon St., Boston MA.

Grossberg, S. and Mingolla, E. (1985). Neural dynamics of form perception: Boundary completion, illusory figures, and neon color spreading. *Psychological Review*, 92(2):173–211.

Haberman, R. (1987). *Elementary applied partial differential equations*. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, second edition.

Huang, T., Yang, G., and Tang, G. (1979). A fast two-dimensional median filtering algorithm. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 27:13–18.

Hummel, A. (1986). Representations based on zero-crossings in scale-space. In Fischler, M. and Firschein., O., editors, *Readings in Computer Vision: Issues, Problems, Principles and Paradigms*. Morgan Kaufmann, Los Angeles.

Koenderink, J. (1984). The structure of images. *Biological Cybernetics*, 50:363–370.

Marr, D. and Hildreth, E. (1980). Theory of edge detection. *Proceedings of the Royal Society of London*, B 207:187–217.

Nitzberg, M. and Shiota, T. (1992). Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):826–833.

Perona, P. and Malik, J. (1987). Scale space and edge detection using anisotropic diffusion. In *Proceedings of the IEEE Computer Society Workshop on Computer Vision*, pages 16–27, Miami, FL.

Perona, P. and Malik, J. (1990). Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639.

Rojer, A. S. and Schwartz, E. L. (1990). Design considerations for a space-variant visual sensor with complex-logarithmic geometry. In *10th International Conference on Pattern Recognition*, volume 2, pages 278–285.

ter Haar Romeny, B. M. (1994). *Geometry Driven Diffusion in Computer Vision*. Kluwer, Monterey, CA.

Werbos, P. J. (1974). *Beyond regression: new tools for prediction and analysis in the behavioral sciences*. Ph.D. thesis, Harvard University.

Whitaker, R. T. and Pizer, S. M. (1991). A multi-scale approach to nonuniform diffusion. *Computer Vision, Graphics and Image Processing*, 57:99–110.

Witken, A. (1983). Scale-space filtering. In *International Joint Conference on Artificial Intelligence*, pages 1019–1021, Karlsruhe, West Germany.