3-1-2005

Multiscale Segmentation with Vector-Valued Nonlinear Diffusions on Arbitrary Graphs

Xiaogang Dong
Ilya Pollak

Follow this and additional works at: http://docs.lib.purdue.edu/ecetr

http://docs.lib.purdue.edu/ecetr/3

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
MULTISCALE SEGMENTATION WITH VECTOR-VALUED NONLINEAR DIFFUSIONS ON ARBITRARY GRAPHS

XIAOGANG DONG
ILYA POLLAK

TR-ECE 05-01
MARCH 2005

PURDUE UNIVERSITY
SCHOOL OF ELECTRICAL
AND COMPUTER ENGINEERING
PURDUE UNIVERSITY
WEST LAFAYETTE, IN 47907-2035
Multiscale Segmentation with Vector-Valued Nonlinear Diffusions on Arbitrary Graphs.

Xiaogang Dong Ilya Pollak*

Abstract

We propose a novel family of nonlinear diffusion equations and apply it to the problem of segmentation of multi-valued images. We show that this family can be viewed as an extension of stabilized inverse diffusion equations (SIDEs) which were proposed for restoration, enhancement, and segmentation of scalar-valued signals and images in [39]. Our new diffusion equations can process vector-valued images defined on arbitrary graphs which makes them well suited for segmentation. In addition, we introduce novel ways of utilizing the shape information during the diffusion process. We demonstrate the effectiveness of our methods on a large number of texture segmentation tasks.

I. INTRODUCTION.

The use of partial differential equations in image processing has been developed over the last twenty years for various tasks such as restoration, reconstruction, and segmentation, among others (see, for example, [32], [37], [44], [45] and references therein). The input image is treated

This work was supported in part by the National Science Foundation (NSF) through CAREER award CCR-0093105 and through grant IIS-0329156, and the Army Research Office through Grant DAAH04-96-1-0445.

The authors are with the School of Electrical and Computer Engineering, Purdue University, 465 Northwestern Ave., West Lafayette, IN 47907, phone 765-494-3465 and 765-494-5916, fax 765-494-3358, e-mail dongx.ipollak@ecn.purdue.edu. Corresponding author’s e-mail: ipollak@ecn.purdue.edu.
as the initial data for a diffusion-like differential equation [24], [35], [56], [58]. The unknown
in this equation is usually a function of three variables: two spatial variables (one for each
image dimension) and \textit{scale}. This function of three variables is called the \textit{scale-space}, and
is alternatively viewed as a collection of 2D images, one image for every value of the scale
parameter.

The scale is sometimes also called \textit{time} because of the similarity of such equations to evolution
equations encountered in physics. In fact, one of the starting points of this line of investigation
was the observation [56], [58] that smoothing an image with Gaussians of varying width is
equivalent to solving the linear heat diffusion equation with the image as the initial condition.
Specifically, the solution to the heat equation at time $t$ is the convolution of its initial condition
with a Gaussian of variance $2t$. Gaussian filtering has been used both to remove noise and as a
pre-processor for edge detection procedures [9]. It has serious drawbacks, however: it displaces
and removes important image features, such as edges, corners, and T-junctions. The interpretation
of Gaussian filtering as a linear diffusion led to the design of other, nonlinear, evolution equations,
which better preserve these features [34], [35], [42], [44], [55]. For example, one motivation for
the work in [35] is achieving both noise removal and edge enhancement through the use of
an equation which in essence acts as an unstable inverse diffusion near edges and as a stable
linear-heat-equation-like diffusion in homogeneous regions without edges.

The anisotropic diffusions introduced in [15], [35] were the point of departure for the de-
velopment of stabilized inverse diffusion equations (SIDEs) in [39]. It was shown in [39] that
SIDEs may be viewed as a conceptually limiting case of the Perona-Malik diffusions. As shown
in [39], the scale-space of such an equation is a family of piecewise-constant approximations of
the original image, with larger values of the scale parameter $t$ corresponding to approximations
at coarser resolutions. Since the approximations are piecewise-constant, the scale-space can also
be viewed as a fine-to-coarse family of segmentations of the image, and the process of evolution can be viewed as a region-merging procedure whereby pairs of regions get recursively merged to result in progressively coarser segmentations. SIDEs are therefore naturally suited to the problem of image segmentation. Their effectiveness for the segmentation of grayscale images was shown in [39]. It was also experimentally demonstrated in [39] that SIDEs are robust to noise outliers and blurring, and their optimality for certain estimation problems was proved in [38], [47]. In addition, SIDEs lend themselves to faster algorithms than other evolution equations, since region merging reduces the dimensionality of the system during evolution.

Since they produce piecewise-constant approximations, however, the original SIDEs are not well suited to the problem of texture segmentation. This difficulty is common to many texture analysis problems, and is typically overcome with a preprocessing step which extracts features from a textured image [6], [10], [11], [14], [21], [22], [26]–[28], [43], [48], [49], [51]–[53]. The goal of feature extraction is to obtain data which is closer to being piecewise-constant than the original textured image. We use the output of a Gabor filter bank as the feature image, following a large body of literature (see, e.g., [6], [14], [21], [22], [26], [43], [48], [49], [51]) which has shown the effectiveness of Gabor features for texture analysis. If there are \( K \) filters in the Gabor filter bank, then each pixel of the feature image has \( K \) values. (We note here that other features can be used, and also that there exist many other situations which can give rise to vector-valued image data, such as color images or outputs from multiple sensors [23].) In Section II, we introduce vector-valued diffusions which can segment such vector-valued images. We propose a novel, flexible way of weighting the features during the evolution which allows us, for example, to weight fine-scale features more heavily at the beginning of the evolution—i.e., for small values of \( t \)—and to weight coarse-scale features more heavily during the later stages of the evolution.
We in addition develop a novel general way of introducing into our diffusion equations the information about the shapes of the image regions. This is done in Section II where our diffusion is interpreted as a gradient descent procedure for a certain energy functional. Both the structure of the underlying space where the gradient descent occurs, and the energy functional itself, are dependent on the image regions.

We illustrate our algorithm in Section III by applying it to segmenting noisy color images. Section IV provides a thorough experimental evaluation of our algorithm. We show that it outperforms several existing methods in a variety of texture segmentation tasks. We give several examples of segmentation of natural images in Section V.

II. SIDEs FOR VECTOR-VALUED IMAGES DEFINED ON AN ARBITRARY DOMAIN.

A. Scalar-Valued Diffusions on an Arbitrary Domain.

We define a real-valued image on an arbitrary finite set \( \mathcal{N} \) of points as any function which assigns a real number to every point in \( \mathcal{N} \). In our segmentation tasks, it is important to define adjacency relationships on the points in \( \mathcal{N} \), and therefore we assume that \( \mathcal{N} \) is the set of nodes of an undirected graph \( \mathcal{G} = (\mathcal{N}, \mathcal{L}) \) where the set \( \mathcal{L} \) of links consists of unordered pairs of distinct nodes. If \( \{m, n\} \) is a link, we say that the nodes \( m \) and \( n \) are neighbors. For example, \( \mathcal{G} \) could be a finite 2D rectangular grid where each node has four neighbors: east, west, north, and south, as in Fig. 1(a).

We let \( N \) be the total number of nodes and, without loss of generality, denote the nodes by the integers \( 1, 2, \ldots, N \), i.e., we assume that \( \mathcal{N} = \{1, 2, \ldots, N\} \). An image \( u \) can then be thought of as an \( N \)-dimensional vector: \( u = (u_1, \ldots, u_N) \in \mathbb{R}^N \). We moreover use \( u(t) \) to denote a parametric family of images defined on the set \( \mathcal{N} \) for all values of a continuous-valued nonnegative scale parameter \( t \), and we call the collection of images \( \{u(t)\}_{t=0}^{\infty} \) a scale-space.
To describe the specific scale-space that we use in this paper, we need the following further definitions. We say that a set of nodes $R \subset \mathcal{N}$ is a region if, for any two nodes $m, n \in R$, there exists a path between $m$ and $n$—i.e., a sequence of links of the form $\{m, m_1\}, \{m_1, m_2\}, \ldots, \{m_k-1, m_k\}, \{m_k, n\}$. Two disjoint regions $R_1, R_2$ are called neighbors if there exist two nodes $m \in R_1, n \in R_2$ which are neighbors, i.e., such that $\{m, n\} \in \mathcal{L}$. Given a partition
$S = \{ R_1, \ldots, R_I \}$ of the set $\mathcal{N}$ into $I$ disjoint regions, we let NBR-PRS be the set of all pairs of neighbor regions in $S$, and we let NBRS($R_i$) be the set of all regions in $S$ that are neighbors of a region $R_i \in S$. For example, a partition $S$ consisting of three regions is shown in Fig. 1(a). In this example, each region is a neighbor of the other two.

For any partition $S = \{ R_1, \ldots, R_I \}$ of $\mathcal{N}$, we let $U_S$ be the set of all piecewise constant images which are constant over each region $R_i \in S$. One such image, for the partition in Fig. 1(a), is shown in Fig. 1(b). We use $\mu_i(u)$ to denote the intensity of any such image $u$ within region $R_i$. Note that $U_S$ is a vector space. To impose a metric on this space, we define the following inner product.

$$\langle u, v \rangle \triangleq \sum_{i=1}^{I} a(R_i) \mu_i(u) \mu_i(v),$$

(1)

where $a(R_i)$ is a positive weight function which enables us to weight the contributions of various regions differently. For example, $a(R_i)$ may encode some information about the shape (or size, or location) of $R_i$.

We consider functionals $\mathcal{E}$ defined for images in $U_S$ which have the following form:

$$\mathcal{E}(u) = \sum_{\{R_i, R_j\} \in \text{NBR-PRS}} b(R_i, R_j) E(|\mu_i(u) - \mu_j(u)|),$$

(2)

where $b(R_i, R_j)$ is a positive weight function with $b(R_i, R_j) = b(R_j, R_i)$, and $E$ is such that $E'(x) > 0$ for $x > 0$ and with $E(0) = 0$. The function $b(R_i, R_j)$ allows us to assign different weights to different pairs of regions, according to, for example, the shape or length or the boundary between $R_i$ and $R_j$.

Given an image $u(0) \in U_S$, we generate a scale-space $u(t)$ by solving the following gradient descent procedure for $t > 0$:

$$\dot{u}(t) = -\nabla \mathcal{E}(u(t)),$$

(3)

where $\dot{u}(t)$ is the derivative of $u(t)$ with respect to the scale parameter $t$, and $\nabla$ stands for the
gradient in the space $U_S$ equipped with the inner product (1). We now show how to implement this descent equation.

**Lemma 1.** For $i = 1, \ldots, I$, let the image $e_i \in U_S$ be the indicator function of the region $R_i$ multiplied by $1/\sqrt{a(R_i)}$, i.e., let

$$
\mu_j(e_i) = \begin{cases} 
\frac{1}{\sqrt{a(R_i)}} & \text{if } j = i \\
0 & \text{if } j \neq i
\end{cases}
$$

for $i, j = 1, \ldots, I$.

These images then constitute an orthonormal basis for $U_S$.

**Proof.** Using Eq. (1), it is easily shown that these $I$ images are orthonormal. Since $U_S$ is an $I$-dimensional space, they form an orthonormal basis for $U_S$. □

The three indicator functions for the regions of Fig. 1(a) are shown in Figs. 1(c-e). They form an orthogonal basis for the space $U_S$ defined by the partition $S$ of Fig. 1(a) (and if $a(R_i) = 1$ then this basis is orthonormal).

**Proposition 1.** The gradient descent procedure (3) can be equivalently written\(^1\) as follows:

$$
\dot{\mu}_i = \frac{1}{a(R_i)} \sum_{R_j \in \text{NBRS}(R_i)} b(R_i, R_j) \frac{\mu_j - \mu_i}{|\mu_j - \mu_i|} E'(||\mu_j - \mu_i||), \text{ for } i = 1, \ldots, I,
$$

where $E'$ is the derivative of $E$.

**Proof.** Note that any image $u(t) \in U_S$ can be written in terms of the basis vectors $e_1, \ldots, e_I$ defined in Lemma 1 and illustrated in Fig. 1(c-e), as follows:

$$
u(t) = \sum_{i=1}^{I} \sqrt{a(R_i)} \mu_i(u(t)) e_i.
$$

Let $c_i$ be the $i$-th coefficient in this expansion:

$$
c_i \triangleq \sqrt{a(R_i)} \mu_i.
$$

\(^1\)When there is no possibility of confusion, we abbreviate $\mu_i(u(t))$ as $\mu_i$. 

In these coordinates, the gradient descent is:

\[
\dot{c}_i = -\frac{\partial E}{\partial c_i} = -\frac{\partial}{\partial c_i} \left\{ \sum_{\{R_i, R_j\} \in \text{NBR-PRS}} b(R_i, R_j) \ E \left( \left| \frac{1}{\sqrt{a(R_i)}} c_i - \frac{1}{\sqrt{a(R_j)}} c_j \right| \right) \right\}
\]

\[
= \sum_{R_j \in \text{NBRS}(R_i)} \frac{b(R_i, R_j)}{\sqrt{a(R_i)}} \cdot \frac{c_i/\sqrt{a(R_i)} - c_j/\sqrt{a(R_j)}}{|c_i/\sqrt{a(R_i)} - c_j/\sqrt{a(R_j)}|} \cdot E'(|c_i/\sqrt{a(R_i)} - c_j/\sqrt{a(R_j)|}).
\]

Rewriting this in terms of \(\mu_i\)'s, we get:

\[
\sqrt{a(R_i)} \mu_i = \sum_{R_j \in \text{NBRS}(R_i)} \frac{b(R_i, R_j)}{\sqrt{a(R_i)}} \cdot \frac{\mu_j - \mu_i}{|\mu_j - \mu_i|} \cdot E'(|\mu_j - \mu_i|),
\]

which is the same as Eq. (4).

**B. Vector-Valued SIDEs on an Arbitrary Domain.**

We have assumed so far that the image intensities are scalars. We now generalize our evolution equations (3,4) to the case when each image intensity is a \(K\)-dimensional vector. In this case, an image \(u\) can be thought of as an \(N \times K\)-dimensional vector: \(u = (\vec{u}_1, \ldots, \vec{u}_N) \in \mathbb{R}^{N \times K}\), where each intensity vector \(\vec{u}_n = (u_{n,1}, \ldots, u_{n,K})\) is in \(\mathbb{R}^K\). An intensity vector can, for example, be the vector of red, green, and blue intensities for a color image, or correspond to \(K\) features extracted from a texture image. Each entry of an intensity vector is called a component.

To define a metric on the space \(\mathbb{R}^K\) of all intensity vectors, we use the following inner product between two intensity vectors \(\vec{u}_m\) and \(\vec{u}_n\):

\[
\langle \vec{u}_m, \vec{u}_n \rangle_w \triangleq \sum_{k=1}^{K} w_k u_{m,k} u_{n,k},
\]

where the positive weights \(w = (w_1, \ldots, w_K)\) allow us to assign different relative importance to different components of an image. We denote the norm corresponding to this inner product by \(\| \cdot \|_w\).

As previously, we fix a partition \(S = \{R_1, \ldots, R_I\}\) of the set \(\mathcal{N}\) of nodes, and let \(U_S\) be the set of all piecewise constant images which are constant over each region \(R_i\) in the partition. We now use \(\vec{\mu}_i(u)\) to denote the vector intensity of any such image \(u\) within region \(R_i\). We define
an inner product on the space $U_S$ by generalizing our definition (1) as follows:

$$\langle u, v \rangle \triangleq \sum_{i=1}^{I} a(R_i) \langle \tilde{\mu}_i(u), \tilde{\mu}_i(v) \rangle_w,$$

where $\langle \cdot \rangle_w$ is the inner product for the intensity vectors defined in Eq. (5).

We again generate a scale-space of images in $U_S$, starting with a given image $u(0) \in U_S$ and using the descent procedure (3) where the functional $E$ is the following generalization of Eq. (2):

$$E(u) = \sum_{\{R_i, R_j\} \in \text{NBR-PRS}} b(R_i, R_j) E(\|\tilde{\mu}_i(u) - \tilde{\mu}_j(u)\|_w).$$

**Proposition 2.** The gradient descent procedure (3) for vector-valued images can be equivalently written as follows:

$$\dot{\tilde{\mu}}_i = \frac{1}{a(R_i)} \sum_{R_j \in \text{NBR}(R_i)} b(R_i, R_j) \frac{\tilde{\mu}_j - \tilde{\mu}_i}{\|\tilde{\mu}_j - \tilde{\mu}_i\|_w} E'(\|\tilde{\mu}_j - \tilde{\mu}_i\|_w), \text{ for } i = 1, \ldots, I.$$ 

**Proof.** Similar to the proof of Proposition 1.

In order to make the evolution (8) an effective segmentation procedure, we use a SIDE energy function $E$ [19], [39], [59]—i.e., a function $E$ such that $E''(x) < 0$ for $x > 0$, as illustrated in Fig. 2(a). In addition, we impose that $\lim_{x \to 0} E'(x) = +\infty$, as shown in Fig. 2(b). For example, $E(x) = \sqrt{x}$ has the desired properties. With this choice of the function $E$, Eq. (8) encourages the merging of pairs of neighbor regions—in other words, the solution of Eq. (8) is attracted to subspaces of $U_S$ which have the form $U_{i,j} = \{v : \tilde{\mu}_i(v) = \tilde{\mu}_j(v)\}$ where $\{R_i, R_j\} \in \text{NBR-PRS}$. 

![Fig. 2. (a) The SIDE energy function $E$. (b) Its derivative $E'$.](image-url)
To see this, note that, if the solution $u(t)$ is very close to one such space $U_{i,j}$ and very far from all other spaces $U_{i,j}$ then $E'(\|\vec{\mu}_j - \vec{\mu}_i\|_w) >> E'(\|\vec{\mu}_j - \vec{\mu}_i\|_w)$, and therefore the dynamics for both $\vec{\mu}_i$ and $\vec{\mu}_j$ will in this case be dominated by the term involving $E'(\|\vec{\mu}_j - \vec{\mu}_i\|_w)$. Denoting $\vec{\nu}_{ij} = \vec{\mu}_j - \vec{\mu}_i$, we then have, from Eq. (8):

$$\dot{\vec{\nu}}_{ij} = -\left(\frac{1}{a(R_i)} + \frac{1}{a(R_j)}\right) b(R_i, R_j) \frac{\vec{\nu}_{ij}}{\|\vec{\nu}_{ij}\|_w} E'(\|\vec{\nu}_{ij}\|_w),$$

which can be shown to imply

$$\dot{x} = -CE'(x),$$

where $x = \|\vec{\nu}_{ij}\|_w$ and $C = \left(\frac{1}{a(R_i)} + \frac{1}{a(R_j)}\right) b(R_i, R_j) > 0$. Since $E'(x) > 0$, we have that $\dot{x} < 0$, and since $E'' < 0$ we have that

$$\ddot{x} = -CE''(x)\dot{x} = C^2 E''(x) E'(x) < 0,$$

which means that $x$ will reach zero in finite time. When this happens, $\mu_i = \mu_j$, and we effectively have a partition which is coarser than the partition $S$ since now $R_i \cup R_j$ can be viewed as a single region. This motivates the following multiscale region-merging segmentation algorithm.

1. Set $t_{init} \leftarrow 0$.

2. Given a partition $S$ of $N$ and an image $u(t_{init}) \in U_S$, evolve Eq. (8) for $t > t_{init}$ until the time $t_{merge}$ when $\vec{\mu}_i(u(t_{merge})) = \vec{\mu}_j(u(t_{merge}))$ (numerically, when $\|\vec{\mu}_i(u(t_{merge})) - \vec{\mu}_j(u(t_{merge}))\|_w < \varepsilon$ where $\varepsilon$ is a small parameter) for some pair of neighbor regions $\{R_i, R_j\}$.

3. Remove regions $R_i$ and $R_j$ from $S$ and add region $R_i \cup R_j$:

$$S \leftarrow S\{R_i, R_j\} \cup \{R_i \cup R_j\}.$$  

4. If the desired number of regions is reached, stop. Else, define the values of the functions $a$ and $b$ for the newly formed region, adjust the weights $w_k$ (if desired), assign $t_{init} \leftarrow t_{merge}$, and go to Step 2.
III. SEGMENTATION OF COLOR IMAGES.

We first illustrate our segmentation algorithm by applying it to color images which are viewed as $\mathbb{R}^3$-valued images with red, green, and blue components. In these experiments, we use $E(x) = \sqrt{x}$, set $a(R_i)$ to be the area $|R_i|$ of region $R_i$ (i.e., the number of nodes in $R_i$), initialize each pixel to be a separate region, and use an eight-neighbor model where the neighbors of pixel $n$ are the eight pixels surrounding $n$. We set $b(R_i, R_j)$ to be the length of the boundary between $R_i$ and $R_j$—i.e., the number of such links $\{m, n\}$ that $m \in R_1$ and $n \in R_2$.

A. Characterizing the Performance of a Segmentation Algorithm.

If the true region is $R_i$ whereas the region extracted by our algorithm is $\hat{R}_i$, we define the mismatch $M(R_i, \hat{R}_i)$ between the true and estimated regions as the set of all nodes which are in one of these regions but not in their intersection:

$$M(R_i, \hat{R}_i) := (R_i \setminus \hat{R}_i) \cup (\hat{R}_i \setminus R_i).$$

We assume in all our experiments that the target number of regions $I$ is given. The mismatch between the ground truth partition $S = \{R_1, \ldots, R_I\}$ and the partition $\hat{S} = \{\hat{R}_1, \ldots, \hat{R}_I\}$ extracted by our algorithm is defined for any permutation $\pi$ of the indexes $1, \ldots, I$ as follows:

$$\text{MISMATCH}(S, \hat{S}, \pi) := \bigcup_{i=1}^{I} M(R_i, \hat{R}_{\pi(i)}).$$

To characterize the performance of our segmentation algorithm, we choose the best match as follows:

$$\pi^* := \arg \min_\pi \left| \text{MISMATCH}(S, \hat{S}, \pi) \right|.$$

Then we define

$$\text{ERROR-SET}(S, \hat{S}) := \text{MISMATCH}(S, \hat{S}, \pi^*),$$

$$\text{ERROR-PERCENTAGE}(S, \hat{S}) := \frac{|\text{ERROR-SET}(S, \hat{S})|}{|N|} \cdot 100\%,$$
i.e., the error percentage is defined as the ratio (in percent) of the number of mismatched pixels to the total number of pixels.

B. Experiment 1: A Simple Shape.

We start by applying our vector-valued SIDE to the color image in Fig. 3. The image in Fig. 3(a) consists of two regions: two of its three color components undergo an abrupt change at the boundary between the regions. More precisely, the \{red, green, blue\} component values are \{0.1, 0.6, 0.9\} for the background and \{0.6, 0.6, 0.5\} for the square. Each component is corrupted with independent white Gaussian noise whose standard deviation is 0.4. The resulting image (normalized in order to make every pixel of every component be between 0 and 1) is shown in Fig. 3(b). We evolve our vector-valued SIDE on the noisy image, until exactly two regions remain. The final boundary, superimposed onto the initial image, is depicted in Fig. 3(c). The algorithm is very accurate in locating the boundary: the error set occupies less than 0.5\% of the pixels for this 100 × 100 image.
Fig. 4. (a) A test image; (b) its noisy version (normalized); (c) detected boundary, superimposed onto the noise-free image.

C. Experiment 2: A Complicated Shape.

A similar experiment, with the same level of noise, is conducted for a more complicated shape, whose image is in Fig. 4(a). The result of processing the noisy image of Fig. 4(b) is shown in Fig. 4(c). In this $100 \times 100$ image, 0.7% of the pixels are errors.

IV. Texture Segmentation.

A. Feature Extraction.

The use of Gabor features [13], [17] is a well established strategy in the texture analysis literature [6], [14], [21], [22], [26], [43], [48], [49], [51]. We adopt Gabor energy features (GEF) from [26], [48], [51]. These features depend on three parameters: orientation $\theta$, scale $\sigma$, and frequency $\omega_0$, which we abbreviate as $\rho \triangleq (\theta, \sigma, \omega_0)$. Given an image $f(x, y)$, the GEF $u(x, y; \rho)$ at orientation $\theta$, scale $\sigma$, and frequency $\omega_0$ is calculated by filtering the image with a pair of so-called GEF filters $h_c(x, y; \rho)$ and $h_s(x, y; \rho)$ to obtain the filtered images $g_c(x, y; \rho)$ and $g_s(x, y; \rho)$, respectively. These filtered images are then combined via:

$$u(x, y; \rho) = \sqrt{g_c(x, y; \rho)^2 + g_s(x, y; \rho)^2}.$$  \hfill (9)
In our experiments, the GEF filter pair is the following pair of Gabor filters with quadrature phase relation [26], [48], [51]:

\[ h_c(x, y; \rho) = e^{-0.5\sigma^{-2}(\bar{x}^2 + \bar{y}^2)} \cos (\omega_0 \bar{x}), \]

\[ h_s(x, y; \rho) = e^{-0.5\sigma^{-2}(\bar{x}^2 + \bar{y}^2)} \sin (\omega_0 \bar{x}), \]

where

\[ \bar{x} = x \cos \theta + y \sin \theta, \]

\[ \bar{y} = -x \sin \theta + y \cos \theta. \]

Our filtering calculations are done in the frequency domain, using the frequency responses of these two filters:

\[ H_c(\omega_x, \omega_y; \rho) = \pi \sigma^2 \left\{ e^{-0.5\sigma^2[\tilde{\omega}_x + \omega_0]^2 + \tilde{\omega}_y^2} + e^{-0.5\sigma^2[(\tilde{\omega}_x - \omega_0)^2 + \tilde{\omega}_y^2]} \right\}, \]

\[ H_s(\omega_x, \omega_y; \rho) = \pi \sigma^2 \sqrt{-1} \left\{ e^{-0.5\sigma^2[\tilde{\omega}_x + \omega_0]^2 + \tilde{\omega}_y^2} - e^{-0.5\sigma^2[(\tilde{\omega}_x - \omega_0)^2 + \tilde{\omega}_y^2]} \right\}, \]

where

\[ \tilde{\omega}_x = \omega_x \cos \theta + \omega_y \sin \theta, \]

\[ \tilde{\omega}_y = -\omega_x \sin \theta + \omega_y \cos \theta. \]

Note that \( H_c(\omega_x, \omega_y; \rho) \) is a real-valued, even function and \( H_s(\omega_x, \omega_y; \rho) \) is a purely imaginary-valued, odd function. Both these filters are bandpass filters centered at \((\omega_0 \cos \theta, \omega_0 \sin \theta)\) in the frequency domain.

Fig. 5 shows the intensity map of a pair of Gabor filters in the space and frequency domains. The even Gabor filter \( h_c(x, y; \rho) \) is presented in Fig. 5(a), and the corresponding odd Gabor filter \( h_s(x, y; \rho) \) is in Fig. 5(c). Fig. 5(b) shows the real part of \( H_c(\omega_x, \omega_y; \rho) \) since it is a real function in the frequency domain. Similarly, Fig. 5(d) shows the imaginary part of \( H_s(\omega_x, \omega_y; \rho) \).

Feature selection for texture analysis is an active research area. For example, features based on
wavelets [10]–[12], [33], [43], [52], [54] and wavelet packets [1], [7], [27] have been proposed. We emphasize that while we use the GEF features in the texture segmentation experiments presented in this paper, our vector-valued diffusion algorithm of Section II can be used in conjunction with any set of features.

B. Parameter Settings.

In order to accurately analyze a repetitive structure such as a texture patch, data points are needed which cover at least a few periods. We therefore assume that in our texture segmentation tasks, the areas of the regions cannot be very small, and design the evolution (8) to encourage the formation of large regions. In all the experiments, the initial segmentation takes every pixel to be a separate region. We assume that the correct number of regions is known, and stop the evolution when this number of regions is reached. At the beginning of the evolution, when most regions are small, we use fine-scale Gabor features. We use coarse-scale Gabor features towards the end of the evolution since, as small regions aggregate into larger ones, coarse-scale features become more informative. In addition, the existence of only large regions towards the end of the evolution allows us to build histograms of Gabor filter outputs within each region as suggested in, for example, [21]. These design considerations motivate the following choices for the parameters.
• To encourage relatively rapid evolution of small regions, we use $a(R_i) = |R_i| + (|R_i|/100)^3$
where the area $|R_i|$ of $R_i$ is the number of nodes in $R_i$. For small regions, $a(R_i) \approx |R_i|$;
however, for large $|R_i|$ the cubic term dominates and slows down the evolution of $\vec{\mu}_i$.

• We evolve Eq. (8) in two stages. Each stage uses a different set of features. In stage 1, we
use eight-dimensional feature vectors consisting of Gabor features at a fine scale $\sigma = 1.5$
and eight orientations. (When selecting the scales and frequencies, we use a procedure
similar to [22] in order to obtain a good coverage of the frequency plane.) In stage 2, we
use 325-dimensional feature vectors constructed as follows. For every region, we take 25-
bin histograms of 12 Gabor features (three scales $\sigma = 1.5, 3, 6$, four orientations per scale),
and in addition we take a 25-bin histogram of the original grayscale image.\footnote{Note that this two-stage procedure is equivalent to running both stages with all 333 features, but setting the weights $w_k$ to zero during each stage for the features that are not used during that stage.}
We stop stage 1 and start stage 2 as soon as the area of every region is at least 500 pixels.

• As in the previous section, we set $E(x) = \sqrt{x}$ and let $b(R_i, R_j)$ be the length of the
boundary between $R_i$ and $R_j$.

We have tried many different parameter settings and found that our segmentation results are not
very sensitive to changes of the parameters.

\textit{C. Experiment 1: Two Textures, Straight Boundary.}

We form 90 two-texture test images using all pairs of ten Brodatz textures [8].\footnote{The Brodatz texture images were obtained from www.ux.his.no/~tranden/brodatz.html.} Each test
image is obtained by concatenating two $256 \times 256$ texture images. Our nonlinear diffusion is
evolved until two regions remain. The results in Table I and in the top two rows of Fig. 6 show
that our segmentations are very accurate, with only 0.6% average error.
D. Experiment 2: Two Textures, Random Boundary.

In the second experiment, 100 test images are generated using two Brodatz textures, D4 and D84, separated by random boundaries which are 100 independent realizations of a correlated Gaussian random process. Examples of such boundaries are shown in the two bottom panels of Fig. 6(c), and the corresponding test images are shown in Fig. 6(a). The results are summarized in Table I and show that, again, our algorithm produces consistently reliable segmentations, with only 1.2% average error. In this experiment, it is not easy to visually discern the boundary; the boundaries extracted by our algorithm (Fig. 6(b)), however, are very close to the actual ones. The mismatches between the extracted boundary and the actual one are shown in Fig. 6(d).
<table>
<thead>
<tr>
<th></th>
<th>Min</th>
<th>Max</th>
<th>Avg</th>
<th>Std</th>
</tr>
</thead>
<tbody>
<tr>
<td>Experiment 1 (straight boundary)</td>
<td>0.2%</td>
<td>3.6%</td>
<td>0.6%</td>
<td>0.4%</td>
</tr>
<tr>
<td>Experiment 2 (random boundary)</td>
<td>0.4%</td>
<td>4.4%</td>
<td>1.2%</td>
<td>0.6%</td>
</tr>
</tbody>
</table>

**TABLE I: Statistics of the errors (in percentages of the total number of pixels) in the two-texture segmentation experiments.**

**E. Experiment 3: Comparison to [21].**

The database used in [21] contains 100 512×512 five-texture images two of which are shown in the left column of Fig. 7(a). The corresponding segmentations produced by our algorithm are shown in Fig. 7(b), and the mismatch between the edges found by our algorithm and the actual edges is in Fig. 7(d). The median error percentage on this database reported in [21] is 2.65%; our median error percentage is 1.48%. The mean error percentage, estimated from the histogram given in [21], is about 5%; our mean error percentage is 2.23%. The histogram of the error percentage in Fig. 8(a) shows that, for most images, our algorithm’s error percentage is below 5%; very few images result in large errors.

We note that the algorithm of [21] operates on 8×8 image blocks rather than on pixels. If we convert our segmentation results to 8×8 blocks by assigning every block to a single region according to the majority of the pixels in the block, our median and mean error percentages go up to 2.21% and 2.79%, respectively.

**F. Experiment 4: Comparison to [43].**

There are 12 different test images in [43] which are used to evaluate many texture segmentation algorithms. For each algorithm and each test image, the segmentation in [43] is obtained
using a supervised classifier trained with training samples from every texture present in the test image. We note here that our segmentation algorithm is unsupervised and is used with the same parameter settings for all our texture segmentation experiments. The two algorithms from [43] whose average error percentages are the lowest, are compared in Table II with our algorithm. For each image, we also provide the mean and the minimum error percentage over all algorithms evaluated in [43]. The average error percentage for our algorithm (rightmost column) is much lower than the best average percentage for [43]. In addition, our algorithm performs well on some images where all algorithms in [43] have very high error percentages. For example, our algorithm’s error percentage for the image shown the top row of Fig. 9 (taken from Fig. 11(f) of [43]) is 5.8%, compared to 35% for the best algorithm in [43].

Segmentation results for three of the 12 images are shown in Fig. 9.

---

6Note that the corrected data for [43]’s Fig. 11(i) comes from www.ux.his.no/~tranden/comments.html.
Fig. 8. (a) The empirical distribution of the percentage of error pixels for the 100 test images from [21]; (b) Comparison of the percentage of error blocks for our algorithm and the algorithm from [21].

Fig. 9. Our segmentation results for images from [43]. Top: Fig. 11(f) from [43]; middle: Fig. 11(h) from [43]; bottom: Fig. 11(i) from [43].
TABLE II: The comparison of error percentages for 12 test images.

<table>
<thead>
<tr>
<th>Number of textures</th>
<th>Two</th>
<th>Five</th>
<th>Ten</th>
<th>Sixteen</th>
</tr>
</thead>
<tbody>
<tr>
<td>Our algorithm</td>
<td>0.5</td>
<td>0.8</td>
<td>4.1</td>
<td>11</td>
</tr>
<tr>
<td>Algorithm f8a(d) in [43]</td>
<td>6.5</td>
<td>7.2</td>
<td>8.7</td>
<td>33</td>
</tr>
<tr>
<td>Algorithm f16b(d) in [43]</td>
<td>8.1</td>
<td>8.2</td>
<td>9.2</td>
<td>35</td>
</tr>
<tr>
<td>Best over all methods in [43]</td>
<td>0.7</td>
<td>2.5</td>
<td>7.2</td>
<td>33</td>
</tr>
<tr>
<td>Mean over all methods in [43]</td>
<td>15</td>
<td>2.8</td>
<td>11</td>
<td>7.3</td>
</tr>
</tbody>
</table>

V. SEGMENTATION OF NATURAL IMAGES.

Segmentation of natural images is important since it is often used as the first stage of image analysis algorithms for various tasks such as database organization and retrieval, classification, detection and recognition of objects in images, compression, etc. Fig. 10 illustrates the performance of our algorithm on several images from the Berkeley segmentation dataset [3], [30]. While precise quantitative evaluation of the performance of our algorithm on such images is beyond the scope of this paper (indeed, analyzing the performance of a segmentation algorithm on a natural image is a challenging open problem [30], [31], [41]), note that these segmentations are comparable to the ones produced by recent algorithms such as [1], [18], [29], [33], [36], [40]. For example, comparing to the results reported in [40], note that our algorithm captures the outline of the small birds in the center of the top left image while the algorithm in [40] does not; in the church image, our algorithm accurately captures the outlines of the two crosses while the algorithm in [40] does not. Our segmentations of the leopard and bear images are very similar to those in [40]; on the other hand, in the deer image the algorithm in [40] is able to segment the legs of the small deer while our algorithm is not.
Fig. 10. Segmentation of natural images from the Berkeley segmentation dataset [3], [30].
VI. CONCLUSIONS, DISCUSSION, AND FUTURE RESEARCH.

We proposed a novel family of vector-valued nonlinear diffusion equations for images on arbitrary graphs and applied it to the problems of color and texture segmentation. We demonstrated the effectiveness of our method by comparing it to other algorithms on a large number of texture segmentation tasks. We also illustrated our algorithm by segmenting a number of natural images.

Our method provides a systematic way of feature-based segmentation based on minimizing an energy functional. The information about the structure of the desired regions and their boundaries can be built into the formulation. In addition, there is flexibility in utilizing various features: different features can be emphasized at different stages of the procedure.

Recent literature on segmentation comes in four broad (and overlapping) categories: graph cut methods [18], [33], [46], [57], statistical methods [5], [7], [11], [12], PDE-based methods [1], [2], [36], [50], and region merging methods [4], [16], [18], [20], [25], [29], [36], [60]. Our framework falls into the last two categories; however, it is different from previously proposed approaches in several important respects. Most recent PDE-based approaches relate image edges to level sets of a (smooth) solution to a PDE. Our system, on the other hand, evolves the image itself, and explicitly forms regions. This allows for a great flexibility in using any information about the sizes, shapes, locations, and boundaries of regions to help guide the evolution. Unlike the traditional region merging methods, however, the fact that our process is the solution to a system of ordinary differential equations ensures certain stability properties, such as robustness to noise and low sensitivity to parameter changes, as shown in [38], [39].

Possible future research questions include probabilistic analysis (to extend the work in [38]), the development of systematic training methods for parameter selection, the design of fast
approximate numerical schemes, and the exploration of different features.

VII. ACKNOWLEDGMENTS.

We would like to thank Profs. Alan Willsky, Hamid Krim, and David Mumford for many helpful discussions during the early stages of this work.

REFERENCES


