

Segmentation of Color, Texture, and Orientation Images

THE preceding chapters were all devoted to the analysis of images and signals which take values in \mathbb{R} . It is often necessary, however, to process vector-valued images where each pixel value is a vector belonging to \mathbb{R}^M , with $M \geq 1$. The entries of this vector could correspond to red, green, and blue intensity values in color images [74], to data gathered from several sensors [33] or imaging modalities [68], or to the entries of a feature vector obtained from analyzing a texture image [10]. In the next section, we generalize our SIDEs to vector-valued images and argue that most properties of scalar-valued SIDEs still apply. We then give several examples of segmentation of color and texture images.

Section 5.2 treats images whose every pixel belongs to a circle \mathbb{S}^1 . Such images arise in the analysis of orientation [48] and optical flow [30].

■ 5.1 Vector-Valued Images.

We use one of the properties discussed in Chapter 3 in order to generalize our evolution equations to vector-valued images. We recall that a SIDE is the gradient descent for the global energy (2.8), which we re-write as follows:

$$\mathcal{E} = \sum_{i,j \text{ are neighbors}} E(\|u_j - u_i\|),$$

where E is a SIDE energy function (Figure 2.6). The norm $\|\cdot\|$ here stands simply for the absolute value of its scalar argument. Now notice that we still can use the above equation if the image under consideration is vector-valued, by interpreting $\|\cdot\|$ as the ℓ^2 norm. To remind ourselves that the pixel values of vector-valued images are vectors, we will use arrows to denote them:

$$\mathcal{E} = \sum_{i,j \text{ are neighbors}} E(\|\vec{u}_j - \vec{u}_i\|), \tag{5.1}$$

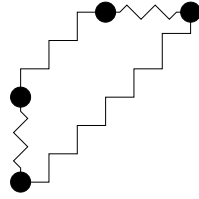


Figure 5.1. Spring-mass model for vector-valued diffusions. This figure shows a 2-by-2 image whose pixels are two-vectors: $(2,2)$, $(0,0)$, $(0,1)$, and $(1,2)$. The pixel values are depicted, with each pixel connected by springs to its neighboring pixels.

where

$$\begin{aligned}\vec{u}_i &= (u_{i,1}, \dots, u_{i,M})^T \in \mathbb{R}^M, \\ \vec{u}_j &= (u_{j,1}, \dots, u_{j,M})^T \in \mathbb{R}^M, \text{ and} \\ \|\vec{u}_j - \vec{u}_i\| &= \sqrt{\sum_{k=1}^M (u_{j,k} - u_{i,k})^2}.\end{aligned}$$

We will call the collection of the k -th entries of these vectors the k -th *channel* of the image. In this case, the gradient descent equation is:

$$\dot{\vec{u}}_i = \frac{1}{m_i} \sum_{j \in A_i} \frac{\vec{u}_j - \vec{u}_i}{\|\vec{u}_j - \vec{u}_i\|} F(\|\vec{u}_j - \vec{u}_i\|) p_{ij}. \quad (5.2)$$

(This notation combines M equations—one for each channel—in a single vector equation). Just as for the scalar images, we merge two neighboring pixels at locations i and j when their values become equal: $\vec{u}_j(t) = \vec{u}_i(t)$. Just as in Chapter 3, m_i and A_i are the area of the i -th region and the set of its neighbors, respectively. The length of the boundary between regions i and j is p_{ij} .

A slight modification of the spring-mass models of Chapter 3, Figures 3.1 and 3.3, can be used to visualize this evolution. We recall that those models consisted of particles forced to move along straight lines, and connected by springs to their neighbors. If we replace each straight line with an M -dimensional space, we will get the model corresponding to the vector-valued equation, with pixel values in \mathbb{R}^M . For example, when $M = 2$, the particles are forced to move in 2-D planes. Another way to visualize the system is by depicting all the particles as points in a single M -dimensional space, as shown in Figure 5.1. Each pixel value $\vec{u}_i = (u_{i,1}, u_{i,2})^T$ is depicted in Figure 5.1 as a particle whose coordinates are $u_{i,1}$ and $u_{i,2}$. Each particle is connected by springs to its neighbors. The spring whose length is v exerts a force whose absolute value is $F(v)$, and which is directed parallel to the spring.

By using techniques similar to those in Chapter 3, it can be verified that Equation (5.2) inherits many useful properties of the scalar equation. Namely, the conservation of mean and the local maximum principle hold for each channel; the equation reaches

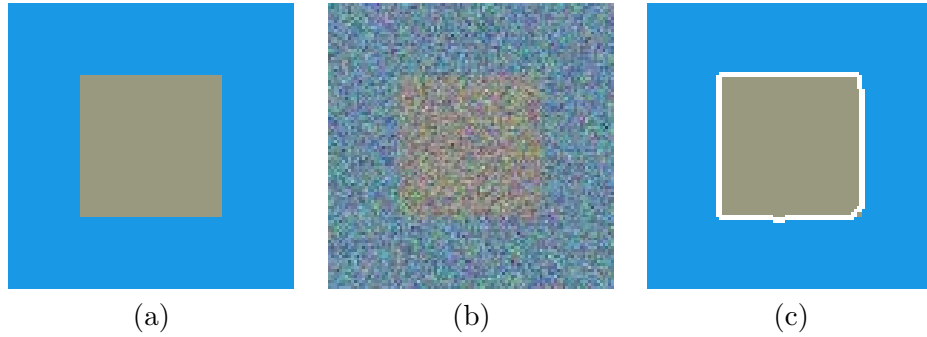


Figure 5.2. (a) A test image; (b) its noisy version (normalized); (c) detected boundary, superimposed onto the noise-free image

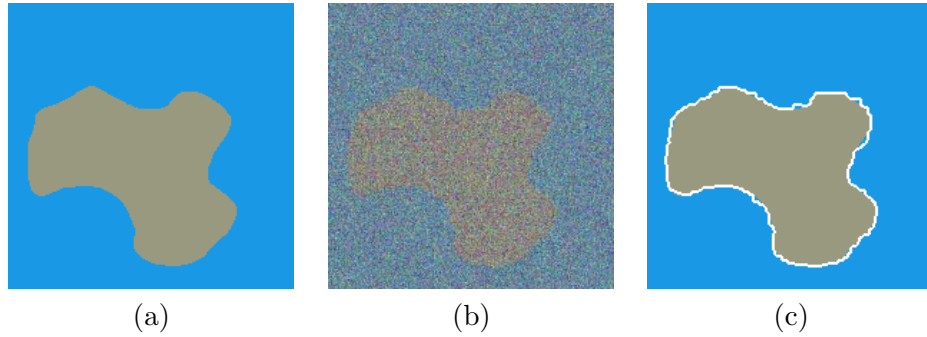


Figure 5.3. (a) A test image; (b) its noisy version (normalized); (c) detected boundary, superimposed onto the noise-free image

the steady state in finite time and has the energy dissipation properties described in Subsection 3.4.2. For usual SIDE force functions, there is no sliding for vector images—just as there was no sliding for scalar images. However, for force functions which are infinite at zero, such as that of Figure 3.8, the sliding property holds, and therefore so does well-posedness. Vector-valued SIDEs are also robust to severe noise, as we show in the experiments.

■ 5.1.1 Experiment 1: Color Images.

We start by applying a vector-valued SIDE to the color image in Figure 5.2. The image in Figure 5.2, (a) consists of two regions: two of its three color channels undergo an abrupt change at the boundary between the regions. More precisely, the {red, green, blue} channel values are {0.1, 0.6, 0.9} for the background and {0.6, 0.6, 0.5} for the square. Each channel 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 channel be between 0 and 1) is shown in Figure 5.2, (b). We evolve a vector-valued SIDE on the noisy image, until exactly two regions remain. The final boundary,

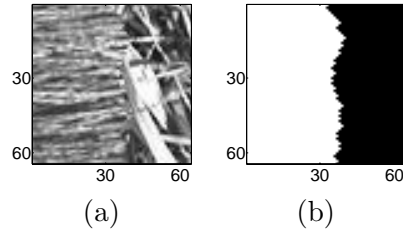


Figure 5.4. (a) Image of two textures: fabric (left) and grass (right); (b) the ideal segmentation of the image in (a).

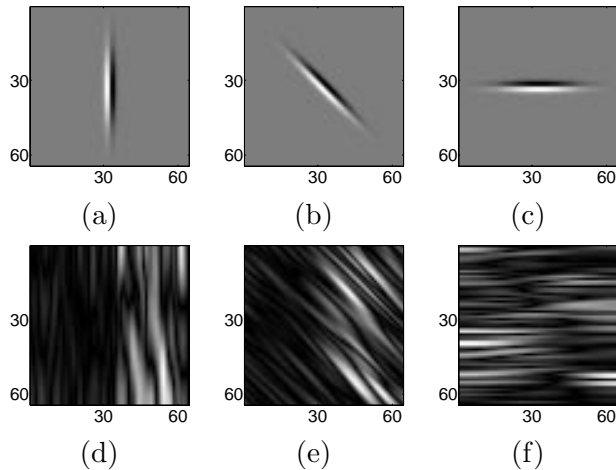


Figure 5.5. (a-c) Filters; (d-f) Filtered versions of the image in Figure 5.4, (a).

superimposed onto the initial image, is depicted in Figure 5.2, (c). Just as in the scalar case, the algorithm is very accurate in locating the boundary: less than 0.2% of the pixels are misclassified in this 100-by-100 image.

A similar experiment, with the same level of noise, is conducted for a more complicated shape, whose image is in Figure 5.3, (a). The result of processing the noisy image of Figure 5.3, (b) is shown in Figure 5.3, (c). In this 200-by-200 image, 0.8% of the pixels are misclassified.

■ 5.1.2 Experiment 2: Texture Images.

Many algorithms for segmenting vector-valued images may be applied to texture segmentation, by extracting features from the texture image and presenting the image consisting of these feature vectors to the segmentation algorithm [36]. This is the paradigm we adopt here. We note that the problem of feature extraction is very important: it is clear that the quality of features will influence the quality of the final segmentation. However, this problem is beyond the scope of the present thesis. The purpose of this experiment is to show the feasibility of texture segmentation using SIDEs, given a

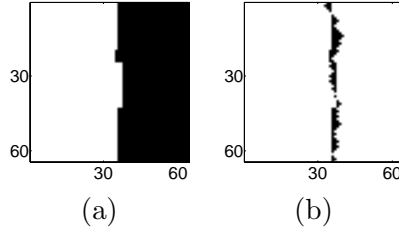


Figure 5.6. (a) Two-region segmentation, and (b) its deviation from the ideal one.

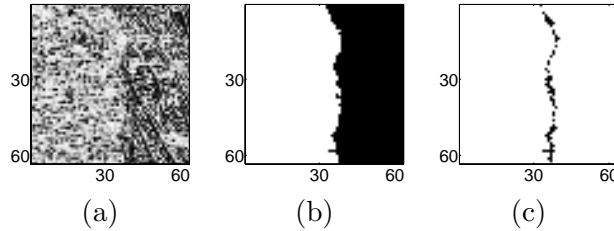


Figure 5.7. (a) A different feature image: the direction of the gradient; (b) the corresponding two-region segmentation, and (c) its deviation from the ideal one.

reasonable set of features.

Our first example is the image depicted in Figure 5.4, (a), which consists of two textures: fabric (left) and grass (right). The underlying texture images whose grayscale versions were used to create the image of Figure 5.4, (a), were taken from the VisTex database of the MIT Media Lab [42]. The correct segmentation is in Figure 5.4, (b). We extract features by convolving our image with the three Gabor functions [40] depicted in Figure 5.5, (a-c), which essentially corresponds to computing smoothed directional derivatives. The results of filtering our image with these functions are shown in Figure 5.5, (d-f). These are the three channels of the vector-valued image which we use as the initial condition for the SIDE (5.2). Figure 5.6, (a) is the resulting two-region segmentation, and its difference from the ideal segmentation is in Figure 5.6, (b). The result is very accurate: about 3.2% of the pixels are classified incorrectly, all of which are very close to the initial boundary.

The next example illustrates the importance of the pre-processing step. We again segment the image of Figure 5.4, (a), but this time we use a different feature. Instead of the three features of Figure 5.5, we use a single feature, which is the absolute value of the gradient direction, depicted in Figure 5.7, (a). More precisely, the (i, j) -th pixel value of the feature image is

$$|\text{angle}(u_{i,j+1} - u_{i,j} + \sqrt{-1}(u_{i+1,j} - u_{i,j}))^2|,$$

where $u_{i,j}$ is the (i, j) -th pixel of the original image. This leads to a significant improvement in performance, shown in the rest of Figure 5.7: the shape of the boundary is much closer to that of the ideal boundary, and the number of misclassified pixels is

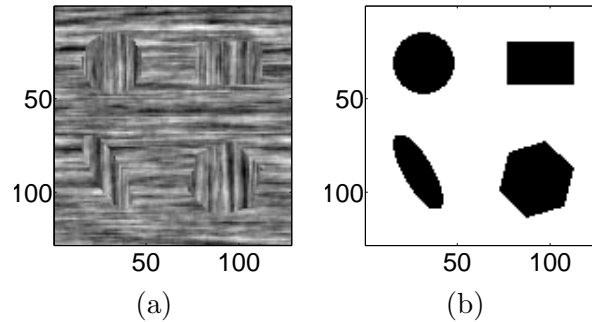


Figure 5.8. (a) Image of two wood textures; (b) the ideal segmentation of the image in (a).

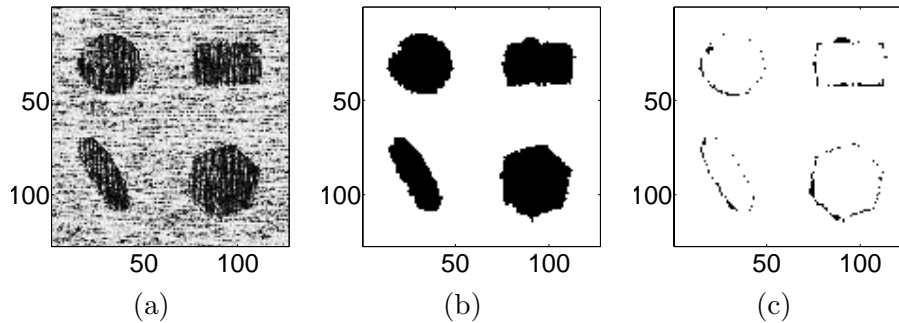


Figure 5.9. (a) Feature image for the wood textures; (b) the corresponding five-region segmentation, and (c) its deviation from the ideal one.

only 2% of the total area of the image.

In our next example, we segment the image of Figure 5.8, which involves two wood textures generated using the Markov random field method of [37]. We again use the direction of the gradient as the sole feature (Figure 5.9, (a)). The five-region segmentation of Figure 5.9, (b) is again very accurate, with the area between our boundary and the ideal one being only 1.6% of the total area of the image (Figure 5.9, (c)).

Our final example of this subsection involves the same wood textures, but arranged in more complicated shapes (Figure 5.10). The feature image and the corresponding segmentation (2.3% of the pixels are misclassified) are shown in Figure 5.11. We will see in the next section that if the two textures only differ in orientation (as in the last two examples), then performance can be improved by using a different version of SIDes, adapted for processing orientation images.

We close this section by noting that in all the examples of this subsection, our segmentation algorithm performed very well, despite the fact that the feature images were of poor quality. No attempt was made to optimize the pre-processing step of extracting the features.

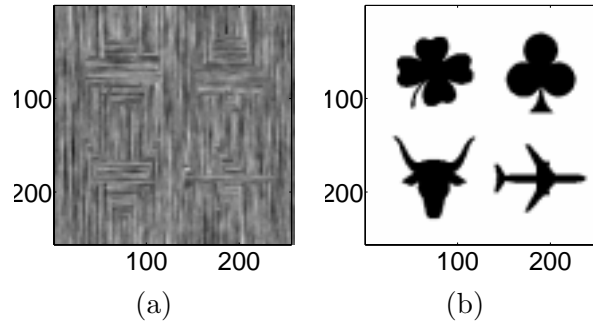


Figure 5.10. (a) Another image of two wood textures; (b) the ideal segmentation of the image in (a).

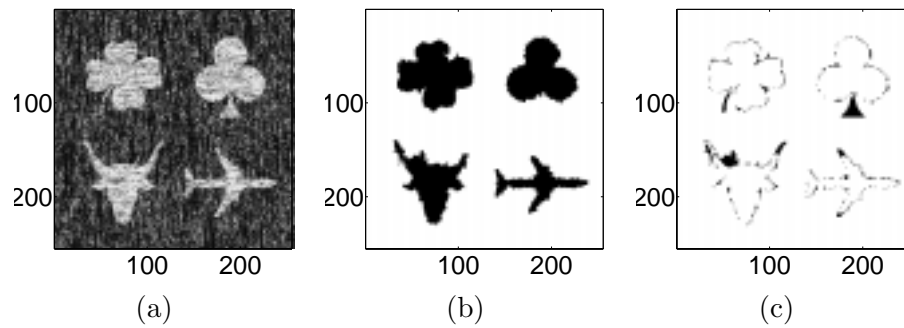


Figure 5.11. (a) Feature image for the wood textures in Figure 5.10, (a); (b) the corresponding five-region segmentation, and (c) its deviation from the ideal one.

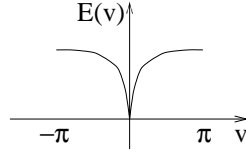


Figure 5.12. A SIDE energy function which is flat at π and $-\pi$ and therefore results in a force function which vanishes at π and $-\pi$.

■ 5.2 Orientation Diffusions.

In [48], Perona used the linear heat diffusion as a basis for developing orientation diffusions. In this section, we follow a similar procedure to adapt SIDEs for processing orientation images. In other words, we consider the case when the image to be processed takes its values on the unit circle \mathbb{S}^1 , i.e., each pixel value is a complex number with unit magnitude. We define the signed distance $\zeta(a, b)$ between two such numbers a and b as their phase difference:

$$\zeta(a, b) \stackrel{\text{def}}{=} -\sqrt{-1} \log \frac{a}{b}, \quad (5.3)$$

where \log is the principal value, i.e., $-\pi \leq \zeta < \pi$.

We would like the force acting between two antipodal points to change continuously as the distance between them changes from $\pi - 0$ to $-\pi + 0$ (i.e., from slightly smaller than π to slightly larger than $-\pi$). We therefore restrict our attention to SIDE energy functions for which $E'(\pi) = E'(-\pi) = 0$, as shown in Figure 5.12. We define the following global energy:

$$\mathcal{E} = \sum_{i, j \text{ are neighbors}} E(|\zeta(u_j, u_i)|). \quad (5.4)$$

We define SIDEs for circle-valued images and signals as the following gradient descent equation:

$$\dot{\mathbf{u}} = -\nabla \mathcal{E},$$

with the i -th pixel evolving according to:

$$\dot{u}_i = -\nabla_i \mathcal{E},$$

where ∇_i is the gradient taken on the unit circle \mathbb{S}^1 . To visualize this evolution, the straight vertical lines of the spring-mass models of Figures 3.1 and 3.3 are replaced with circles: each particle is moving around a circle. After taking the gradients, simplifying, and taking into account merging of pixels, we obtain that the differential equation governing the evolution of the phase angles of u_i 's is very similar to the scalar SIDE:

$$\dot{\theta}_i = \frac{1}{m_i} \sum_{j \in A_i} F(\zeta(u_j, u_i)) p_{ij}, \quad (5.5)$$

where θ_i is the phase angle of u_i (we use the convention $0 \leq \theta_i < 2\pi$, and identify $\theta_i = 2\pi$ with $\theta_i = 0$). The rest of the notation is the same as in Chapter 3. Two neighboring pixels are merged when they have the same phase angle.

While this evolution has many similarities to its scalar and vector-valued counterparts, it also has important differences, stemming from the fact that it operates on the phase angles. Thus, it is not natural to talk about the mean of the (complex) values of the input image; instead, this evolution preserves the sum of the phases, modulo 2π . This is easily verified by summing up the equations (5.5).

Property 5.1 (Total phase conservation). *The phase angle of the product of all pixel values stays invariant throughout the evolution of (5.5).*

Another important distinction from the scalar and vector SIDEs is the existence of unstable equilibria. Since $F(\pi) = E'(\pi) = 0$, it follows that if two neighboring pixels have values which are two antipodal points on \mathbb{S}^1 , these points will neither attract nor repulse each other. Thus, unlike the scalar and vector-valued equations whose only equilibria were constant images, many other equilibrium states are possible here. The next property, however, guarantees that the only stable equilibria are constant images.

Property 5.2 (Equilibria). *Suppose that all the pixel values of an image \mathbf{u} have the same phase. Then \mathbf{u} is a stable equilibrium of (5.5). Conversely, such images \mathbf{u} are the only stable equilibria of (5.5).*

Proof. If all the phases are the same, then the whole image \mathbf{u} is a single region, which therefore has no neighbors and is not changing. Moreover, if a pixel value is perturbed by a small amount, forces exerted by its neighbors will immediately pull it back. Thus, it is a stable equilibrium.

Suppose now that an equilibrium image \mathbf{u} has more than one region. Let us pick an arbitrary region, call its value u_* , and partition the set of its neighboring regions into two subsets: $U = \{u_1, \dots, u_p\}$, whose every element pulls u_* in the counter-clockwise direction (i.e., U is comprised of those regions for which the phase of $\frac{u_i}{u_*}$ is positive and strictly less than π), and the set $V = \{v_1, \dots, v_q\}$, whose elements pull u_* in the clockwise direction (i.e., those regions for which the phase of $\frac{v_i}{u_*}$ is negative and greater than or equal to $-\pi$). One of the sets U, V —but not both—can be empty.

Since our system (5.5) is in equilibrium, it means that the resultant force acting on u_* is zero—i.e., the right-hand side of the corresponding differential equation is zero. Suppose now that u_* is slightly perturbed in the clockwise direction. Since the force function F is monotonically decreasing, this means that the resultant force exerted on u_* by the regions comprising the set V will increase, and the resultant force exerted by U will decrease. The net result will be to further push u_* in the clockwise direction. A similar argument applies if u_* is perturbed in the counter-clockwise direction. Thus, the equilibrium is unstable, which concludes the proof. ■

For any reasonable probabilistic model of the initial data, the probability of attaining an unstable equilibrium during the evolution is zero. In any case, a numerical

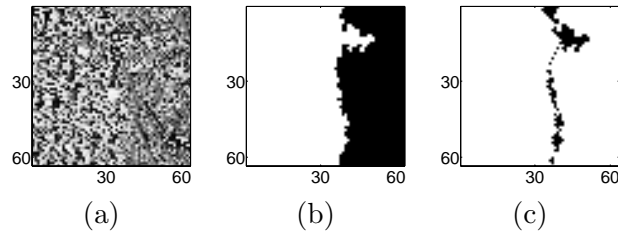


Figure 5.13. (a) The orientation image for Figure 5.7, (a); (b) the corresponding two-region segmentation, and (c) its deviation from the ideal one.

implementation can be designed to avoid such equilibria. Therefore, in the generic case, the steady state of the evolution is a stable equilibrium, which, by the above property, is a constant image. This corresponds to the coarsest segmentation (everything is one region), just like in the scalar-valued and vector-valued cases.

Since there is no notion of a maximum on the unit circle, there is no notion of a “maximum principle”, either. This means, moreover, that we cannot mimic the proof of Property 3.2 (finite evolution time) of the scalar evolutions. This property does hold for the evolutions on a circle, but the proof is different. Specifically, Property 3.7 holds here, with a similar proof, which means that between two consecutive mergings, the global energy is a concave decreasing function of time (Figure 3.7). It will therefore reach zero in finite time, at which point the evolution will be at one of its equilibria.

Property 5.3 (Finite evolution time). *The SIDE (5.5) reaches its equilibrium in finite time, starting with any initial condition.*

■ 5.2.1 Experiments.

To illustrate segmentation of orientation images, we use the same texture images which we used in the previous section. To extract the orientations, we use the direction of the gradient. The (i, j) -th pixel value of the orientation image is

$$\text{angle}[(u_{i,j+1} - u_{i,j} + \sqrt{-1}(u_{i+1,j} - u_{i,j}))]^2,$$

where $u_{i,j}$ is the (i, j) -th pixel value of the raw texture image. (Note that the absolute value of this orientation image was used as a feature image in the previous section.) The expression in the above formula is squared so as to equate the phases which differ by π .

The orientations for the fabric-and-grass image are shown in Figure 5.13, (a). We present this image as the initial condition to the circle-valued SIDE (5.5) and evolve it until two regions remain. The resulting segmentation is depicted in Figure 5.13, (b), and its difference from the ideal one is in 5.13, (c). About 4.3% of the total number of pixels are classified incorrectly. It is not surprising that this method performs slightly worse on this example than the evolutions of the previous section. Indeed, if a human were asked to segment the original texture image based purely on orientation, he might

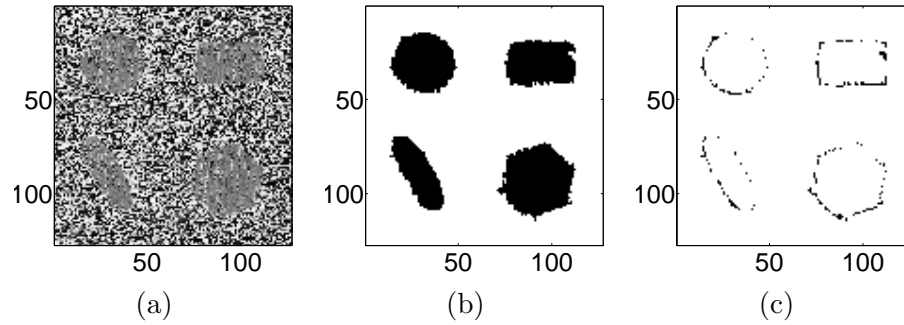


Figure 5.14. (a) The orientation image for Figure 5.9, (a); (b) the corresponding five-region segmentation, and (c) its deviation from the ideal one.

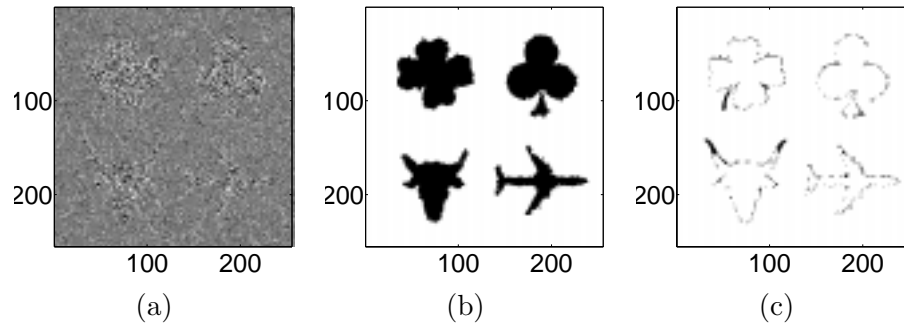


Figure 5.15. (a) The orientation image for Figure 5.10, (a); (b) the corresponding five-region segmentation, and (c) its deviation from the ideal one.

make similar errors. Note in particular that the protrusion in the upper portion of the boundary found by the SIDE corresponds to a horizontally oriented piece of grass in the original image, which can be mistaken for a portion of the fabric.

In the second example, however, the orientation information is very appropriate for characterizing and discriminating the two differently oriented wood textures. The orientation image is in Figure 5.14, (a). The resulting five-region segmentation of Figure 5.13, (b) incorrectly classifies only 252 pixels (1.5%) in the 128-by-128 image, which is 14 pixels better than the method of the previous section. A more dramatic improvement is achieved in the example of Figure 5.15, which shows the five-region segmentation of the image in Figure 5.10, (a), using the circle-valued SIDE (5.5). Only 1.5% of the pixels are misclassified, as compared to 2.3% using the method of the previous section.

■ 5.3 Conclusion.

In this chapter, we generalized SIDEs to the situations when the image to be processed is not scalar-valued. We described the properties of the resulting evolutions and demonstrated their application to segmenting color, texture, and orientation images.