Article

Edge-based Color Image Segmentation using Particle Motion in a Vector Image Field Derived from Local Color Distance Images

Wutthichai Phornphatcharaphong [1] and Nawapak Eua-Anant [1,*]

[1] Department of Computer Engineering, Faculty of Engineering, Khon Kaen University, 40002, Thailand; wutthichai@kkumail.com (W.P.); nawapak@kku.ac.th (N.E.)

* Correspondence: nawapak@kku.ac.th

Abstract: This paper presents an edge-based color image segmentation approach, derived from the method of particle motion in a vector image field, which could previously be applied only to monochrome images. Rather than using an edge vector field derived from a gradient vector field and a normal compressive vector field derived from a Laplacian-gradient vector field, two novel orthogonal vector fields were directly computed from a color image, one parallel and another orthogonal to the edges. These were then used in the model to force a particle to move along the object edges. The normal compressive vector field is created from the collection of the center-to-centroid vectors of local color distance images. The edge vector field is later derived from the normal compressive vector field so as to obtain a vector field analogous to a Hamiltonian gradient vector field. Using the PASCAL Visual Object Classes Challenge 2012 (VOC2012), the Berkeley Segmentation Data Set and Benchmarks 500 (BSDS500), the benchmark score of the proposed method is provided in comparison to those of the traditional particle motion in a vector image field (PMVIF), Watershed, simple linear iterative clustering (SLIC), K-means, mean shift, and J-value segmentation (JSEG). The proposed method yields better Rand index (RI), global consistency error (GCE), normalized variation of information (NVI), boundary displacement error (BDE), Dice coefficients, faster computation time, and noise resistance.

Keywords: color image segmentation; particle motion; local color distance images; normal compressive vector field; edge vector field

1. Introduction

In digital image processing, image segmentation that reduces the amount of unnecessary data and preserves the important information needed for analysis plays an important role in image analysis. In general, image segmentation gathers pixels displaying similar characteristics within the same areas and converts them into regions. Among the various techniques, image segmentation methods can be divided into two main groups: machine learning image segmentation and classical image segmentation. First, machine learning image segmentation is a method by which a program can learn and segment an object by itself, without adjusting the program further. There are three types of machining approaches: supervised, unsupervised, and reinforcement learning methods. Supervised methods use a training dataset containing ground truth data to train artificial neural networks to map between input images and segmented results (see the survey [1]). However, the training process is computationally intensive and the ground truth construction, that requires manual labeling by experts, is labor-intensive. Additionally, when a new object class is added, the whole training dataset must be thoroughly reconstructed and the time-consuming training process must be repeated. In contrast, the unsupervised method does not require a dataset for training. Instead, the result of each iteration is recursively input to the program to adjust its parameters. This type of approach, such as K-means [2][3], mean shift [4][5], and JSEG [6][7], etc., is often more effective and more tolerant of unusual or unpredictable situations. However, the unsupervised methods are usually time-consuming due to its iterative processes embedded in the methods. Finally, the reinforcement learning method uses the reward and punishment techniques from environmental analysis for learning to drive the agent to the target. This method requires a large number of iterations for training of the agent to get a reward [8][9][10]. Second, classical image segmentation is a low-level image processing approach that tries to extract information without knowing the truth. Although, nowadays machine learning image segmentation is state-of-the-art [11], classical image segmentation is still necessary in cases in which segmentation does not have ground truth images or there is a time constraint. Classical image segmentation also helps to create ground truth data in training datasets for machine learning techniques. Classical image segmentation techniques comprise of thresholding-based, edge-based, region-based and graph-based techniques. Thresholding-based techniques are divided into three types [12]: global thresholding, local thresholding and adaptive thresholding. First, global thresholding weighs the distribution of intensity in the histogram to determine the threshold for separating objects from the background [13][14][15]. Second, local thresholding is used when a single threshold is not possible for images with uneven illumination or shadows. In such a case, it is necessary to use a sub-image to select the threshold [16]. Third, adaptive thresholding makes a calculation of the threshold by a window to find the intensity from the neighbor pixel [17][18]. The edge-based techniques, such as zero-crossing [19], Active Canny [20], PMVIF [21][22][23], EdgeFlow [24], and PointFlow [25], extract object boundaries in the image by creating contours around the objects. The region-based techniques, such as watershed [26][27] are based on the principle of grouping pixels with similar properties into the same regions or the same objects. Finally, the graph-based techniques, such as graph cuts [28][29], normalized cuts [30][31], Superpixel [32][33] and SLIC [34][35] proceed by grouping pixels according to graph theory. The methods mentioned above have various strengths and weaknesses, such as segmentation accuracy, processing time, flexibility, ease of use, and robustness to noise. For example, some algorithms can be applied only to grayscale images while some are only available in the RGB color space. Some methods are not suitable for real-time use or have to adjust too many parameters.

This paper introduces an edge-based classical image segmentation algorithm for a color image using particle motion in a vector image field derived from local color distance images (PMLCD). It is developed from the PMVIF algorithm that is known to have a fast computation time and yields closed boundaries but can be applied only to grayscale images. In the PMVIF algorithm, two vector fields, namely, the normal compressive vector field and the edge vector field, derived from derivatives of grayscale images, are used to force the particle to move along the object edges which results in closed particle trajectories that resemble the object boundaries. In order to extend this principle to a color image segmentation task, the new formulae for computing the normal compressive vector field and the edge vector field, derived from the local color distance images, are introduced. The method proposed in this paper can be used not only with color images but also multichannel images such as hyperspectral images.

The rest of the paper is organized as follows: Section 2 describes the principle of the PMVIF algorithm; Section 3 describes the developed color image segmentation using particle motion in a vector image field derived from local color distance images; Section 4 presents the experimental validations and benchmarking of the proposed algorithm; finally, conclusions are drawn in Section 5.

2. Background to Particle Motion in a Vector Image Field

This section describes the principle of a traditional boundary extraction algorithm based on particle motion in a vector image field (PMVIF), which is an edge-based classical image segmentation approach. In general, in an N-dimensional space, a boundary can be explicitly represented by a manifold of dimension N-1 interfacing between regions of different attributes; for example, a close curve in a two-dimensional space. However, in a discretized image where a set of pixels or voxels is the only class that can exist, explicit representations of region boundaries, such as a curve or a surface, are difficult to encode. In this case, a normal compressive vector field [21][22][23], where all vectors are normal and point to the nearest interface, providing information about the direction to the nearest boundary, is more suitable to be used as an implicit boundary representation. Nevertheless, the normal compressive vector field itself only provides information about the location of the boundary but cannot offer any clues regarding the direction for tracking edges. In order to be able to locate and track a boundary simultaneously, another vector field containing vectors parallel to edges—namely an edge vector field—combined with the normal compressive vector field is required. The concept of using two such orthogonal vector fields for boundary extraction in a grayscale image was introduced in the PMVIF algorithm, where the gradient–Laplacian vector field used as a normal compressive vector field and the Hamiltonian gradient vector field used as an edge vector field are given as follows:

$$\vec{n} = \frac{1}{c}\nabla P \cdot \nabla^2 P \tag{1}$$

and

$$\vec{e} = -\frac{\partial P}{\partial y}\hat{i} + \frac{\partial P}{\partial x}\hat{j} \tag{2}$$

where c is a normalization factor, and $\hat{i}$, $\hat{j}$ are unit vectors in x and y directions, respectively.

In general, in Equations 1 and 2, partial derivatives can be approximated using difference operators such as Sobel operators. Figure 1 illustrates examples of the gradient ∇P, Laplacian $\nabla^2 P$, edge vector field $\vec{e}$, and the gradient–Laplacian vector field $\vec{n}$. In order to extract object boundaries, sequences of boundary points were obtained from trajectories of a particle driven by the combined force field $\alpha\vec{e} + \beta\vec{n}$, computed as follows:

$$\vec{P}_{k+1} = \vec{P}_k + \alpha\vec{e}_k + \beta\vec{n}_k \tag{3}$$

where $\vec{P}_k$ is the k^{th} particle position vector; $\vec{e}_k$ is the edge vector, interpolated at the k^{th} particle position; $\vec{n}_k$ is the normal compressive vector, interpolated at the k^{th} particle position; α is a tangential stepping factor, with $\alpha > 0$ for a particle moving in a clockwise direction and $\alpha < 0$ for a particle moving in a counter-clockwise direction; and β, $\beta > 0$, is a normal stepping factor allowing the trajectory to converge to a boundary line.

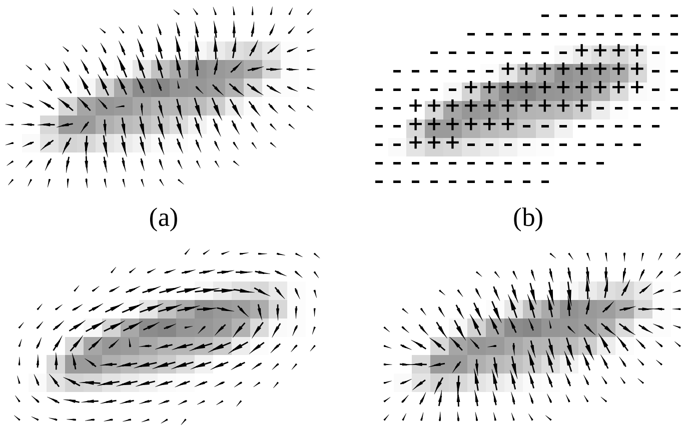

Figure 1. (**a**) ∇P, (**b**) $\nabla^2 P$, (**c**) an edge vector field $\vec{e}$, and (**d**) a normal compressive vector field $\vec{n}$.

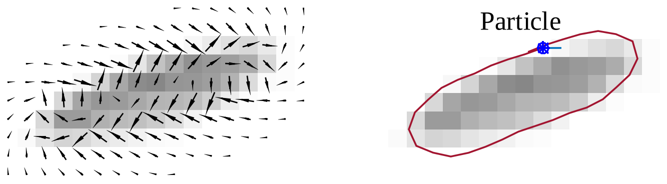

Figure 2. (**a**) A combined vector field, $\alpha\vec{e} + \beta\vec{n}$, $\alpha = 0.5$, $\beta = 0.5$, and (**b**) a boundary extraction result.

Figure 2 demonstrates a combined vector field, $\alpha\vec{e} + \beta\vec{n}$, $\alpha = 0.5$, $\beta = 0.5$, and a boundary extraction result obtained from a particle trajectory, according to Equation 3, as applied to the image in Figure 1. The PMVIF works well in extracting boundaries of regions with a constant intensity in grayscale images, providing subpixel resolution results. Nevertheless, the limitation of the PMVIF method is that the edge and normal compressive vector fields are derived from partial derivative operations that can only be applied to a scalar or intensity image. In the case of color or multispectral images in which each pixel is considered as a vector, there is no exact definition of gradient and Laplacian operators, limiting the application of the PMVIF method to color images. To overcome this limitation, a new scheme to generate a normal compressive vector field and an edge vector field for the vector image is required.

3. Methodology

The PMVIF algorithm requires both normal compressive and edge vector fields as particle driving forces. Due to the gradient definition that is applied only to a scalar image, the original PMVIF method can be applied only to intensity images. In this paper, the PMLCD method for finding the normal compressive and edge vector fields for color images using the center to centroid vectors of local color distance images is presented below.

3.1. Image moments

For a discrete image $I(x, y)$, a two-dimensional moment of order (p, q) [36] is defined as

$$M_{pq} = \sum_x \sum_y x^p y^q I(x, y) \tag{4}$$

Analogous to a center of gravity in classical mechanics, the centroid $(\bar{x}, \bar{y})$ of an image $I(x, y)$ can be calculated as follows:

$$\begin{aligned}(\bar{x}, \bar{y}) &= \left(\frac{M_{10}}{M_{00}}, \frac{M_{01}}{M_{00}}\right) \\ &= \left(\frac{\sum_x \sum_y xI(x,y)}{\sum_x \sum_y I(x,y)}, \frac{\sum_x \sum_y yI(x,y)}{\sum_x \sum_y I(x,y)}\right)\end{aligned} \tag{5}$$

The displacement between a center and a centroid of an image indicates an unbalanced pixel intensity distribution in a spatial domain.

3.2. Local color distance images

In general, image segmentation can be viewed as a process to determine in which region each pixel should be located. For a multispectral image I, one feature which is widely used to determine whether or not pixels should belong to the same region is the color distance between two pixels, defined as

$$D_c((x,y),(i,j)) = \sqrt{(I_1(x,y) - I_1(i,j))^2 + (I_2(x,y) - I_2(i,j))^2 + ... + (I_n(x,y) - I_n(i,j))^2} \quad (6)$$

where $I_n(x,y)$ and $I_n(i,j)$ are the n^{th} color components of pixels (x,y) and (i,j), respectively. In the data classification aspect, the color distance functions as a dissimilarity measurement between two pixels. Using the concept of a moving window, a local color distance image (LCD) of the pixels surrounding pixel (i,j) can be computed as

$$LCD(x-i, y-j) = \underset{(x,y)\in N(i,j)}{D_c((x,y),(i,j))} \quad (7)$$

where $N(i,j)$ is a neighbor area of a center pixel (i,j).

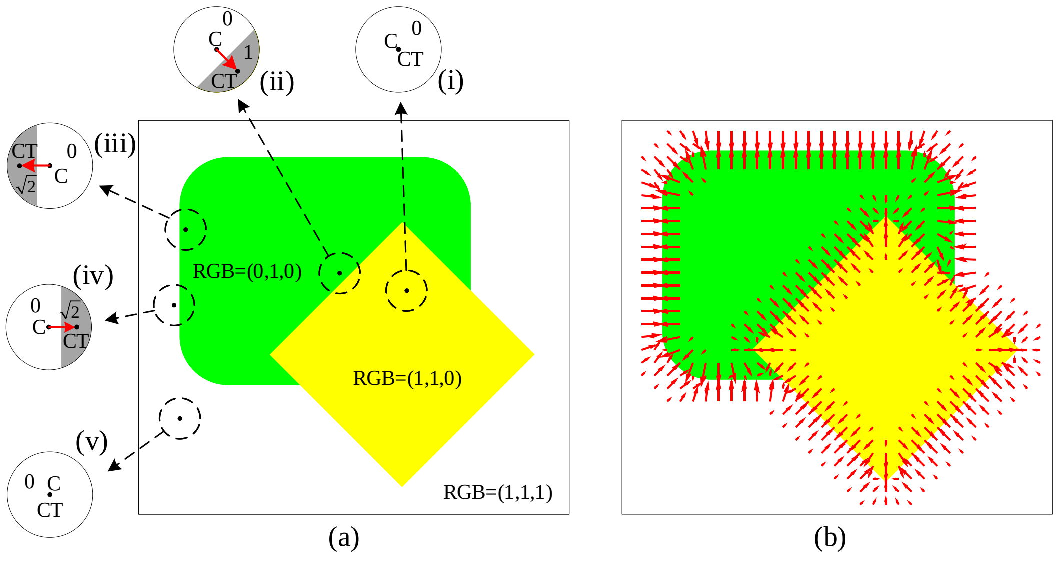

Figure 3. (**a**) Local color distance images and (**b**) a normal compressive vector field obtained from $C - to - CT$ vectors.

Each pixel in $LCD(x-i, y-j)$ represents a color distance between a neighboring pixel (x,y) and the center pixel (i,j). Figure 3(a) illustrates examples of RGB local color distance images (i)-(v), obtained using a circular moving window computed at various places in a simple two-object image. As seen in the (i) and (v) cases, if a circular window is placed entirely inside one region, a local color distance image contains all zero pixels. Conversely, if a circular window is located at the border between two regions, the obtained local color distance image comprises pixels, with large values packed to one side of the image, as shown in cases (ii)-(iv) in 3(a). As a result, the centroid CT of the local color distance image computed using Equation 5 is shifted from the center C toward a high color distance area belonging to an adjacent region. Thus, for a local color distance image located in the proximity of a boundary, a vector from center C to a centroid CT points in the direction of the nearest boundary, independent of the side of the center of the local color distance on which the image is; for example, cases (iii) and (iv) in Figure 3(a).

3.3. The normal compressive vector field

By gathering $C-to-CT$ vectors of local color distance images obtained at all valid positions in an original image, a normal compressive vector field $\vec{n}$ can be computed as

$$\vec{n}(i,j)=\frac{1}{C}\begin{bmatrix}\bar{x}_{(i,j)}-i\\ \bar{y}_{(i,j)}-j\end{bmatrix} \tag{8}$$

where C is a normalization factor making $max\,|\vec{n}(i,j)|=1$, and $(\bar{x}_{(i,j)},\bar{y}_{(i,j)})$ is a centroid, computed using Equation 5, of $LCD(x-i,y-j)$ computed using Equation 7. Figure 3(b) demonstrates the $\vec{n}$ of the image in Figure 3(a). By combining Equations 5-7, $\vec{n}(i,j)$ can be directly computed as

$$\vec{n}(i,j)=\frac{1}{C}\begin{bmatrix}\sum\limits_{(x,y)\in N(i,j)}(x-i)D_c\big((x,y),(i,j)\big)\Big/\sum\limits_{(x,y)\in N(i,j)}D_c\big((x,y),(i,j)\big)\\ \sum\limits_{(x,y)\in N(i,j)}(y-j)D_c\big((x,y),(i,j)\big)\Big/\sum\limits_{(x,y)\in N(i,j)}D_c\big((x,y),(i,j)\big)\end{bmatrix} \tag{9}$$

It is worthy of note that, in this vector field, the phenomenon that a vector on one side always points in the opposite direction to a vector on another side is called the normal compressive property. In the PMVIF technique, the normal compressive property of the vector field causes a particle to cling to the object boundary. The difference in the $\vec{n}$ from Equation 1 and Equation 9 is that the vector size obtained from Equation 1 is smaller than Equation 9, as shown in Figure 4. As Equation 9 uses the principle of LCD while Equation 1 only uses grayscale images that include intensity for each band collapsed together and using a gradient, resulting in a smaller vector size, Equation 9 is more suitable when used with color images.

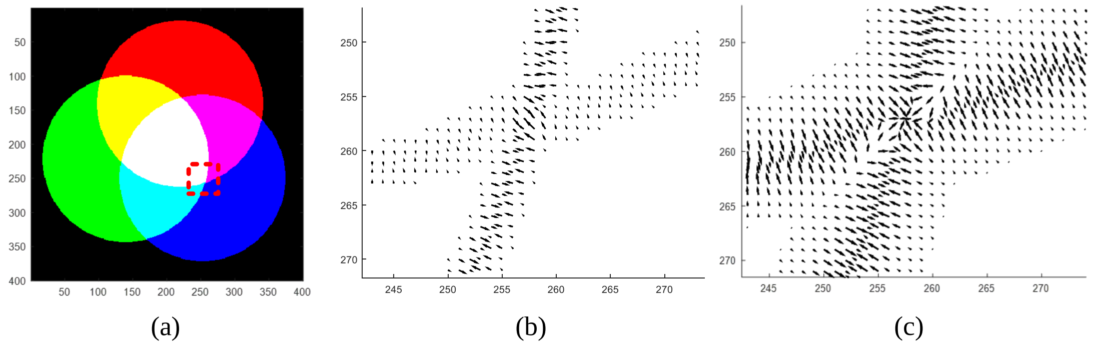

Figure 4. (**a**) Original color image, (**b**) $\vec{n}$ obtained from Equation 1, and (**c**) $\vec{n}$ obtained from Equation 9

3.4. The edge vector field

The edge vector field in the original PMVIF method, used to drive a particle to move in a direction parallel to object edges in a grayscale image, is derived from a Hamiltonian gradient vector field. However, such a vector field cannot be generated in the case of vector images such as color images where each pixel is represented by a color vector. In order to create a vector field analogous to the edge vector field, firstly, a vector-to-scalar conversion scheme must be applied to a color image to achieve a unique condition, ensuring that different colors, normally represented by vectors, are represented by different scalar values. The linearization technique used to convert a color image into a scalar auxiliary image, based on the number base system, is proposed in this paper as follows:

$$Aux(x,y)=m^{(n-1)}I_n(x,y)+m^{(n-2)}I_{n-1}(x,y)+...+m^2I_3(x,y)+mI_2(x,y)+I_1(x,y) \tag{10}$$

where m is the maximum intensity level of each color component. The auxiliary image is created to determine whether a neighbor pixel (x, y) has the same color as the center pixel (i, j).

Thus, only a difference between $Aux(x, y)$ and $Aux(i, j)$ is sufficient to determine whether both pixels (x, y) and (i, j) have the same color.

To obtain a gradient-like vector field, in a normal compressive vector field, as demonstrated in Figure 3(b), vectors outside objects must be reverted while vectors inside objects retain the same direction. Thus, Equation 9 is modified by multiplying the local color distance with the sign of a difference between auxiliary image pixels as follows:

$$\begin{aligned}\vec{G}(i,j) &= \begin{bmatrix} G_x \\ G_y \end{bmatrix} \\ &= \frac{1}{C}\begin{bmatrix} \sum\limits_{(x,y)\in N(i,j)} (x-i)sign(Aux(x,y)-Aux(i,j))D_c((x,y),(i,j)) \\ \sum\limits_{(x,y)\in N(i,j)} (y-i)sign(Aux(x,y)-Aux(i,j))D_c((x,y),(i,j)) \end{bmatrix}\end{aligned} \tag{11}$$

where C is a normalization factor so that $max\left|\vec{G}(i,j)\right| = 1$ and $sign(A) = \begin{cases} 1 & A \geq 0 \\ -1 & A < 0 \end{cases}$.

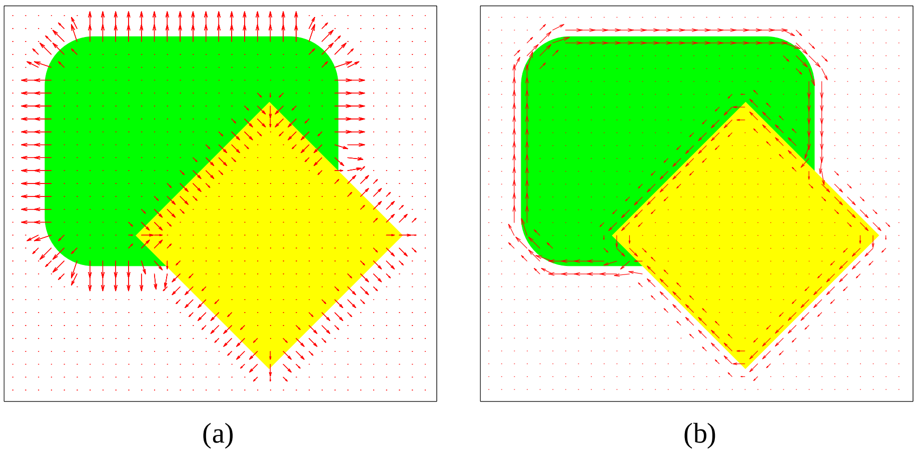

Figure 5. (**a**) A gradient-like vector field $\vec{G}$ obtained from Equation 11, and (**b**) an edge vector field $\vec{e}$ obtained from Equation 12.

As a result, the normal compressive property of $\vec{n}$ in Equation 9—i.e., a vector on one side always points in a direction opposite to a vector on another side, as shown in Figure 3(b)—is transformed to a gradient-like property of $\vec{G}$ where vectors on both sides of objects always point in the same direction, as shown in Figure 5(a). Next, by rotating all vectors in $\vec{G}$ by 90°, an edge vector field, similar to a Hamiltonian gradient vector field, is obtained as

$$\vec{e}(i,j) = \begin{bmatrix} -G_y \\ G_x \end{bmatrix} \tag{12}$$

as shown in Figure 5(b). Notice that vectors in $\vec{e}$ in the proximity of boundaries are always larger than areas farther away. Therefore, the magnitude of $\vec{e}$ can be used as a measurement for localizing object edges.

3.5. Particle Motion in a Vector Image Field derived from Local Color Distance Images

The proposed boundary extraction algorithm is based on particle motion in a vector image field derived from local color distance images (PMLCD), using a normal compressive vector field that is calculated using Equation 9 while the edge vector field is calculated using Equation 12 so that particle trajectories, calculated using Equation 3, can be obtained. The object boundaries can then be extracted from a collection of these trajectories. The remaining steps of the PMLCD method are as for those of the PMVIF method.

3.6. Appropriate PMLCD parameter setting

The PMLCD method has three parameters: $T_{|\vec{e}|}$, α, and β. The $T_{|\vec{e}|}$ parameter sets the threshold of $|\vec{e}|$ to determine the starting points of the particle, while α is the strength of the particle moving in the direction parallel to the object edges, and β is the strength in which the particle attaches to the object edges. These parameters are difficult to adjust. This article, therefore, suggests methods for adjusting all three parameters as follows:

$T_{|\vec{e}|}$ is determined using the Otsu's threshold method [14]. α and β are related as follows:

$$\alpha = \frac{1}{2} + R \tag{13}$$

$$\beta = \frac{1}{2} - R \tag{14}$$

$$R = \mid (\overline{V_c} V_{\vec{e}}) - V_{\vec{n}} \mid * 100\gamma \tag{15}$$

where

$\overline{V_c}$ is the mean of the normalized variance of the color image.

$V_{\vec{e}}$ is the normalized variance of $|\vec{e}|$.

$V_{\vec{n}}$ is the normalized variance of $|\vec{n}|$.

γ is the ratio of α and β, $\begin{cases} 0 < \gamma < 1 & \alpha > \beta \\ \gamma = 0 & \alpha = \beta \\ -1 < \gamma < 0 & \alpha < \beta \end{cases}.$

The variance (V) of each channel, converted to a vector A, is made up of scalar observation defined as

$$V = \frac{1}{N-1} \sum_{i=1}^{N} \mid A_i - \overline{A} \mid^2 \tag{16}$$

3.7. *Overall boundary extraction method*

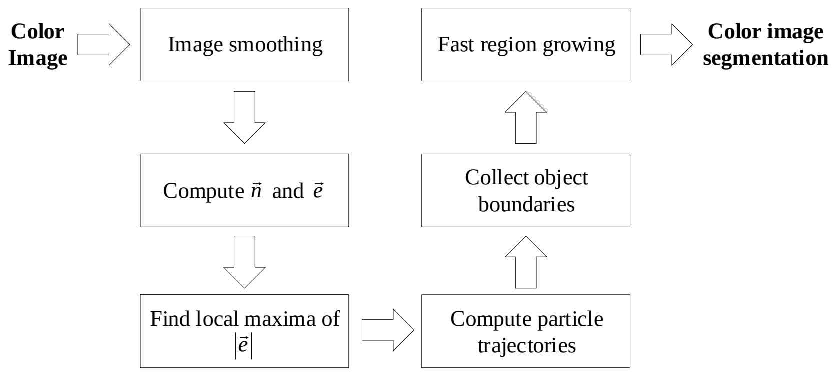
Color Image
Image smoothing
Compute $\vec{n}$ and $\vec{e}$
Find local maxima of $|\vec{e}|$
Compute particle trajectories
Collect object boundaries
Fast region growing
Color image segmentation

Figure 6. Block diagram of the proposed boundary extraction process.

The overall segmentation algorithm for color images, as illustrated in Figure 6, is described here. First, an image is smoothed to remove noise using a Gaussian low pass filter. The normal compressive vector field $\vec{n}$ and edge vector field $\vec{e}$ are then calculated using Equations 9 and 12, respectively, using a circular moving window of radius R. Local maximum points of $|\vec{e}|$, that are greater than a threshold, are chosen as candidates for the starting points of the boundary extraction process. The suitable threshold value is determined by Otsu's threshold method of $|\vec{e}|$. Commencing at each starting point, under the influence of a compressing edge vector field, a particle is forced to move along object edges, according to Equation 3, in both clockwise ($\alpha > 0$) and counter-clockwise directions ($\alpha < 0$) with a subpixel step size until it reaches a starting point or other previously extracted paths. Consequently, boundaries are collected from all obtained particle trajectories. Consequently, a complete edge map is achieved by quantizing the extracted boundaries. Finally, these boundaries are labeled with the fast region growing algorithm to produce the color image segmentation result.

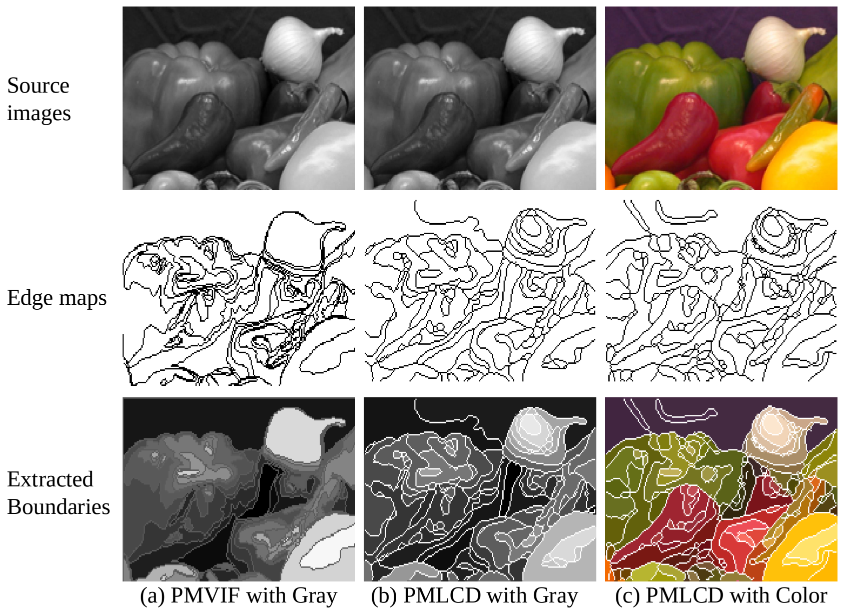

Figure 7. Image segmentation results obtained using (**a**) PMVIF and (**b**) PMLCD are evaluated using the same grayscale image and (**c**) The PMLCD result evaluated using the original color image.

Figure 7 illustrates image segmentation results obtained using both PMVIF and PMLCD evaluated using the same grayscale image and the original color image. Parameters used in all cases were $T_{|\vec{e}|}$ = 0.08, α = 0.5, β = 0.2 (for both PMVIF and PMLCD), and the radius of LCD = 1 (for PMLCD). As seen in Figures 7(b) and (c), PMLCD can be applied to both grayscale and color images. In addition, when compared to the results evaluated using the grayscale image, PMLCD evaluated using the original color image provided the best results with least fault contours.

Figure 8 shows the simulation of particle motion in a vector image field derived from Equation 3 using the following parameters: radius of LCD = 1, $T_{|\vec{e}|}$ = 0.25, γ = 0.05 (α = 0.55, β = 0.45).

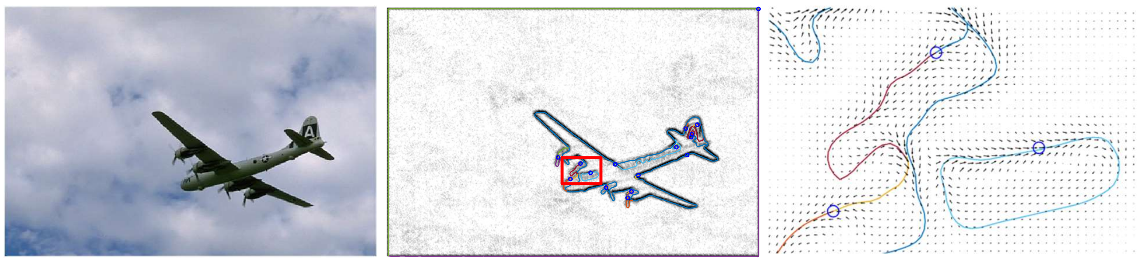
(a) (b) (c)

Figure 8. (**a**) BSDS500 #3096, (**b**) particle trajectory obtained from Equation 3, and (**c**) zoom of (b)

4. Experimental Results and Discussion

The experimental results of color image segmentation using MATLAB 2019b with a CPU Intel Core i7-4710HQ, the VOC2012 dataset [37] and the BSDS500 dataset [38] collected to measure the performance of PMLCD compared with other unsupervised machine learning methods including K-means [2][3], mean shift [4][5], and JSEG [6][7] and classical methods including the grayscale PMVIF [21][22][23], grayscale watershed [26][27], and SLIC [34][35] are given in this section. Benchmarks used in this paper include the Rand Index (RI) [39], Global Consistency Error (GCE) [39], Normalized Variation of information (NVI) [40], Boundary Displacement Error (BDE) [39], Dice coefficient [39], computation time and noise tolerance. Figure 9 shows the experimental color image segmentation results obtained from all methods using the image #2007_000063 from VOC2012. Figure 10 shows similarities between the object chosen from the ground truth image (a dog) and the corresponding segmented regions obtained from all methods in Figure 9. Figure 11 shows the results of the same experiment as those of Figure 9 and Figure 10 for the images randomly selected from VOC2012 and BSDS500. As shown in Table 1, the parameters of all methods for each image in the experiment have been adjusted to achieve high RI and high Dice coefficients. The average benchmarking results show that the method with the highest average RI is PMLCD, at 0.78 (0.11). The methods with the lowest average GCE are PMLCD, at 0.13 (0.05), and Watershed, at 0.13 (0.08). The methods with the lowest average NVI are JSEG, at 0.12 (0.01), and PMLCD, at 0.12 (0.04). The method with the lowest average BDE is SLIC, at 11.82 (4.12). The method with the fastest average calculation time is Watershed, at 0.06 (0.01) seconds. The method with the highest average Dice coefficient is PMLCD, at 0.93 (0.03). Briefly, the PMLCD method yields the four best average values for the RI, GCE, NVI, and Dice coefficient. Figure 12 shows the graphs of computation times used to segment image #3096 from the BSDS500, interpolated to achieve various image sizes. As seen, the Watershed, PMVIF, and PMLCD methods are the fastest, respectively, but the PMLCD is the only true color image segmentation method. Figure 13 demonstrates the result of the noisy color image segmentation of the image #2007_001289 from VOC2012 with additive white Gaussian noise (signal-to-noise ratio (SNR) 0 dB (σ_{noise} = 0.21)) obtained using the PMLCD algorithm with the following parameters: radius of LCD = 3, $T_{|\vec{e}|}$ = 0.27 derived from Otsu's method, γ = -0.14 resulting in α = 0.34 and β = 0.66 obtained from Equations 13 and 14, respectively. The result gives the following benchmarks: RI = 0.91, GCE = 0.01, NVI = 0.01, BDE = 90.23 and computation time = 0.72 seconds. Figure 14 shows the SNR-performance graph of the PMLCD applied to this image, reflecting the high noise tolerance of the PMLCD method.

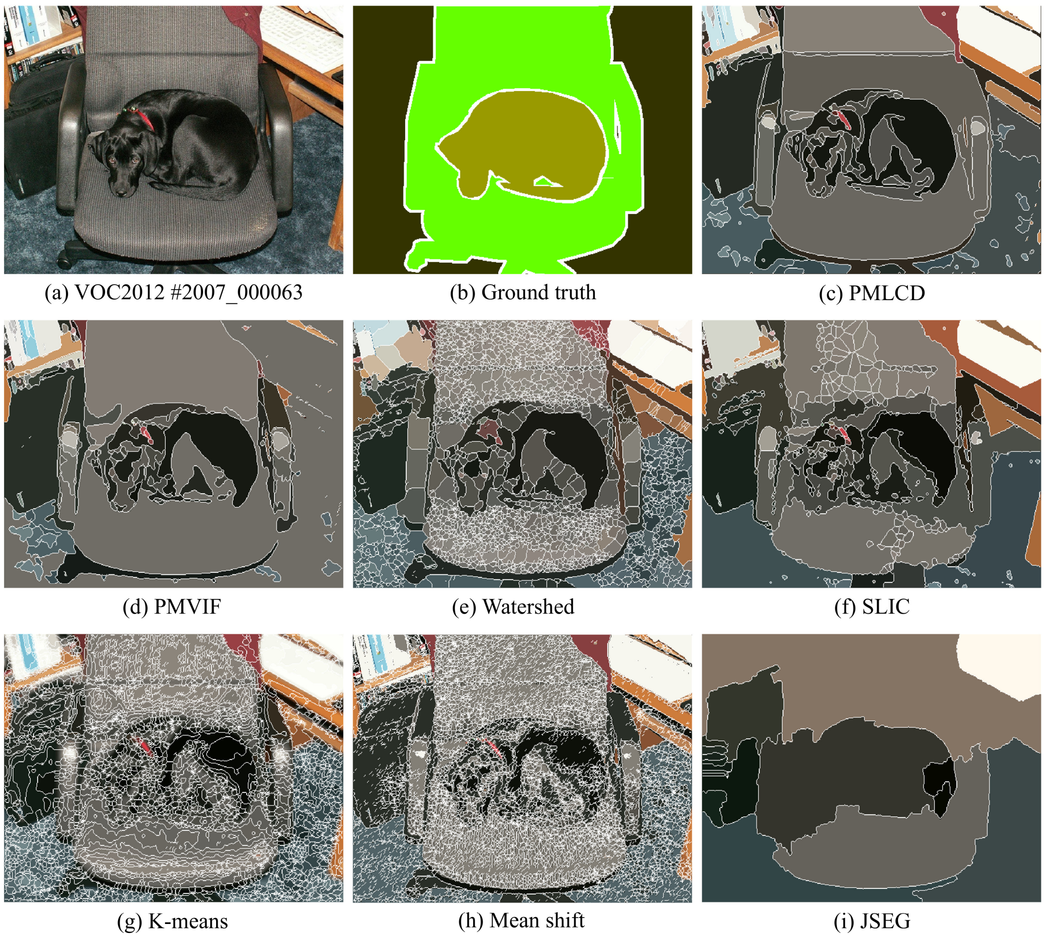

Figure 9. Color image segmentation results of the image #2007_000063 from VOC2012.

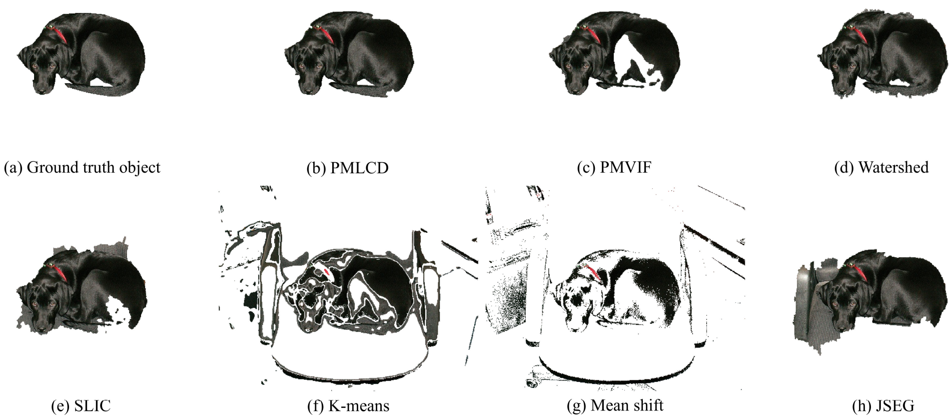

Figure 10. The ground truth object in Figure 9 and the corresponding segmented regions.

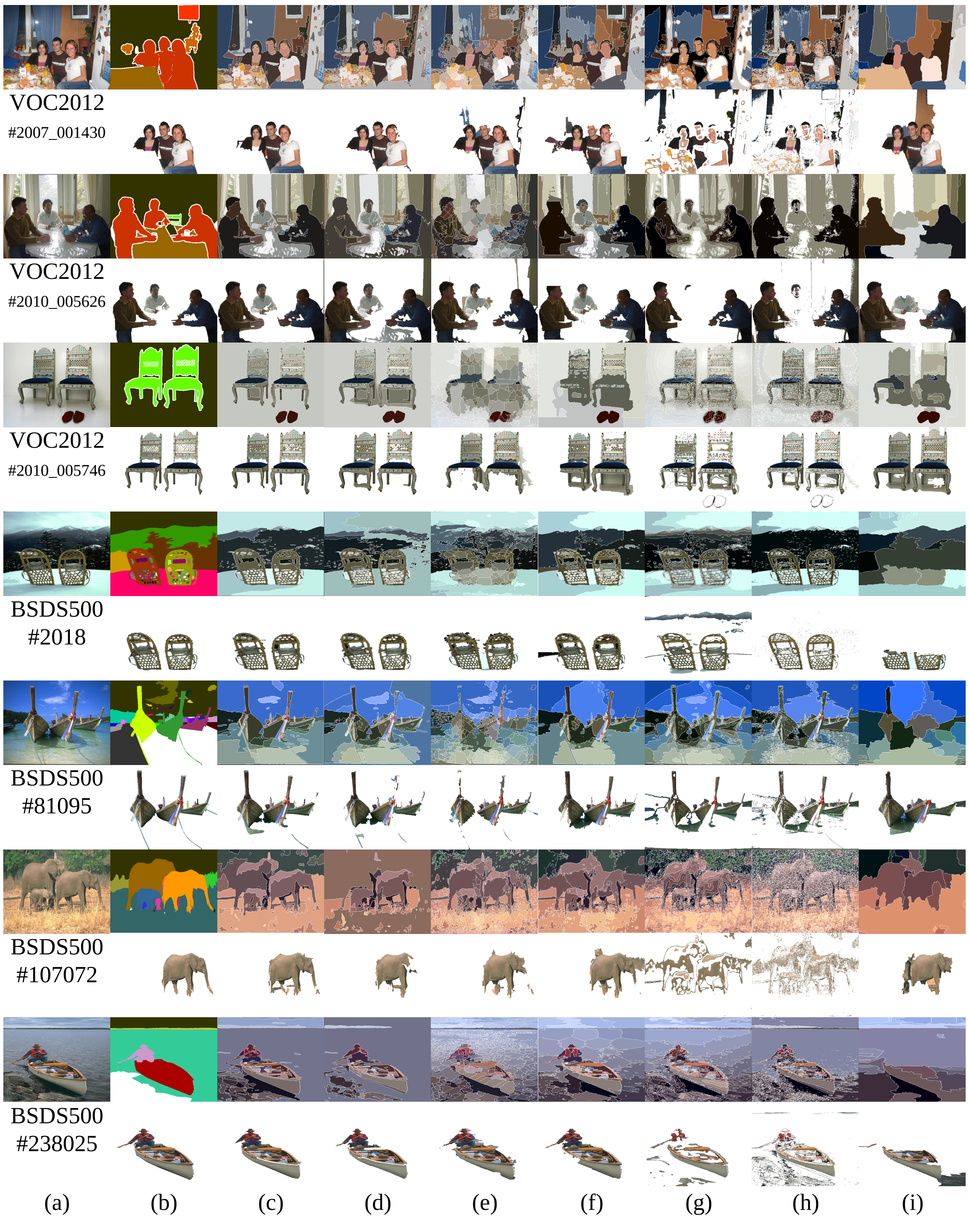

Figure 11. Color image segmentation results. (**a**) Dataset images, (**b**) ground truths; results of (**c**) PMLCD, (**d**) PMVIF, (**e**) Watershed, (**f**) SLIC, (**g**) K-means, (**h**) mean shift, and (**i**) JSEG.

Table 1. The performance of each method from Figure 9, 10 and 11

Dataset	Method	By Image					By Object	Parameter
		RI	GCE	NVI	BDE	Time	Dice	
VOC2012 #2007_000063	PMLCD	**0.67**	0.09	0.15	16.66	0.21s	**0.97**	LCD radius 1, $T_{\vert\vec{e}\vert}$0.17(*Otsu*), γ0.06(α0.52,β0.48)
	PMVIF	0.63	0.30	0.15	14.87	0.49s	0.87	$T_{\vert\vec{e}\vert}$0.16(*Otsu*), γ0.06(α0.50,β0.50)
	Watershed	0.62	**0.06**	0.34	13.96	**0.08s**	0.96	Level 0.10
	SLIC	0.64	0.15	0.20	**13.87**	12.99s	0.88	Number of SuperPixels 20
	K-means	0.63	0.23	0.24	14.02	12.78s	0.63	Number of clusters 100
	Mean shift	0.62	0.29	0.28	13.92	147.71s	0.47	Bandwidth 0.02
	JSEG	0.66	0.30	**0.12**	18.69	84.38s	0.79	Color quantization 20
VOC2012 #2007_001430	PMLCD	**0.62**	0.09	0.17	18.75	0.41s	**0.95**	LCD radius 2, $T_{\vert\vec{e}\vert}$0.22(*Otsu*), γ0.00(α0.50,β0.50)
	PMVIF	0.60	0.14	0.20	16.59	0.75s	0.90	$T_{\vert\vec{e}\vert}$0.13(*Otsu*), γ0.00(α0.50,β0.50)
	Watershed	0.60	**0.08**	0.23	16.85	**0.07s**	0.90	Level 0.08
	SLIC	0.60	0.11	0.18	**16.05**	3.92s	0.90	Number of SuperPixels 30
	K-means	0.58	0.49	0.16	16.98	2.12s	0.41	Number of clusters 8
	Mean shift	0.57	0.47	0.18	18.11	53.50s	0.43	Bandwidth 0.07
	JSEG	**0.62**	0.16	**0.13**	21.92	60.72s	0.83	Color quantization 10
VOC2012 #2010_005626	PMLCD	**0.65**	**0.12**	0.17	17.43	0.39s	**0.90**	LCD radius 2, $T_{\vert\vec{e}\vert}$0.16(*Otsu*), γ0.30(α0.70,β0.30)
	PMVIF	0.60	0.24	0.20	16.24	0.64s	0.78	$T_{\vert\vec{e}\vert}$0.12(*Otsu*), γ0.30(α0.50,β0.50)
	Watershed	0.60	0.17	0.27	19.33	**0.07s**	0.83	Level 0.05
	SLIC	0.64	0.13	0.17	**15.54**	4.35s	0.88	Number of SuperPixels 20
	K-means	0.63	0.40	0.14	16.73	2.03s	0.78	Number of clusters 8
	Mean shift	**0.65**	0.38	**0.09**	19.75	1.96s	0.79	Bandwidth 0.25
	JSEG	0.64	0.20	0.14	20.19	71.54s	0.85	Color quantization 19
VOC2012 #2010_005746	PMLCD	**0.82**	**0.05**	**0.09**	7.65	0.19s	**0.93**	LCD radius 1, $T_{\vert\vec{e}\vert}$0.21(*Otsu*), γ0.00(α0.50,β0.50)
	PMVIF	0.73	**0.05**	0.12	**4.26**	0.45s	0.91	$T_{\vert\vec{e}\vert}$0.18(*Otsu*), γ0.00(α0.50,β0.50)
	Watershed	0.37	0.11	0.30	19.40	**0.07s**	0.82	Level 0.01
	SLIC	0.46	0.16	0.15	7.59	4.13s	0.75	Number of SuperPixels 5
	K-means	0.37	0.16	0.24	9.28	10.62s	0.74	Number of clusters 100
	Mean shift	0.41	0.17	0.23	13.43	373.74s	0.71	Bandwidth 0.02
	JSEG	0.46	0.14	0.14	14.47	36.85s	0.77	Color quantization 2
BSDS500 #2018	PMLCD	**0.90**	**0.21**	**0.09**	5.43	0.20s	**0.92**	LCD radius 1, $T_{\vert\vec{e}\vert}$0.24(*Otsu*), γ0.35(α0.86,β0.14)
	PMVIF	0.75	0.46	0.14	4.32	0.49s	**0.92**	$T_{\vert\vec{e}\vert}$0.19(*Otsu*), γ0.35(α0.52,β0.48)
	Watershed	0.84	0.31	0.16	7.88	**0.05s**	0.83	Level 0.04
	SLIC	0.89	0.22	0.10	**3.75**	3.54s	0.88	Number of SuperPixels 10
	K-means	0.82	0.61	0.16	4.14	2.33s	0.69	Number of clusters 10
	Mean shift	0.74	0.36	0.12	5.01	8.34s	0.73	Bandwidth 0.10
	JSEG	0.81	0.37	0.11	18.94	69.68s	0.59	Color quantization 30
BSDS500 #81095	PMLCD	**0.90**	0.17	**0.09**	9.55	0.35s	**0.89**	LCD radius 2, $T_{\vert\vec{e}\vert}$0.21(*Otsu*), γ0.12(α0.64,β0.36)
	PMVIF	0.81	0.26	0.14	9.64	0.38s	0.86	$T_{\vert\vec{e}\vert}$0.18(*Otsu*), γ0.12(α0.50,β0.50)
	Watershed	0.85	**0.15**	0.18	10.63	**0.06s**	0.84	Level 0.05
	SLIC	0.85	0.20	0.11	9.99	2.50s	0.86	Number of SuperPixels 10
	K-means	0.81	0.49	0.16	**8.77**	2.77s	0.64	Number of clusters 14
	Mean shift	0.82	0.50	0.14	10.41	38.64s	0.63	Bandwidth 0.08
	JSEG	0.84	0.30	0.11	16.63	47.30s	0.80	Color quantization 12
BSDS500 #107072	PMLCD	**0.85**	0.18	**0.10**	12.78	0.16s	**0.93**	LCD radius 1, $T_{\vert\vec{e}\vert}$0.22(*Otsu*), γ-0.05(α0.47,β0.53)
	PMVIF	0.48	0.33	0.13	15.34	0.27s	0.47	$T_{\vert\vec{e}\vert}$0.23(*Otsu*), γ-0.05(α0.50,β0.50)
	Watershed	0.75	**0.12**	0.25	15.50	**0.05s**	0.92	Level 0.15
	SLIC	0.76	**0.12**	0.16	12.27	3.89s	0.88	Number of SuperPixels 25
	K-means	0.75	0.34	0.17	12.33	3.61s	0.43	Number of clusters 20
	Mean shift	0.74	0.32	0.27	13.58	79.77s	0.41	Bandwidth 0.02
	JSEG	0.82	0.22	**0.10**	**8.99**	47.27s	0.73	Color quantization 10
BSDS500 #238025	PMLCD	**0.86**	0.10	**0.09**	13.60	0.37s	**0.96**	LCD radius 2, $T_{\vert\vec{e}\vert}$0.19(*Otsu*), γ0.15(α0.64,β0.36)
	PMVIF	0.69	0.31	0.10	15.95	0.28s	0.93	$T_{\vert\vec{e}\vert}$0.18(*Otsu*), γ0.15(α0.50,β0.50)
	Watershed	0.65	**0.06**	0.29	19.30	**0.06s**	0.94	Level 0.05
	SLIC	0.67	0.08	0.17	15.49	3.73s	0.95	Number of SuperPixels 30
	K-means	0.68	0.35	0.15	**12.80**	2.94s	0.67	Number of clusters 16
	Mean shift	0.71	0.47	0.12	13.41	50.11s	0.61	Bandwidth 0.05
	JSEG	0.73	0.24	0.11	15.27	42.22s	0.61	Color quantization 10
Average (Standard Deviation)	**PMLCD**	**0.78**	**0.13**	**0.12**	12.73	0.29s	**0.93**	LCD radius 1.50(0.50), γ0.14(0.15)
		(0.11)	(0.05)	(0.04)	(4.52)	(0.10)	(0.03)	
	PMVIF	0.66	0.26	0.15	12.15	0.47s	0.83	γ0.14(0.15)
		(0.10)	(0.12)	(0.03)	(4.98)	(0.16)	(0.14)	
	Watershed	0.66	**0.13**	0.25	15.36	**0.06s**	0.88	Level 0.0.07(0.04)
		(0.15)	(0.08)	(0.06)	(4.03)	(0.01)	(0.05)	
	SLIC	0.69	0.15	0.16	**11.82**	4.88s	0.87	Number of SuperPixels 18.75(8.93)
		(0.13)	(0.04)	(0.03)	(4.12)	(3.11)	(0.05)	
	K-means	0.66	0.38	0.18	11.88	4.90s	0.62	Number of clusters 34.50(38.01)
		(0.14)	(0.14)	(0.04)	(4.05)	(3.99)	(0.13)	
	Mean shift	0.66	0.37	0.18	13.45	94.22s	0.60	Bandwidth 0.08(0.07)
		(0.13)	(0.10)	(0.07)	(4.21)	(113.90)	(0.14)	
	JSEG	0.70	0.24	**0.12**	16.89	57.50s	0.75	Color quantization 14.13(8.01)
		(0.12)	(0.07)	(0.01)	(3.78)	(15.60)	(0.09)	

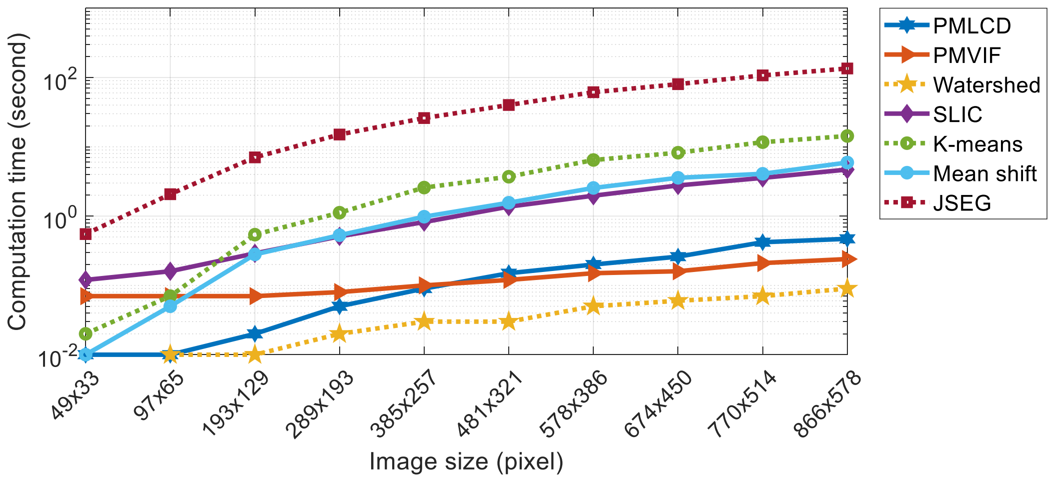

Figure 12. The comparison of the computation time of PMLCD and other methods.

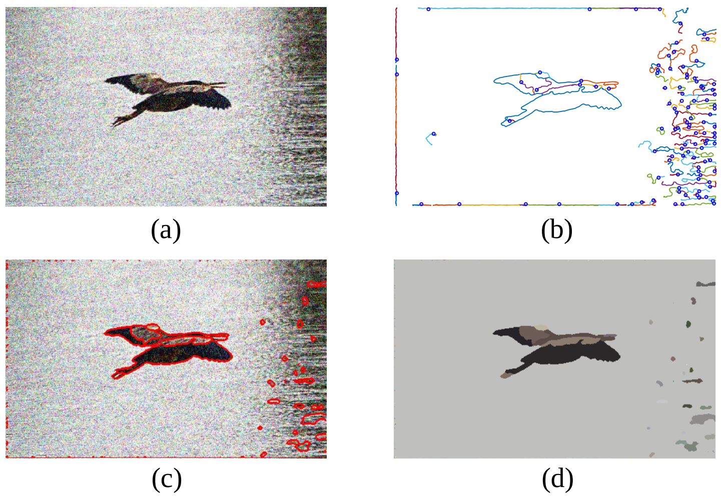

Figure 13. Noisy image segmentation results: (**a**) VOC2012 #2007_001289 image with SNR = 0 dB (σ_{noise} = 0.21), (**b**) particle trajectories obtained using PMLCD with a radius of LCD = 3, $T_{|\vec{e}|}$ = 0.27, γ = -0.14 (α = 0.34, β = 0.66), (**c**) extracted boundaries (red lines) and (**d**) segmented regions.

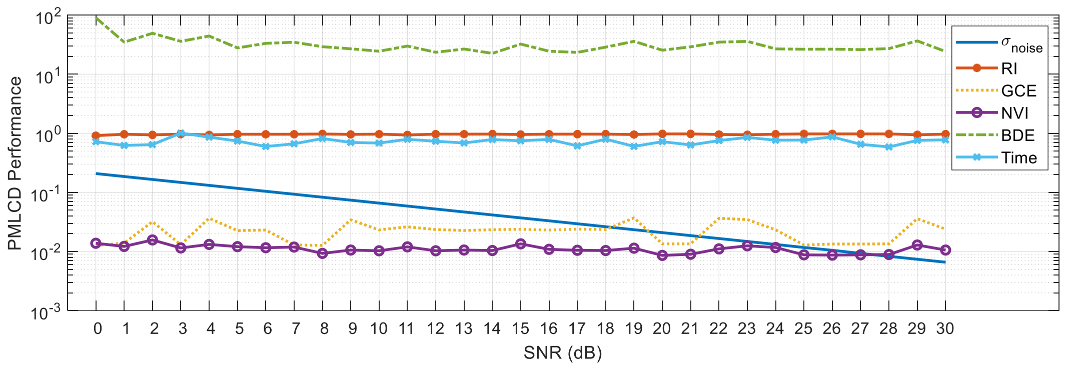

Figure 14. The benchmarks of the PMLCD method applied to the noisy VOC2012 #2007_001289 image.

5. Conclusions

The PMLCD color image segmentation algorithm is developed from the traditional method of particle motion in a vector image field (PMVIF) which uses two vector fields orthogonal to each other—namely a normal compressive vector field and an edge vector field—to force a particle to travel along object boundaries. Unlike the formulae previously used in the original PMVIF method, a normal compressive vector field is derived from the center-to-centroid vectors of local color distant images, whereas a gradient-like vector field is derived from center-to-centroid vectors of local color distant images in which each pixel is multiplied by the difference of the auxiliary pixels. An edge vector is then obtained by rotating each vector in a gradient-like vector field by 90° to achieve a Hamiltonian gradient-like field. In addition, for ease of use, a method for adjusting parameters related to particle movement, including $T_{|\vec{e}|}$, α, and β, is introduced. Experimental results show that the proposed method yields promising results, with better RI, GCE, NVI, and Dice measures as well as a faster computation time and good noise resistance. Since the proposed algorithm is based on color distance measurement, which can be applied to both scalar and vector images, it outperforms other grayscale-based methods, especially in regions in which edge information cannot be visualized in the grayscale image domain. Moreover, the method is not only useful for segmenting color images, but also can be used for all types of color space and vector images including multispectral and hyperspectral images.

Author Contributions: Conceptualization, W.P. and N.E.; methodology, N.E.; software, W.P.; validation, W.P. and N.E.; formal analysis, W.P.; investigation, W.P.; resources, N.E.; data curation, W.P. and N.E.; writing—original draft preparation, W.P.; writing—review and editing, N.E.; visualization, W.P.; supervision, N.E.

Funding: This research received no external funding.

Acknowledgments: The authors would like to thank the lecturers and staff of the Department of Computer Engineering. Faculty of Engineering Khon Kaen University for providing advice and support for this research.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Minaee, S.; Boykov, Y.; Porikli, F.; Plaza, A.; Kehtarnavaz, N.; Terzopoulos, D. Image Segmentation Using Deep Learning: A Survey, 2020, [arXiv:cs.CV/2001.05566].
2. MacQueen, J. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics; University of California Press: Berkeley, Calif., 1967; pp. 281–297.
3. Hamada, M.A.; Kanat, Y.; Abiche, A.E. Multi-Spectral Image Segmentation Based on the K-means Clustering. *International Journal of Innovative Technology and Exploring Engineering* **2019**, *9*, 1016–1019. doi:10.35940/ijitee.k1596.129219.
4. Fukunaga, K.; Hostetler, L.D. The Estimation of the Gradient of a Density Function, with Applications in Pattern Recognition. *IEEE Transactions on Information Theory* **1975**, *21*, 32–40. doi:10.1109/TIT.1975.1055330.
5. Xiao, W.; Zaforemska, A.; Smigaj, M.; Wang, Y.; Gaulton, R. Mean shift segmentation assessment for individual forest tree delineation from airborne lidar data. *Remote Sensing* **2019**, *11*, 1–19. doi:10.3390/rs11111263.
6. Deng, Y.; Manjunath, B.S. Unsupervised segmentation of color-texture regions in images and video. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2001**, *23*, 800–810. doi:10.1109/34.946985.
7. Aloun, M.S.; Hitam, M.S.; Yussof, W.N.J.H.W.; Abdul Hamid, A.A.K.; Bachok, Z. Modified JSEG algorithm for reducing over-segmentation problems in underwater coral reef images. *International Journal of Electrical and Computer Engineering* **2019**, *9*, 5244–5252. doi:10.11591/ijece.v9i6.pp5244-5252.
8. Sahba, F.; Tizhoosh, H.R.; Salama, M.M. A reinforcement learning framework for medical image segmentation. *IEEE International Conference on Neural Networks - Conference Proceedings* **2006**, pp. 511–517. doi:10.1109/ijcnn.2006.246725.
9. Wang, Z.; Sarcar, S.; Liu, J.; Zheng, Y.; Ren, X. Outline Objects using Deep Reinforcement Learning. *arXiv preprint arXiv:1804.04603* **2018**.

10. Casanova, A.; Pinheiro, P.O.; Rostamzadeh, N.; Pal, C.J. Reinforced active learning for image segmentation. International Conference on Learning Representations, 2020.
11. Alom, M.Z.; Taha, T.M.; Yakopcic, C.; Westberg, S.; Sidike, P.; Nasrin, M.S.; Hasan, M.; Van Essen, B.C.; Awwal, A.A.; Asari, V.K. A state-of-the-art survey on deep learning theory and architectures. *Electronics (Switzerland)* **2019**, *8*, 1–67. doi:10.3390/electronics8030292.
12. K. Bhargavi.; S. Jyothi. A Survey on Threshold Based Segmentation Technique in Image Processing. *International Journal of Innovative Research & Development* **2014**, *3*, 234–39.
13. Doyle, W. Operations Useful for Similarity-Invariant Pattern Recognition. *J. ACM* **1962**, *9*, 259–267. doi:10.1145/321119.321123.
14. Otsu, N. A Threshold Selection Method From Gray-Level Histograms. *IEEE Trans Syst Man Cybern* **1979**, *SMC-9*, 62–66. doi:10.1109/tsmc.1979.4310076.
15. Wang, W.; Duan, L.; Wang, Y. Fast Image Segmentation Using Two-Dimensional Otsu Based on Estimation of Distribution Algorithm. *Journal of Electrical and Computer Engineering* **2017**, *2017*, 1735176. doi:10.1155/2017/1735176.
16. Nakagawa, Y.; Rosenfeld, A. Some experiments on variable thresholding. *Pattern Recognition* **1979**, *11*, 191–204. doi:10.1016/0031-3203(79)90006-2.
17. Chow, C.; Kaneko, T. Boundary Detection of Radiographic Images by a Threshold Method. Frontiers of Pattern Recognition, 1971, pp. 1530–1535. doi:10.1016/B978-0-12-737140-5.50009-9.
18. Chow, C.K.; Kaneko, T. Automatic Boundary from Detection of the Cineangiograms " Left Ventricle. *Image (Rochester, N.Y.)* **1972**, pp. 388–410.
19. Kimmel, R.; Bruckstein, A.M. Regularized Laplacian zero crossings as optimal edge integrators, 2003. doi:10.1023/A:1023030907417.
20. Baştan, M.; Bukhari, S.S.; Breuel, T. Active Canny: Edge detection and recovery with open active contour models. *IET Image Processing* **2017**, *11*, 1325–1332, [1609.03415]. doi:10.1049/iet-ipr.2017.0336.
21. Eua-Anant, N.; Udpa, L. A novel boundary extraction algorithm based on a vector image model. Midwest Symposium on Circuits and Systems, 1996, Vol. 2, pp. 597–600. doi:10.1109/mwscas.1996.587797.
22. Eua-anant, N.; Lalita, U.; Upda, L. Boundary extraction algorithm based on particle motion in a vector image field. Proceedings of International Conference on Image Processing, 1997, Vol. 2, pp. 732–735 vol.2. doi:10.1109/ICIP.1997.638600.
23. Eua-Anant, N.; Udpa, L. Boundary detection using simulation of particle motion in a vector image field, 1999. doi:10.1109/83.799884.
24. Ma, W.Y.; Manjunath, B.S. EdgeFlow : A Technique for Boundary Detection and Image Segmentation, 2000.
25. Yang, F.; Bruckstein, A.M.; Cohen, L.D. PointFlow: A model for automatically tracing object boundaries and inferring illusory contours, 2018. doi:10.1007/978-3-319-78199-0_32.
26. Beucher, S.; Lantuejoul, C. Use of Watersheds in Contour Detection, 1979. doi:citeulike-article-id:4083187.
27. Kornilov, A.S.; Safonov, I.V. An overview of watershed algorithm implementations in open source libraries. *Journal of Imaging* **2018**, *4*. doi:10.3390/jimaging4100123.
28. Greig, D.M.; Porteous, B.T.; Seheult, A. Exact Maximum A Posteriori Estimation for Binary Images. *Journal of the Royal Statistical Society, Series B* **1989**, *51*, 271–279. doi:10.1111/j.2517-6161.1989.tb01764.x.
29. Wang, T.; Ji, Z.; Sun, Q.; Chen, Q.; Ge, Q.; Yang, J. Diffusive likelihood for interactive image segmentation. *Pattern Recognition* **2018**, *79*, 440–451. doi:10.1016/j.patcog.2018.02.023.
30. Shi, J.; Malik, J. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **2000**, *22*, 888–905. doi:10.1109/34.868688.
31. Xu, J.; Janowczyk, A.; Chandran, S.; Madabhushi, A. A weighted mean shift, normalized cuts initialized color gradient based geodesic active contour model: applications to histopathology image segmentation. Medical Imaging 2010: Image Processing, 2010, Vol. 7623, p. 76230Y. doi:10.1117/12.845602.
32. Ren, X.; Malik, J. Learning a classification model for segmentation. *Proceedings of the IEEE International Conference on Computer Vision* **2003**, *1*, 10–17. doi:10.1109/iccv.2003.1238308.
33. Daoud, M.I.; Atallah, A.A.; Awwad, F.; Al-Najjar, M.; Alazrai, R. Automatic superpixel-based segmentation method for breast ultrasound images. *Expert Systems with Applications* **2019**, *121*, 78–96. doi:10.1016/j.eswa.2018.11.024.

34. Achanta, R.; Shaji, A.; Smith, K.; Lucchi, A. SLIC Superpixels Compared to State-of-the-Art Superpixel Methods. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* **2012**, *34*, 1–1. doi:10.1109/vr.2012.6180941.
35. Li, Z.; Chen, J. Superpixel segmentation using Linear Spectral Clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* **2015**, *07-12-June*, 1356–1363. doi:10.1109/CVPR.2015.7298741.
36. Hu, M.K. Visual Pattern Recognition by Moment Invariants, 1962. doi:10.1109/TIT.1962.1057692.
37. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (VOC) challenge. *International Journal of Computer Vision* **2010**, *88*, 303–338. doi:10.1007/s11263-009-0275-4.
38. Martin, D.; Fowlkes, C.; Tal, D.; Malik, J. A Database of Human Segmented Natural Images and its Application to Evaluating Segmentation Algorithms and Measuring Ecological Statistics. Proc. 8th Int'l Conf. Computer Vision, 2001, Vol. 2, pp. 416–423. doi:10.1109/ICCV.2001.937655.
39. Majid, H.; Hadi Yazdani, B. Color Image Segmentation Metrics. In *Encyclopedia of Image Processing*; CRC Press, 2019. doi:10.1201/9781351032742-140000183.
40. Reichart, R.; Rappoport, A. The NVI clustering evaluation measure. *CoNLL 2009 - Proceedings of the Thirteenth Conference on Computational Natural Language Learning* **2009**, pp. 165–173. doi:10.3115/1596374.1596401.