Hindawi
Journal of Electrical and Computer Engineering
Volume 2017, Article ID 7347421, 15 pages
https://doi.org/10.1155/2017/7347421

Research Article

Visual Sensor Based Image Segmentation by Fuzzy Classification and Subregion Merge

Huidong He,[1] Xiaoqian Mao,[1] Wei Li,[2] Linwei Niu,[3] and Genshe Chen[4]

[1] *School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China*
[2] *Department of Computer & Electrical Engineering and Computer Science, California State University, Bakersfield, CA 93311, USA*
[3] *Department of Math and Computer Science, West Virginia State University, Institute, WV 25112, USA*
[4] *Intelligent Fusion Technology, Inc., Germantown, MD 20876, USA*

Correspondence should be addressed to Wei Li; wli@csub.edu

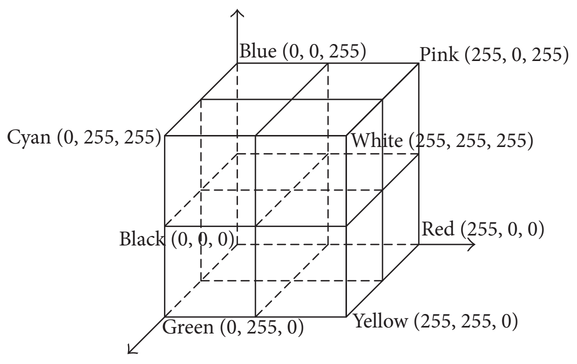
Received 16 June 2017; Revised 4 August 2017; Accepted 9 August 2017; Published 25 September 2017

Academic Editor: Ping Feng Pai

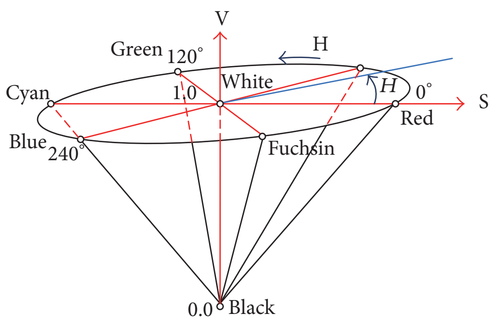
The extraction and tracking of targets in an image shot by visual sensors have been studied extensively. The technology of image segmentation plays an important role in such tracking systems. This paper presents a new approach to color image segmentation based on fuzzy color extractor (FCE). Different from many existing methods, the proposed approach provides a new classification of pixels in a source color image which usually classifies an individual pixel into several subimages by fuzzy sets. This approach shows two unique features: the spatial proximity and color similarity, and it mainly consists of two algorithms: CreateSubImage and MergeSubImage. We apply the FCE to segment colors of the test images from the database at UC Berkeley in the RGB, HSV, and YUV, the three different color spaces. The comparative studies show that the FCE applied in the RGB space is superior to the HSV and YUV spaces. Finally, we compare the segmentation effect with Canny edge detection and Log edge detection algorithms. The results show that the FCE-based approach performs best in the color image segmentation.

1. Introduction

Image segmentation is a fundamental issue in image processing and computer vision as it is a preliminary process in many applications that need to segment an image into meaningful regions. Applications of segmenting gray-level images are very limited so studies on color image segmentation become interesting because color images provide much more information than gray-level ones. A number of image segmentation techniques have been proposed in past decades [1–3]. A physical model in [4] is proposed for color image understanding. A widely used technique for color image segmentation is based on the idea of region growing. A color segmentation algorithm combining region growing and region merging processes is discussed in [5]. The fuzzy growing and expanding approach presented in [6] uses histogram tables for fine segmentation. A method for segmentation of color texture region using the defined class maps is presented in [7]. An approach to contrast analysis using fuzzy region growing is documented in [8]. Color image segmentation using seeded region growing is presented in [9–11]. Most pixel clustering algorithms work in the RGB color space in which the pixels are represented by red, green, and blue components. Clustering methods [12, 13], for example, c-means fuzzy classification [14], are often considered as an unsupervised classification of pixels as a prior knowledge of the image. Both supervised and unsupervised segmentation of textured color images are presented in [15, 16]. The level setting approaches to process a color image for segmentation are discussed in [17, 18]. All of the proposed techniques above provide a crisp segmentation of images, where each pixel is classified into a unique subset. This classification may not reflect an understanding of images by humans very well, as the work [1] stated "the image segmentation problem is basically one of psychophysical perception, and therefore not susceptible to a purely analytical solution." Probably, one of



(a) RGB color space



(b) HSV color space

Figure 1: Color distribution in the RGB and HSV color spaces.

the difficulties in color image segmentation is how to define color conception, since an understanding of colors in an image varies by different observers.

Considering uncertainty and ambiguity in many real applications, this paper presents an approach to color image segmentation based on a fuzzy color extractor (FCE). Its preliminary version appears in [19]. The FCE is directly extended from the fuzzy gray-level extractor, which was applied to recognize landmarks on roads for autonomous vehicle navigation in [20–22]; that is, the fuzzy rules for extracting a road from a gray-level image are as follows:

If the gray level of a pixel is close to the gray-level of the road,

Then one can make it black, or

Else one can make it to white.

If these are modified to the rules for extracting an object from a color image, then

if the color components of a pixel closely matches the color components of an object,

then one can extract this pixel as a *Matched* pixel of the object, or

else one can extract this pixel as an *Unmatched* pixel.

The FCE viewed as a mapping operator of measuring colors is able to extract colors that are similar to a color pattern (seed). If a pixel is extracted it is defined as a *Matched* pixel; otherwise it is defined as an *Unmatched* pixel. In doing this, we develop procedures IniSeedList and UpdateSeedList for automatic selection of seeds. Our approach mainly consists of two algorithms: CreateSubImage and MergeSubImage. The CreateSubImage algorithm always extracts the seed as the first *Matched* pixel into a subimage and continually uses the FCE to check neighbors of each *Matched* pixel to grow the subimage. The subimage is created when neighbors' colors of all *Matched* pixels are "not close to" the seed's color. The CreateSubImage algorithm iteratively invokes UpdateSeedList to select new seeds to create next subimages until all the pixels in the image are processed. Different from all of the color segmentation techniques discussed above, the CreateSubImage algorithm most likely classifies an individual pixel into several subimages by fuzzy sets. As a result, two created subimages may share a common area, which can be considered as a color similarity metric for merging the subimages. The MergeSubImage algorithm first removes insignificant subimages from the set of subimages and merges significant subimages in order to extract meaningful segmented images. The insignificant subimages usually have a "too small" or a "too large" ratio of their area sizes over the source image. A "too small" insignificant subimage is mainly caused by a tiny fragment or a noise pixel in the source image, while a "too large" insignificant image is created by a "bad" seed. Then, the algorithm merges two subimages together according to their color similarity described by the size of their common pixels. We tested the proposed approach using exemplar color images from the database at UC Berkeley [23] in three different color spaces and discussed its effectiveness and robustness by comparing segmentation effect with two edge detection algorithms, namely, Canny edge detection and Log edge detection.

2. Three Color Spaces

In many color-based computer vision applications, it is very important to segment colors in an image, but it is very difficult due to the fact that colors are ambiguous in uncertain environments. In this study, we apply the FCE to segment colors in the image represented in the RGB, HSV, and YUV spaces.

2.1. RGB Color Space. RGB is the most traditional color space and is widely used in image segmentation because of its simplicity and fast processing [24, 25]. Colors consist of red (R), green (G), and blue (B) components, as shown in Figure 1(a).

2.2. HSV Color Space. HSV is another color space widely used for image processing, where a color is decomposed into hue (H), saturation (S), and value (V). The distribution of color in the HSV color space is shown in Figure 1(b). The color components in the RGB space can be transformed into the HSV space by

$$\begin{aligned}
\max &= \max(\mathrm{R},\mathrm{G},\mathrm{B}),\\
\min &= \min(\mathrm{R},\mathrm{G},\mathrm{B}),\\
\mathrm{V} &= \max\\
\mathrm{S} &= \frac{(\max-\min)}{\max}\times 255,\\
\mathrm{H} &= \begin{cases} \dfrac{(\mathrm{G}-\mathrm{B})}{(\max-\min)}\times 60 & \mathrm{R}=\max\\ 120+\dfrac{(\mathrm{B}-\mathrm{R})}{(\max-\min)}\times 60 & \mathrm{G}=\max\\ 240+\dfrac{(\mathrm{R}-\mathrm{G})}{(\max-\min)}\times 60 & \mathrm{B}=\max \end{cases}\\
\mathrm{H} &= \mathrm{H}+360, \quad \text{if } \mathrm{H}<0,\\
\mathrm{H} &= \mathrm{H}\times 255\div 360.
\end{aligned} \tag{1}$$

Usually, the RGB values are between 0 and 255, so the HSV values are also between 0 and 255 through the transformation by (1).

2.3. YUV Color Space. YUV is the color space which is used in an analog color TV system, which translates color image components obtained in the RGB space into the luminance Y and the colors B -Y (U) and R - Y (V) by

$$\begin{aligned}
\mathrm{Y} &= 0.299\mathrm{R}+0.587\mathrm{G}+0.114\mathrm{B},\\
\mathrm{U} &= (-0.1687\mathrm{R}-0.3313\mathrm{G}+0.500\mathrm{B})+128,\\
\mathrm{V} &= (0.500\mathrm{R}-0.4187\mathrm{G}-0.0813\mathrm{B})+128.
\end{aligned} \tag{2}$$

The YUV values are between 0 and 255 through the transformation by (2). The advantage of using the YUV color space for color segmentation is that the luminance Y is separated from the colors U and V.

3. Fuzzy Color Extractor

The fuzzy color extractor proposed here enables us to deal with such uncertainty. We will only discuss how to use the FCE-based algorithm defined in the RGB color space to segment color images, because the algorithm can directly process the color images in the HSV and YUV spaces, which are converted from the RGB space. Figure 1(a) shows the RGB space, where colors are represented by their red, green, and blue components in an orthogonal Cartesian space. The upper-left corner in the image is the origin. The color of each pixel $p(n,m)$ denoted by $p(n,m)_{\mathrm{RGB}}$ is processed to separate its red, green, and blue components ($p(n,m)_{\mathrm{R}}$, $p(n,m)_{\mathrm{G}}$, $p(n,m)_{\mathrm{B}}$). The FCE extracts a cluster of colors based on a defined color pattern (seed) denoted by $\mathrm{seed}_{\mathrm{RGB}}$, $\mathrm{seed}_{\mathrm{RGB}}$ and is either directly defined by its RGB components ($\mathrm{seed}_{\mathrm{R}}$, $\mathrm{seed}_{\mathrm{G}}$, $\mathrm{seed}_{\mathrm{B}}$) or determined by a pixel in the image. The color components' differences between a pixel $p(n,m)_{\mathrm{RGB}}$ and $\mathrm{seed}_{\mathrm{RGB}}$ are calculated as follows:

$$\begin{aligned}
\mathrm{dif}(n,m)_{\mathrm{R}} &= p(n,m)_{\mathrm{R}}-\mathrm{seed}_{\mathrm{R}},\\
\mathrm{dif}(n,m)_{\mathrm{G}} &= p(n,m)_{\mathrm{G}}-\mathrm{seed}_{\mathrm{G}},\\
\mathrm{dif}(n,m)_{\mathrm{B}} &= p(n,m)_{\mathrm{B}}-\mathrm{seed}_{\mathrm{B}},\\
&\qquad 0\le m<M,\ 0\le n<N.
\end{aligned} \tag{3}$$

M and N represent the size of an array which holds an image. The corresponding fuzzy rules can be obtained as follows:

If $\mathrm{dif}(n,m)_{\mathrm{R}}$ and $\mathrm{dif}(n,m)_{\mathrm{G}}$ and $\mathrm{dif}(n,m)_{\mathrm{B}}$ are *Zero*,
Then $p(n,m)$ is *Matched*.
If $\mathrm{dif}(n,m)_{\mathrm{R}}$ or $\mathrm{dif}(n,m)_{\mathrm{G}}$ or $\mathrm{dif}(n,m)_{\mathrm{B}}$ is *Negative* or *Positive*,
Then $p(n,m)$ is *Unmatched*.

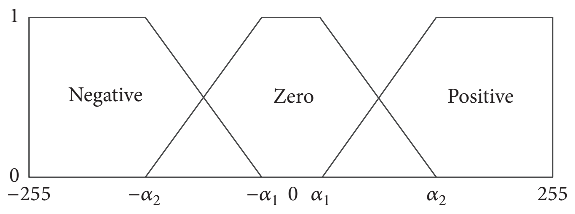
Both the rules indicate that $p(n,m)$ belongs to the object to be extracted, if the Euclidean distances between $p(n,m)_{\mathrm{RGB}}$ and $\mathrm{seed}_{\mathrm{RGB}}$ along the three axes in RGB coordinate system are small enough; otherwise, $p(n,m)$ does not belong to the object.

Figure 2(a) shows the membership functions ($\mu_{\mathrm{N}}(x)$, $\mu_{\mathrm{Z}}(x)$, $\mu_{\mathrm{P}}(x)$) for the input fuzzy variables *(Negative, Zero, Positive)* defined by

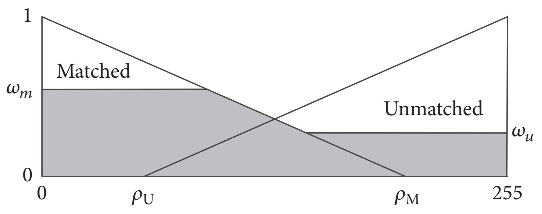
$$\begin{aligned}
\mu_{\mathrm{N}}(x) &= \begin{cases} 1 & -255\le x<-\alpha_2\\ \dfrac{x+\alpha_1}{\alpha_1-\alpha_2} & -\alpha_2\le x<-\alpha_1\\ 0 & -\alpha_1\le x\le 255, \end{cases}\\
\mu_{\mathrm{Z}}(x) &= \begin{cases} 0 & -255\le x<-\alpha_2\\ \dfrac{x+\alpha_2}{\alpha_2-\alpha_1} & -\alpha_2\le x<-\alpha_1\\ 1 & -\alpha_1\le x\le \alpha_1\\ \dfrac{x-\alpha_1}{\alpha_2-\alpha_1} & \alpha_1\le x<\alpha_2\\ 0 & \alpha_2\le x\le 255, \end{cases}\\
\mu_{\mathrm{P}}(x) &= \begin{cases} 0 & -255\le x<-\alpha_1\\ \dfrac{x+\alpha_2}{\alpha_2-\alpha_1} & \alpha_1\le x<\alpha_2\\ 1 & \alpha_2\le x\le 255, \end{cases}
\end{aligned} \tag{4}$$

and Figure 2(b) shows the membership functions ($\mu_{\mathrm{M}}(x)$, $\mu_{\mathrm{U}}(x)$) for the output fuzzy variables *(Matched, Unmatched)* defined by

$$\begin{aligned}
\mu_{\mathrm{M}}(x) &= \begin{cases} \dfrac{\rho_{\mathrm{M}}-x}{\rho_{\mathrm{M}}} & 0\le x<\rho_{\mathrm{M}}\\ 0 & \rho_{\mathrm{M}}\le x\le 255, \end{cases}\\
\mu_{\mathrm{U}}(x) &= \begin{cases} 0 & 0\le x<\rho_{\mathrm{U}}\\ \dfrac{x-\rho_{\mathrm{U}}}{255-\rho_{\mathrm{U}}} & \rho_{\mathrm{U}}\le x\le 255, \end{cases}
\end{aligned} \tag{5}$$

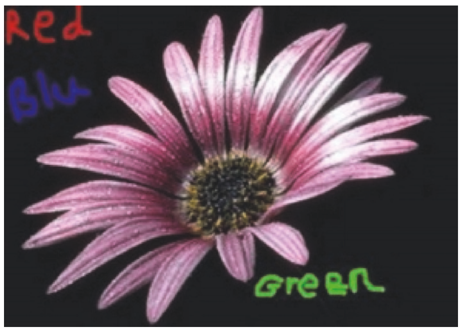


(a) Membership functions for color differences

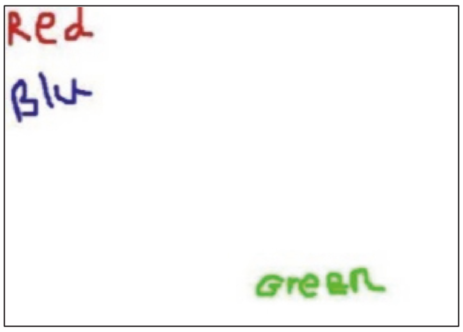


(b) Membership functions for defuzzification

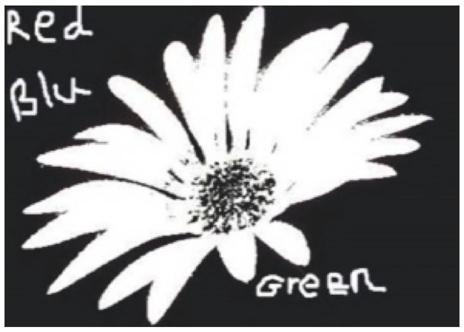
FIGURE 2: Definition of membership functions for the FCE.



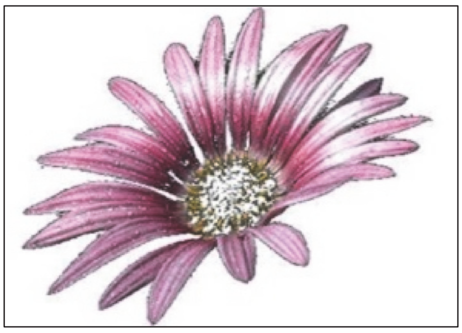
(a) I_{FLOWER}



(b) $\{I_{\text{RED}}, I_{\text{GREEN}}, I_{\text{BLUE}}\}$



(c) $I_{\text{BACKGROUND}}$



(d) I_{PETAL}

FIGURE 3: Extraction of colors in I_{FLOWER} image by FCE.

where $\rho_M + \rho_U = 255$. Based on $\text{dif}(n,m)_R$, $\text{dif}(n,m)_G$, and $\text{dif}(n,m)_B$, the fuzzy rules produce the weight ω_m for *Matched* and the weight ω_u for *Unmatched* by

$$\omega_m = \min\{\mu_Z(R), \mu_Z(G), \mu_Z(B)\},$$
$$\omega_u = \max\{\mu_N(R), \mu_N(G), \mu_N(B), \mu_P(R), \mu_P(G), \mu_P(B)\}. \quad (6)$$

Figure 2(b) shows the produced areas in the output domain, while ω_m and ω_u cut $\mu_M(x)$ and $\mu_U(x)$. A crisp output value, $\Delta\rho_F$, is calculated by the centroid defuzzification method:

$$\Delta\rho_F = \frac{\int \mu_{\text{out}}(x)\,x\,dx}{\int \mu_{\text{out}}(x)\,dx}, \quad (7)$$

where $\mu_{\text{out}}(x)$ represents the envelope function of the areas cut by ω_m and ω_u in fuzzy output domain. If $\Delta\rho_F < \sigma$, $p(n,m)$ is extracted, $p(n,m)$ is not extracted, where σ is a threshold. The FCE can be understood as a mapping operator between Euclidean distances $\{\text{dif}(n,m)_R, \text{dif}(n,m)_G, \text{dif}(n,m)_B\}$ in the RGB space and a difference $\Delta\rho_F$ in the intensity space under a fuzzy metric.

The example in Figure 3 shows how to create some subimages with interesting colors from the source image I_{FLOWER} by the FCE. First, we define three seeds: $\text{seed}_{\text{RED}} = (255, 0, 0)$, $\text{seed}_{\text{GREEN}} = (0, 255, 0)$, and $\text{seed}_{\text{BLUE}} = (0, 0, 255)$ to create three subimages $\{I_{\text{RED}}, I_{\text{GREEN}}, I_{\text{BLUE}}\}$ that hold the words "red," "green," and "blue" from the source image I_{FLOWER}, as shown in Figure 3(b). Also, we select a seed $p(0,0)_{\text{RGB}} = (6, 6, 8)$ at the origin to remove the background from the source image I_{FLOWER} as shown in Figure 3(c). Figure 3(d) shows $I_{\text{PETAL}} = I_{\text{FLOWER}} - (I_{\text{RED}} \cup I_{\text{GREEN}} \cup I_{\text{BLUE}} \cup I_{\text{BACKGROUND}})$ that holds the color components of the flower petal.

Table 1: Definitions of the eight subspaces shown in Figure 1(a).

Vertex i	Subspace	Red-axis	Green-axis	Blue-axis
${v^1}_{\text{RGB}}$ (0, 0, 0)	ν_black	[0, 127]	[0, 127]	[0, 127]
${v^2}_{\text{RGB}}$ (0, 0, 255)	ν_blue	[0, 127]	[0, 127]	[128, 255]
${v^3}_{\text{RGB}}$ (0, 255, 0)	ν_green	[0, 127]	[128, 255]	[0, 127]
${v^4}_{\text{RGB}}$ (0, 255, 255)	ν_cyan	[0, 127]	[128, 255]	[128, 255]
${v^5}_{\text{RGB}}$ (255, 0, 0)	ν_red	[128, 255]	[0, 127]	[0, 127]
${v^6}_{\text{RGB}}$ (255, 0, 255)	ν_pink	[128, 255]	[0, 127]	[128, 255]
${v^7}_{\text{RGB}}$ (255, 255, 0)	ν_yellow	[128, 255]	[128, 255]	[0, 127]
${v^8}_{\text{RGB}}$ (255, 255, 255)	ν_white	[128, 255]	[128, 255]	[128, 255]

```
IniSeedList ($I_{\text{IMG}}$)
  for $i \leftarrow 1$ to 8 do
      pixel_num$_i \leftarrow 0$;
  for $n \leftarrow 1$ to $N$ do
      for $m \leftarrow 1$ to $M$ do
          Calculate $d_i(n, m)$ in (8);
          $d_k(n, m) \leftarrow \min\{d_i(n, m)\}$;
          pixel_num$_k \leftarrow$ pixel_num$_k + 1$;
  Find seed_cand$^{(i)}$ in each subspace;
      // with the shortest distance to its vertex $i$.
  Push seed_cand$^{(i)}$ into SeedList;
  Sort seed_cand$^{(i)}$;
      // according to pixel number classified in the subspaces.
return SeedList;
```

Pseudocode 1

4. Creating Subimages Using FCE

This section discusses the CreateSubImage algorithm for creating a set of subimages from a color image using the FCE in the RGB space. A critical step in this algorithm is to select seeds for the FCE. For this purpose, we evenly divide the RGB space into the eight subspaces {ν_black, ν_red, ν_green, ν_blue, ν_yellow, ν_pink, ν_cyan, ν_white}, which are defined by the well-defined colors: {black, red, green, blue, yellow, pink, cyan, white}, at their corresponding vertices of the RGB space, as shown in Figure 1(a). Table 1 lists their definitions; for example, ν_black is defined by a cubic $[0, 127] \times [0, 127] \times [0, 127]$ on the red-axis, green-axis, and blue-axis.

We calculate the distances of each pixel's RGB components to the eight well-defined colors by

$$d_i(n, m) = \sum \left[p(n, m)_{\text{RGB}} - {v^i}_{\text{RGB}}\right]^2 \quad i = 1, 2, \ldots, 8, \quad (8)$$

where n and m are the index of a pixel in the image, $p(n, m)_{\text{RGB}}$ is the color components of the pixel located at (n, m), ${v^i}_{\text{RGB}}$ is the color components of vertex i in the RGB space, and $d_i(n, m)$ is the distance between the pixel's color and vertex i.

We developed two procedures, IniSeedList and Update-SeedList, to initialize a seed list and to update it. For an input image, IniSeedList classifies each pixel into one of the eight subspaces according to its minimum distance to vertex i that is a well-defined color in the RGB space. In each subspace, IniSeedList selects such a pixel as a seed candidate, seed_cand$^{(i)}$, that has the shortest distance to its vertex i. Then, IniSeedList inserts seed_cand$^{(i)}$ into the seed list in order of the number of pixels classified in the subspaces and returns it. UpdateSeedList checks if seed_cand$^{(i)}$ is popped out or is extracted by the FCE. If so, UpdateSeedList updates seed_cand$^{(i)}$ in its corresponding subspace. The pseudocode of the IniSeedList procedures is shown in Pseudocode 1 and the pseudocode of the Update-SeedList is shown in Pseudocode 2.

Table 2 shows an example of a seed list generated by IniSeedList from the color image, I_{BIRD}. Columns 1 and 2 in Table 2 indicate the defined subspaces and the number of the classified pixels in the subspaces, column 3 shows the order of seed_cand$^{(i)}$ in the list, and columns 4 and 5 list the RGB color components and locations of seed_cand$^{(i)}$ in I_{IMG}, respectively. The top seed in Table 2 is seed_cand$^{(1)}$ = (244, 251, 243) with location (269, 271) from the subspace ν_white holding 100,369 pixels. After the algorithm pops out this seed for creating ${I_{\text{M}}}^{(i)}$, UpdateSeedList produces a new seed candidate in the subspace ν_white from ${I_{\text{U}}}^{(1)}$ and inserts it on the top of the list. The new seed_cand$^{(1)}$ = (240, 251, 247) is located at (283, 267) in ${I_{\text{U}}}^{(1)}$ and is also from the subspace ν_white with 105,045 pixels. Please note that the

Table 2: The seed list generated from the color image, I_{BIRD}.

Subspace	Number of pixels	seed_cand$^{(i)}$ order	seed_cand$^{(i)}$ colors	seed_cand$^{(i)}$ locations (n, m)
ν_white	100369	1	(244, 251, 243)	(269, 271)
ν_black	19393	2	(0, 6, 0)	(193, 233)
ν_cyan	70	3	(151, 100, 57)	(216, 190)
ν_red	26	4	(206, 166, 107)	(225, 204)
ν_blue	0	N/A	N/A	N/A
ν_yellow	0	N/A	N/A	N/A
ν_green	0	N/A	N/A	N/A
ν_pink	0	N/A	N/A	N/A

```
UpdateSeedList ($I_{\text{IMG}}$, SeedList)
    subspace_index ($i$) ← 0;
    for $i$ ← 1 to 8 do
        if (seed_cand$^{(i)} \in I_{\text{IMG}}$)
            subspace_index ($i$) ← 1;
        else
            Remove seed_cand$^{(i)}$ from SeedList;
    for $i$ ← 1 to 8 do
        if subspace_index ($i$) = 0
            Find seed_cand$^{(i)}$ in subspace $i$;
                // with the shortest distance to its vertex $i$.
            Push seed_cand$^{(i)}$ into SeedList;
    Sort seed_cand$^{(i)}$;
        // according to pixel number classified in the subspaces.
return SeedList;
```

Pseudocode 2

input image to IniSeedList is $I_{\text{IMG}} = I_{\text{BIRD}} - I_{\text{CENTER}}$, instead of I_{BIRD} itself. I_{CENTER} is an image extracted by $\text{seed}_{\text{CENTER}} = (127, 127, 127)$ that is the center of the RGB space. It is difficult to determine in which subspace the center color is classified, so any seed close to the center may not be able to differentiate colors very well, as shown in Figure 4(a). In order to guarantee to get a "good" seed, our approach does not select any seed falling in I_{CENTER}. It is also possible that there is no pixel in a subspace; that is, the seed candidates are less than eight. For example, the seed candidates for I_{BIRD} are not available in the subspaces ν_blue, ν_yellow, ν_green, and ν_pink.

The pseudocode of the CreateSubImage procedures is listed as shown in Pseudocode 3.

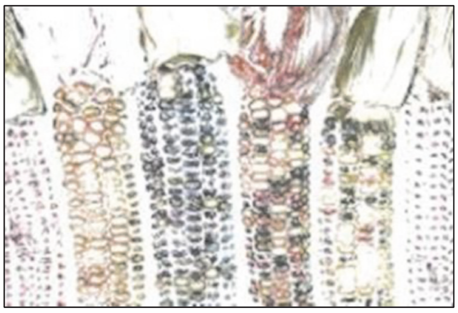
CreateSubImage creates a set of subimages from a color source image. CreateSubImage invokes mainly three procedures: IniSeedList, UpdateSeedList, and FuzExtractor. The pseudocode of the FuzExtractor is shown in Pseudocode 4.

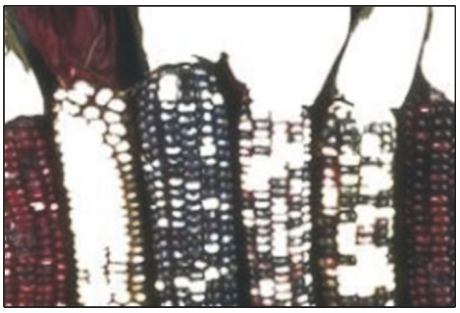
CreateSubImage has two arguments: I_{SRC} and center, where I_{SRC} is a source image and center is a Boolean variable. If center = true, I_{CENTER} is created by $\text{seed}_{\text{CENTER}}$, otherwise I_{CENTER} is assigned by the empty image $\emptyset$. FuzExtractor is mainly implemented based on (3) and extracts a subimage I_{TEMP}. The FuzExtractor procedure starts with seed_{RGB} to generate I_{MATCH} with region growing. Because seed_{RGB} belongs to its subimage I_{MATCH}, FuzExtractor pushes its four neighbor pixels into a queue and checks the first-in element, $\text{pixel}_{\text{FIRST}}$, in the queue to see if it belongs to the subimage I_{MATCH}. If $\text{pixel}_{\text{FIRST}}$ belongs to I_{MATCH}, its new four neighbors are pushed into the queue. The FuzExtractor procedure repeats this process until the queue is empty, which indicates that there are no more pixels in the source image I_{SRC} matching the seed_{RGB} color. After FuzExtractor returns I_{MATCH}, CreateSubImage assigns I_{MATCH} to I_{TEMP} as described.

CreateSubImage uses two conditions to check if I_{TEMP} is significant or not. The first condition is to check $I_{\text{TEMP}}/I_{\text{SRC}}$ that it is a ratio of I_{TEMP} area over I_{SRC} one. If this ratio is too small, CreateSubImage ignores I_{TEMP} as I_{TEMP} holds too few pixels extracted from the source image I_{SRC}. I_{TEMP} is most likely affected by noise and is not significant for any object in the source image. CreateSubImage also uses the condition, $(I_{\text{TEMP}}/I_{\text{SRC}})$ & var (I_{TEMP}), to check I_{TEMP}, where var (I_{TEMP}) is a variance defined by

$$\varpi = E\left[\left(I_{\text{TEMP}}\right)^2\right],$$
$$\operatorname{var}\left(I_{\text{TEMP}}\right) = E\left[\left(I_{\text{TEMP}} - \varpi\right)^2\right]. \quad (9)$$

A big $I_{\text{TEMP}}/I_{\text{SRC}}$ indicates a large area size of I_{TEMP} and a big var (I_{TEMP}) implies that I_{TEMP} contains the color components


(a) I_{CENTER}


(b) I_{TEMP}

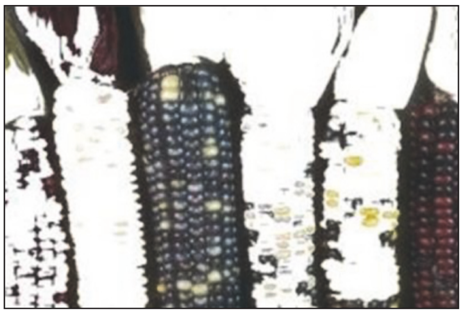

(c) $I_{\text{UNCLASSIED}}$

Figure 4: (a) The image I_{CENTER} extracted by the seed, $\text{seed}_{\text{CENTER}} = (127, 127, 127)$. (b) An insignificant image, I_{TEMP}, extracted by a "bad" seed. (c) The image $I_{\text{UNCLASSIED}}$.

```
CreateSubImage ($I_{SRC}$, center)
    $i \leftarrow 0$; $I_M^{(i+1)} \leftarrow ø$; $I_U^{(i)} \leftarrow I_{SRC}$;
    If (center = true)
        $I_{CENTER} \leftarrow$ FuzExtractor ($I_{SRC}$, $seed_{CENTER}$);
    else $I_{CENTER} \leftarrow ø$;
    $I_{IMG} \leftarrow I_U^{(i)} - I_{CENTER}$;
    SeedList ← IniSeedList ($I_{IMG}$);
    while (SeedList ≠ ø) do
        $seed_{RGB} \leftarrow$ The first element in SeedList;
        $I_{TEMP} \leftarrow$ FuzExtractor ($I_{SRC}$, $seed_{RGB}$);
        if $I_{TEMP}/I_{SRC} > \varepsilon$
            if ($I_{TEMP}/I_{SRC} < \gamma_1$ & var ($I_{TEMP}$) $< \gamma_2$)
                $I_M^{(i+1)} \leftarrow I_{TEMP}$;
        $I_U^{(i+1)} \leftarrow I_{SRC} - (I_M^{(i+1)} \cup I_M^{(i-1)} \cdots I_M^{(1)})$;
        $I_{IMG} \leftarrow I_U^{(i+1)} - I_{CENTER}$;
        SeedList ← IniSeedList ($I_{IMG}$);
        $i \leftarrow i + 1$;
    end
return {$I_M^{(i+1)}$};
```

Pseudocode 3

with a wide range distribution, as shown in Figure 4(b). Such a subimage is also considered as insignificant. If I_{TEMP} is significant, CreateSubImage assigns I_{TEMP} to a *Matched* subimage $I_{\text{M}}^{(i+1)}$ and creates an *Unmatched* $I_{\text{U}}^{(i+1)}$, where $I_{\text{M}}^{(i+1)}$ holds the color components matching the color of seed_{RGB} and $I_{\text{U}}^{(i+1)}$ holds unextracted pixels in the source image. CreateSubImage iteratively processes $I_{\text{U}}^{(i)}$ until $I_{\text{U}}^{(i)} = ø$; that is, each pixel in the image is processed.

There may exist tiny holes in the subimage $I_{\text{M}}^{(i)}$. Because these tiny fragments may be insignificant for object pattern

```
FuzExtractor ($I_{\mathrm{SRC}}$, seed$_{\mathrm{RGB}}$)
    Move seed$_{\mathrm{RGB}}$ into $I_{\mathrm{MATCH}}$;
    Push four neighbor pixels of seed$_{\mathrm{RGB}}$ into queue;
    while (queue $\neq \emptyset$) do
            pixel$_{\mathrm{FIRST}}$ $\leftarrow$ the top element in queue;
            Calculate $\Delta\rho_{\mathrm{F}}$ using (3);
            if ($\Delta\rho_{\mathrm{F}} < \sigma$)
                Move pixel$_{\mathrm{FIRST}}$ into $I_{\mathrm{MATCH}}$;
                Push unduplicated neighbors of pixel$_{\mathrm{FIRST}}$ into queue;
        end
return $I_{\mathrm{MATCH}}$;
```

Pseudocode 4

```
MergeSubImage ($\{I_{\mathrm{M}}{}^{(i)}\}$)
    $K \leftarrow I$;
    for $i \leftarrow 1$ to $I$ do
        for $l \leftarrow i - 1$ to $I$ do
            if $(I_{\mathrm{M}}{}^{(i)} \cap I_{\mathrm{M}}{}^{(l)})/I_{\mathrm{M}}{}^{(i)} > \tau$;
                $I_{\mathrm{M}}{}^{(l)} \leftarrow I_{\mathrm{M}}{}^{(i)} \cup I_{\mathrm{M}}{}^{(l)}$;
                Remove $I_{\mathrm{M}}{}^{(i)}$ from the list;
                $K \leftarrow K - 1$;
                break;
    for $k \leftarrow 1$ to $K$ do
        $I_{\mathrm{OBJ}}{}^{(k)} \leftarrow I_{\mathrm{M}}{}^{(k)}$;
        $I_{\mathrm{SEG}} \leftarrow I_{\mathrm{OBJ}}{}^{(k)} \cup I_{\mathrm{OBJ}}{}^{(k-1)} \cdots I_{\mathrm{OBJ}}{}^{(1)}$;
return $I_{\mathrm{SEG}}$;
```

Pseudocode 5

features, the algorithm removes them by filling up such tiny holes with their original pixels. Usually, these tiny fragments are caused by a small RGB distance of two pixels that have the same RGB values and are located in the same row or in the same column, so the algorithm checks such a distance if it is less than a threshold, $\delta = 7$. If so, the algorithm fills the tiny holes when creating the subimage, $I_{\mathrm{M}}{}^{(i)}$. The algorithm generates a seed from the *Unmatched* subimage $I_{\mathrm{U}}{}^{(i)}$ but always creates a subimage $I_{\mathrm{M}}{}^{(i)}$ from the source image I_{SRC} so it is very probable that the algorithm extracts an individual pixel into multiple subimages. We use a list to maintain $\{I_{\mathrm{M}}{}^{(i)}\}$ according to a sending order of their sizes. Table 3 shows an example of the proposed algorithm that creates 20 subimages from the source image I_{BIRD} in Figure 5(a). Column 1 in Table 3 contains $I_{\mathrm{M}}{}^{(i)}$ ID, columns 2 and 3 list the locations and the color components of the seeds for extracting the subimages $I_{\mathrm{M}}{}^{(i)}$ using the FCE, column 4 shows the number of pixels extracted into the subimages $I_{\mathrm{M}}{}^{(i)}$, and column 5 records the variance of $I_{\mathrm{M}}{}^{(i)}$.

Table 3: Subimages created from I_{BIRD}.

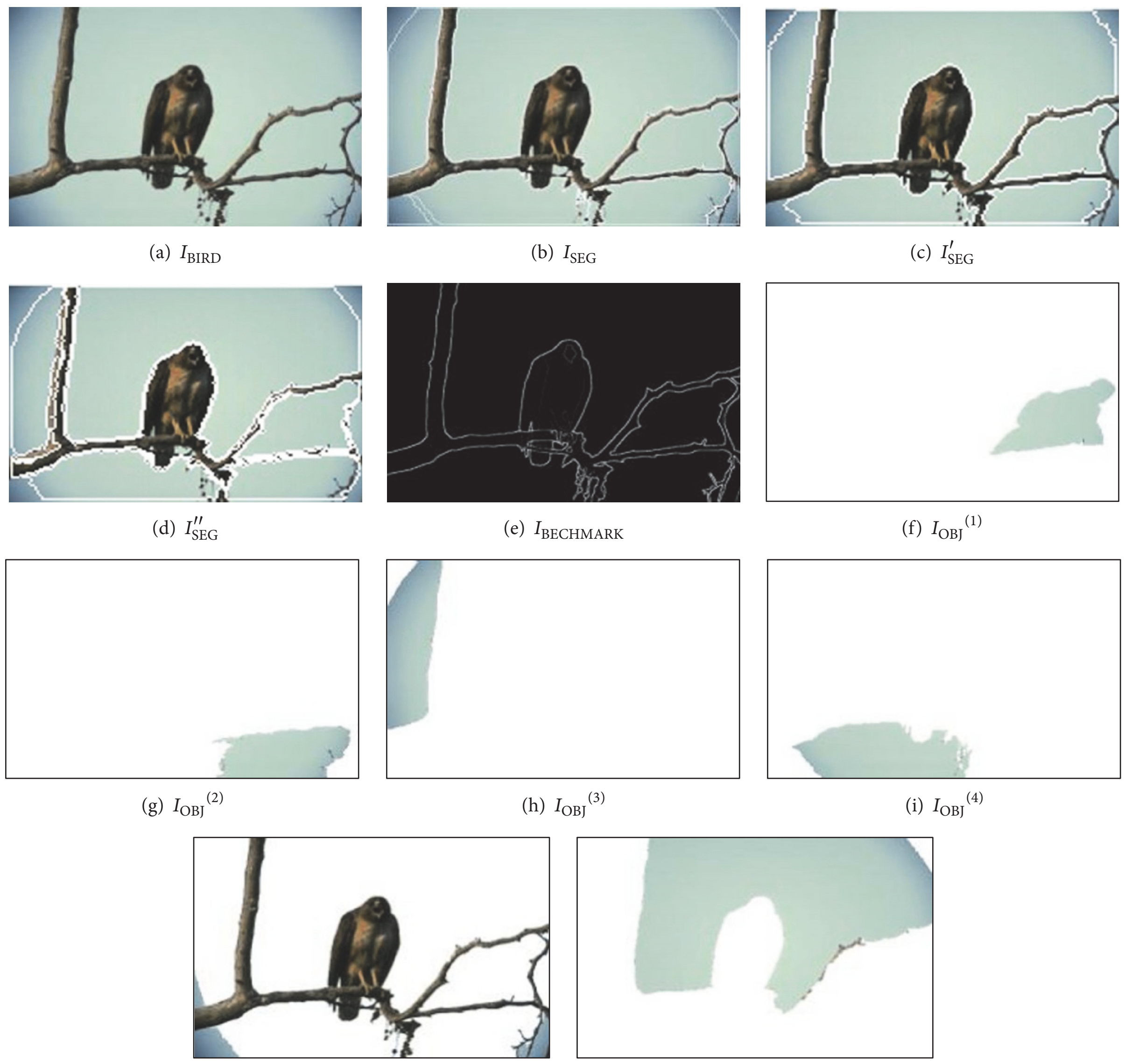
ID	Locations	RGB colors	Pixels	Variance
(1)	(269, 271)	(244, 251, 243)	14308	148
(2)	(283, 267)	(240, 251, 247)	9674	114
(3)	(267, 243)	(230, 252, 239)	64924	150
(4)	(193, 233)	(0, 6, 0)	15643	917
(5)	(358, 174)	(228, 246, 232)	10492	157
(6)	(276, 280)	(229, 232, 221)	15307	240
(7)	(292, 302)	(209, 222, 213)	10894	372
(8)	(58, 141)	(197, 216, 212)	13815	590
(9)	(292, 296)	(1, 12, 4)	3003	1419
(10)	(290, 302)	(11, 14, 5)	3529	1602
(11)	(464, 285)	(7, 22, 25)	3941	1591
(12)	(275, 252)	(17, 30, 13)	17645	1311
(13)	(320, 261)	(20, 38, 26)	4831	1961
(14)	(68, 138)	(54, 39, 16)	26009	2416
(15)	(216, 190)	(151, 100, 57)	2916	2394
(16)	(357, 158)	(50, 53, 46)	26769	2703
(17)	(293, 294)	(185, 198, 189)	11018	534
(18)	(352, 171)	(206, 191, 162)	67427	375
(19)	(49, 238)	(89, 72, 42)	28510	3311
(20)	(1, 320)	(58, 86, 108)	26845	4777

5. Merging Subimages

The proposed algorithm merges significant subimages into a set of meaningful segmented objects by the procedure MergeSubImage. The pseudocode of the MergeSubImage procedure is shown in Pseudocode 5.

The inputs to the procedure MergeSubImage are the subimages $\{I_{\mathrm{M}}{}^{(i)}\}$, and the output from MergeSubImage is the segmented image I_{SEG}. Figures 5(a)-5(b) show a color image, I_{BIRD}, and the segmented image in the RGB space, I_{SEG}, which is a union of 6 objects, $I_{\mathrm{OBJ}}{}^{(i)}$, $i = 1, \ldots, 6$, shown in Figures 5(f)–5(k). Figure 5(e) shows the manually made benchmark, I_{BECHMARK}.

MergeSubImage merges the subimages according to the overlap sizes of their color components to construct some objects in the source image. In the process of the Merge-SubImage, the FCE usually classifies an individual pixel into multiple subimages. It causes these subimages to share a common color area in the RGB space, and even probably a



(a) I_{BIRD} (b) I_{SEG} (c) I'_{SEG} (d) I''_{SEG} (e) I_{BECHMARK} (f) $I_{\mathrm{OBJ}}{}^{(1)}$ (g) $I_{\mathrm{OBJ}}{}^{(2)}$ (h) $I_{\mathrm{OBJ}}{}^{(3)}$ (i) $I_{\mathrm{OBJ}}{}^{(4)}$ (j) $I_{\mathrm{OBJ}}{}^{(5)}$ (k) $I_{\mathrm{OBJ}}{}^{(6)}$

FIGURE 5: Source image, I_{BIRD}; segmented images in the RGB, HSV, and YUV space, I_{SEG}, I'_{SEG}, and I''_{SEG}; manually made benchmark, I_{BECHMARK}; and partitioned objects in the RGB space, $I_{\mathrm{OBJ}}{}^{(i)}$.

subimage may fully cover another subimage. The common area of two subimages implies two features: spatial proximity and color similarity, because the two subimages connect together and have a partition with the same color if they share the common area. Therefore, our algorithm uses this unique characteristic to merge subimages. MergeSubImage uses the union $(I_{\mathrm{M}}{}^{(i)} \cap I_{\mathrm{M}}{}^{(l)})/I_{\mathrm{M}}{}^{(i)}$ to check the common area size of two subimages $I_{\mathrm{M}}{}^{(i)}$ and $I_{\mathrm{M}}{}^{(l)}$ in the RGB space. A bigger value of $(I_{\mathrm{M}}{}^{(i)} \cap I_{\mathrm{M}}{}^{(l)})/I_{\mathrm{M}}{}^{(i)}$ means that $I_{\mathrm{M}}{}^{(i)}$ and $I_{\mathrm{M}}{}^{(l)}$ share a larger common area; that is, their color components are more similar. A data structure for maintaining the subimages $\{I_{\mathrm{M}}{}^{(i)}\}$ is a list that sorts $\{I_{\mathrm{M}}{}^{(i)}\}$ in a sending order of their sizes. Technical implementation of merging $\{I_{\mathrm{M}}{}^{(i)}\}$ is shown in Figure 6. The merging process starts with the last element $I_{\mathrm{M}}{}^{(i)}$ with $i = I$, located at the list tail, which is the subimage with the smallest area size, and checks $I_{\mathrm{M}}{}^{(l)}$ with each subimage $\{I_{\mathrm{M}}{}^{(l)},\ l = i-1, \ldots, 1\}$ via $(I_{\mathrm{M}}{}^{(i)} \cap I_{\mathrm{M}}{}^{(l)})/I_{\mathrm{M}}{}^{(i)}$. If $(I_{\mathrm{M}}{}^{(i)} \cap I_{\mathrm{M}}{}^{(l)})/I_{\mathrm{M}}{}^{(i)} > \tau$, the algorithm merges $I_{\mathrm{M}}{}^{(i)}$ and $I_{\mathrm{M}}{}^{(l)}$ together, removes $I_{\mathrm{M}}{}^{(i)}$ from the list, and replaces $I_{\mathrm{M}}{}^{(i)}$ with $I_{\mathrm{M}}{}^{(i)} \cup I_{\mathrm{M}}{}^{(l)}$, where $I_{\mathrm{M}}{}^{(l)}$ size is bigger than $I_{\mathrm{M}}{}^{(i)}$ one. After processing all the subimages in the list, the list holds the partitioned objects $\{I_{\mathrm{OBJ}}{}^{(i)}\}$ from the input source image I_{SRC}. MergeSubImage returns the partitioned objects $\{I_{\mathrm{OBJ}}{}^{(k)}\}$ as well as its union, $I_{\mathrm{OBJ}}{}^{(k)} \cup I_{\mathrm{OBJ}}{}^{(k-1)} \cdots I_{\mathrm{OBJ}}{}^{(1)}$, as the segmented image I_{SEG}.

Table 4 lists the extracted subimages from the source image I_{BIRD} in Figure 5(a). MergeSub- Image merges these

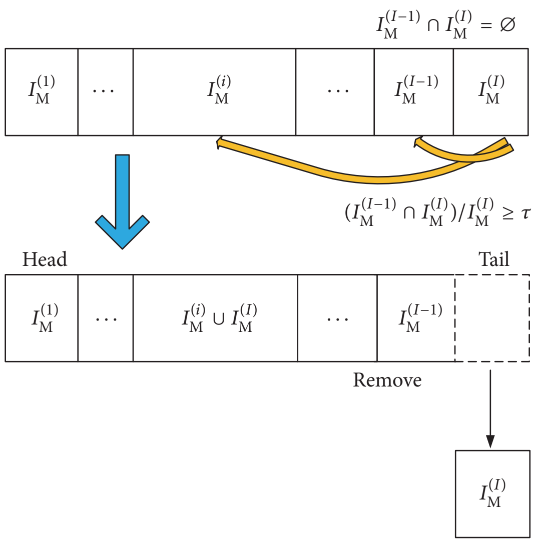


Figure 6: Process for merging subimages ${I_M}^{(i)}$ and ${I_M}^{(l)}$.

Table 4: ${I_{OBJ}}^{(i)}$ merged by subimages created from I_{BIRD}.

Segmented objects	Sub-images
${I_{OBJ}}^{(1)}$	${I_M}^{(11)}$
${I_{OBJ}}^{(2)}$	${I_M}^{(2)}, {I_M}^{(19)}, {I_M}^{(22)}$
${I_{OBJ}}^{(3)}$	${I_M}^{(22)}$
${I_{OBJ}}^{(4)}$	${I_M}^{(22)}, {I_M}^{(22)}$
${I_{OBJ}}^{(5)}$	${I_M}^{(22)}, {I_M}^{(22)}, {I_M}^{(22)}, {I_M}^{(22)}, {I_M}^{(22)}, {I_M}^{(22)}$, ${I_M}^{(22)}, {I_M}^{(22)}, {I_M}^{(22)}, {I_M}^{(22)}, {I_M}^{(22)}$
${I_{OBJ}}^{(6)}$	${I_M}^{(22)}, {I_M}^{(22)}$

20 subimages into 6 partitioned objects $\{{I_{OBJ}}^{(i)}\}$ in Figures 5(f)–5(k). Table 4 also shows information on which and how many subimages are merged into which object. For example, the partitioned object ${I_{OBJ}}^{(1)}$ shown in Figure 5(f) comes from a single subimage ${I_M}^{(11)}$, while the partitioned object ${I_{OBJ}}^{(5)}$ in Figure 5(j) is produced by merging 11 subimages $\{{I_M}^{(9)}$, ${I_M}^{(22)}, {I_M}^{(23)}, {I_M}^{(27)}, {I_M}^{(29)}, {I_M}^{(32)}, {I_M}^{(34)}, {I_M}^{(36)}, {I_M}^{(38)}, {I_M}^{(41)}$, ${I_M}^{(43)}\}$. Figure 5(b) shows the segmented image I_{SEG} by a union operation of ${I_{OBJ}}^{(6)} \cup {I_{OBJ}}^{(5)} \cdots {I_{OBJ}}^{(1)}$.

6. Discussions and Conclusions

In this section, we present our selection of the membership functions' parameters of the FCE defined in (4)–(7) and settings of the CreateSubImage and MergeSubImage procedures used for our tests. We empirically selected the parameters of the membership functions as $\alpha_1 = 60$, $\alpha_2 = 200$, $\rho_M = 20$, $\rho_M + \rho_U = 255$, and $\sigma = 90$. A smaller α_1 may more precisely segment color components in an input image but produces more extracted subimages. A slight change of α_2 does not affect the segmentation result, so it is not sensitive. ρ_M should be at least less than 127.5. The threshold, σ, is important for classifying pixels into a *Matched* subimage ${I_M}^{(i)}$ or an *Unmatched* subimage ${I_U}^{(i)}$. We select the parameters in CreateSubImage and MergeSubImage as $\varepsilon = 0.01$, $\gamma_1 = 0.25$, $\gamma_2 = 500$, and $\tau = 0.6$. The function of ε is to remove insignificant subimages with a "too small" area that may be caused by noise. Both parameters γ_1 and γ_2 are defined to remove another type of insignificant subimages, for example, I_{TEMP} shown in Figure 4(b). This subimage is not extracted very well due to a "bad seed," $\text{seed}_{RGB} = (64, 28, 28)$. Such a subimage is featured by its large area size and a large variance of its color components. The large variance indicates that the subimage holds a wide range of color distribution in the RGB space, so this subimage should be removed. τ is the threshold to check the overlap area over the subimage with a smaller area. In this paper, $\tau = 0.6$ herein means that the overlap area is 60% of the subimage with the smaller area. The bigger τ is, the more similar the color components of the two subimages are. Changing τ affects the number of partitioned objects generated by MergeSubImage. As an extreme example, $\tau = 1.0$ means that only if the larger subimage fully overlaps the smaller subimage, the two subimages will be merged together.

In our implementation, the algorithm invokes the CreateSubImage procedure twice. For the first invocation, an input image, I_{IMG}, to IniSeedList and UpdateSeedList, is $I_{SRC} - I_{CENTER}$, where I_{CENTER} is extracted by $\text{seed}_{CENTER} = (127, 127, 127)$. Because the first call is to process a source image I_{SRC}, the algorithm does not select a seed close to the RGB space center. As discussed in Section 4, the seed seed_{CENTER} cannot differentiate color components which are far from the well-defined eight colors. For example, the extracted image I_{CENTER} shown in Figure 4(a) holds many color components of six corn cobs. Because the algorithm does not select any seed from I_{CENTER}, it is very possible that some of the pixels in the source image remain unclassified. In

Table 5: Statistics of subimages and partitioned objects of five color images.

ID	I_{BIRD}	I_{CORN}	I_{LAKE}	I_{MAN}	I_{BABOON}
${I_{\mathrm{M}}}^{(i)}$	20	41	16	22	85
${I_{\mathrm{OBJ}}}^{(i)}$	6	13	6	54	6

order to process the unclassified pixels, we create an image $I_{\mathrm{UNCLASSIED}} = I_{\mathrm{SRC}} - \cup {I_{\mathrm{OBJ}}}^{(i)}$, as shown in Figure 4(c). The algorithm calls the CreateSubImage a second time to process $I_{\mathrm{UNCLASSIED}}$, but the input image to IniSeedList and UpdateSeedList is $I_{\mathrm{UNCLASSIED}}$ itself. The variance of $I_{\mathrm{UNCLASSIED}}$ is smaller than that of its original source image I_{SRC} as most pixels with a large range of color distribution are classified after first calling CreateSubImage. For this reason, we halve the parameter $\alpha_1 = 30$ of the membership functions to process $I_{\mathrm{UNCLASSIED}}$. The proposed approach segments the color image I_{CORN} in Figure 7(a) into 12 objects. Figure 7(b) shows the segmented image. Figures 7(f)–7(q) display the first 12 partitioned objects. ${I_{\mathrm{OBJ}}}^{(3)}$ and ${I_{\mathrm{OBJ}}}^{(6)}$ in Figures 7(h) and 7(k) are extracted from $I_{\mathrm{UNCLASSIED}}$ in Figure 4(c).

In order to test the robustness of the proposed algorithm, we applied the algorithm with the same settings defined in the first paragraph of this section to process the five color images $\{I_{\mathrm{BIRD}}, I_{\mathrm{CORN}}, I_{\mathrm{LAKE}}, I_{\mathrm{MAN}}, I_{\mathrm{BABOON}}\}$, in the RGB color space as shown in Figures 5(a), 7(a), 8(a), 9(a), and 10(a), respectively. Their corresponding segmented images are displayed in Figures 5(b), 7(b), 8(b), 9(b), and 10(b), respectively. The first four color images are from the database at UC Berkeley [23]. These selected color images represent different features and diverse scenes that might be taken under different light conditions. The source image I_{BIRD} in Figure 5(a) holds a bird on a tree trunk under the sky. The source image I_{CORN} in Figure 7(a) holds six corns with many small multicolor kernels and its segmented image holding the six colorful corns. The source image I_{LAKE} in Figure 8(a) shows a lake and a mountain under the sky at evening. Figures 9(a) and 10(a) show the color images I_{MAN} and I_{BABOON}. Table 5 lists the subimages, and the partitioned objects from the 5 color images. 85 subimages are extracted from I_{BABOON}, but the partitioned objects are only 6.

There is a lack of a strong mathematical model for precisely expressing the human's understanding of colors due to psychophysical perception. For a given color, different observers may perceive its color definition differently. That is why the FCE-based approach classifying an individual pixel into several subimages is reasonable. The proposed approach to color image segmentation is easy to use, and it does not apply any procedure where one needs to preprocess source images. Removing subimages with "too small" areas is filtering short edges and noise in a color image. The approach automatically partitions a color image into an appropriate number of meaningful objects or regions. The manually made benchmarks are downloaded from the database at UC Berkeley [23], as shown in Figures 5(e), 7(e), 8(e), 9(e), 11(e), and 12(e), respectively. These benchmarks display desired segmentation results represented by the images' boundaries. Although the proposed approach does not reach the benchmarks' results, it is able to partition the color images into meaningful objects, for example, the I_{CORN} segmentation results as shown in Figure 7.

In this study, we also compared the performance of the FCE-based algorithm by segmenting the same test images $\{I_{\mathrm{BIRD}}, I_{\mathrm{CORN}}, I_{\mathrm{LAKE}}, I_{\mathrm{MAN}}, I_{\mathrm{HORSE}}, I_{\mathrm{VEGGI}}\}$ in the RGB, HSV, and YUV spaces, respectively. In this case, we only transformed the color image representation from the RGB space into the HSV and YUV spaces according to (1) and (2). The test images are selected randomly from the database at UC Berkeley [23], as shown in Figures 5(a), 7(a), 8(a), 9(a), 11(a), and 12(a). Their corresponding segmented images in the RGB, HSV, and YUV spaces are displayed in Figures 5(b)–5(d), 7(b)–7(d), 8(b)–8(d), 9(b)–9(d), 11(b)–11(d), and 12(b)–12(d). We find that the segmented images in the HSV space I'_{SEG} are the most different from $I_{\mathrm{BENCHMARK}}$ by comparison. So the FCE-based algorithm defined in the HSV space is not adopted. In some cases, the segmentation results in the YUV space are superior to the RGB spaces, such as the "face" of I_{MAN} in Figure 9 and the "boxes" in I_{VEGGI} in Figure 12. But in most cases, the segmented images in the RGB space I_{SEG} are the most similar to $I_{\mathrm{BENCHMARK}}$. It is proven that the FCE-based algorithm defined in the RGB space is the most robust. The possible reason for the results is the transformation of the color image representation from the RGB space into the HSV and YUV spaces may cause color errors so that the segmentation is influenced.

In order to show the advantages of the new approach to color image segmentation, we applied Canny edge detection and Log edge detection algorithms to segment the test images $\{I_{\mathrm{LAKE}}, I_{\mathrm{MAN}}, I_{\mathrm{HORSE}}, I_{\mathrm{VEGGI}}\}$, respectively, and the segmented images were shown in Figures 8(f)-8(g), 9(f)-9(g), 11(f)-11(g), and 12(f)-12(g). Then we compared I_{Canny} and I_{Log} with the segmented images through the FCE-based approach, namely, I_{SEG}, I'_{SEG}, and I''_{SEG}, and meanwhile I_{BECHMARK} can be used as a metric to evaluate the segmentations. As shown in Figures 8 and 9, the edges in I_{Canny} and I_{Log} are "too many" or "too few" compared with I_{BECHMARK}, which proves that the segmentation effect with Canny edge detection and Log edge detection is inferior to the FCE-based approach. Although the segmentation effect with Log edge detection is superior to the FCE-based approach applied in the HSV and YUV spaces, it performs not well as the proposed approach applied in the RGB color space, as shown in Figures 11 and 12. Overall, compared with Canny edge detection and Log edge detection, the new approach based on FCE defined in the RGB color space presents the best performance in the color image segmentation.

The approach is effective in tracking or following a certain color because the FCE-based approach does not need a precise color pattern to extract a set of colors. We apply this algorithm to extract chemical plumes in images taken undersea at very poor illumination conditions [26]. In this paper, we start selecting seeds near the eight well-defined colors, that is, at the vertices of the RGB space, and move their selection from the vertices to the RGB center. Our further research will investigate more effective methods for selecting seeds because seed selection plays an important

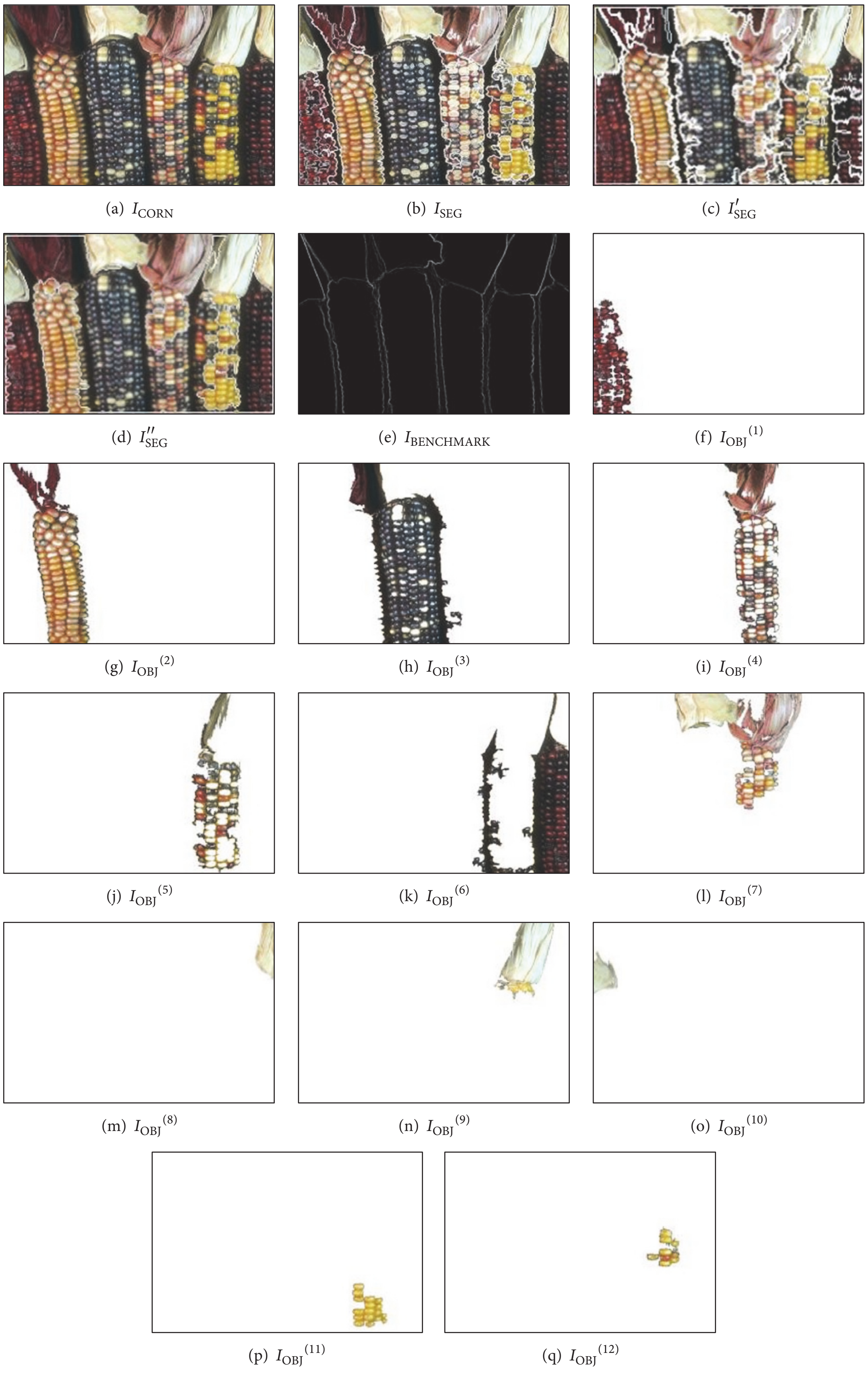
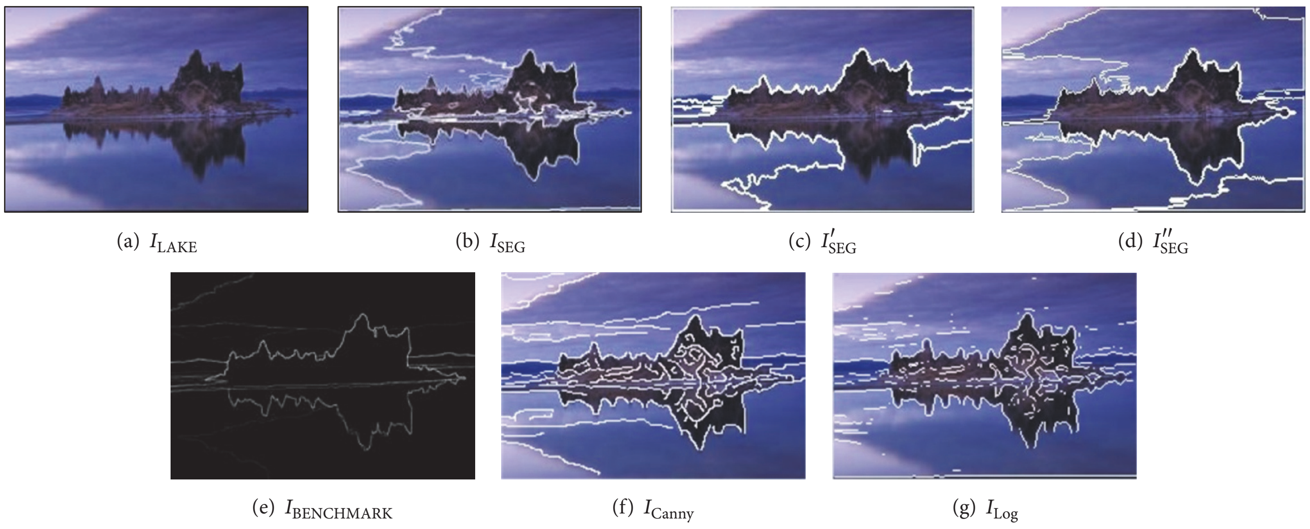
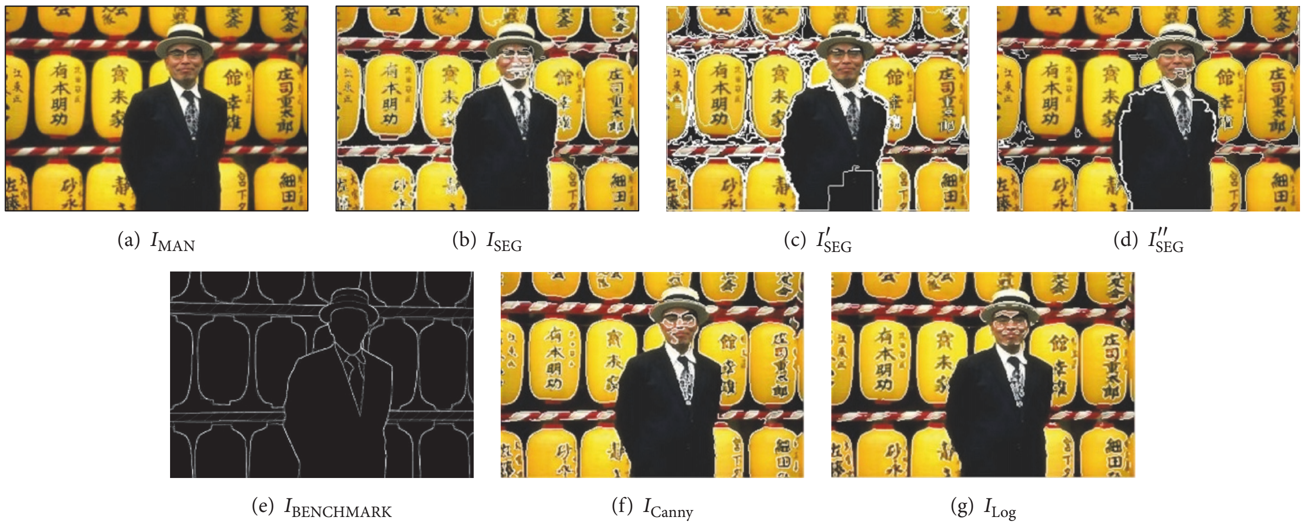

(a) I_{CORN}
(b) I_{SEG}
(c) I'_{SEG}
(d) I''_{SEG}
(e) $I_{\text{BENCHMARK}}$
(f) $I_{\text{OBJ}}{}^{(1)}$
(g) $I_{\text{OBJ}}{}^{(2)}$
(h) $I_{\text{OBJ}}{}^{(3)}$
(i) $I_{\text{OBJ}}{}^{(4)}$
(j) $I_{\text{OBJ}}{}^{(5)}$
(k) $I_{\text{OBJ}}{}^{(6)}$
(l) $I_{\text{OBJ}}{}^{(7)}$
(m) $I_{\text{OBJ}}{}^{(8)}$
(n) $I_{\text{OBJ}}{}^{(9)}$
(o) $I_{\text{OBJ}}{}^{(10)}$
(p) $I_{\text{OBJ}}{}^{(11)}$
(q) $I_{\text{OBJ}}{}^{(12)}$

Figure 7: Source image, I_{CORN}; segmented images in the RGB, HSV, and YUV space, I_{SEG}, I'_{SEG}; and I''_{SEG}; manually made benchmark, I_{BECHMARK}; and partitioned objects in the RGB space, $I_{\text{OBJ}}{}^{(i)}$.



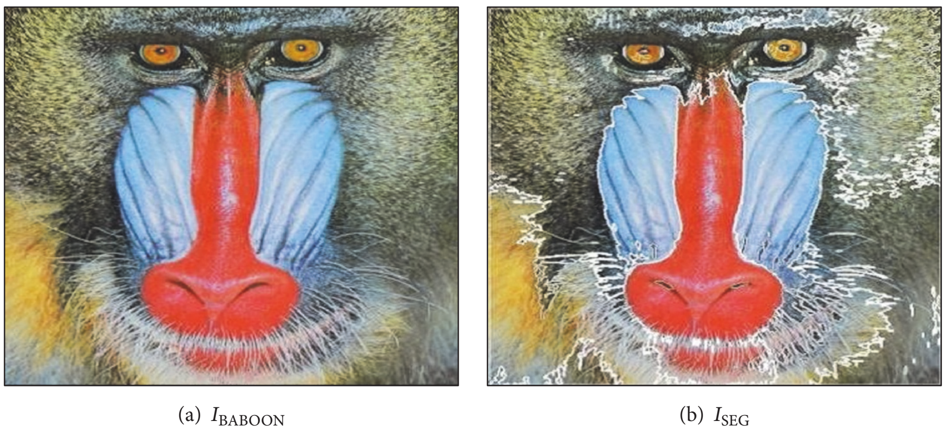
(a) I_{LAKE} (b) I_{SEG} (c) I'_{SEG} (d) I''_{SEG} (e) $I_{\text{BENCHMARK}}$ (f) I_{Canny} (g) I_{Log}

FIGURE 8: Source image, I_{LAKE}; segmented images in the RGB, HSV, and YUV space, I_{SEG}, I'_{SEG}, and I''_{SEG}; manually made benchmark, I_{BECHMARK}; segmented images with Canny edge detection and Log edge detection, I_{Canny}, I_{Log}.



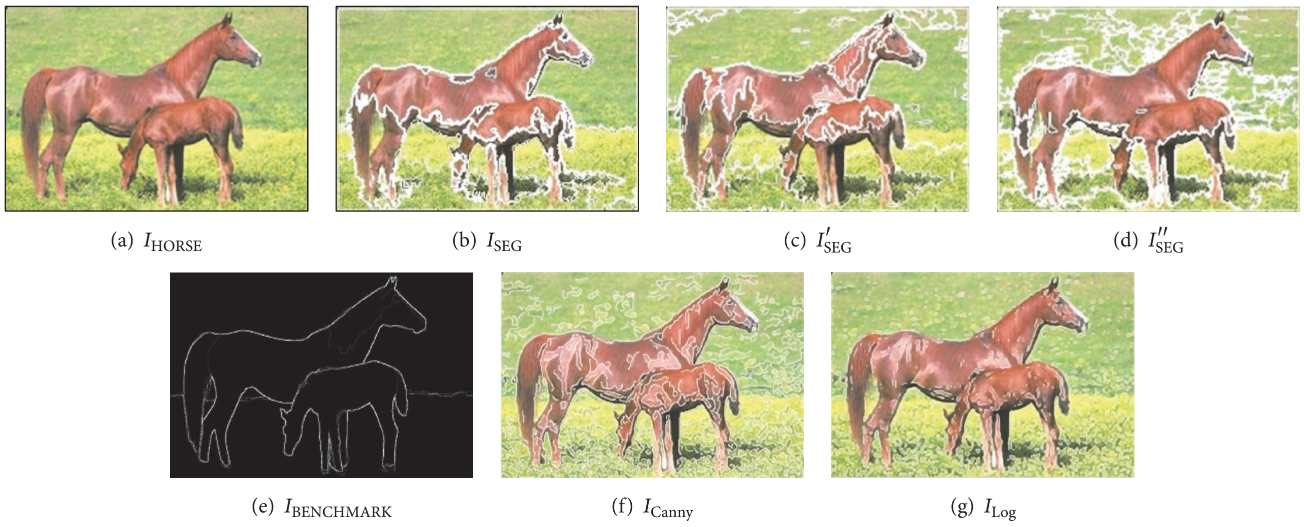
(a) I_{MAN} (b) I_{SEG} (c) I'_{SEG} (d) I''_{SEG} (e) $I_{\text{BENCHMARK}}$ (f) I_{Canny} (g) I_{Log}

FIGURE 9: Source image, I_{MAN}; segmented images in the RGB, HSV, and YUV space, I_{SEG}, I'_{SEG}, and I''_{SEG}; manually made benchmark, I_{BECHMARK}; segmented images with Canny edge detection and Log edge detection, I_{Canny}, I_{Log}.



(a) I_{BABOON} (b) I_{SEG}

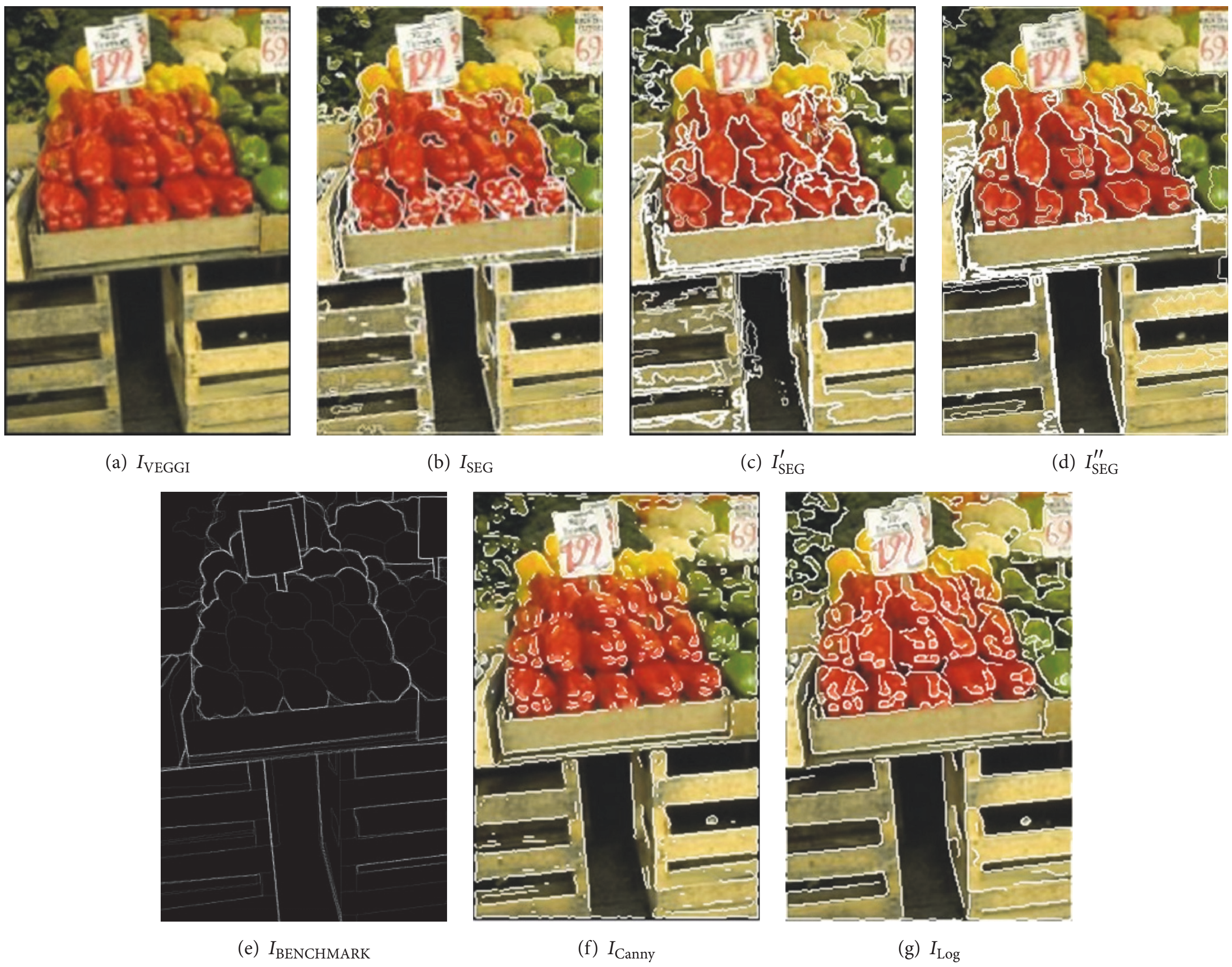
FIGURE 10: Source image I_{BABOON} and its segmented image I_{SEG}.



(a) I_{HORSE} (b) I_{SEG} (c) I'_{SEG} (d) I''_{SEG}

(e) $I_{\text{BENCHMARK}}$ (f) I_{Canny} (g) I_{Log}

FIGURE 11: Source image, I_{HORSE}; segmented images in the RGB, HSV, and YUV space, I_{SEG}, I'_{SEG}, and I''_{SEG}; manually made benchmark, I_{BECHMARK}; segmented images with Canny edge detection and Log edge detection, I_{Canny}, I_{Log}.



(a) I_{VEGGI} (b) I_{SEG} (c) I'_{SEG} (d) I''_{SEG}

(e) $I_{\text{BENCHMARK}}$ (f) I_{Canny} (g) I_{Log}

FIGURE 12: Source image, I_{VEGGI}; segmented images in the RGB, HSV, and YUV space I_{SEG}, I'_{SEG}, and I''_{SEG}; manually made benchmark, I_{BECHMARK}; segmented images with Canny edge detection and Log edge detection, I_{Canny}, I_{Log}.

role in achieving a good segmentation performance. Another issue we will address in our study is how to automatically determine the parameters of fuzzy sets based on an input color image.

Conflicts of Interest

The authors declare no conflicts of interest.

Authors' Contributions

All the authors contributed equally to this work.

Acknowledgments

This research work was funded by the National Natural Science Foundation of China under Grant no. 61473207.

References

[1] K. S. Fu and J. K. Mui, "A survey on image segmentation," *The Journal of the Pattern Recognition Society*, vol. 13, no. 1, pp. 3–16, 1981.

[2] N. R. Pal and S. K. Pal, "A review on image segmentation techniques," *Pattern Recognition*, vol. 26, no. 9, pp. 1277–1294, 1993.

[3] H. D. Cheng et al., "Color image segmentation: Advances & Prospects," *Pattern Recognition*, vol. 34, no. 12, pp. 2257–2281, 2001.

[4] G. J. Klinker, S. A. Shafer, and T. Kanade, "A physical approach to color image understanding," *International Journal of Computer Vision*, vol. 4, no. 1, pp. 7–38, 1990.

[5] A. Tremeau and N. Borel, "A region growing and merging algorithm to color segmentation," *Pattern Recognition*, vol. 30, no. 7, pp. 1191–1203, 1997.

[6] A. Moghaddamzadeh and N. Bourbakis, "A fuzzy region growing approach for segmentation of color images," *Pattern Recognition*, vol. 30, no. 6, pp. 867–881, 1997.

[7] Y. Deng and B. S. Manjunath, "Unsupervised segmentation of color-texture regions in images and video," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 8, pp. 800–810, 2001.

[8] Y. F. Ma and H. J. Zhang, "Contrast-based image attention analysis by using fuzzy growing," in *Proc. Eleventh ACM International Conference on Multimedia.(ACM '03)*, pp. 374–381, New York, NY, USA, 2003.

[9] R. Adams and L. Bischof, "Seeded region growing," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 16, no. 6, pp. 641–647, 1994.

[10] J. Fan, D. K. Y. Yau, A. K. Elmagarmid, and W. G. Aref, "Automatic image segmentation by integrating color-edge extraction and seeded region growing," *IEEE Transactions on Image Processing*, vol. 10, no. 10, pp. 1454–1466, 2001.

[11] F. Y. Shih and S. Cheng, "Automatic seeded region growing for color image segmentation," *Image and Vision Computing*, vol. 23, no. 10, pp. 877–886, 2005.

[12] Y. W. Lim and S. U. Lee, "On the color image segmentation algorithm based on the thresholding and the fuzzy c-means techniques," *Pattern Recognition*, vol. 23, no. 9, pp. 935–952, 1990.

[13] D. Mújica-Vargas, F. J. Gallegos-Funes, and A. J. Rosales-Silva, "A fuzzy clustering algorithm with spatial robust estimation constraint for noisy color image segmentation," *Pattern Recognition Letters*, vol. 34, no. 4, pp. 400–413, 2013.

[14] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, Advanced Applications in Pattern Recognition, New York, NY, USA, 1981.

[15] N. Paragios and R. Deriche, "Geodesic active regions and level set methods for supervised texture segmentation," *International Journal of Computer Vision*, vol. 46, no. 3, pp. 223–247, 2002.

[16] D. K. Panjwani and G. Healey, "Markov Random Field Models for Unsupervised Segmentation of Textured Color Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 10, pp. 939–954, 1995.

[17] L. A. Vese and T. F. Chan, "A multiphase level set framework for image segmentation using the Mumford and Shah model," *International Journal of Computer Vision*, vol. 50, no. 3, pp. 271–293, 2002.

[18] J. Liu and Y. H. Yang, "Multiresolution color image segmentation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 16, no. 7, pp. 689–700, Jul 1994.

[19] G. Zhao, Y. Li, G. Chen et al., "A fuzzy-logic based approach to color segmentation," in *Proceedings of the SPIE 8739, Sensors and Systems for Space Applications VI, 873910*, Baltimore, Md, USA, 2013.

[20] W. Li, G. Lu, and Y. Wang, "Recognizing white line markings for vision-guided vehicle navigation by fuzzy reasoning," *Pattern Recognition Letters*, vol. 18, no. 8, pp. 771–780, 1997.

[21] W. Li, X. Jiang, and Y. Wang, "Road recognition for vision navigation of an autonomous vehicle by fuzzy reasoning," *Fuzzy Sets and Systems*, vol. 93, no. 3, pp. 275–280, 1998.

[22] W. Li, F. M. Wahl, J. Z. Zhou, H. Wang, and K. Z. He, "Vision-Based Behavior Control of Autonomous Systems by Fuzzy Reasoning," *Sensor Based Intelligent Robots*, vol. 1724, pp. 311–325, 1999.

[23] http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/segbench/.

[24] X. C. Li, H. K. Liu, F. Wang, and C. Y. Bai, "The survey of fuzzy clustering method for image segmentation," *Journal of Image and Graphics*, vol. 17, no. 4, pp. 447–458, 2012.

[25] J. Zhang, M. Li, and Z. L. Yang, "Color image segmentation based on fuzzy color extraction," *Journal of Tianjin University of Technology*, no. 6, pp. 33–38, 2014.

[26] Y. Tian, X. Kang, Y. Li et al., "Identifying rhodamine dye plume sources in near-shore oceanic environments by integration of chemical and visual sensors," *Sensors*, vol. 13, no. 3, pp. 3776–3798, 2013.