Image Segmentation Via Color Clustering

Kaveh Heidary
Department of Electrical Engineering and Computer Science
Alabama A&M University
4900 Meridian Street, Huntsville, AL 35810 USA

***Abstract*—This paper develops a computationally efficient process for segmentation of color images. The input image is partitioned into a set of output images in accordance to color characteristics of various image regions. The algorithm is based on random sampling of the input image and fuzzy clustering of the training data followed by crisp classification of the input image. The user prescribes the number of randomly selected pixels comprising the trainer set and the number of color classes characterizing the image compartments. The algorithm developed here constitutes an effective preprocessing technique with various applications in machine vision systems. Spectral segmentation of the sensor image can potentially lead to enhanced performance of the object detection, classification, recognition, authentication and tracking modules of the autonomous vision system.**

Keywords—Clustering; Classification; Image Segmentation; Machine Vision

I. INTRODUCTION

Background removal and image segmentation constitute fundamental components of many autonomous vision systems. Segmentation is utilized in order to separate regions or entities of potential interest from each other and from inconsequential image background for further processing. This paper presents an operationally robust and computationally efficient algorithm for segmentation of the input image based on color. The complex image at the sensor output, which is presented to the autonomous vision system, is partitioned into multiple less complicated images prior to further processing [1-5]. Following the segmentation phase, pertinent members of the resultant image set are processed by the corresponding target classification, recognition, identification, and authentication layers of the machine vision system.

The image segmentation process at lower levels entails ascribing to each pixel the appropriate class label, while at the higher levels segmentation involves utilization of lower level information for associating salient parts of the image with known objects of interest [6-10]. Target detection, classification, recognition, and authentication procedures which are based on two dimensional spatial signatures acquired with various modalities including infrared and electro optical imagery involve utilization of spatial filters [11-15]. The spatial filters may be applied directly to the image at the sensor output or to the resultant images following the segmentation stage. This paper provides the formulation and implementation of an efficient lower level image segmentation algorithm. Image pixels are classified in accordance to their color attributes regardless of spatial relationships. The color classifier is computed using a set of randomly selected pixels, obtained from the input image, which are partitioned using a fuzzy clustering procedure. A set of prototype color vectors are computed from the resultant fuzzy sets and are subsequently utilized to segment the input image.

II. BAKGROUND

Image segmentation is used in order to partition the input image into its salient components for further processing. Segmentation is utilized in various machine vision applications such as object recognition and tracking as well as image compression, editing, and retrieval. Segmentation involves clustering the image feature vectors such as pixel intensity levels and colors [16-18]. In top-down image segmentation the input image is partitioned in accordance to the relationship between the image content and the images of various objects in the database including object shapes, contours, textures, and colors. The bottom-up image segmentation, on the other hand, utilizes the intensity, color, texture, and region boundaries to break up the image into its more basic components. Despite the impressive results of recently reported bottom-up image based segmentation algorithms, they often fail to capture fundamental relationships among image elements. The inherent difficulty encountered by low-level image based segmentation algorithms is due to potentially sharp intensity and color variations within the object boundaries. High-level segmentation algorithms rely on image features such as contours and shapes as segmentation primitives in order to reduce the computational complexity. Detection of edges and contours in the input image is achieved through convolving the grayscale image with local derivative filter operators [19-20]. Different regions that are circumscribed by distinct closed contours are subsequently recognized as the respective image segments. This Paper presents an unsupervised learning algorithm for segmentation of color images.

III. CLUSTERING ALGORITHM

Given a set of N data points in M-dimensional space, and a user specified integer representing the number of clusters (classes) Q, the algorithm described here computes a set of Q prototypes and a $Q \times N$ membership matrix. Each prototype is a vector in M-space and is the optimal representation of the corresponding class. Each element of the membership matrix represents the degree of membership (association) of a data point in the respective cluster.

$$\mathbf{X} = \left\{ \mathbf{X}_n : 1 \le n \le N \right\} \quad (1)$$

$$\mathbf{X}_n = \left[x_{mn} : 1 \le m \le M \right] \quad ; 1 \le n \le N \quad (2)$$

$$\mathbf{Y} = \left\{ \mathbf{Y}_q : 1 \le q \le Q \right\} \quad (3)$$

$$\mathbf{Y}_q = \left[y_{mq} : 1 \le m \le M \right]; \quad 1 \le q \le Q \quad (4)$$

$$x_{mn}, y_{mq} \in \mathbb{R} \quad (5)$$

Where $\mathbf{X}, \mathbf{Y}$ represent, respectively, the set of data points and prototype vectors in M-space, and $\mathbb{R}$ is the set of real numbers. Our objective is to utilize data points in Eq. (1) in order to partition M-space into Q distinct regions with each region represented by a prototype vector $\mathbf{Y}_q$. In the operation phase, an unlabeled vector is classified based on its distance with respect to the prototypes. In crisp classification, for example, the input vector is assigned uniquely to the class with the closest prototype with respect to the input vector. In fuzzy classification, on the other hand, the input vector is assigned to all classes with varying degrees of association.

Each original data point in the trainer set will be linked to all Q regions (classes) with varying degrees of association determined by elements of the membership matrix. The initial membership matrix is generated by assigning random numbers drawn from independent and identically distributed uniform probability functions to each matrix element. We will describe an iterative algorithm for computation of the prototype vectors. The prototype vectors are then used to make hard decisions with regard to new input data points. A new data point is associated with the prototype (class) to which it is closest in accordance to some predefined distance metric.

$$\mathbf{S} = \left[s_{qn} \right]; 1 \le q \le Q, 1 \le n \le N \quad (6)$$

$$s_{qn} = \frac{\alpha_n}{\sum_{p=1}^{Q} \left(d_{qn} / d_{pn} \right)^{\frac{2}{u-1}}} \quad (7)$$

$$\alpha_n = \frac{1}{\sum_{q=1}^{Q} \frac{1}{\sum_{p=1}^{Q} \left(d_{qn} / d_{pn} \right)^{\frac{2}{u-1}}}} \quad (8)$$

$$d_{pn} = \left\| \mathbf{Y}_p \overset{p=1}{-} \mathbf{X}_n \right\| \quad (9)$$

Where, $\mathbf{S}$ is the membership (association) matrix, s_{qn} denotes the degree with which data point-n is associated with (is member of) cluster-q, and d_{qn} is the distance between data point-n and prototype-q. Here, Euclidean distance is used as a measure of distance between vectors in M-space. The exponent parameter $u \in \{[1, \infty]\}$ is user-specified and determines the fuzziness of the clustering process. It is noted from Eq. (8) that the membership matrix is normalized such that sum of each column is equal to one. When $u = \infty$, each data point belongs to all clusters uniformly and $s_{qn} = 1/Q, 1 \le n \le N, 1 \le q \le Q$. when $u = 1$, however, clustering is not fuzzy and each data point is associated with a unique cluster. For crisp (hard) clustering, elements of the membership matrix are given as follows: $s_{qn} = 1$, $d_{qn} < d_{pn}$ $\forall p \ne q$ and $s_{qn} = 0$ otherwise. In hard clustering, $u = 1$, each column of $\mathbf{S}$ contains a single one and the rest of entries for that column are zero. The value of u affects the rate of convergence of the algorithm. In experiments conducted on diverse sets of RGB images, we have found that setting $u = 2.5$, in general, leads to fast convergence and accurate results.

The process starts with generating a random membership matrix, called the zero-order membership matrix $\mathbf{S}^{(0)}$. Matrix elements are chosen from a uniform probability distribution function $s_{qn}^{(0)} \in [0,1]$. The matrix is then normalized by setting the sum of each column to one. The randomly generated membership matrix is then utilized to compute Q zero-order prototype vectors, one for each cluster. A particular prototype vector is computed as the weighted sum of the entire set of data points, where each data point is weighted in accordance to its association to (membership in) the respective cluster.

$$\mathbf{Y}^{(0)} = \left\{ \mathbf{Y}_q^{(0)} : 1 \le \mathrm{q} \le \mathrm{Q} \right\} \quad (10)$$

$$\mathbf{Y}_q^{(0)} = \frac{\sum_{n=1}^{N} \left(s_{qn}^{(0)} \right)^u \mathbf{X}_n}{\sum_{n=1}^{N} \left(s_{qn}^{(0)} \right)^u} \quad ; 1 \le q \le Q \quad (11)$$

Where, $\mathbf{Y}_q^{(0)}$ represents the zero-order prototype vector associated with cluster-q, $\mathbf{X}_n$ is the nth data vector denoting a typical trainer, $s_{qn}^{(0)} (1 \le q \le Q, 1 \le n \le N)$ are elements of the randomly generated zero-order membership matrix, and u is the user-specified exponential parameter. The zero-order prototype vectors are then utilized to compute the first-order membership matrix as shown below.

$$\mathbf{S}^{(1)} = \left[s_{qn}^{(1)} \right]; \quad 1 \le q \le Q, \ 1 \le n \le N \quad (12)$$

$$s_{qn}^{(1)} = \frac{\alpha_n^{(0)}}{\sum_{p=1}^{Q} \left(d_{qn}^{(0)} / d_{pn}^{(0)} \right)^{\frac{2}{u-1}}} \quad (13)$$

$$\alpha_n^{(0)} = \frac{1}{\sum_{q=1}^{Q} \frac{1}{\sum_{p=1}^{Q} \left(d_{qn}^{(0)} / d_{pn}^{(0)} \right)^{\frac{2}{u-1}}}} \quad (14)$$

$$d_{pn}^{(0)} = \left\| \mathbf{Y}_p^{(0)} \overset{p=1}{-} \mathbf{X}_n \right\| \quad (15)$$

$$\mathbf{G}^{(1)} = \left[g_{qn}^{(1)} \right] \quad ; g_{qn}^{(1)} = \left| s_{qn}^{(1)} - s_{qn}^{(0)} \right| \quad (16)$$

$$\delta^{(1)} = \max_{q,n} \left(g_{qn}^{(1)} \right) \quad (17)$$

Where, $\mathbf{S}^{(1)}$ and $\mathbf{G}^{(1)}$ denote, respectively, the first order membership and gradient matrices, and $\delta^{(1)}$ is the first order gradient. Next, the computed first order membership matrix is used in order to compute the first order prototype vectors using Eq. (11), where the superscript 0 is replaced with 1. subsequently, the computed first order prototype vectors are

utilized to compute the second order membership matrix and the gradient as shown in Eq. (13) and Eq. (17), respectively.

The iterative process described above continues until a user prescribed stopping criterion is met. The stopping criterion may be the maximum number of iterations (orders), in which case the process is terminated when the number of iterations is reached. One may also use the gradient value or the relative change of gradient between two consecutive iterations as the stopping criterion. For the experiments in this paper the iteration process terminates when the gradient falls below the user prescribed threshold, i.e. $\delta^{(r)} < T = 0.001$.

IV. Tests with Simulated Data

The clustering algorithm described above was used to partition various synthetically generated data sets into classes, numbers of which were prescribed by the user. The algorithm computes a prototype for each class and the membership matrix for the entire data set. Each trainer may then be assigned to a unique class by binarizing the computed membership matrix. Likewise, new unlabeled input data are classified based on distance between the data point and the computed prototypes.

In the example of Figure 1, the input data set is comprised of points in the xy-plane of the Cartesian coordinate system. the data points were generated by a pair of 2D Gaussian distributions with means at (1,3), (-2,-1) and equal standard deviations set to one. The xy components of each data point were generated by independent distributions. Fifty points were randomly selected from each distribution and were combined to form the unlabeled set of one-hundred input data points. Parameters of the clustering algorithm were set to $Q = 2, u = 2, T = 0.001$, and the iteration process converged after ten rounds. Figure 1 shows the evolution of two prototypes. Both prototype vectors started very close to each other at the proximity of the center of gravity of the entire data set. As the iterations proceed, it is seen that prototypes move toward centers of the respective distributions, where final values of the computed prototypes are shown as triangles.

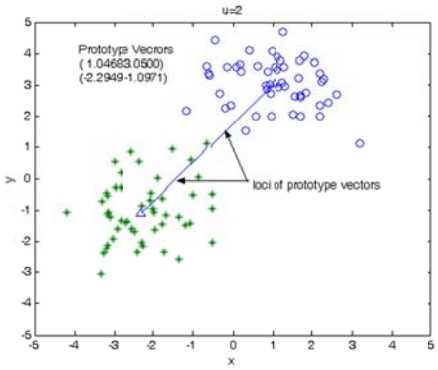

Fig. 1. Evolution of prototype vectors in a two-class problem.

In Fig 1, using circles and stars to denote the data points associated with two classes is for illustration purpose only, and the algorithm is entirely oblivious to the class dispositions of input data points. Despite complete lack of knowledge about origins of the data set members in Figure 1, it is seen that the algorithm finds appropriate prototypes for two classes. Figure 2 shows the degree of association of various data points (1-100) to each prototype (class), and is an illustration of the computed membership matrix for the fuzzy classifier. It is seen that the first fifty data points are more strongly associated with the first prototype (q=1), whereas the last fifty points have higher association to the second prototype (q=2), as expected. Fuzzy classification may be utilized for assignment of classes to new unlabeled input data, where each input vector is assigned probabilities of membership in respective classes. In some applications, crisp classification of the input data may be desired, where a typical input vector is assigned exclusively to the class whose prototype is closest, based on Euclidian distance, to the input vector.

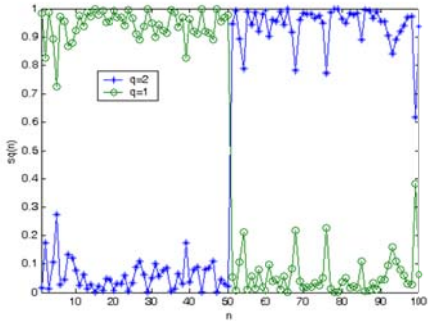

Fig. 2. Trainer data association factor.

In the example of Figure 3, the input data set comprises 125 points in the xy-plane, randomly chosen from three Gaussian distributions with means at (1,3), (-2,-6), (3,-2), and different variances along two axes. The clustering algorithm was tasked to partition the above unlabeled set of data using parameters $Q = 3, u = 2, T = 0.001$. As expected, all three prototype vectors are initially very close to each other and are situated virtually at the center of gravity of the entire input data set. As the iteration process proceeds the prototypes traverse the xy-plane toward their final destinations denoted as triangles. It is noted that the computed prototypes in this example are not equal to mean vectors of the respective classes. This is a byproduct of dataset composition and does not affect ability of the computed prototype vectors to accurately classify new and unlabeled input data.

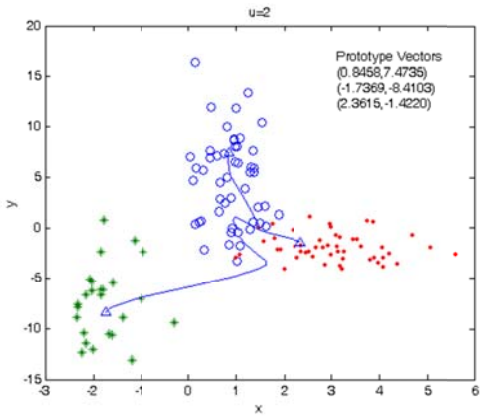

Fig. 3. Prototype evolutions in a three-class problem

In order to obtain quantitative performance results for the fuzzy clustering algorithm the following tests were conducted. We started with two Gaussian distributions in 2D-space, where the x and y components of each sample point were obtained from independent distributions with equal variances.

In the first example, classes A and B were obtained from two 2D distributions with equal variances $\sigma_A=\sigma_B$. Equal numbers of randomly generated trainers from each class were combined and were subsequently utilized as the unlabeled trainer set.

Fuzzy clustering was applied to the above set of unlabeled trainers in order to evolve two prototypes. A large number of test points were then generated from the distribution functions described above. Binary classification was utilized to label all the test vectors in accordance to their Euclidean distances with respect to the above two computed prototypes.

Figure 4 shows the classification error rate as function of the separation factor with the number of unlabeled trainers utilized from each class as parameter.

$$SF = \frac{\left\| \mathbf{m}_A - \mathbf{m}_B \right\|}{\sqrt{\sigma_A^2 + \sigma_B^2}} \qquad (18)$$

Where, SF represents the separation factor between two distributions, and $\mathbf{m}, \sigma$ denote, respectively, the mean-vector and the standard deviation of the particular data set. For each test case, the number of trainers was fixed and the separation factor was varied from 0.25 to 4.

Error rate is the percentage of input test vectors that are misclassified. As expected, the classifier performance improves as the separation factor increases. It is noted that number of trainers has virtually no effect on classifier performance.

In the example of Figure 5, the two Gaussian distributions have unequal standard deviations such that $\sigma_B = 2\sigma_A$ and all other parameters are same as before. It is seen that the number of trainers in this case has an slightly more pronounced effect of the classifier performance.

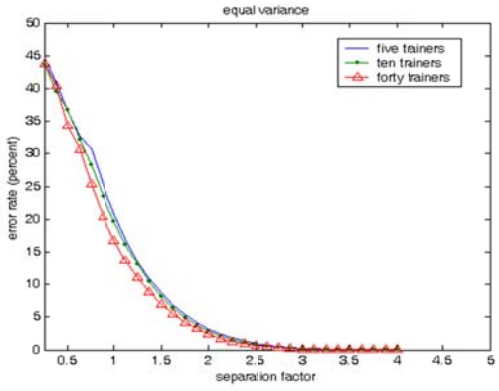

Fig. 4. Effect of SF on classification error rate

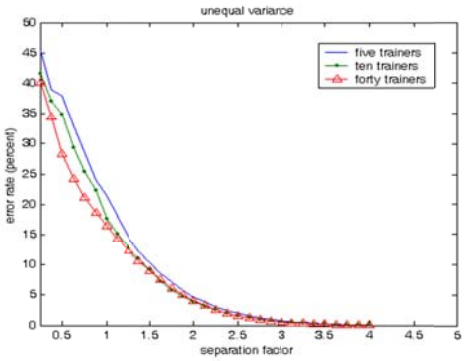

Fig. 5. Effect of SF on classification error rate

V. Experiments with Real Data

In this section the fuzzy clustering algorithm described above is applied to the task of segmentation of color images. The set of RGB vectors associated with a group of randomly selected pixels of the input color image constitute the trainer set. The computed prototypes comprise a set of color vectors which are subsequently utilized to partition the input image.

The example of Figure 6 shows the input image (upper-left), and the result of color segmentation. In this example one-hundred pixels (N=100) were randomly selected from the input image, comprising the unlabeled trainers which were used as the input of the fuzzy clustering process. The algorithm was tasked to partition the training set into three classes (Q=3). Figures 7 and 8 show, respectively, the trainers and the evolution of class prototypes in the RGB-space. It is noted that all three prototypes are initially very close to each other and are proximate to centroid of the training set. The prototypes migrate toward their factual positions and true prototypes are evolved as shown in Figure 8. The initial and final values of the RGB coordinates of the prototype vectors for three classes are listed in Table 1. In this example it took twenty iterations for all three prototypes to reach their final destinations. The computed prototypes were then utilized for crisp classification of all the input image pixels. One of three possible labels were assigned to each pixel of the input image. The images of Figure 6 show results of the filtering process.

Fig. 6. Original input image (upper-left), and color-segmented images. Classes-one (upper right), two (lower-left), and three (lower-right)

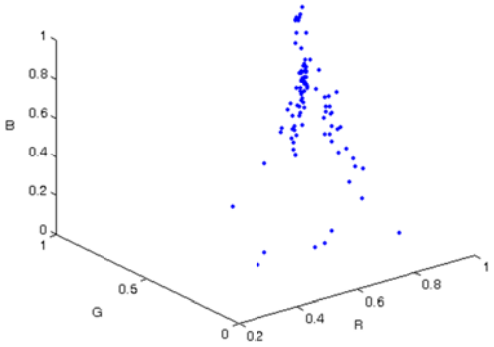

Fig. 7. Training set comprised of one-hundred randomly selected pixels from the original input image

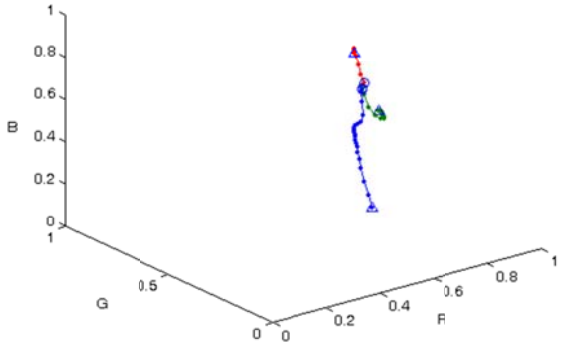

Fig. 8. Evolutions of prototypes for classes-one through three. Initial estimates of prototypes are denoted as circles and final estimates are triangles.

TABLE I. INITIAL AND FINAL PROTOTYPE VECTORS.

	Initial Prototypes			Final Prototypes		
	R	G	B	R	G	B
Class-one	0.853	0.678	0.492	0.566	0.255	0.233
Class-two	0.861	0.687	0.492	0.946	0.716	0.345
Class-three	0.880	0.700	0.507	0.885	0.757	0.623

In the next example the trainer set consisted of one-hundred pixels randomly selected from the Mondrian painting of Figure 9. Fuzzy clustering was used to partition the trainer set into four classes, and the respective RGB prototype vectors were computed.

All pixels of the input image were subsequently classified crisply in accordance to the prototype with the smallest Euclidean distance with respect to the corresponding pixels. The images of Figure 10 show the result of input image segmentation, where pixels of the corresponding class are turned on while pixels of all other classes are set to white.

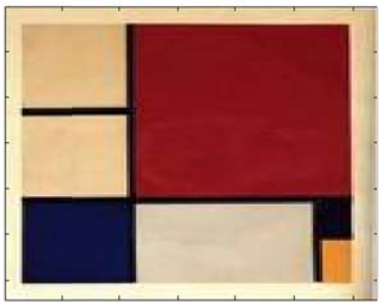

Fig. 9. Input image

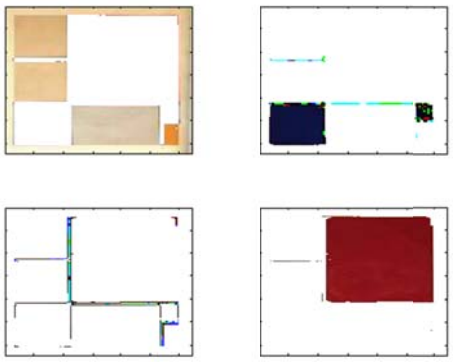

Fig. 10. Output images.

Figures 11 and 12 show the set of one-hundred training vectors and the evolution of four prototype vectors for the Mondrian of Figure 9. As before, all four prototypes are initially close to the center of mass of the training set. In this experiment, The process converged after six iterations. Circles and triangles in Figure 12 denote, respectively, the initial and final values of the prototype vectors. The above four computed prototype RGB vectors were used to assign each pixel of the input Mondrian to one exclusive class, characterized by the prototype with the smallest Euclidean distance with respect to the RGB vector of the pixel. Each one of the filtered images in Figure 10 shows the pixels of the respective class with all other pixels set to white.

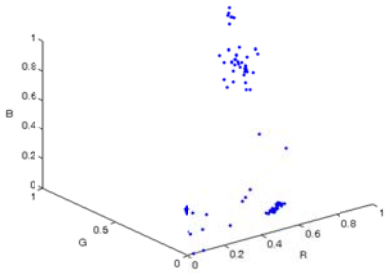

Fig. 11. Training set comprised of one-hundred randomly selected pixels of the Mondrian

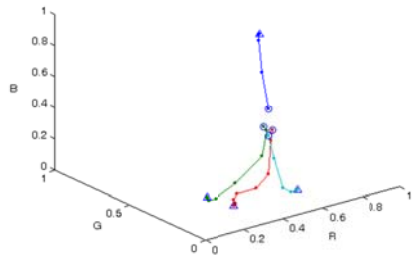

Fig. 12. Evolutions of prototypes for classes-one through four. Initial estimates of prototypes are denoted as circles and final estimates are triangles.

The images of Figure 13 show an input image and the resultant filtered images which are produced by the prototype vectors computed from fuzzy clustering of the trainer set into three groups. The one-hundred element trainer set, shown in Figure 14, was obtained by random sampling of the input image. Figure 15 shows the evolution of the prototype vectors.

Fig. 13. Upper left shows the input image. The input image is filtered using a three-class filter.

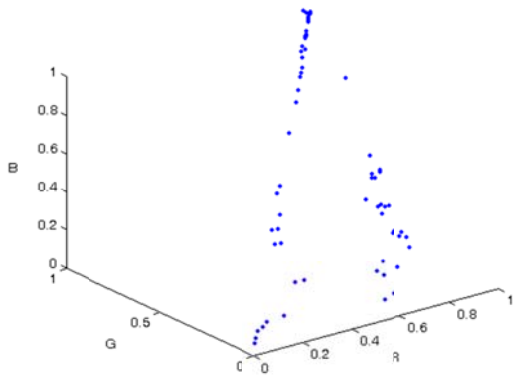

Fig. 14. Training set comprised of one-hundred randomly selected pixels of the input image.

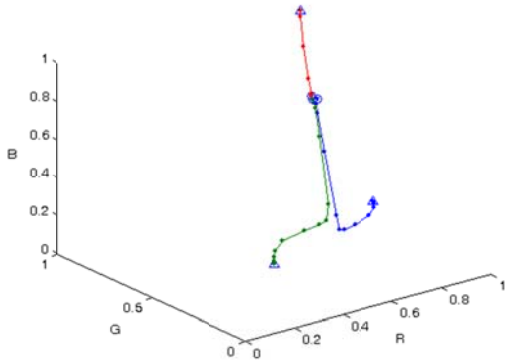

Fig. 15. Evolutions of prototypes for classes-one through three. Triangles denote final prototypes.

In the next example the input image is partitioned in a hierarchical manner. First, the image is sampled randomly and the samples are grouped into two classes using the fuzzy clustering algorithm. This leads to computation of two prototypes, which are used to carry out crisp segmentation of the input image. This process produces two images, each comprised of the input image pixels that belong to the respective class with all other pixels set to white. Each of the two generated images is treated as a new input image and is partitioned into two classes, resulting in four new images. The process continues for a user specified number of partition rounds.

The original set of training pixels were selected randomly from the input image of Figure 16, and were partitioned into two classes using fuzzy clustering. The images of Figure 17 show the result of this two-class segmentation process. The Class-one image, consisting of the leaf and bug only which constitute the foreground in the original image, was then sampled randomly to form a new set of trainers which were partitioned using fuzzy clustering, resulting in computation of two new prototypes. The image (Class-one) was subsequently partitioned using the computed prototypes. The images of Figure 18 show the segmentation results.

VI. CONCLUSIONS

This paper provides a computationally efficient and operationally robust algorithm for segmentation of color images. Tests using synthetically generated data sets as well as real RGB images have demonstrated the efficacy of the image segmentation procedure developed here. The algorithm has practical applications in machine vision systems where partitioning the sensor images in accordance to color characteristics of various image regions can precede higher level processing layers such as recognition and tracking of targets. Future work will include utilization of different distance measures such as Mahalanobis distance and applications to multi-spectral image segmentation.

Fig. 16. Input image.

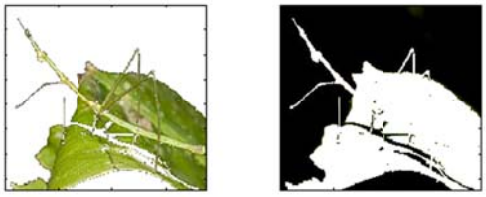

Fig. 17. The result of segmentation of the input image into two classes, foreground and background.

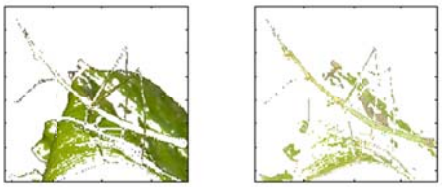

Fig. 18. The result of segmentation of the foreground into two classes

VII. ACKNOWLEDGEMENTS

Partial sponsorship for this work was provided by Department of Defense through US Army RDECOM W911NF-13-1-0136.

REFERENCES

[1] Carson, C., Belongie, S., Greenspan, H., Malik, J.: Blobworld: image segmentation using expectation-maximization and its application to image querying. IEEE Trans. Pattern Anal. Mach. Intell. **24** (8), 2002, pp. 1026–1038.

[2] Cheng, H.D., Jiang, X.H., Sun, Y.,Wang, J.: Color image segmentation: advances and prospects. Pattern Recog. **34** (12), 2001, pp. 2259–2281.

[3] Heidary, K., Caulfield, J. :Color Classification using margin-setting with ellipsoids. Signal Image and Video Processing. 2012, DOI 10.1007/s11760-012-0349-6.

[4] Heidary, K., Caulfield, J. : Presmoothing effects in Artificial Color image segmentation. Computer Vision and Image Understanding **117**, 2013, pp. 195-201.

[5] Heidary, K., Caulfield, J. :Discrimination among similar looking noisy color patches using Margin Setting. Optics Express **15** (1), 2007, pp. 62-75.

[6] Batchelor, B.G. (Editor), Machine Vision Handbook, Springer, 2012.

[7] Steger, C., Ulrich, M., Wiedemann, C.: Machine Vision Algorithms and Applications, Wiley -VCH, 2007.

[8] Davies, E.R.: Computer and Machine Vision: Theory, Algorithms, Practicalities, Academic Press, 2012.

[9] Acharya, T., Ray, A.K.: Image Processing - Principles and Applications, Wiley, 2006.

[10] Zhu, H., Zheng, J., Cai, J., Thalman, N.M.: Object-level image segmentation using low level cues. IEEE Transactions on Image Processing **22** (10) 2013, pp. 4019-4027.

[11] Belongie, S., Malik, J., Puzicha, J.: Shape matching and object recognition using shape contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (24) 2002, pp. 509-522.

[12] Bhanu, B.: Automatic target recognition: State of the art survey. IEEE Transactions on Aerospace and Electronic Systems **22** (4) 1986, pp. 364-379.

[13] Heidary, K.: Distortion tolerant correlation filter design. Applied Optics **52** (12) 2013, pp. 2570-2576.

[14] Heidary, K.: Synthetic template: effective tool for target classification and machine vision. International Journal of Advanced Computer Science and Applications **4** (10) 2013, pp. 22-31.

[15] Heidary, K., Caulfield, H.J.: Needles in a haystack: fast spatial search for targets in similar-looking backgrounds. Journal of the Franklin Institute **349** 2012, pp. 2935-2955.

[16] Martin, D., Fowlkes, C., Malik, J.: Learning to detect natural image boundaries using local brightness, color, and texture cues. IEEE Transactions on Pattern Analysis and Machine Intelligence **26** (5) 2004, pp.530–549.

[17] Yu, S., Shi, J.: Multiclass spectral clustering. Proceedings of International Conference on Computer Vision, 2003.

[18] Lezoray, O., Charrier, C.: Color image segmentation using morphological clustering and fusion with automatic scale selection. Pattern Recognition Letters **30**, pp. 397-406, 2009.

[19] Marr, D.C., Hildreth, E.: Theory of edge detection. Proceedings of Royal Society of London, 1980.

[20] Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and Hierarchical Image Segmentation. IEEE Trans. Pattern Analysis Machine Intelligence **33** (5) 2011 pp. 898-916.