

Competitive Learning Techniques for Color Image Segmentation

Aaron Mavrinas
University of Windsor
mavrin1@uwindsor.ca

Jonathan Wu
University of Windsor
jwu@uwindsor.ca

Xiang Chen
University of Windsor
xchen@uwindsor.ca

Kemal Tepe
University of Windsor
ktepe@uwindsor.ca

Abstract

A method for color image segmentation using a competitive learning clustering scheme is examined, and some basic improvements are made. Two important aspects of the color image segmentation problem, namely color space selection and oversegmentation, are discussed in the context of the algorithm, with comments about suitability and effectiveness of choices for various applications. A variety of settings are tested and compared to highlight performance.

1. Introduction

Color image segmentation is an important part of many image processing and computer vision problems, including recognition, image retrieval, and compression. It has long been recognized that segmenting image pixels in a three-dimensional color space is a much different problem than segmenting grayscale pixels (where one can use relatively simple thresholding techniques). Clustering techniques based on the least-sum-of-squares criterion, such as the K -means algorithm [1], have been shown to perform well in classifying multivariate observations like color image pixels.

Uchimaya and Arbib [6] propose using the competitive learning clustering technique, in which the positions of weight vector units in the feature space are updated when they “win” (e.g., are closest in Euclidean distance) among the other units given a random input vector. To avoid the problem of a few units monopolizing the input space, they modify the algorithm to start with a single unit and divide units based on a win-count threshold until the prescribed number of units has been generated.

Some slight modifications to the original algorithm in [6] yield improvements in the theoretical and actual speed, as discussed in Section 2, and are therefore included in all

subsequent tests and comparisons.

2. Basic Algorithm Improvements

For reference, the modified competitive learning algorithm from [6] is summarized here.

The parameters required are the total number of iterations N_{max} , the weight vector splitting threshold θ_t , and the learning rate α . Experimental results from [6] offer values of N_{max} and θ_t which result in good solutions:

$$\theta_t = 400\sqrt{n} \quad (1)$$

$$N_{max} = (2n - 3)\theta_t(n + 7) \quad (2)$$

Note that the sufficiency condition for n units to be generated is $N_{max} \geq (2n - 3)\theta_t$.

A single weight vector \mathbf{W}_0 is placed at $\bar{\mu}_0$ (the global center of mass of the input set X), and its *wincount* variable and N_R (the current number of iterations) are initialized to zero. The competitive learning algorithm then proceeds as follows:

1. Select a random input vector \mathbf{X} from X .
2. Find weight vector \mathbf{W}_w such that the squared Euclidean distance $\|\mathbf{X} - \mathbf{W}_w\|^2$ is a minimum for all \mathbf{W} (select one randomly in case of a tie).
3. Update \mathbf{W}_w by $\Delta\mathbf{W}_w = \alpha(\mathbf{X} - \mathbf{W}_w)$.
4. Increment the *wincount* of \mathbf{W}_w . If greater than θ_t , reset the *wincount* and generate a new weight vector equal to \mathbf{W}_w .
5. Increment N_R . If $N_R = N_{max}$, stop. Otherwise, repeat from 1.

The rest of this section explains two slight modifications to the original algorithm which yield improvements in speed.

2.1. Starting With Two Units

The algorithm's strength as a clustering method comes from the fact that weight vector units move into high-density regions and subsequently split into multiple units, reducing the number of iterations required as compared to the case where the weight vector units are all initialized and available from the start.

The original algorithm prompts initializing the system with a single weight vector unit. However, since this first unit's initial position is at $\vec{\mu}_0$, the centroid of the entire set of input vectors, and since it is guaranteed to win and be adjusted by random input vectors for the first θ_t iterations, statistically, it will not move significantly from its initial position before it splits into two units. In other words, the statistically probable location of the first split into two weight vector units is $\vec{\mu}_0$.

Therefore, one may initialize the system with two units rather than one, and reduce the number of iterations by θ_t , with no adverse effect. This reduces the sufficiency condition for producing the required n vectors to $N_{max} = (2n - 4)\theta_t$.

2.2. Winner Selection in Case of a Tie

The algorithm also suggests randomly selecting from among winning weight vector units in case of a tie. Using a reasonably precise data format for the storage of vector positions in the color space, for example, IEEE 754 double-precision floating point triplets, it is exceedingly unlikely that weight vectors will tie unless they are experiencing their first win following a split. Since in such a case the vectors would be identical for all practical purposes (same location and same *wincount*), there is no effective difference between choosing randomly from between these units and simply choosing the first one encountered. In implementation, it removes the delay and programming complexity of storing an arbitrary number of ties and then randomly selecting from among them.

3. Color Space Selection

A very important factor in color image segmentation is the selection of color space in which the color of each pixel will be represented and clustered. There are many possible three-dimensional representations of a color which represent useful properties of the color, and only the application can determine which would result in the most accurate segmentation. There is also a trade-off between segmentation accuracy and computational speed, since converting between color spaces may involve complex non-linear transformations.

The equations for converting between these color spaces can be found in [5].

3.1. RGB

The most familiar color space is that used to represent pixels in most image data formats, the RGB space. This format simply combines intensity values of red, green and blue to represent the image. It does not define colors exactly, as it is not an absolute color space; instead, it relies on the exact shades of red, green and blue (primaries) used to define the colors in the image.

Obviously, this is the least computationally intensive representation, since converting the R, G and B levels into real values in $[0, 1]$ requires, at most, a scalar multiplication. However, this color space suffers for two reasons: first, there is no direct representation of luminance, an important factor in the perceptual difference between colors, and second, distances in this color space do not directly correspond to perceptual differences (a property known as *perceptual uniformity*).

The RGB color space is primarily useful for image compression by color quantization, as it provides a reasonable approximation of the original image without requiring conversion to and from another color space.

3.2. HSL and HSV

Two color spaces commonly used in image processing are the HSL (hue-saturation-lightness or hue-saturation-luminance) and HSV (hue-saturation-value) color spaces. Like RGB, HSL and HSV are not absolute color spaces.

Conversion from RGB values involves a non-linear transformation. However, these color spaces offer some components that may be more useful than RGB for some applications, such as the hue and luminance.

The HSL and HSV color spaces are primarily useful in applications where segmentation depends heavily on certain features of the colors rather than overall perceptual difference. Particularly, the individual components can be weighted differently by modifying the range in which they are expressed after conversion from RGB.

3.3. CIE XYZ

CIE 1931 XYZ is an absolute color space based on measurements of human perception, and thus the tristimulus X , Y , and Z values correspond closely to the red, green and blue wavelengths detected by the eye.

Conversion from RGB values involves a linear matrix transformation dependent upon the RGB color space primaries and white point. In this implementation, the sRGB color space and D65 white point are assumed.

While segmentation using the CIE XYZ color space is perfectly valid and may in fact yield better results than using RGB, it serves primarily as a basis for conversion to other, more useful color spaces.

3.4. CIE L*u*v* and L*a*b*

The CIE 1976 L*u*v* and CIE 1976 L*a*b* color spaces are based on the CIE XYZ color space. They are defined to have the properties of *perceptual linearity*, meaning that a change of a certain amount in the space corresponds to a change of the same visual importance, and *perceptual uniformity*, meaning that such changes have the same effect across the entire color space.

Conversion from CIE XYZ tristimulus values involves a fairly intensive non-linear transformation, which causes markedly slower performance. However, the perceptual linearity and uniformity of these color spaces make them the optimal choice for segmenting images based on the perceived differences in color regions.

Generally, the CIE L*u*v* and L*a*b* color spaces are ideal for most applications, unless the trade-off in computation speed is unacceptable or particular properties only offered by other color spaces are desired.

4. The Oversegmentation Problem

The modified competitive learning algorithm, like most clustering methods, requires *a priori* knowledge of the nominal number of clusters in order to find an appropriate solution. In color space segmentation where the number of distinct color regions is not obvious or practical to count from the image, the problems of *undersegmentation* or *oversegmentation* may occur, causing rapid degradation of performance.

If the prescribed number of clusters is too high, the algorithm will be allowed to produce multiple weight vector units in an area containing only one actual distinct cluster. The effect is an oversegmented output image in which pixels that would otherwise be part of a single color cluster are split between two or more very similar clusters.

Two methods for reducing this effect are discussed here. Note that these two methods are inherently incompatible and thus cannot be combined to achieve a greater overall effect.

4.1. Oversegmentation Thresholding

One simple and effective method for correcting oversegmentation once the clustering algorithm is complete is to check for weight vector units relatively close to one another and merge them into a single unit. In this implementation, the following algorithm is used:

1. Find \mathbf{W}_i and \mathbf{W}_j such that the squared Euclidean distance $\|\mathbf{W}_i - \mathbf{W}_j\|^2$ is a minimum for all \mathbf{W} and $i \neq j$.
2. If $\|\mathbf{W}_i - \mathbf{W}_j\|^2 \geq t$, where t is the supplied minimum distance threshold, stop.
3. Move \mathbf{W}_i to the average location $\mathbf{W}_i - \mathbf{W}_j$.
4. Remove \mathbf{W}_j from the set of weight vector units.
5. Repeat.

The primary drawback of this method is that the minimum distance threshold must be specified numerically, and the relationship between a given visual difference in color and this numerical threshold is not obvious. In fact, strictly speaking, this method requires the properties of perceptual linearity and uniformity in the color space being segmented, and of the six color spaces used in this implementation, only the CIE L*u*v* and CIE L*a*b* color spaces have this property. Even given such a color space, the actual threshold must be determined experimentally, unless additional methods are used to estimate it automatically.

4.2. Rival Penalization

A different approach for avoiding oversegmentation, this one occurring during the learning portion of the clustering algorithm, is a technique known as *rival penalization* [4]. This method modifies the competitive learning algorithm to not only move winning weight units closer to the input vector by a factor of the learning rate, but also to push the next-closest weight unit, or *rival*, farther away by a factor of the learning rate and the rival's relative distance from the input vector. In theory, if there are reasonably well-defined clusters in the data, the correct number of weight units should be allowed to settle in their centers, and the remaining weight units should be pushed far away from the clusters.

To implement rival penalization, the following steps are added to the weight update step of the competitive learning algorithm:

1. Find a second weight vector \mathbf{W}_r such that the squared Euclidean distance $\|\mathbf{X} - \mathbf{W}_r\|^2$ is a minimum for all $\mathbf{W} \neq \mathbf{W}_w$.
2. Update \mathbf{W}_r by $\Delta \mathbf{W}_r = -\alpha p_r (\mathbf{X} - \mathbf{W}_r)$.

The rival penalization strength p_r is defined in [2] as:

$$p_r(\mathbf{X}) = \frac{\min(d_{cr}, d_{ci})}{d_{cr}} \quad (3)$$

where d_{cr} and d_{ci} are distance measuring functions, such as the Euclidean distance used in this implementation:

$$d_{cr} = \|\mathbf{W}_r - \mathbf{W}_w\| \quad (4)$$

$$d_{ci} = \|\mathbf{X} - \mathbf{W}_w\| \quad (5)$$

In practice, the effectiveness of rival penalization in color image segmentation varies. If the image has well-defined color regions (clusters of pixels in a given color space), rival penalization works very well. If, however, the data points are distributed relatively smoothly in the color space, which is the case in some images (particularly in the compression application), rival penalization may actually cause the algorithm to take longer to converge.

5. Experimental Results

A variety of images¹ were tested for clustering performance by changing the number of clusters and color space used. The oversegmentation thresholding and rival penalization methods were also tested. All experiments conducted used $\alpha = 0.015$, $\theta_t = 400\sqrt{n}$, and $N_{max} = (2n^2 + 11n - 20)\theta_t$.

5.1. Segmentation Accuracy

A test of the accuracy of the solution found by the segmentation algorithm is an example image undergoing color space quantization for the purpose of image compression. In this case, we are looking for the optimal palette of 16 colors and classification of pixels into this palette, for which the new image most faithfully represents the old in terms of the visual information contained in it.

The test example (fig. 1) is an image of a nebula taken by the Hubble Space Telescope, which contains a variety of colors.

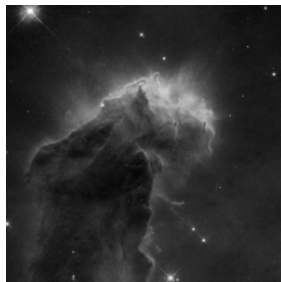


Figure 1. Nebula (Original)

Segmenting this image into 16 clusters with a perceptually linear and uniform color space, in this case the CIE $L^*u^*v^*$ space, yields good detail from the original image using only 16 colors (fig. 2).

¹The original images used, obviously, are color images, and therefore some detail is unavoidably lost in greyscale as they are presented here.

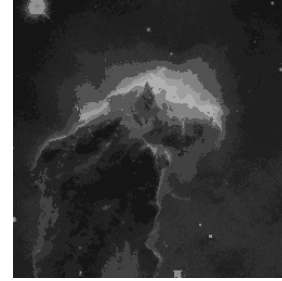


Figure 2. Nebula ($n = 16, L^*u^*v^*$)

5.2. Color Space Selection

Different applications are interested in different regions of images, described in turn by different properties. As discussed in Section 3, selection of the color space is an important consideration to ensure that the appropriate color features are being segmented.

The importance of the properties of perceptual linearity and uniformity are evident in the following example, in which we want to locate the goldfish in the water (fig. 3) by segmenting the image into two clusters.

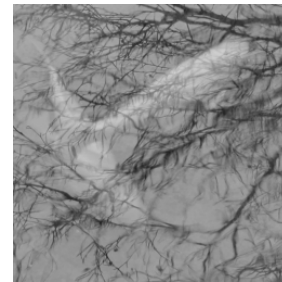


Figure 3. Goldfish (Original)

If the image is segmented using the RGB color space, the results are not what is intuitively expected: rather than segmenting the goldfish from the water, the algorithm segments the branch reflections from the rest of the water and merges the goldfish into the same cluster as the water (fig. 4).

However, if the image is segmented using the perceptually linear and uniform CIE $L^*u^*v^*$ color space, the results conform much closer to human perception and the goldfish are properly segmented from the water (fig. 5).

In another example of the selection of color space having an important impact, in the following image of the sun obscured by cloud (fig. 6), suppose we wish to segment the sun itself from the rest of the image despite its obscure edge.

The image is segmented into 4 clusters in the HSV color

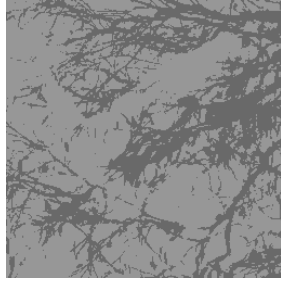


Figure 4. Goldfish ($n = 2$, RGB)



Figure 5. Goldfish ($n = 2$, L*u*v*)



Figure 6. Sun (Original)

space, selected to cluster the sun's pixels by their distinct properties in this space (fig. 7). This particular result cannot be achieved using the RGB/XYZ or L*u*v*/L*a*b* color spaces.

5.3. Oversegmentation

The effectiveness of the oversegmentation thresholding and rival penalization techniques at reducing oversegmentation is best tested by selecting an image with a known finite number of clearly defined color segments. In this case, an image of a sign with three distinct components (the border and symbol, the fill, and the background) is used (fig. 8).

For reference, the image is first segmented using the

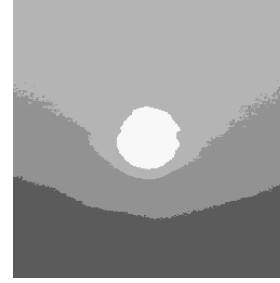


Figure 7. Sun ($n = 4$, HSV)



Figure 8. Sign (Original)

known number of clusters (3). The CIE L*u*v* color space is used for this as well as subsequent segmentations on this image (fig. 9).

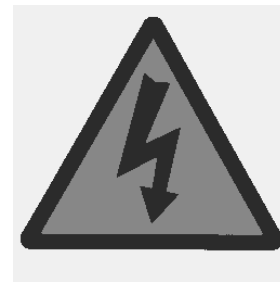


Figure 9. Sign ($n = 3$, L*u*v*)

The image is segmented into 16 clusters with oversegmentation thresholding enabled, and by trial and error the threshold is adjusted until the segmentation merges into 3 clusters. A threshold of 3000 was used to obtain the following segmentation (fig. 10).

The image is again segmented into 16 clusters, this time with rival penalization enabled (fig. 11). In this case, there are still 4 significant clusters, which is not the desired solution, but it is reasonable and did not require manual adjustment of a threshold as in the previous case.

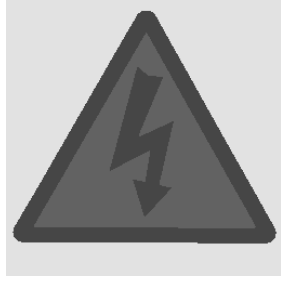


Figure 10. Sign ($n = 16$, $L^*u^*v^*$, OT 3000)

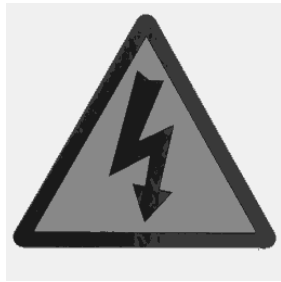


Figure 11. Sign ($n = 16$, $L^*u^*v^*$, RP)

6. Conclusion

The modified competitive learning algorithm proposed by Uchimaya and Arbib [6] is a powerful clustering method which lends itself very well to color image segmentation.

When modified for improved performance as proposed in Section 2, used with a color space appropriate to the application as described in Section 3, and combined with methods for compensating for oversegmentation as proposed in Section 4, it becomes an excellent solution to a wide variety of color image segmentation problems.

Acknowledgement

The authors would like to thank the Natural Sciences and Engineering Research Council of Canada for providing funding for this research.

References

- [1] E. Alpaydin. *Introduction to Machine Learning*. The MIT Press, Cambridge, MA, 2004.
- [2] Y. M. Cheung. Rival penalization controlled competitive learning for data clustering with unknown cluster number. In *Proc. 9th Intl. Conf. on Neural Information Processing*, pages 467–471, November 2002.
- [3] H. P. L. Shafarenko and J. Kittler. Histogram-based segmentation in a perceptually uniform color space. *IEEE Trans. on Image Processing*, 7(9):1354–1358, September 1998.
- [4] L. Law and Y. Cheung. Color image segmentation using rival penalized controlled competitive learning. In *Proc. Intl. Joint Conf. on Neural Networks*, pages 108–112, July 2003.
- [5] B. J. Lindbloom. Useful color equations. <http://brucelindbloom.com/Math.html>, 2007.
- [6] T. Uchimaya and M. A. Arbib. Color image segmentation using competitive learning. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(12):1197–1206, December 1994.