A Study of Variables Affecting K-Means and Fuzzy C-Means Color Image Segmentation Performance by Means of Quantitative and Qualitative Analysis

Connor S. Maynes
Rochester Institute of Technology
Electrical Engineering Department

***Abstract*—The purpose of these experiments was to segment the 300 images in the Berkeley segmentation dataset (BSDS) using the K-Means (KM) and Fuzzy C-Means (FCM) clustering algorithms, while varying several parameters (*color space*, maximum iterations (*max_iter*), number of clusters (*n_clusters*), and fuzziness index (*m*)) and monitoring segmentation accuracy through use of the normalized probabilistic Rand Index. The results suggested a near-optimal number of clusters of 7 for KM and 8 for FCM, a color space of LAB for KM and HSV for FCM, a maximum number of iterations of 30 for KM and 8 for FCM, and a fuzziness factor of 1.01 for FCM. KM provides superior segmentation results over FCM in every test. These experiments demonstrate the value of performance tuning algorithm parameters by showing the significant gains in performance that can be had.**

***Index Terms*—Fuzzy Systems, K-Means, Fuzzy C-Means, Fuzzy Logic, Clustering, Color Image Segmentation**

I. Introduction

IMAGE segmentation is generally defined as the process of subdividing and grouping image pixels in such a way as to simplify the image representation for the purposes of further automated processing and analysis, commonly with the goal of identifying and labeling objects and boundaries [1]. Image segmentation began with gray-scale and black-and-white images, due to a combination of inefficient segmentation algorithms and low computing power in previous decades. More recently, color image segmentation has become commonplace as the efficiency of segmentation algorithms has improved and computing power has increased. The applications of image segmentation are myriad, from medical image processing, to basic object recognition and computer vision tasks, and even to real-time automated motor vehicle navigation systems.

The two algorithms researched for these experiments – K-Means and Fuzzy C-Means – were chosen for their speed, simplicity, and how commonly they have been used. While more accurate and even faster algorithms have more recently been in use in some fields, such as density-based spatial clustering of applications with noise (DBSCAN) for the creation of superpixels, KM and FCM remain important early algorithms which still find use today in one form or another [2], [3], [4], [5].

The purpose of these experiments was to vary some of the available parameters in both KM and FCM clustering algorithms and monitor performance when applied to the problem of color segmentation to determine if any performance improvements could be achieved by using one set of parameter values over another. The Berkeley standard segmentation dataset, consisting of 300 different images, was used for this analysis because it has been cited and used in numerous other works with the Rand Index calculated based on the ground truth segmentations (GTSs) provided in it [6].

II. Related Works

A. Color Space

Color space was chosen as a parameter to vary, because it has previously proven to function as a differentiator of performance for KM. For example, one recent study found that the hue-shade-value (HSV) color space provided superior results in terms of mean-squared-error (MSE) and peak signal-to-noise ratio (PSNR) over the luminance and A and B color channels color space (LAB), when using KM [7]. While the study demonstrated that HSV provided superior segmentation results regarding the metrics, it didn't consider the Rand Index nor other color spaces, such as the common red-green-blue (RGB) or luminance-color value (LUV), both of which were analyzed in these experiments.

Another recent study analyzed a wider range of color spaces (RGB, HSV, LAB, XYZ, and YCbCr), but only regarding how they impacted segmentation of land-sea imagery and only on the metric of segmentation rate [8]. While the study was somewhat limited, they concluded the HSV color space to provide superior results, followed closely by LAB and YCbCr. The segmentation algorithm used was Fusion of Over Segmentation (FOOS), while these experiments focus on the study of KM and FCM.

Another study utilized the YCbCr color space in combination with KM for segmenting images of hand gestures to good effect [9]. The rationale behind the success of this combination of color space and KM for this set of imagery was that by separating chrominance and luminance using the YCbCr color space, the effect of complex scene illumination on segmentation

Manuscript received December 19, 2018.

Connor S. Maynes is a student of the Kate Gleason College of Engineering in the Electrical Engineering department at the Rochester Institute of Technology, Rochester, NY 14623 USA (phone: 802-324-7516; e-mail: csm2737@rit.edu).

results was minimized. The current experiments sought to verify that color spaces which separate luminance from chrominance do in fact achieve better segmentation performance, albeit using the LUV and LAB color spaces instead of YCbCr.

Overall, studies regarding color space have agreed that breaking apart luminance from chrominance, such as in LAB, LUV, and YCbCr, provide superior color segmentation results. The body of work on color space and its relation to color image segmentation performance has been found somewhat fragmented and lacking in a robust study on a wide range of imagery and across multiple types of color spaces for KM and FCM. For these reasons, these experiments were conducted to close the knowledge gap and aide in performance tuning in future research.

B. *Maximum Iterations*

The maximum iterations variable defines the maximum number of times the KM or FCM algorithm can run. With each iteration of either algorithm, the cluster centers and the membership matrix are recalculated. Ideally, each iteration should move the cluster centers towards their ideal values, but this may not be the case [10]. Unfortunately, there seem to be no studies on the impact of the maximum number of iterations of KM or FCM on image segmentation performance. These experiments provided the needed study to aide tuning of both KM and FCM in future applications and future research.

C. *Number of Clusters*

The number clusters selected by the user of either KM or FCM is critical to the success of each clustering algorithm. Each algorithm will seek to find the number of clusters provided, no more and no less. Given the number of clusters selected may be crucial to segmentation performance – as providing too large a number may over-segment an image or providing too few may under-segment an image, a great deal of research has gone into finding methods of automatically selecting the number of clusters on an image-by-image basis. For example, one recent study used a fuzzy silhouette on dynamic data to determine an optimal number of clusters [11]. While this study and others like it using different approaches provide a useful basis for a fully automated version of both KM and FCM, there are always cases where simpler and often faster and more maintainable segmentation algorithms are desired. For this reason, these experiments once again sought to close the knowledge gap on the optimal number of clusters to use, by providing a study using the large and robust BSDS.

D. *Fuzziness Factor*

The fuzziness factor used in FCM controls how fuzzy or varied the final membership matrix is for each data point or pixel in the dataset or image. Obviously, by varying this parameter significant changes in the final membership matrix can be had. To date, there appears to be no study on the effects of using different fuzziness factors in color image segmentation. These experiments sought to close this knowledge gap by testing a wide range of fuzziness factors of varying resolution to gain insight into appropriate values to use.

III. Data

The data used in this study were downloaded from the standard Berkeley color segmentation dataset of 300 unique images. This dataset has been successfully used in a wide number of studies for gauging performance of color segmentation algorithms against GTSs provided by a set of human participants who manually segmented the images [12]. Each image in the dataset has a different number assigned to it, which is used as the *image name*. The GTS files are stored in a format like run-length encoding, where the start and stop row and column indices of where a segmentation label starts and stops on each row are stored instead of the full membership matrix. The codebase used in this study abides this format developed by [12] in its segmentation files, which also have a *.seg* file extension.

IV. Proposed method

A. *Overview*

In these experiments, the proposed method is divided into the following steps: parameter selection and variation, either K-Means (KM) or Fuzzy C-Means (FCM) segmentation, and segmentation evaluation. The flowchart is shown in Fig. 1.

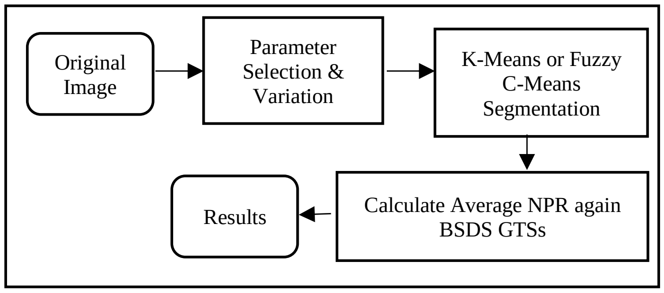
Original Image
Parameter Selection & Variation
K-Means or Fuzzy C-Means Segmentation
Calculate Average NPR again BSDS GTSs
Results

Fig. 1 Methodology Flowchart

B. *Metrics*

The original Rand Index (RI) was first introduced with the goal of evaluating clustering algorithms and was not specifically designed for measuring the performance of image segmentations [13]. The value of RI lies between [0, 1]. RI works in a simple manner, essentially by finding the ratio of the number of pairs of elements in two ordered sets which agree to the total number of pairs of elements. The problem with RI is that it tends to favor segmentations with more clusters, which may or may not be a valid assumption of the nature of the dataset in question [14]. Since not all datasets require many clusters, the Adjusted Rand Index (ARI) – also known as the Normalized Probabilistic Rand Index (NPR) – was proposed to counteract this problem [15]. The value of NPR lies between [0, 1], usually, but may become negative if the two segmentations fail to match up to the degree expected by random chance. A higher NPR indicates better performance. The equation for NPR is described in (1), (2), (3), (4), (5), and (6).

$$a = \frac{\sum_{r=1}^{R}\sum_{c=1}^{C} t_{rc}^2}{2} \quad (1)$$

$$b = a - \frac{N}{2} \quad (2)$$

$$c = \frac{\sum_{r=1}^{R} t_{r+}^2}{2} - a \quad (3)$$

$$d = \frac{\sum_{c=1}^{C} t_{+c}^2}{2} - a \quad (4)$$

$$e = a + \frac{N^2 - \sum_{r=1}^{R} t_{r+}^2 - \sum_{c=1}^{C} t_{+c}^2}{2} \quad (5)$$

$$NPR = \frac{\binom{N}{2}(b+e) - [(b+c)(b+d)+(d+e)(c+e)]}{\binom{N}{2}^2 - [(b+c)(b+d)+(d+e)(c+e)]} \quad (6)$$

The variable t_{rc} represents the number of pixels segmented in the r^th subset of one segmentation of the image, R, and the c^th subset of the other segmentation of the image. b represents the number of pairs of elements which have the same label in R and C; c represents the number of pairs of elements which have the same label in R but a different label in C; d represents the number of pairs of elements with the same label in C and a different label in R; and e represents the number of pairs of elements with different labels in both R and C.

The NPR metric was chosen as the metric for these experiments, because it provides an accurate method of comparing against the ground truth segmentations in the BSDS which are considered the gold standard in segmentation of these images. The BSDS contains multiple segmentations for each image each created by a different human user, because segmentation of any given image can be interpreted in slightly different ways. For the purposes of these experiments, the NPR was calculated between the algorithm's segmentation and each GTS in the BSDS and then averaged, forming the average NPR (ANPR).

C. Color Space

Four different color spaces were used in these experiments, RGB, HSV, LUV, and LAB. The rationale behind using each color space as well as how to convert to each color space from RGB is described.

The RGB color space is the color space most images are stored in and is the color space in which all BSDS images are stored. RGB can be thought of as representing all possible colors which can be created between the three channels of red, green, and blue. RGB combines color tone, saturation, and chromaticity, which can lead to poor segmentation results when the subject of the image is poorly or strangely lit. RGB is commonly used to display and represent images in electronic systems. Because RGB is the default color space in many systems, it was included in these experiments. No process is needed to convert to the RGB color space for these experiments as all BSDS images are already in this color space.

HSV is far different from RGB, because it attempts to separate hue (color tone), saturation (the strength of colors), and value (the amount of light). The value component of HSV can be used to draw a distinction between dark and light colors. The process for converting from RGB to HSV is described in equations (7) through (12).

$$C_{min} = min(r, g, b) \quad (7)$$

$$C_{max} = max(r, g, b) \quad (8)$$

$$delta\ =\ C_{max} - C_{min} \quad (9)$$

$$h = \begin{cases} 60 * ((g-b)/delta) & if\ r\ = C_{max} \\ 60 * (2 + (b-r)/delta) & if\ g\ = C_{max} \\ 60 * (4 + (r-g)/delta) & else \end{cases} \quad (10)$$

$$s = \begin{cases} delta/C_{max} & if\ C_{max}\ != \ 0 \\ 0 & else \end{cases} \quad (11)$$

$$v = C_{max} \quad (12)$$

r, g, and *b* in the equations above represent the three features - red, green, and blue - of the original RGB color space of the image, while *h, s,* and *v* represent the three features of the HSV color space. The formulas above will successfully convert from RGB to HSV, except in the case where the RGB value is 0,0,0, in which case the HSV value would be -1,0,C_{max}.

The LUV color space attempts to break apart a color into three components of luminance and two chromaticity components. The LUV color space is meant to more closely approximate what the human eye perceives as color. In other words, the LUV color space is an attempt to linearize color space differences [16]. By separating luminance from chrominance, there is an opportunity to reduce segmentation distortion caused by uneven lighting in the image. RGB must first be converted to the XYZ color space and then converted to the LUV color space. Simple matrix multiplication can be used to convert from RGB to XYZ, as follows in equation (13).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.412453 & 0.357580 & 0.180423 \\ 0.212671 & 0.715160 & 0.072169 \\ 0.019334 & 0.119193 & 0.950227 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \quad (13)$$

R, G, and *B* in equation (13) again represent the three components of the RGB color space, while X, Y, and Z represent the three components of the XYZ color space. The XYZ color space is meant to be a device-invariant color space, which is to say that a color in this space should look the same regardless of the device it is displayed on. The XYZ color space is also the base from which many other color spaces have been derived, because of its device-invariant display property. The transformation from XYZ to LUV is described by equations (14) through (16).

$$L = \begin{cases} 116 * Y^{1/3} - 16 & if\ Y_r > 0.008856 \\ 903.3 * Y_r & else \end{cases} \tag{14}$$

$$U = 13 * L * \left(\frac{4 * X}{X + 15 * Y + 3 * Z} - \frac{4 * X_r}{X_r + 15 * Y_r + 3 * Z_r}\right) \tag{15}$$

$$V = 13 * L * \left(\frac{9 * Y}{X + 15 * Y + 3 * Z} - \frac{4 * Y_r}{X_r + 15 * Y_r + 3 * Z_r}\right) \tag{16}$$

Once again, *X*, *Y*, and *Z* represent the three components of the XYZ color space, X_r, Y_r, and Z_r represent a reference white value, and *L*, *U*, and *V* represent the components of the LUV color space converted from XYZ.

The LAB color space is like the LUV color space. The LAB color space expresses color as a combination of lightness (L) and two-color components, A for green-red, and B for blue-yellow. The LAB color space was designed with the goal of being perceptually uniform for human observers, meaning that values represented in the LAB color space should closely represent those perceived by the human eye. The primary differences between LAB and LUV are the chromaticity components, which are calculated in a different way that may yield different segmentation results. One of the goals of these experiments was to determine whether the LAB and LUV color spaces yield significantly different segmentation results. The process for converting from RGB to LAB also requires XYZ as an intermediate color space, as described in equation (13). Converting from XYZ to LAB also requires a reference white, X_r, Y_r, and Z_r. The process for converting from XYZ to LAB is described in equations (17) through (22).

$$f_1 = \begin{cases} {X_r}^{1/3} & if\ X_r > 0.008856 \\ \dfrac{903.3 * X_r + 16}{116} & else \end{cases} \tag{17}$$

$$f_2 = \begin{cases} {Y_r}^{1/3} & if\ Y_r > 0.008856 \\ \dfrac{903.3 * Y_r + 16}{116} & else \end{cases} \tag{18}$$

$$f_3 = \begin{cases} {Z_r}^{1/3} & if\ Z_r > 0.008856 \\ \dfrac{903.3 * Z_r + 16}{116} & else \end{cases} \tag{19}$$

$$L = 116 * f_2 - 16 \tag{20}$$

$$A = 500 * (f_1 - f_2) \tag{21}$$

$$B = 200 * (f_2 - f_3) \tag{22}$$

Each of the four-color spaces were used in these experiments, holding other variables constant and applying KM and FCM in separate trial runs, processing all 300 images of the BSDS. During the trials where color space was varied, *max_iter* was held constant at 50 for KM and 8 for FCM, *n_clusters* was held constant at 8, and *m* was held constant at 2 for FCM.

D. K-Means Segmentation Algorithm

The K-Means (KM) algorithm is one of the older clustering algorithms, but it is still in wide use today because of its speed and simplicity. KM seeks to split any dataset into k-clusters, where the number of clusters is pre-defined by the user of the algorithm. KM stores k-centroids, which describe the centers of the k-clusters in the dataset. A point in the dataset, in this case a pixel, is considered a member of a cluster if it is closer to the centroid of that cluster than it is to any of the other centroids in any of the other clusters. Each centroid is simply calculated as the geometric mean of that cluster. The centroids were initialized randomly for the purposes of these experiments. KM should generally move the centroids to where a human observer of the data might expect them to be located regardless of starting position, so the initial centroid location is not always critical to performance; however, initially well-placed centroids can dramatically speed up convergence to the solution.

With each iteration of KM, the membership matrix – defining what cluster each pixel belongs to – is updated and the centroids recalculated. There are two possible stop criteria for KM: (a) the maximum number of iterations allowed by the user is reached; (b) the previous centroids are the same as the current centroids, within some margin, and thus there is no reason to continue to the next iteration of the algorithm. The implementation of KM used for these experiments used a Euclidean distance measurement for determining distance from each pixel to each centroid when calculating the membership matrix.

The objective equation in (23) is minimized by the K-Means algorithm.

$$J(V) = \sum_{i=1}^{K}\sum_{j=1}^{K_i}\left(\|x_i - v_j\|\right)^2 \tag{23}$$

$\|x_i - v_j\|$ represents the Euclidean distance between a data point or pixel x_i and a centroid v_j of one of the clusters. K represents the total number of clusters, while K_i represents the total number of data points or pixels in cluster *i*.

The two variables which were varied in these experiments were the maximum number of iterations of the algorithm, *max_iter*, and the number of clusters, *n_clusters*. Through some initial experimentation, an appropriate range of values to test for each variable were determined. For *n_clusters*, values between 6 and 10 were tested against on the entire BSDS, while all other variables were held constant, with a *color_space* of RGB and *max_iter* of 50. For *max_iter*, values of 5, 10, 20, 30, 40, and 50 were tested, while all other variables were held constant, with a *color_space* of RGB and *n_clusters* of 8. Finally, as previously mentioned, the *color_space* was varied between RGB, HSV, LUV, and LAB, while *n_clusters* was held constant at 8 and *max_iter* was held constant at 50.

E. Fuzzy C-Means Segmentation Algorithm

The Fuzzy C-Means (FCM) segmentation algorithm has a similar reasoning behind it, relative to KM, but with the key difference that its membership matrix allows every point in the dataset to belong a little bit to each centroid or cluster. In other words, the closer a datapoint is to one centroid over another, the more that datapoint belongs to that cluster over another. The number of elements of a membership matrix for one datapoint equals the number of clusters, *n_clusters*, and the sum of the membership matrix for one datapoint equals one. As in KM, the membership matrix and cluster centers are recalculated with each iteration of the algorithm. The maximum number of iterations of FCM is defined by the user, *max_iter*. The membership matrix can be calculated through application of the formula in (24).

$$\mu_{ij} = \frac{1}{\sum_{k=1}^{C}\left(\frac{d_{ij}}{d_{ik}}\right)^{\frac{2}{m}-1}} \tag{24}$$

d_{ij} represents the Euclidean distance between the ith data point and the jth cluster center, d_{ik} represents the Euclidean distance between the ith data point and the kth cluster center, *m* represents the fuzziness index, *C* represents the number of clusters, and μ_{ij} represents the amount of membership datapoint *i* has to cluster *j*. The cluster centers are recalculated by means of equation (25).

$$v_j = \frac{\sum_{i=1}^{N} x_i * \left(\mu_{ij}\right)^m}{\sum_{i=1}^{N}\left(\mu_{ij}\right)^m} \tag{25}$$

N represents the number of datapoints, x_i represents the ith datapoint in the set, v_j represents the jth cluster center, and all other variables have the same meaning as in equation (24). The centroids are recalculated based on a weighted sum of all datapoints each weighted by their degree of membership in each centroid. The maximum number of iterations, *max_iter*, is one of the stopping conditions for the algorithm. Another stopping condition is if the *error*, defined as the norm of the difference between the previous membership matrix and the current one is less than some user-defined value.

For the purposes of these experiments, only the maximum number of iterations – *max_iter* – the number of clusters - *n_clusters* – and the fuzziness factor – *m* – were varied and studied. Some initial experimentation was used to determine appropriate ranges of values to test for each variable. The *error* was held constant at 0.005. For when *n_clusters* was tested between 6 and 10, *max_iter* was held constant at 8, *m* was held constant at 2, and *color_space* at RGB. For when *max_iter* was tested between 6 and 10, *n_clusters* was held constant at 8, *m* was held constant at 2, and *color_space* at RGB. For when *m* was tested for values of 0.01, 0.5, 0.8, 0.9, 0.95, 0.99, 1.01, 1.1, 1.2, 1.5, 2, and 5, *n_clusters* was held constant at 8, *max_iter* was held constant at 8, and *color_space* was held constant at RGB. Finally, for when *color_space* was tested for values of RGB, HSV, LUV, and LAB, *n_clusters* was held constant at 8, *m* was held constant at 2, and *max_iter* was held constant at 8. Since FCM produces a fuzzy membership, ultimate segmentation labels were determined based on what segment each data point most belonged to.

F. Implementation

All experiments were carried out in the Python programming environment [17]. For the purposes of these experiments, preexisting implementations of KM and FCM from the sklearn Python library were utilized [18]. The NPR for each combination of algorithm segmentation and each version of ground truth segmentation for the same image was calculated using the adjusted Rand Index function of the sklearn library and then averaged together to arrive at the average NPR for each image.

V. Results

A. K-Means Algorithm

This section outlines the results obtained during these experiments, with all segmentation results evaluated based on NPR through comparison against the ground truth segmentations provided in the BSDS as well as some qualitative analysis. The results of varying *color_space*, *max_iter*, and *n_clusters* for KM, follow in Fig. 2, Fig. 3, and Fig. 4, respectively.

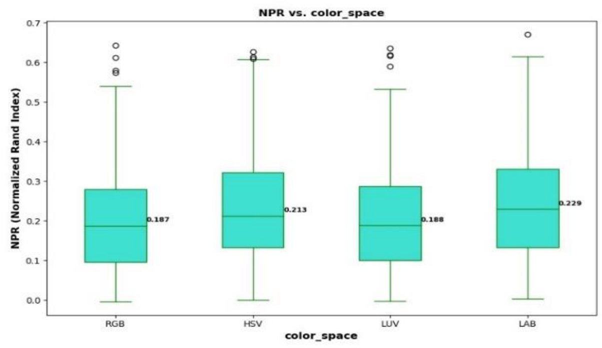

Fig. 2 K-Means NPR vs. Color Space

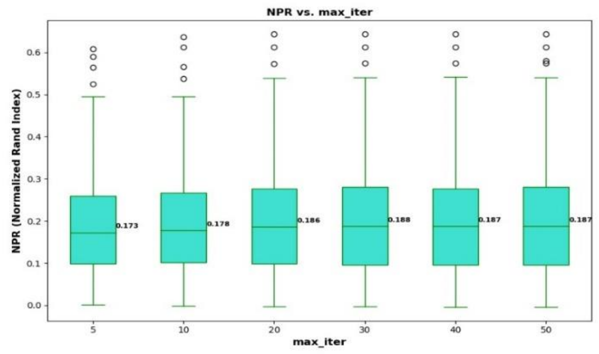

Fig. 3 K-Means NPR vs. Maximum Number of Iterations

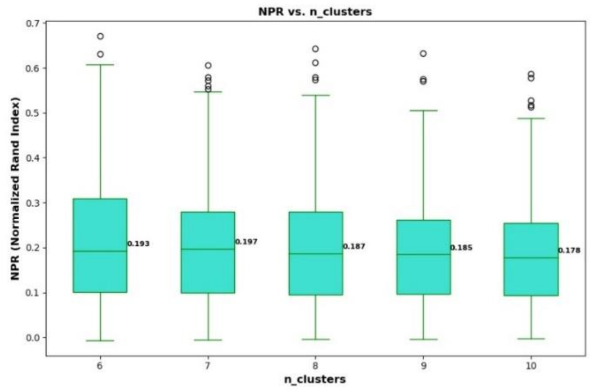

Fig. 4 K-Means NPR vs. Number of Clusters

The boxplot in Fig. 2 shows that both the HSV and LAB color spaces provide superior segmentation results in terms of average NPR, with LAB leading. There is a significant gap between RGB and LUV, with values of 0.187 and 0.188 respectively, and HSV and LAB, with values of 0.213 and 0.229 respectively. In other words, from best performing to worst performing color spaces, there is LAB, HSV, LUV, and RGB. This indicates that, at least for the K-Means algorithm, the LAB color space provides better segmentation results on average, by a margin of ~0.016 relative the next best option of HSV.

The plot in Fig. 3 shows NPR measurements across the entire BSDS as the allowed maximum number of iterations of the K-Means algorithm was varied from 5 to 50. The results of these trials were quite uniform, except for when the maximum number of iterations exceeded 20, as there was a jump of about 0.009 in the average NPR between maximum iteration values of 10 and 20. After *max_iter* exceeds 20, there is little difference in performance, suggesting a near-optimal value for the maximum number of iterations of KM is near 30, as this is where the average NPR peaks.

The plot in Fig. 4 shows NPR values calculated for when different numbers of clusters were specified for KM. The results suggest an optimal value for *n_clusters* is 7. Application-specific values for *n_clusters* may be needed in production settings, but this value of 7 provides a good default for general use-cases of KM in color image segmentation. It is also worth noting that the spread of NPR values decreases as *n_clusters* increases, providing tighter and perhaps more reliable and consistent segmentation results, albeit with an average less than that provided by a value of 7 for *n_clusters*.

To aide in further visualizing and understanding the segmentation results presented in Fig. 2, Fig. 3, and Fig. 4, displays of some of the segmentations from each experiment are represented in Fig. 5, Fig. 6, and Fig. 7.

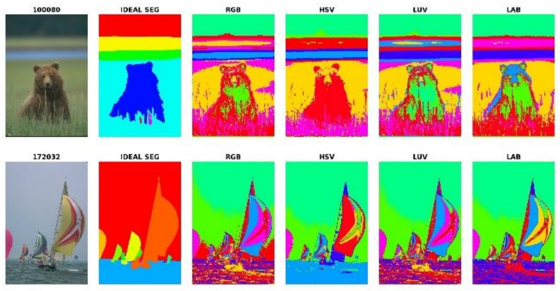

Fig. 5 K-Means Color Space Sample Segmentations

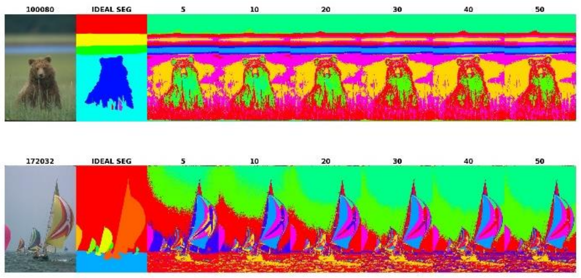

Fig. 6 K-Means Maximum Number of Iterations Sample Segmentations

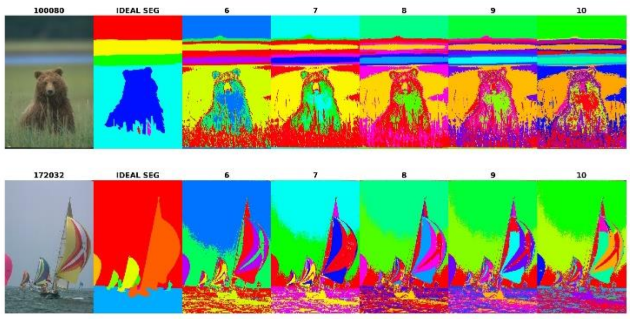

Fig. 7 K-Means Number of Clusters Sample Segmentations

The images in Fig. 5 support what the NPR boxplots in Fig. 2 suggested, which is that the HSV and LAB color spaces provide superior segmentation results. By comparing the ideal segmentation of the bear image to the segmentation results when using HSV or LAB, it is clear both color spaces do a reasonably good job segmenting the background and grass in the bear image, with HSV perhaps performing better when segmenting the bear itself, as the bear appears to be of just two segments for HSV but of more than 2 for LAB. Similarly, the image of the sailboat is well-segmented in the HSV and LAB color spaces, again perhaps more so in the HSV color space as the water is clearly of just one segment for HSV but of two or three for LAB and the ideal segmentation uses just one segment for the water. These results generally indicate HSV and LAB perform well in segmenting background from foreground. LAB appears to have some trouble in segmenting the bear in the bear image, due to shadows near the bear. All color spaces seemed to have issue with segmenting the sail of the sailboat, as they all used at least three different segments because of the three different colors used in the sail. Depending on the application, segmentation of each color in the sail may or may not be desired so this result should be considered on an application-specific basis.

The segmentation samples in Fig. 6 also support the quantitative data, at least for the sailboat image, as there is gradual improvement in segmentation of the background up until *max_iter* reaches 30 and then little changes. The reason little changes is probably that the centroids had largely stabilized in their locations by 30 iterations and additional iterations only produce minute shifts in the clusters barely noticeable to the human eye. It is worth noting that segmentation of the background in the bear image appears to change little, while segmentation of the background in the sailboat image changes significantly based on the maximum number of iterations allowed.

Finally, the segmentation samples in Fig. 7 clearly show how the bear is over-segmented at the extreme *n_clusters* values of 6 and 10 and achieves the best segmentation at 8. Segmentation of the bear appears better when *n_clusters* is 8, but segmentation of the grass in the bear image is better when the variable is 7. Similarly, the sail and water in the sailboat image appear the least over-segmented when *n_clusters* is 7.

Based on all the results for KM, the near-optimal values of the algorithm's variables are to use the LAB color space, a *max_iter* value of 30, and a *n_clusters* value of 7. The results of using these near-optimal settings are represented in Fig. 8.

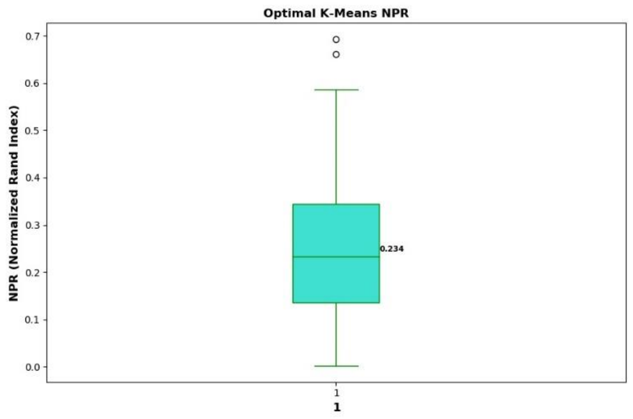

Fig. 8 K-Means NPR Using Near-Optimal Variable Settings of *n_clusters*=7, *max_iter*=30, and *color_space*=LAB

The segmentation results in Fig. 8 show a relative improvement of the average NPR – 0.234 – by 0.047 from the average NPR calculated when using all the default variable settings – 0.187. This supports all the observations made previously about what the near-optimal values of the different parameters are, as this new average NPR is better than all previous. Some sample segmentations resulting from the optimal settings are presented in Fig. 9.

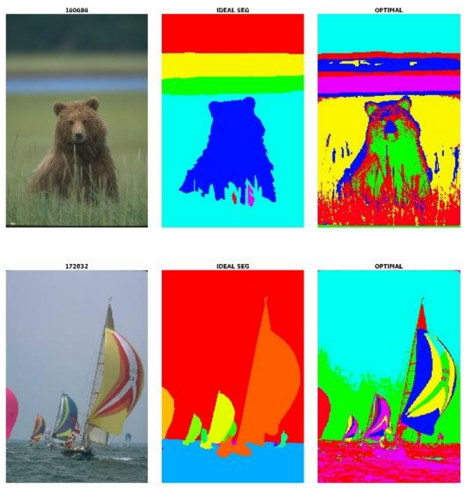

Fig. 9 K-Means Sample Segmentations Using Optimal Parameter Values

The segmentations in Fig. 9 demonstrate the reasonably good quality of segmentation which can be achieved when using the near-optimal parameter values determined through these experiments for KM. The bear is over-segmented, but the background segmentation is like the ground truth segmentation, with the exception that the grass in front of the bear is again in a different segment than the rest of the background, suggesting KM may be useful in just separating foreground from background. The sailboat sail is once again over-segmented, but this could be up for debate depending on the application.

B. *Fuzzy C-Means Algorithm*

The segmentation results of varying *color_space*, *max_iter*, *n_clusters*, and *m* for FCM, are represented in Fig. 10, Fig. 11, Fig. 12, and Fig. 13.

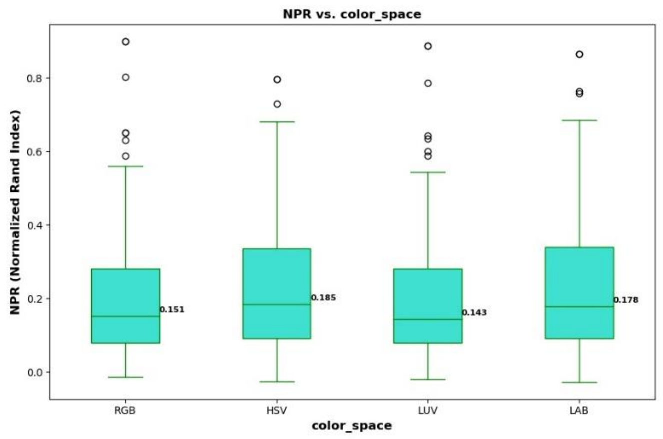

Fig. 10 Fuzzy C-Means NPR vs. Color Space

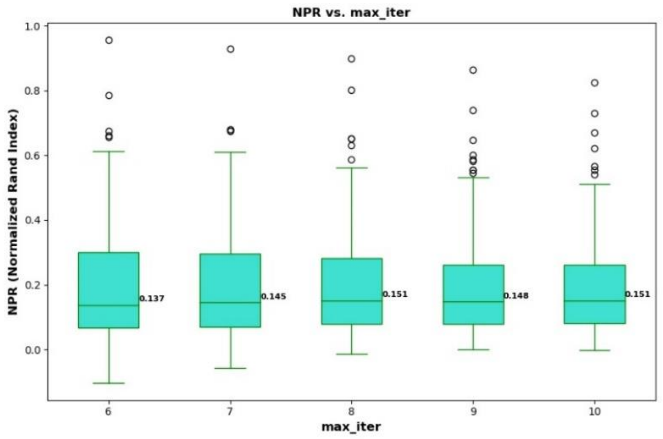

Fig. 11 Fuzzy C-Means NPR vs. Maximum Number of Iterations

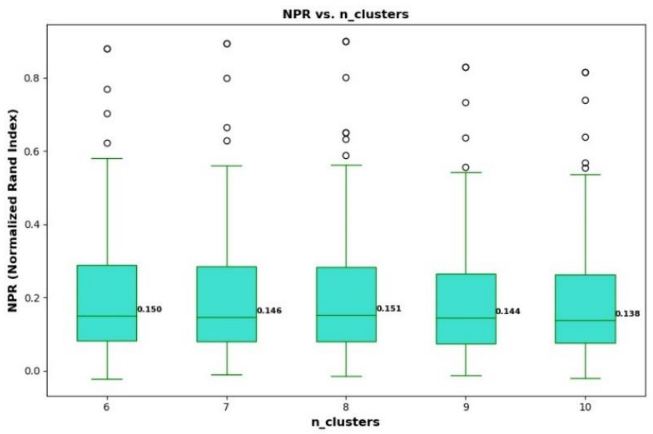

Fig. 12 Fuzzy C-Means NPR vs. Number of Clusters

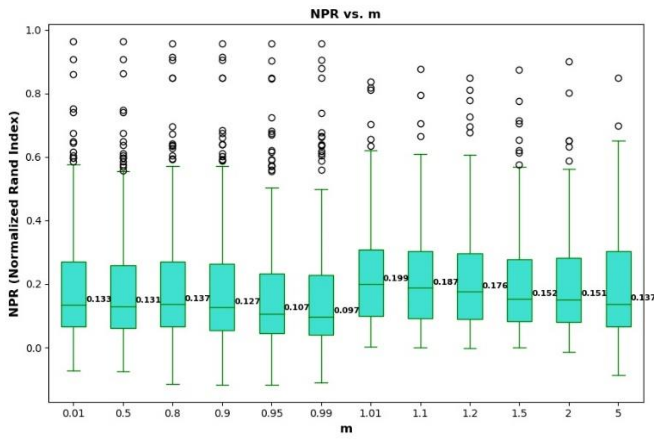

Fig. 13 Fuzzy C-Means NPR vs. Fuzziness Factor

The plot in Fig. 10 shows the FCM segmentation results versus various color spaces. The first thing to note about these results is that they are significantly worse than the KM results. In fact, the worst average NPR of KM – 0.187 – which came from using the RGB color space beats the best average NPR of FCM – 0.185. From best performing to worst performing color space for FCM, there is HSV, LAB, RGB, and LUV. These results are interesting relative to the KM results in Fig. 2, because the positions of HSV and LAB have flipped as well as those of RGB and LUV in the list of best performing to worst performing color spaces.

The plot in Fig. 11 represents the segmentation results from varying the maximum number of iterations allowed for the FCM algorithm and suggests a *max_iter* value of 8 is near optimal. The average NPR value remains pretty much flat for *max_iter* values between 8 and 10, and since every iteration of the algorithm increases computation time, the lowest value of 8 should be used. Once again, the FCM algorithm performs far worse than KM, with a margin of ~0.022 between the best average NPR for FCM and the worst average NPR for KM for the same test.

The plot in Fig. 12 shows the NPR values calculated from varying the number of clusters, *n_clusters*, between 6 and 10, with a peak in the average NPR of 0.151 at 8. The average NPR tends not to change too much between *n_clusters* values of 6 and 8, so 6 may be suitable as there is only a 0.001 differential between the NPR at 6 and the NPR at 8. Once again, FCM performs far worse than KM for the same test with a differential of 0.027 between the best-case average NPR for FCM and worst-case average NPR for KM.

Finally, the plot in Fig. 13 shows NPR values calculated while varying the fuzziness factor, *m*, which is specific to FCM and not a variable of KM. The results suggest a value of *m* just barely exceeding 1 provides superior segmentation results, possibly 1.01. The results of this test are interesting because they almost appear asymptotic near a value of 1 for *m*, but this makes some sense as *m* is part of exponentials in equations (24) and (25) used to calculate the membership matrices and cluster centers, respectively. It is also worth noting that the NPR distributes become tighter after *m* exceeds 1.

Once again, to better understand the quantitative results, some sample segmentations from application of the FCM algorithm are presented in Fig. 14, Fig. 15, Fig. 16, and Fig. 17.

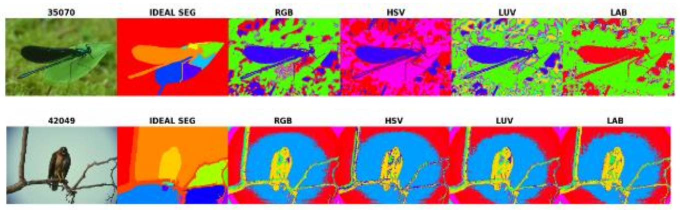

Fig 14 Fuzzy C-Means Color Space Sample Segmentations

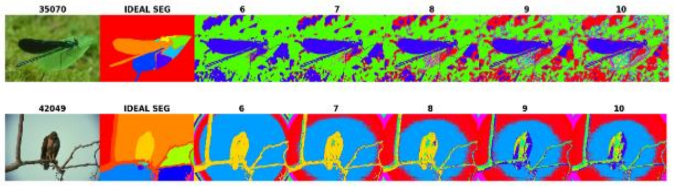

Fig 15 Fuzzy C-Means Maximum Number of Iterations Sample Segmentations

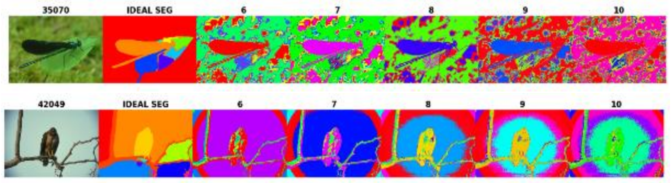

Fig 16 Fuzzy C-Means Number of Clusters Sample Segmentations

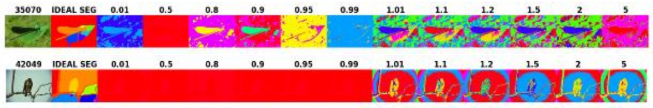

Fig 17 Fuzzy C-Means Fuzziness Factor Sample Segmentations

The plots in Fig. 14 of the dragonfly support the quantitative results, as the dragonfly image appears the least over-segmented when the HSV color space is used. The image of the falcon is segmented in pretty much the same way regardless of color space, suggesting some weakness in all the color spaces in that they all result in a segmentation including the bright circle of light surrounding the bird, which is not segmented this way in the ideal segmentation.

In Fig. 15, the images of the falcon show the background as being over-segmented when the maximum number of iterations is 6 and the bird over-segmented when the maximum number of iterations is 10, supporting the quantitative data in picking 8 as a generally good choice. On the other hand, segmentation of the dragonfly image only appears to get worse as the maximum number of iterations increased, with a value of 6 providing good segmentation between the dragonfly and the background.

Similarly, Fig. 16 shows over-segmentation of the background for low values for the number of clusters and over-segmentation of the bird for high values, with 7 or 8 being a happy medium. It is interesting that as the number of clusters increases in the image of the Falcon, lighting near the falcon becomes more and more of a problem, eventually leading to over-segmentation of and near the bird.

Finally, Fig. 17 shows sample segmentations for when the fuzziness factor was varied. It is interesting to point out the tremendous differences in segmentation results between the image of the dragonfly and that of the falcon. For the first value of *m* of 0.01, the dragonfly is well separated from the background, but the falcon image appears heavily under-segmented and consists of only one segment. The same pattern repeats for *m* values of 0.8 and 0.9. It isn't until *m* is 1.01 that both images achieve decent segmentation results at the same time. This highlights the challenge of image segmentation, because creating a one-size-fits-all set of parameters may not be possible and that is why a great deal of performance tuning is done on an per-application basis.

Overall, the plots in Fig. 10, Fig. 11, Fig. 12, and Fig. 13, provide a set of near-optimal settings for the FCM algorithm. The results of these experiments suggest near-optimal variable settings of *color_space*=HSV, *n_clusters*=8, *max_iter*=8, and *m*=1.01. The results of using these near-optimal settings for the FCM algorithm are represented in Fig. 18.

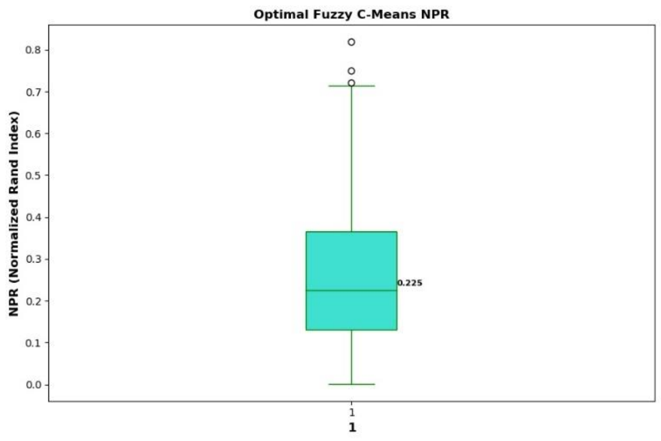

Fig. 18 Fuzzy C-Means NPR Using Near-Optimal Variables Settings of *color_space*=HSV, *n_clusters*=8, *max_iter*=8, and *m*=1.01

The plot in Fig. 18, like Fig. 8, shows a tremendous improvement in segmentation performance when the near-optimal variable settings determined through these experiments are used. The differential between the average NPR – 0.225 – of Fig. 18 and the average NPR when the default settings are used – 0.151 – of Fig. 10 is 0.074, producing FCM performance far closer to KM peak performance of 0.234 for average NPR. Some sample segmentations resulting from the optimal settings are presented in Fig. 19.

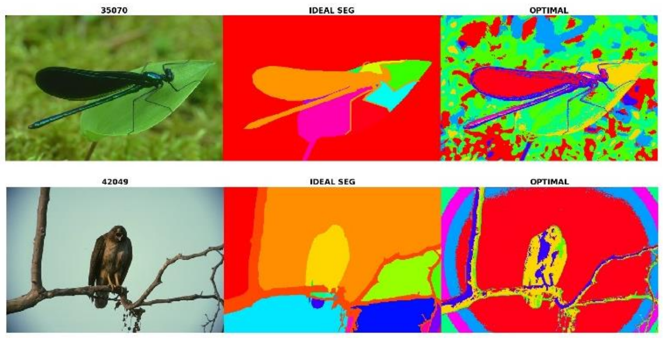

Fig. 19 Fuzzy C-Means Sample Segmentations Using Optimal Parameter Values

The segmentations in Fig. 19 again demonstrate the reasonably good performance improvements which can be gained by using the near-optimal settings determined through these experiments. There are still problems with these segmentations though, as both the image of the dragonfly and of the falcon are over-segmented and lighting in both images is negatively impacting segmentation performance, most notably in the image of the falcon where a set of rings have sprung up around the bird and segmentation of the tree branch and bird varies based on lighting. One last note is that FCM did a good job in finding the edges of the dragonfly, even if it did over-segment it.

VI. CONCLUSION

Setting optimal parameter values for K-Means and Fuzzy C Means segmentation algorithms can be a challenging task. The current work tests the impact of color space, the maximum number of iterations, the number of clusters, and the fuzziness factor on segmentation accuracy using the Normalized Probabilistic Rand Index as well as some qualitative analysis. The results show that the LAB and HSV color spaces provide superior segmentation results to the RGB and LUV color spaces, while also showing a good value for the number of clusters is 7 or 8, a good value for the maximum number of iterations is 30 for KM and 8 for FCM, and a good fuzziness factor is ~1.01. The results of this study demonstrate the gains which can be had in segmentation performance when parameter values are optimized for as well as provide a set of good default parameter values to use with these algorithms. Further research could go into development and testing of genetic algorithms for use in parameter tuning on a per-application basis to optimize for images of specific subject matter.

ACKNOWLEDGMENT

Connor Maynes thanks Professor Saber of the Rochester Institute of Technology for his support and guidance of this research.

REFERENCES

[1] *Encyclopedia of Information Science and Technology*. Ed. Mehdi Khosrow-Pour. Vol. 7. 2nd. ed. Hershey, PA: Information Science Reference, 2009. p3224.W.-K. Chen, *Linear Networks and Systems* (Book style). Belmont, CA: Wadsworth, 1993, pp. 123–135.

[2] Real-Time Superpixel Segmentation by DBSCAN Clustering Algorithm - IEEE Journals & Magazine. (2018, January 1). Retrieved December 16, 2018, from https://ieeexplore-ieee-org.ezproxy.rit.edu/document/7588062

[3] Rajaby, E., Ahadi, S. M., & Aghaeinia, H. (2016). Robust color image segmentation using fuzzy c-means with weighted hue and intensity. Digital Signal Processing, 51, 170–183. https://doi.org/10.1016/j.dsp.2016.01.010

[4] Image segmentation based on adaptive K-means algorithm - ProQuest. (2000, January 1). Retrieved December 16, 2018, from https://search-proquest-com.ezproxy.rit.edu/docview/2082506910?pq-origsite=summon

[5] Dhanachandra, N., & Chanu, Y. J. (2015). Image Segmentation Method Using K-Means Clustering Algorithm for Color Image. Advanced Research in Electrical and Electronic Engineering, 2.

[6] C. Fowlkes, D. Martin, J. Malik. "Local Figure/Ground Cues are Valid for Natural Images" Journal of Vision, 7(8):2, 1-9.

[7] Bora, D. J., Gupta, A. K., & Khan, F. A. (2015). Comparing the performance of L* A* B* and HSV color spaces with respect to color image segmentation. arXiv preprint arXiv:1506.01472.

[8] Toure, S., Diop, O., Kpalma, K., & Maiga, A. S. (2018, July). Best-Performing Color Space for Land-Sea Segmentation. In 2018 41st International Conference on Telecommunications and Signal Processing (TSP) (pp. 1-5). IEEE.

[9] Qiu-yu, Z., Jun-chi, L., Mo-Yi, Z., Hong-xiang, D., & Lu, L. (2015). Hand gesture segmentation method based on YCbCr color space and K-means clustering. Interaction, 8, 106-116.

[10] Abbas, A. W., Minallh, N., Ahmad, N., Abid, S. A. R., & Khan, M. A. A. (2016). K-Means and ISODATA clustering algorithms for landcover classification using remote sensing. Sindh University Research Journal-SURJ (Science Series), 48(2).

[11] Subbalakshmi, C., Krishna, G. R., Rao, S. K. M., & Rao, P. V. (2015). A method to find optimum number of clusters based on fuzzy silhouette on dynamic data set. Procedia Computer Science, 46, 346-353.

[12] Arbelaez, P., Maire, M., Fowlkes, C., & Malik, J. (2011). Contour detection and hierarchical image segmentation. IEEE transactions on pattern analysis and machine intelligence, 33(5), 898-916.

[13] W. M. Rand, “Objective criteria for the evaluation of clustering methods,” Journal of the American Statistical Association, vol. 66, pp. 846–850, 1971.

[14] Vinh, N. X., Epps, J., & Bailey, J. (2010). Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. Journal of Machine Learning Research, 11(Oct), 2837-2854.

[15] Hubert, L., & Arabie, P. (1985). Comparing partitions. Journal of Classification, 2(1), 193–218. https://doi.org/10.1007/bf01908075

[16] Color Conversion Algorithms. (2000, January 1). Retrieved December 18, 2018, from https://www.cs.rit.edu/~ncs/color/t_convert.html#RGB%20to%20HSV%20&%20HSV%20to%20RGB

[17] "G. van Rossum, Python tutorial, Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, May 1995."

[18] Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.