Comparing Clustering Algorithms Project 1 Group 6 CAP 6610

Matthew Carr, Pallavi Raiturkar, Dylan Stewart, Weihuang Xu

***Abstract*—In this paper we study a variety of clustering algorithms using an RGB and a Hyperspectral dataset. We begin by introducing the clustering algorithms. Methods for applying Principal Component Analysis (PCA) are explained to assist in segmentation of the dataset. Parameters of the algorithms are also discussed along with their intricacies. Methods for parameter selection and evaluation are then developed. We lastly discuss the results on the our dataset based upon a quantitative evaluation metric known as the Object Consistency Error (OCE) and elaborate on qualitative evaluation of images without ground truth. Overall, the results show that each algorithm has a very similar mean OCE score when averaged over the RGB dataset, and they were mainly distinguished by the standard deviation of their OCE scores.**

***Index Terms*—Image Segmentation, Machine Learning, Statistical Learning, Machine Vision**

I. Introduction

In recent years, machine learning has become increasingly important in pattern recognition. As complex imagery increases in quality and supply, many previously intractable problems are being solved in the fields of image processing, computer vision, and pattern recognition. The problem we will focus on is image segmentation. With a large image filled with many objects it can be difficult to segment it into objects of interest efficiently. The algorithms we discuss in this paper are meant to address this problem. Before we introduce the specific algorithms, it is important to further explain the need for a solution.

Image segmentation is useful in any situation where semantic information about the contents of an image needs to be extracted in an automated way. For example, a medical image could be segmented into regions of distinct tissue types, or even into regions of cancerous versus healthy tissue. This example demonstrates the need for accuracy in segmentation algorithms; if cancerous tissue is incorrectly classified as healthy, a malignancy could go unnoticed. Segmentation could also be applied to machine vision. An image of a street taken from an automated vehicle could be segmented into regions which cover the other vehicles on the road, the lane the vehicle is traveling down, as well as pedestrians and other potential obstruction. This last example demonstrates the need for efficient segmentation algorithms; an automated vehicle would need to compute segmentations in real time in order to be able to react safely to road conditions.

II. Methods

In this section we will discuss the algorithms under consideration, their parameters, and their quantitative and qualitative analysis.

A. Data and Code

The dataset for this paper consists of the RGB images and segmentations from the file ImsAndSegs.zip and the hyperspectral images PaviaHyperIm.mat and SanBarHyperIm.mat. For the evaluation of hyperspectral images, we chose to run 10 experiments on each image for each algorithm to provide enough measurements for quantitative analysis. All code for this project was derived either from built in MATLAB functions or from algorithms presented in [1].

B. Algorithms

The following algorithms will be evaluated in our study: Fuzzy c-Means (FCM), Gaussian Mixture Models (GMM), K-means (Kmeans), Self-Organizing Map (SOM), and Spectral clustering (Spectral).

1) K-Means: Given a set of observations $(x_1, x_2, ..., x_n)$, where each observation is a d-dimensional real vector, k-means clustering will partition the n observations into k sets $\mathbf{S} = S_1, S_2, ..., S_k$ to minimize the within cluster sum of squares. The objective is shown in equation 1.[2]

$$\min_{\mathbf{S}} \sum_{i=1}^{k} \sum_{x \in S_i} ||x - \mu_i||^2 \tag{1}$$

To minimize this objective, an assignment step (equation 2) followed by an update step (equation 3) is computed.

2) Fuzzy C-Means(FCM): Let Y be a sample of N observations in $\mathbb{R}^N$ where y_k is the k-th feature vector; y_{kj} the j-th feature of y_k. An objective function J_m can be developed to produce a soft partition of an image:

$$J_m(U, v) = \sum_{k=1}^{N} \sum_{i=1}^{c} (u_{ik})^m ||y_k - v_i||_A^2 \tag{2}$$

Where u_{ik} is the membership of the ith pixel in the kth cluster and the following constrains are in order for the weights.

$$\sum_{i=1}^{N} u_{ik} > 0 \quad \forall i \tag{3}$$

$$\sum_{i=1}^{N} u_{ik} = 1 \quad \forall k \tag{4}$$

Where v_i is the mean vector for the ith cluster and U is the fuzzy partition matrix of Y containing labels [3].

3) Gaussian Mixture Models(GMM)[4]: Gaussian mixture model (GMM) is the most widely used mixture model, where each base distribution in the mixture is a multivariate Gaussian with mean μ_k and covariance matrix Σ_k. The GMM is a parametric probability density function represented as a weighted sum of Gaussian component densities. The equation of GMM is:

$$p\left(\mathbf{X}_i|\lambda\right) = \sum_{k=1}^{K} \pi_k g\left(\mathbf{X}_i|\mu_k, \Sigma_k\right) \tag{5}$$

Where $\mathbf{X}_i$, i=1,...N, are a D-dimensional data vectors; π_k k = 1,...K, are the weights of each component, which satisfy the constraint that $\sum_{k=1}^{K} \pi_k = 1$; $g\left(\mathbf{X}_i|\mu_k, \Sigma_k\right)$ are the component Gaussian densities. Each component is a D-variate Gaussian function of the form:

$$g\left(\mathbf{x}|\mu_k, \Sigma_k\right) = \frac{\exp\left(-\frac{1}{2}\left(\mathbf{x}-\mu_k\right)^T \Sigma_k^{-1}\left(\mathbf{x}-\mu_k\right)\right)}{\left(2\pi\right)^{D/2}\left|\Sigma_k\right|^{1/2}} \tag{6}$$

with mean vector μ_k and covariance matrix Σ_k.

The GMM is parameterized by the mean vectors, covariance matrices and mixture weights from all components. Here, the covariance matrices can be full rank or diagonal. Full covariance matrices indicate that the components are correlated. Diagonal covariance matrices represent the case that the components are uncorrelated. Also, all the components could share the same covariance matrices which means all the distributions have the same covariance. [4]

Expectation-maximization(EM) algorithm is used to estimates all the parameters in GMM. The weights are initialized equally at the beginning. Then, new model is estimated with new parameters such that $p(\mathbf{x}|\lambda^*) >= p(\mathbf{x}|\lambda)$. The new model will become the initial model for the next iteration and the process is repeated until convergence threshold is reached. The following equations are used to update parameters:

$MixtureWeights$:

$$\pi_k^* = \frac{1}{N}\sum_{i=1}^{N} Pr(k|\mathbf{X}_i, \lambda). \tag{7}$$

$Means$:

$$\mu_k^* = \frac{\sum_{i=1}^{N} Pr(k|\mathbf{X}_i, \lambda)\mathbf{X}_i}{\sum_{i=1}^{N} Pr(k|\mathbf{X}_i, \lambda)} \tag{8}$$

$Variance(diagonal)$:

$$\Sigma_k^* = \frac{\sum_{i=1}^{N} Pr(k|\mathbf{X}_i, \lambda)\mathbf{X}_i\mathbf{X}_i^T}{\sum_{i=1}^{N} Pr(k|\mathbf{X}_i, \lambda)} - (\mu_k^*)^2 \tag{9}$$

where posteriori probability for each component k is:

$$Pr(k|\mathbf{X}_i, \lambda) = \frac{\pi_k g\left(\mathbf{X}_i|\mu_k, \Sigma_k\right)}{\sum_{k'=1}^{K} \pi_k' g\left(\mathbf{X}_i|\mu_k', \Sigma_k'\right)} \tag{10}$$

4) Self-Organizing Map(SOP): The self-organizing map algorithm is a machine learning algorithm which uses an artificial neural network to perform dimensionality reduction (in particular classification). Suppose we have $D_1, ..., D_t \in \mathbb{R}^D$, a set of training data. Our goal is to learn a map $F : \mathbb{R}^D \to \mathbb{R}^d$, where $d < D$. Let Let $N_1, ..., N_v \in \mathbb{R}^d$, this is the set of neurons. To neuron N_k we assign $W_k(s) \in \mathbb{R}^D$, its *weight*, where s is the step index. The weights evolve from one step to the next according to the rule

$$W_k(s+1) = W_k(s) + \theta(k, \ell, s) \cdot \alpha(s) \cdot (D_i - W_k(s)),$$

where α is a monotonically decreasing function, and θ (called a *neighborhood function*) is a decreasing function of s and decreases as $||N_k - N_\ell||$ increases. Notice that this update rule depends on a choice of i. This choice is important and has an impact on the convergence rate of the network. Common strategies for choosing i is to pick randomly (called bootstrapping) or to systematically scan through i to find the index which produces the largest change, as well as other methods. The above rule is applied until the network "converges" or until the implementer gives up and sends SIGTERM.

Suppose that training ends after λ steps. Then the mapping of a novel input $X \in \mathbb{R}^D$ is computed by finding the weight $W_k(\lambda)$ which is closest to X and then declaring $F(X) = N_k$.

5) Spectral Clustering: Consider a set of observations $\{x_1, x_2, ..., x_n\}$, where each observation is a d-dimensional real vector. Let W denote the similarity matrix of these observations and k be the number of clusters we wish to construct. The k eigenvectors $\{v_1, v_2...v_k\}$ corresponding to the smallest k eigenvalues of the matrix are computed, and are used to construct a matrix V with the eigenvectors as its columns. The rows of V are treated as new observations Z_i, which are then clustered using the K means algorithm described previously.

C. Evaluation Metrics

1) Object Consistency Error: We used a quantitative measure of segmentation consistency called the Object Consistency Error (OCE) to evaluate the performance of each algorithm. The OCE is defined in [5]. This algorithm takes as input two segmentations A and B and outputs a real number E such that $0 \leq E \leq 1$. The number E has the property that $E = 0$ if $A = B$. So the smaller E is the more similar A and B are. To compute E we first define a partial error as follows:

$$E(A, B) = \sum_{j=1}^{M}\left(1 - \sum_{i=1}^{N}\frac{|A_j \cap B_i|}{|A_j \cup B_i|}W_{ji}\right)W_j, \tag{11}$$

where M and N are the number of segments in A and B respectively, and

$$W_{ji} = \frac{\bar{\delta}(|A_j \cap B_i|)|B_i|}{\sum_{k=1}^{N}\bar{\delta}(|A_j \cap B_k|)|B_k|} \tag{12}$$

$$W_j = \frac{|A_j|}{\sum_{\ell=1}^{M}|A_\ell|}. \tag{13}$$

Algorithm	$1000 \cdot \mu$	$1000 \cdot \sigma$
SOM	824	120
GMM	827	124
Spectral	836	140
Kmeans	872	94
FCM	883	91

TABLE I
MEAN AND STANDARD DEVIATION OF THE OCE SCORES FOR EACH ALGORITHM

Here

$$\bar{\delta}(x) := \begin{cases} 1 & \text{if } x \neq 0 \\ 0 & \text{if } x = 0 \end{cases}.$$

Then

$$E := \min(E(A,B),\, E(B,A)).$$

2) Qualitative Analysis: Due to a lack of ground truth for the Santa Barbara images, we must look to qualitatively analyze these images. Our way of dealing with this issue is to develop a bounding box image with boxes around what we consider "pure and separate regions." This can be seen in figure (insert figure number). We drew bounding boxes around regions of Santa Barbara that have objects or groups of objects that should be separated from their neighbors. Although this is in our view a very convoluted way to compare segmentations with our base image, an absence of ground truth does not present the authors with many options at this time.

D. Principal Component Analysis

When the input data is high dimensional, it is a good idea to look into lowering the dimensionality to speed up computation. One technique that is widely used for dimensionality reduction is Principal Component Analysis (PCA), also known as the Karhunen-Lo'eve transform. PCA is an orthogonal projection of the data onto a lower dimensional linear space, so that the variance of the projected data is maximized [1]. For our purposes we performed PCA on the input data and used a total number of components that contained greater than 95 percent of the variance. The way we determined this was by summing the first highest eigen values of the covariance matrix and dividing this sum by the total sum of the eigenvalues. The first N eigen vectors are then used to project the data to a lower dimensional space. The proof for this method can be found in Bishop's Pattern Recognition Book and is too lengthy to provide in this paper.

III. EXPERIMENTS

A. RGB Experiments

To evaluate the performance of the algorithms, a segmentation was computed for each of the 198 images in ImsAndSegs.zip using each algorithm. The number of clusters to use in the segmentation was determined by the number of clusters in the variable Seg1. So in MATLAB code we have

```
NumClusts = length(unique(Seg1));
```

These segmentations were then compared to each of the three ground truths using the OCE. The minimum of the three OCE values was taken as the score for that algorithm on that image. The mean and standard deviations of these scores were then computed for each algorithm and are summarized in Table I.

The mean of these scores is 848 with a standard deviation of 27. So the performance of each algorithm was very similar when averaged over the whole dataset, with Self-Organizing Map having the lowest average score.

B. Hyperspectral Experiments

TABLE II
HYPERSPECTRAL SCORES

Alg:	KM μ	KM σ	FCM μ	FCM σ	SP μ	SP σ	GM μ	GM σ	SOM μ	SOM σ
All D:	860	97	869	0	0	0	766	0	764	0
PCA:	850	96	844	0	0	0	810	0	837	0

1) Pavia: For the Pavia hyperspectral image, we ran 10 experiments with each algorithm and computed the Martin index. To compute the Martin index we took the output cluster image (ClusterIm) and masked it using the ground-truth mask (PaviaGrTruthMask) and compared it to the ground-truth (PaviaGrTruth). With algorithms such as spectral clustering where we downsampled the image we also downsampled the ground truth and ground truth mask for a fair comparison. For the PCA experiments we used 4 components which were computed to contain greater than 95% variance. From the 10 experiments we computed the mean and variance for each method as seen in table 3. The pseudo-code for the experiments is shown below in Algorithm 1.

```
Data: Pavia Hyperspectral Image
Result: Martin Index Scores
if Compute PCA then
    Use first 3 components;
else
    Use all components;
end
for i = 1:10 do
    for Each Algorithm do
        Compute 9 Clusters;
        if Not Converging then
            Downsample Image;
            Compute 9 Clusters;
            Downsample Ground Truth and Mask;
            Compute Martin Index;
        else
            Compute Martin Index With Full Ground
             Truth;
        end
    end
end
Compute means and standard deviations;
```

Algorithm 1: Pseudo-Code for Pavia Experiments

Fig. 1. Santa Barbara Labeled

TABLE III
ALGORITHM RANKING TABLE

Ranks	RGB Rankings	Hyperspectral Rankings
1(Best):	SOM	GMM
2:	GMM	SOM
3:	Spectral	Kmeans
4:	Kmeans	FCM
5:	FCM	Spectral

2) Santa Barbara: Without ground truth information we decided to use a bounding box image as described in the methods *2.3.2* to compare the best visually looking result from each algorithm out of our 10 trials.

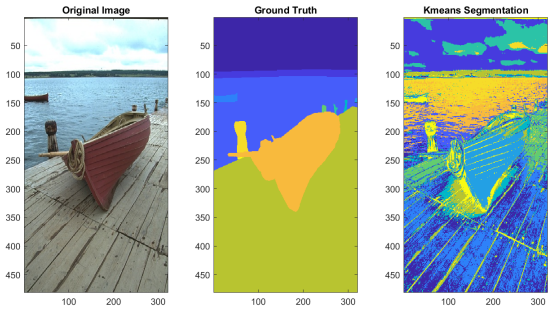

Fig. 2. K-means Segmentation Comparison

IV. OBSERVATIONS

A. K-Means Advantages and Disadvantages

The major advantage of K-Means is the speed of algorithm. Actually, it's the fastest algorithm among all five clustering algorithms. Especially when we applied it to hyperspectral images which have a lot of dimensions. No need for downsampling the images with PCA. Also, using PCA only slightly reduces performance. Another advantage of K-means we found in the experiment is that K-means has less chance to lose clusters, which means it usually return a labeled image which has the same number of cluster as you input. Even though some of those clusters may have a tiny area. However, other algorithm like GMM may lose several clusters due to the low possibility. The major disadvantage of K-Means is that it highly depends on the initialization. The initial centers will affect accuracy a lot. When we ran K-Means on the same images several times during experiments, we found that the labeled images are different each time. We need to run multiple times and find the best results. This is also the reason why K-Means has a high variance. Second disadvantage of K-Means is that the computation is all based on euclidean distances and the data may not bode well for simple distance measures. Third disadvantage of D-Means is that it requires prior knowledge of the number of clusters K which we usually don't know. Last, K-means has difficulty

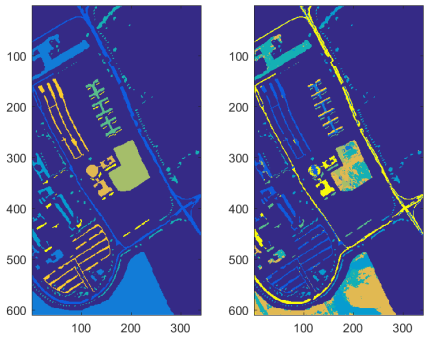

Fig. 3. Pavia K-means Results

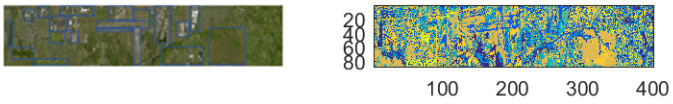

Fig. 4. Santa Barbara K-means Results 10 Clusters

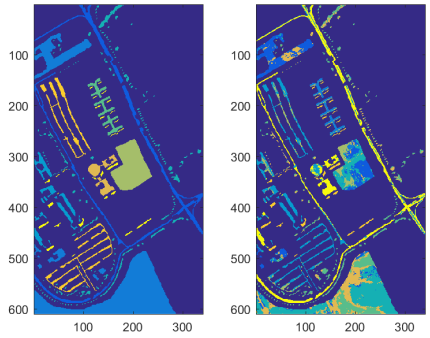

Fig. 6. Pavia FCM Results

B. *Fuzzy c-Means Advantages and Disadvantages*

FCM is very similar to K-Means. The only difference is that FCM introduce the idea of fuzzy membership. Instead of assigning a point to a specific class, FCM assign factors to each data point which will define how strongly the data point belongs to that cluster. The results we got from FCM are close to K-Means. The major advantage of FCM is speed. Even though FCM requires slightly more computation time than K- Means because of the fuzzy measures calculations involvement in the algorithm, FCM still takes much less time than the other algorithms. Other advantages are similar to K-Means, FCM can work on multi-dimensional data without downsample. The disadvantages of FCM are identical to K-Means such as requiring prior knowledge of the number of clusters K, sensitive to initialization, etc.

Fig. 7. Santa Barbara FCM Results 14 Clusters

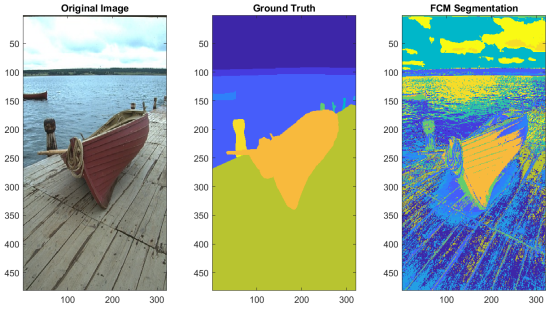

Fig. 5. FCM Segmentation Comparison

C. *Spectral Clustering Advantages and Disadvantages*

Advantages of Spectral Clustering are that when it does work correctly it can separate difficult areas into clusters. It also seems to work well with PCA. One disadvantage is the need to solve so many eigenvectors makes us needing to downsample images. Also, the Martin Index is really small because we had to compare the Martin Index on a very small image (roughly 10% of original size). The qualitative results of the images are also quite poor due to this work around.

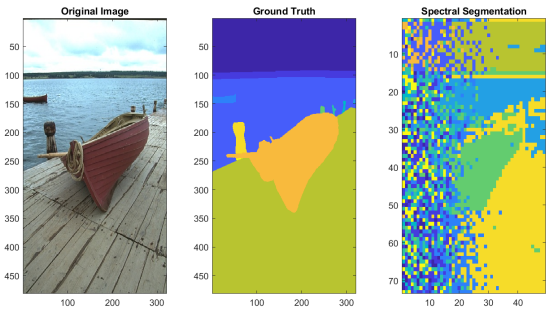

Fig. 8. Spectral Segmentation Comparison

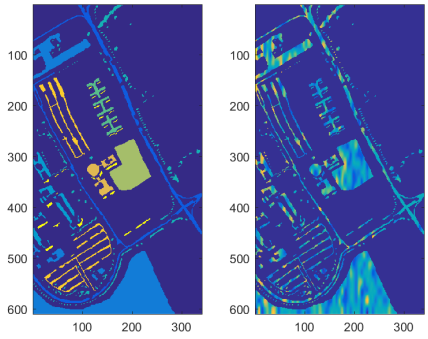

Fig. 9. Pavia Spectral Results

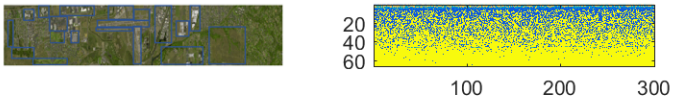

Fig. 10. Santa Barbara Spectral Results 10 Clusters

D. Gaussian Mixture Models Advantages and Disadvantages

The performance of GMM is good on both RGB images and hyperspectral images as shown in Table IV. The major advantage of GMM is that each cluster can have unconstrained covariance structure(a more general case). For example the rotated and/or elongated distribution of points in a cluster, instead of spherical as in Kmeans. As a result, cluster assignment is much more flexible in GMM. The second advantage is that GMM allows for mixed membership of points to clusters like FCM. Depending on the application, mixed membership may be more appropriate or not. Also, the Martin index shows little to no variance of segmentations, providing insight to repeatability. Last, the initialization of parameters won't affect GMM as much as K-Means. The first disadvantage of GMM is that it is a bit slower than Kmeans. If the dimensionality of the input is too high, GMM may fail to work due to computational reasons. The second disadvantage is that it is hard to estimate the covariance matrices, if one has insufficiently many points per mixture. And the algorithm is known to diverge and find solutions with infinite likelihood unless one regularizes the covariances artificially.

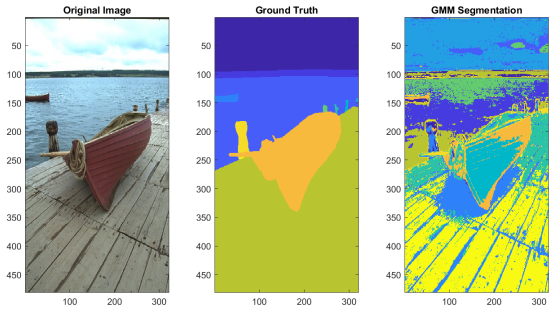

Fig. 11. GMM Segmentation Comparison

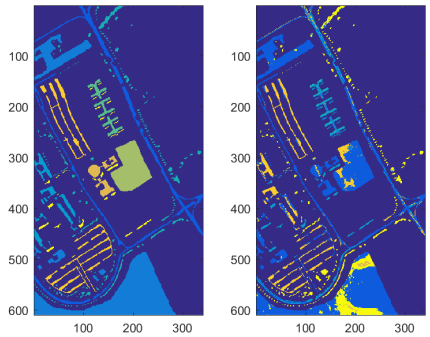

Fig. 12. Pavia Gaussian Mixture Models Results

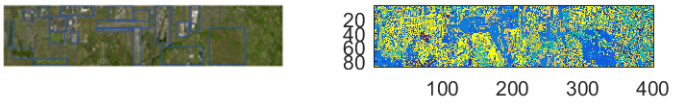

Fig. 13. Santa Barbara GMM Results 10 Clusters

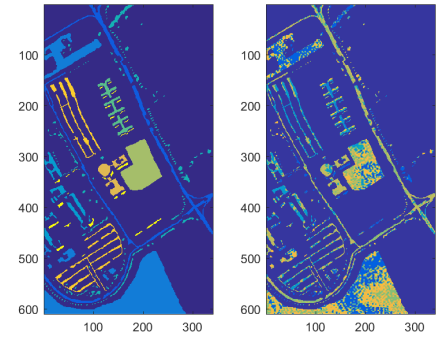

Fig. 15. Pavia Self-Organizing Map Results

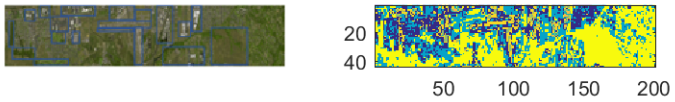

Fig. 16. Santa Barbara SOM Results 13 Clusters

E. Self Organizing Map Advantages and Disadvantages

Advantages of Self Organizing Map are not needing to downsample images and using PCA improves performance. Also, the Martin index shows little to no variance of segmentations, providing insight to repeatability. Disadvantages of Self Organizing Map is that it is a bit slower than Kmeans.

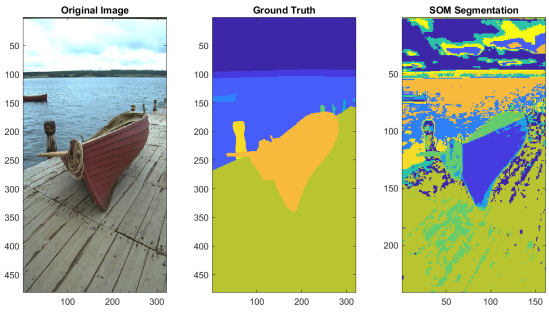

Fig. 14. SOM Segmentation Comparison

F. Relative Performance of Algorithms

While all the algorithms performed similarly when averaged over the RGB dataset, the SOM algorithm had to lowest mean OCE score and tended to provide segmentations which looked subjectively better. That being said, it was one of the slower algorithms in the set, and all that extra runtime didn't seem to translate into a proportionally better result. The algorithm which demonstrated the best runtime performance was definitely Kmeans. Considering the fact that on most images Kmeans produced segmentations which had similar OCE values compared to SOM and the others, unless accuracy is critical, Kmeans seems to be a good choice. It was also noted that certain images were just easier to segment than others, with all the algorithms doing well or all failing miserably. It seems that if an image is hard to segment, its just hard, and so it may be better to compute an answer quickly rather than wasting time trying to generate a marginally better

segmentation.

REFERENCES

[1] Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*.

[2] Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2):129–137, 1982.

[3] JC Bezdek. Pattern recognition with fuzzy objective function algorithms plenum press, new york, 1981. *MATH CrossRef Google Scholar*.

[4] Geoffrey McLachlan and David Peel. *Finite mixture models*. John Wiley & Sons, 2004.

[5] Minghong Pi Mark Polak, Hong Zhang. An evaluation metric for image segmentation of multiple objects. *Image and Vision Computing*, 27:1223–1227, 2009.