K-Means Clustering Using Localized Histogram Analysis

Michael Bryson

University of South Carolina, Department of Computer Science
Columbia, SC
brysonm@cse.sc.edu

Abstract. The first step required for many applications and algorithms is the segmentation or clustering of an image. This intuitively easy problem is actually very difficult for a computer to automatically perform. While having an effective solution to this problem would be valuable for all types of image processing, it would be especially useful in medical imaging. One method that addresses this problem is the K-means clustering algorithm. This paper presents a simple case of applying this algorithm to images, and extends it by redefining the distance measure to consider texture information. Results are shown for both algorithms, with the improvements from the modified algorithm clearly visible.

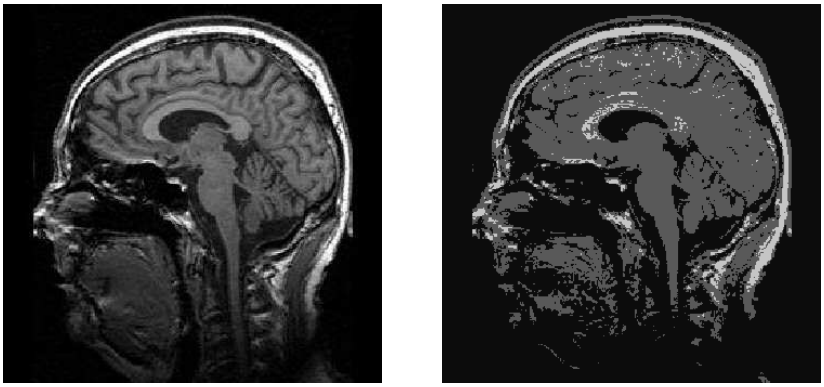
1 Introduction

A common problem that must be addressed for many applications and algorithms is that of image segmentation. Image segmentation attempts to divide an image into distinct regions either completely automatically or with manual assistance. There are several classes of segmentation algorithms including clustering, histogram based, region growing, graph partitioning, and model based. This paper focuses on a standard clustering algorithm that is fully automatic, and extends it to provide better results.

While image segmentation is a general problem which can be applied to many different types of images, it can be especially valuable in medical imaging. It can be used to differentiate between different tissues or structures, measure the size or volume of structures, locate tumors or other irregularities, etc. These identifications can be useful during diagnosis and treatment planning, as well as computer guided surgery. Additionally segmentation can be used to aid with image registration, which is itself a large research problem.

1.1 Clustering

Clustering is the process of "attempting to determine which components of a data set naturally belong together." [1] In the case of image clustering, each pixel or voxel represents a single data component. The objective is to group each pixel with other pixels that are 'close' using some definitive measure of



(a) Original Image (b) Simple Clustering

Fig. 1: These two images demonstrate a simple clustering of a brain MRI image into three different regions. Each region is denoted by a particular grayscale value, which is the average grayscale value of all the pixels in that cluster. Notice that the regions are not necessarily continuous, as they are divided into many disjoint areas.

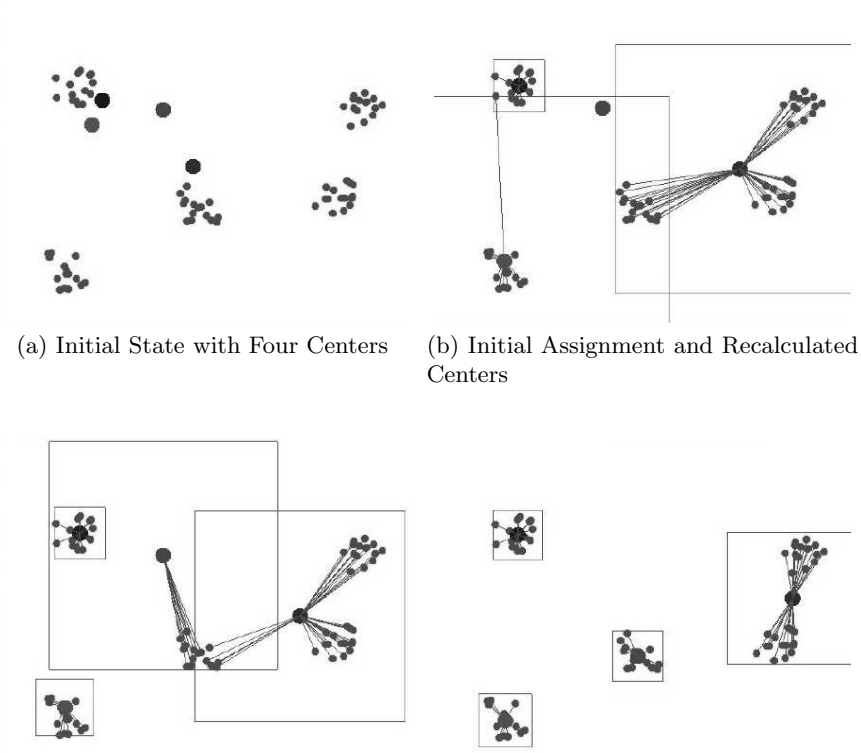
distance. The definition of 'close' and the method used to determine the best grouping can vary depending on the particular algorithm being used.

Some segmentation algorithms produce distinctly continuous regions, such as the watershed algorithm. However, clustering may produce disjoint regions that are scattered throughout the image if locality is not taken into consideration (see Fig. 1). In certain cases this may actually make sense. For example, a medical image that contains the cross section of several bones (such as a hand or foot) may have several disjoint areas that all represent bone. If the objective is to identify different tissues or structures (such as muscle or bone) the clustering of all disjoint areas corresponding to each particular material into a single region may be desired.

1.2 K-Means Algorithm

One common clustering algorithm is K-Means. [2] This algorithm applied to an image takes a number of clusters k, and attempts to group the pixels into the given number of clusters. Figure 1 above shows a simple example of K-Means with k equal to 3. The algorithm begins with some k initial centers, which can be provided manually or generated randomly or based on some heuristic. Each pixel is assigned to the 'closest' center, based on some measure defining distance. Each center is then recalculated to be the average of every pixel assigned to its cluster. The algorithm then iteratively repeats until convergence, which is guaranteed. Upon convergence, the variance among the clusters has been minimized. The final state may vary based on the initial centers however, so a global optimal solution is not guaranteed. The steps of the algorithm are summarized below.

1. Define the number of clusters k
2. Generate k centers randomly or heuristically, or have them provided
3. Assign each pixel to the closest cluster center
4. Calculate new centers based on current cluster contents
5. Repeat steps 3 and 4 until the clusters no longer change



(a) Initial State with Four Centers

(b) Initial Assignment and Recalculated Centers

(c) Assignment to Recalculated Centers

(d) Final Centers and Assignment

Fig. 2: These four figures show two iterations of the K-Means clustering algorithm. Fig. 2a shows the initial state, with each data component shown as a small dot and each initial center shown as a large dot. Fig. 2b shows the assignment of each data component to the closest center, with each center having been recalculated. Closeness in this example is simply the geometric distance between each point and a center. The boxes indicate variance within each cluster. Fig. 2c shows the assignment to the newly calculated centers. Fig. 2d shows the final centers and assignment, with minimal variance. *Images adapted from those found on the Wikipedia page describing K-means. [3]*

2 Methodology

The description of the K-Means algorithm given above is generally incomplete. Two items have been left undefined: the distance measure and the meaning of average. One simple metric is the geometric distance as demonstrated in Figure 2 and given in Equation 1, where x and y are the coordinates of a data point, and x_{ci} and y_{ci} are the coordinates of center $i, 1 \leq i \leq k$. The average in this case can easily be defined as the average of the x and y coordinates of all data points in each cluster.

$$\sqrt{(x - x_{ci})^2 + (y - y_{ci})^2} \tag{1}$$

This metric has no usefullness in segmenting an image when used alone, however. This can clearly be seen since it does not take into consideration the actual image information, or pixel intensities. Therefore given any two different images and starting the algorithm with the same centers on both images, the same results will be obtained. This metric could be used as a weight added to another distance function however, attempting to create clusters which are localized.

2.1 Intensity Difference

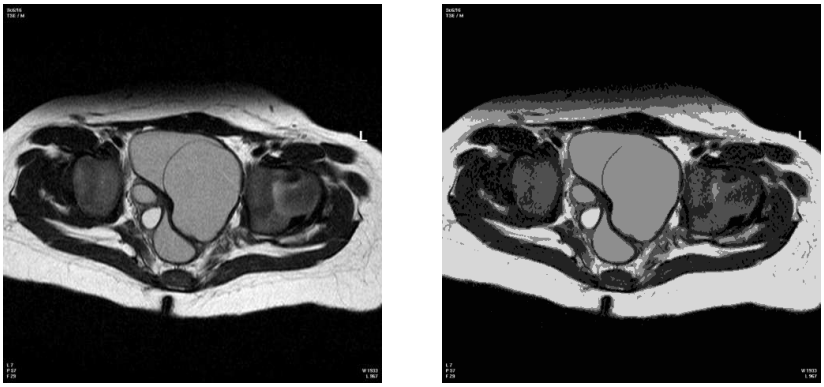
A different measure that could be used to define distance for image segmentation is the difference between a pixel's intensity and the intensity of a center. This is defined in Equation 2, where $F(x, y)$ is the intensity of the pixel with coordinates x and y, and C_i is value of center $i, 1 \leq i \leq k$. In this case the average would be defined as the average intensity of all the pixels in each cluster. Therefore each center does not correspond to an actual pixel of the image, as a center may be calculated which does not correspond to the intensity of any pixel in the image. Since the intensity is all that is needed however, this does not create any issues. An example of this method is displayed in Figure 3.

$$|F(x, y) - C_i| \tag{2}$$

The result shown in Figure 3b demonstrates good segmentation of structures that are homogeneous (such as the center structure), but performs poorly on structures with varying intensity (such as the two structures to the left and right sides of the image). In this case different parts of particular structures were assigned to three or more clusters.

2.2 Neighborhood Averaging

One way this problem could be addressed is to consider the texture of structures, instead of simply a single pixel intensity. Consider the three sample shown in Figure 4, all of which have a center pixel with similar intensity, but clearly come from different structures. Each of these three center pixels would likely end up in the same cluster, even though their neighboring pixels indicate that they should



(a) Original Image (b) Intensity Difference Clustering

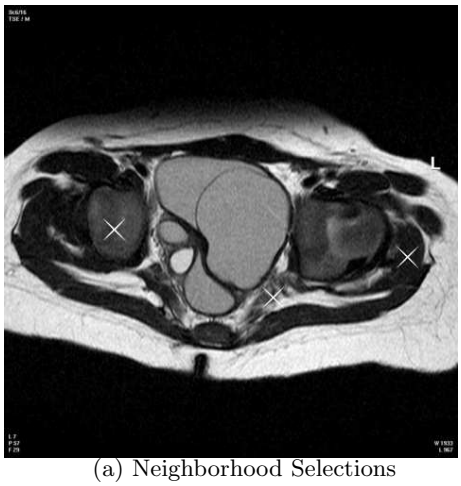
Fig. 3: These two images demonstrate a simple clustering into five different regions based on intensity difference. Each region is denoted by a particular grayscale value, which is the average grayscale value of all the pixels in that region. While some of the structures are effectively clustered, many structures are divided into multiple clusters. This is particularly evident in the two regions on the left and right side of the image. These two structures have varying intensity, so different parts of each structure are assigned to different clusters.

not. A simple way to quantify the neighborhood of a pixel is to look at the average intensity of all the included pixels. This would help in some cases, but still proves ambiguous in seemingly obvious cases, such as that shown in Figure 5.

2.3 Local Histograms

Another measure that can be used to quantify texture comes from looking at the histogram of a pixel's neighbors. This method adds two new parameters to the algorithm in addition to k. The number of surrounding pixels to be considered, or neighborhood size must be defined. The bin size for the histogram must also be declared, which determines what range of intensities should be grouped together. For example, if a bin size of 16 is used, there will be a total of 16 bins (assuming 256 different grayscale values). The first bin will count how many pixels in the neighborhood have pixel intensity of 0-15, the second bin will count the pixels with intensity 16-31, etc. If a neighborhood size of 9x9 is used, a total of 81 pixels will be considered. Therefore the sum of all the bins for any given histogram would also be 81.

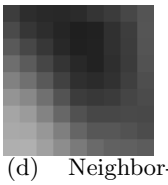
These histograms provide a measure that is invariant to rotation, translation, and scale assuming that the neighborhood is large enough to encompass a representative sample of pixels. The histograms from three neighborhoods, each with a center pixel intensity of 100 but coming from different structures, are displayed in Figure 6. From these histograms it should be possible to distinguish



(a) Neighborhood Selections

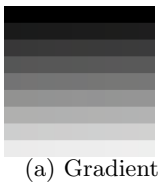
(b) Neighborhood 1

(c) Neighborhood 2

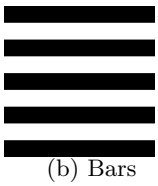


(d) Neighborhood 3

Fig. 4: Three pixels were chosen from the original image shown in Fig. 4a, with an X marking each location. These three pixels clearly come from different structures, but their center pixels all have similar intensity.



(a) Gradient



(b) Bars

Fig. 5: These two neighborhoods have obviously different texture, but have the same average intensity. Clustering based on the average intensity would therefore put these two neighborhoods in the same cluster.

between different structures even if there is texture, such as the two structures on the left and right of Figure 3a.

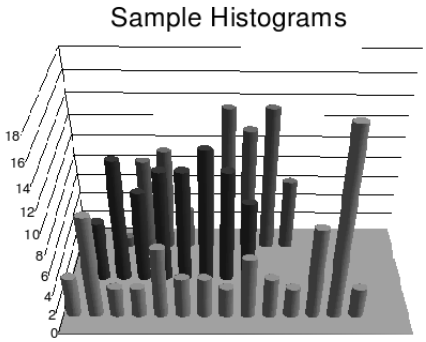


Fig. 6: The histogram of three different neighborhoods, each with a center pixel intensity of 100 but located in different structures. Even though the center pixel intensity cannot distinguish between these three neighborhoods, the histogram provides a 'fingerprint' that can be used for comparison.

Once the histograms for each pixel have been established, the distance between two histograms can be defined as the root mean square distance (RMSD) between each pair of bins. This is shown in Equation 3, where $H(i)$ is the number of pixels in the i^{th} bin. The center of a cluster can then be calculated as the binwise average of the histograms of each pixel in the cluster. This is shown in Equation 4 where $H_c(i)$ is the i^{th} bin of the histogram for center c, $H_j(i)$ is the i^{th} bin of the histogram for pixel j which is in cluster c, and N_c is the number of pixels in cluster c.

$$\sqrt{\sum_i (H_1(i) - H_2(i))^2} \tag{3}$$

$$H_c(i) = \frac{\sum_{j \in c} H_j(i)}{N_c} \tag{4}$$

3 Implementation

The algorithm described in this paper was implemented using GNU Octave [4], which is a free numerical computation package similar to and mostly compatible with MATLAB. This package was chosen due to its built in functions for reading and writing images, as well as its extensive library of vector functions

that would save much time. Additionally all memory management and other tedious overhead work is eliminated, allowing programs to be quickly created and modified. The downside however is that programs run significantly slower than other possible implementations, such as C/C++. Therefore numerical computations packages are good for prototyping and experimenting, but a production implementation would probably be best suited to a more optimizable language.

The basic K-Means algorithm operating only with each pixel's individual intensity value was able to be implemented almost purely with vector operators. This created a fast running program, which could perform 25 iterations with a k of 5 on a image with 480x480 pixels in approximately 10 seconds on a standard PC. The histogram based K-Means algorithm requires the use of more complicated looping functions due to its consideration of each pixel's neighborhood instead of just the individual pixel's value. Therefore this implementation runs significantly slower, requiring around 5 minutes for an image of the same size with the same number of iterations. This runtime is still acceptable for the purposes of this paper however.

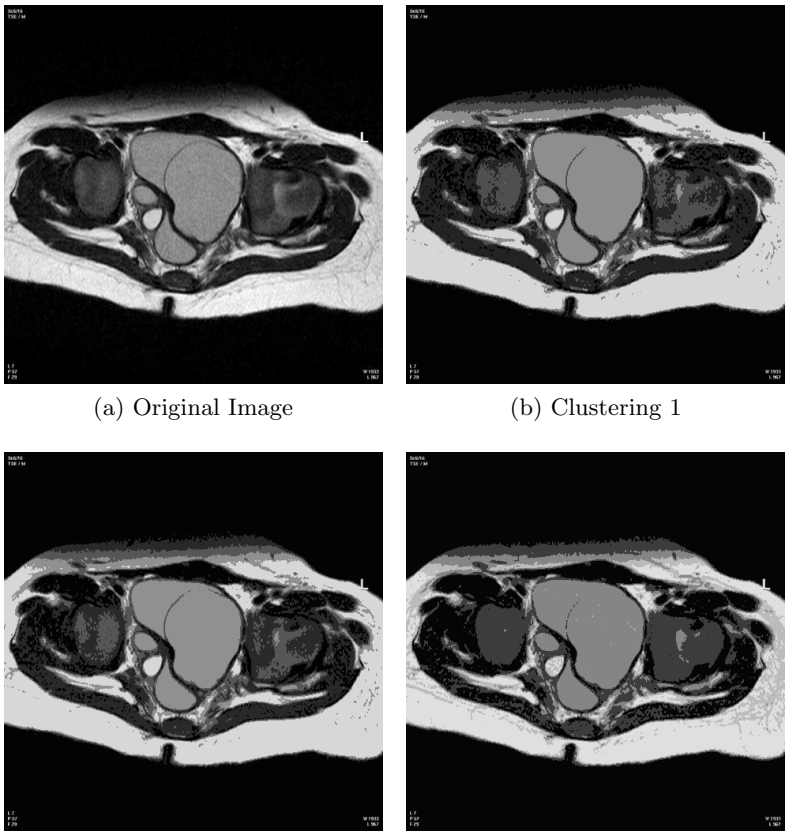
4 Results

The first set of results shown in Figure 7 shows the performance of using the intensity difference method described in Section 2.1 on images sized 480x480 pixels. The number of centers k was set to 5, and the centers were chosen randomly. The only constraint on the random centers is that they had to be unique. Interestingly, the algorithm converged to Figure 7b a large majority of the time, indicating that this method is generally insensitive to the initial centers. Two other convergences are also shown in Figures 7c and 7d. In each of these results, the segmentation performance on homogeneous structures is good, but issues are evident with textured structures. The final centers are listed in Table 1, corresponding to Figure 7. The first two clusterings have very similar final centers, while the third clustering deviated somewhat to a different local optimum. These differences in the third clustering can be seen in the left and right structures of Figure 7d.

Table 1: Final centers after convergence using intensity distance clustering.

	C1	C2	C3	C4	C5
Clustering 1	2	34	78	142	214
Clustering 2	3	40	85	144	215
Clustering 3	5	59	133	186	223

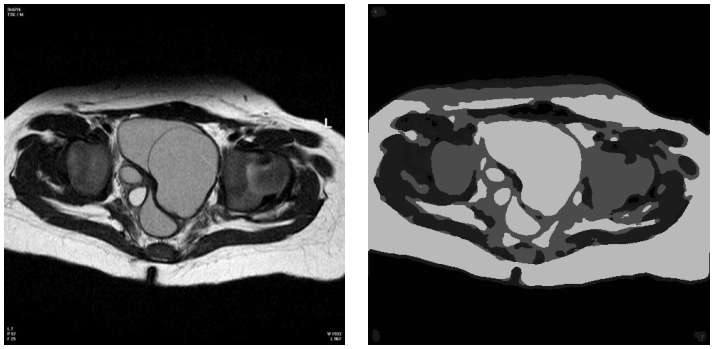
The second set of results shown in Figure 8 shows the performance of the local histogram based method described in Section 2.3 on images sized 480x480 pixels. The number of centers k was set to 5, with a binsize of 16 and neighborhood



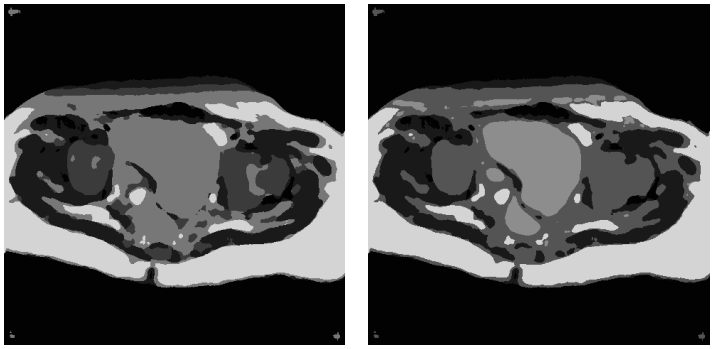
(a) Original Image (b) Clustering 1

(c) Clustering 2 (d) Clustering 3

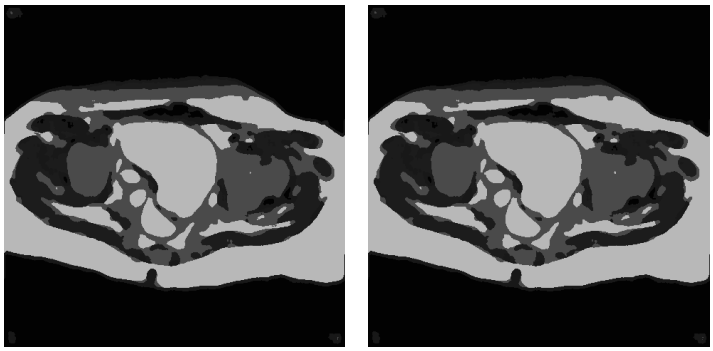
Fig. 7: Three different clusterings of Fig. 7a are shown here, using the intensity difference based method defined in Sect. 2.1 with $k = 5$. Each image is 480x480 pixels. The initial 5 centers were randomly generated, leading to the three different results. A large majority of the time the algorithm converged to the result shown in Fig. 7b however.



(a) Original Image (b) Clustering 1



(c) Clustering 2 (d) Clustering 3



(e) Clustering 4 (f) Clustering 5

Fig. 8: Five different clusterings of Fig. 7a are shown here, using localized histograms as described in Sect. 2.3. Each image is 480x480 pixels. The number of clusters k was set to 5, with a binsize of 16 and a neighborhood size of 9x9. The initial 5 centers were randomly generated, leading to the five different results. Note how much these 5 results vary, compared to the differences between each of the results in Fig. 7.

size of 9x9. The algorithm was run for 10 iterations in each case. To generate the initial centers, 5 random pixels were chosen in the image. The center was then defined to be the neighborhood histogram of each of these pixels. These histograms were also guaranteed to be unique. The most immediate observation is that the clustered regions are much more continuous, with less speckling between the clusters. Each of the results also varies much more than the results shown in Figure 7, ranging from not much better than simple intensity differences to accurately segmenting the textured regions. This indicates that this method is much more sensitive to the initial centers. This can be contributed to the fact that this method is 16 dimensional (since there are 16 bins in each histogram) instead of single dimensional as was the case using intensity difference.

5 Conclusion

This paper discusses a general image segmentation algorithm based on the standard K-Means clustering algorithm. Although applicable to a wide range of images, these algorithms can be particularly useful with medical images. The method presented has several advantages, such as being fully automatic, fast (with a production implementation), and requiring no training data. This algorithm can also be applied to many different types of medical images, not just a single set (such as heart images or brain images). These advantages largely come from the fact that no high-level information is used, such as shape models. This can prove disadvantageous as well, however. Some of the results currently stray far from an ideal segmentation of individual structures that may be achieved by other higher level algorithms.

The local histogram method described in Section 2.3 generally outperforms the intensity based method described in Section 2.1. There are still issues with this method, however, with the most obvious being the choice of initial centers. Choosing random pixels to provide the initial centers creates a wide range of results. There are heuristic methods that could be used to improve the initial selection of centers while still remaining fully automatic. Instead of merely requiring that each initial center histogram is unique, it could generate several more localized histograms than are needed. It could then select a subset of these which have the greatest overall distance, ensuring that numerous initial centers are not located within the same region. Another possibility would be to start with the intensity based clustering, and then use the final center values to select pixels to generate the initial histograms. This would help ensure that each center starts in a distinct region.

References

1. Forsyth, D.A., Ponce, J.: Computer Vision - A Modern Approach. Prentice Hall, New Jersey (2003) p. 304
2. MacQueen, J.B.: Some Methods for classification and Analysis of Multivariate Observations. Proceedings of 5-th Berkeley Symposium on Mathematical Statistics and Probability. Berkeley, University of California Press (1967) 1:281-297
3. http://en.wikipedia.org/wiki/Kmeans
4. http://www.gnu.org/software/octave