|  |
| --- |
| cSCI 362 |
| k means |
| Algorithm Report |
|  |
| **Naeem Tai** |
| **12/4/2013** |

|  |
| --- |
| This report will explain the algorithms used to implement the ‘simple k means’ and the ‘probabilistic k means’ and their results. |

* *Language used to implement algorithms: Matlab*
* **Simple k means Algorithm:**

k means is an algorithm that attempts to separate pixels of an image into k number of clusters based on their centroid values. First, we convert an image to grayscale. Then, we pick k random points from an image, and take their pixel values as centroids. Then we examine the distance of each pixel from each centroid. The distance is simply the difference between the centroid value and the pixel value. Then we determine the minimum distance between the centroid and the pixel, and assign the pixel to the corresponding cluster. After separating pixels into clusters, we recalculate the centroid of each cluster by taking an average of the pixels that belong in the cluster. We repeat this process until the cluster assignment converges, meaning that the pixels stop moving between clusters. In order to do so, we keep track of each pixel’s current cluster assignment and the previous cluster assignment. When the current cluster assignment matches the previous cluster assignment, we conclude that the pixels have stopped moving between clusters. At the final step we change each pixel’s value to the centroid value of the cluster it belongs to.

The image choice and number of clusters are flexible so that users can determine their inputs. Following are the images generated after running ‘k means’ with different number of clusters:

Grayscale Image

Original Image





Final Assignment Matrix (k = 3)

Image after Final Assignment (k = 3)



Final Assignment Matrix (k = 5)

Image after Final Assignment (k = 5)

Final Assignment Matrix (k = 8)

Image after Final Assignment (k = 8)



* **Probabilistic k means Algorithm:**

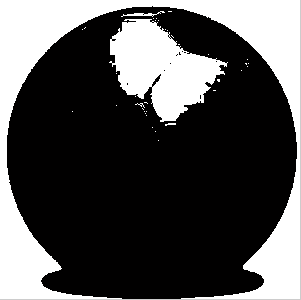
In this algorithm we repeatedly run the ‘k means’ on an image for 100 times. Each time, we generate different random centroids to start with. What we are interested in this algorithm is the probability of each pixel belonging to a cluster with specific intensity range. We construct k different matrices to keep track how many times each pixel belongs to that cluster. We also keep the centroids sorted in ascending order. The first centroid is associated with the first cluster matrix and the kth centroid is associated with the kth cluster matrix. This way we associate the centroid with lowest value with the matrix keeping track of pixels belonging to lowest color intensity, and the centroid with highest value with the matrix keeping track of pixels belonging to a cluster with highest color intensity. At the end of each k means execution, we examine the cluster assignment of each pixel and increment the times it belongs to a specific cluster in the corresponding cluster matrix. We perform this action 100 times to get an accurate measure of how likely is each cluster to belong in a cluster with lower or higher color intensity.

Again, we keep the image choice and number of clusters are flexible so that users can determine their inputs. The program generates k number of probability images, one for each cluster. Following are the images generated after running ‘probabilistic k means’ with different number of clusters:

****

Original Image

Grayscale Image



Probability Image of Cluster2 (k =3)

Probability Image of Cluster1 (k =3)



Probability Image of Cluster3 (k =3)

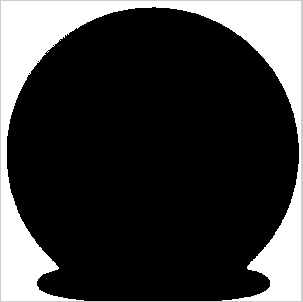
Probability Image of Cluster3 (k =5)

Probability Image of Cluster2 (k =5)

Probability Image of Cluster1 (k =5)

Probability Image of Cluster5 (k =5)

Probability Image of Cluster4 (k =5)



* **Running Time:**

For both algorithms described above, the running time depends on the size of the image and number of clusters. This is because all the looping in the program is either dependent on the number of pixels in the image or the number of cluster to form. For a 300x300 image of a soccer ball displayed above:

on average the ‘k means’ algorithm only takes about 1 to 2 seconds to form clusters and generate images.

‘probabilistic k means’ takes about 15 seconds to finalize 3 clusters and 35 seconds to finalize 5 clusters over 100 runs of k means.