COSC450: k-means Segmentation

Ashley Manson

October 6, 2015

Department of Computer Science
University of Otago

1 Program and Code

The code is compiled with the included Makefile, if the setPath.sh has run sucessfully on the lab machines.

The program takes 7 command line arguments in this order:

1. The input image to perform the segmentation on.
2. An output image to save the result.
3. k for the number of clusters.
4. Cluster type, which is a number between 0 and 2.
    a. 0 is for random selection.
    b. 1 is for random clustering.
    c. 2 is for k-means++.
5. Distance type is a number between 0 and 1.
    a. 0 is for comparing RGB values.
    b. 1 is for comparing HSV values.
6. Termination type is a number between 0 and 2.
    a. 0 is for giving the maximum iterations before terminating.
    b. 1 is for terminating once the clusters have stopped making changes. A value at the top of the file CLUSTER_MIN_CHANGE can augment this.
    c. 2 is for terminating once the labels have stopped making changes. A value at the top of the file LABEL_MIN_CHANGE can augment this.
7. Max iterations is for giving a maximum number of iterations before the program will terminate. Will be used if 0 is given for the end type.

Once the first image has been displayed, a key press will begin the k-means segmentation. Output will be printed to stdout of the progress so far. Once the k-means algorithm completes, the

segmented image will be displayed. Sometimes it displays behind the original, so moving the original image will reveal the segmented image. Another key press once the segmented image is displayed will close both windows. The segmented image will be saved to the given command line argument for viewing later.

2 Implementation

2.1 Initialisation Methods

I have implemented the three initialisation methods that were proposed in the Assignment 2 specifications; random selection, random clustering, and k-means++.

2.1.1 Random Selection

This method of initialisation is the fastest. For each k cluster point, it will get a random x and y coordinate that corresponds to a pixel in the image. This is then added to a vector of Vec3b for computation later.

2.1.2 Random Clustering

This initialisation method is slower than selection as it goes through all the pixels in the image. For each pixel in the image, it is assigned a random k cluster. Once all pixels from the image have been assigned a k cluster, it goes through each cluster, calculating the average colour across all the pixels. Once the average has been calculated, the colour average is added to the clusters vector for computation later.

2.1.3 k-Means++

This method of initialisation takes the longest due to the random distribution that contributes to the selection of clusters. The first cluster is selected at random from the image. This serves as the first cluster point. The first loop then goes over the rest of the clusters that need to be selected.

The first inner loop then goes over the image calculating the distances from the selected cluster, storing the result in its corresponding index in a vector and adding to a count. The second inner loop then goes on until another cluster is added to the cluster vector. Here there is a chance for one of the distances in the vector to be selected based on a random distribution of all distances calculated. Once a new k cluster has been found, it is added to the clusters vector, and this loop exits. All of this is repeated until all k clusters are found.

2.2 Distance Metrics

I have implemented two different ways of calculating the distance of two pixels, one being the raw RGB values, and the second being the HSV of a pixel, ignoring the value. For each pixel in the image, I go through and calculate the distance from it and each of the k cluster points. If the distance is less than the previous distance, that pixel label gets updated and it gets a new minimum distance set for it. For RGB, I simply calculate the difference between the corresponding RGB values. For HSV I scale the hue by dividing by 180 and than multiply by 256. The reason for this is due to OpenCV storing its hue value between 0 and 179. The scale ensures that the distance calculations are correct. When calculating the distance I ignore the value from HSV.

2.3 Termination Conditions

I have implemented three different ways for the k-means program to terminate. It can be passed in a parameter to set the maximum number of iterations that the program will run for. It can be set to check the differences in cluster changes. It can be set to check how many labels are changing between iterations.

2.3.1 Maximum Iterations

The program can be set to run for a set number of iterations, no matter if it has stabilised or still has yet to stabilise.

2.3.2 Cluster Changes

The program can be set to check for changes in cluster points. With every iteration, the difference between cluster changes is calculated, and if it is less than a prespecified minimum change, then the algorithm will terminate.

2.3.3 Label Changes

The program can be set to check for changes in labels for each pixel. With every iteration, the number of changed labels is summed up, and if it is less than a prespecified minimum change, then the program will terminate.

3 Experiments

I have experimented with Random Selection, Random Clustering and k-Means++, as well as testing each initialisation method with both RGB and HSV. I have ran each with k clusters of 4, 6, and 8. The termination case was the change of clusters. I have taken the average of each to find out which is fastest to segment the image. In the following charts, RS means random selection, RC means random clustering, and KM means k-means++. The error bars on the charts are of the standar deviation or error for each average.

3.1 Average Iterations

3.1.1 mms.jpg

Figure 2 shows that HSV in combination with Random Selection or k-Means++ performs better than the other distance metrics and initialisation method combination. Probably due to there being a wider range of colour in this image for the initialisation methods to select. I believe that the reasont he clustering falls short is there is too much colour for it to average over, so it takes longer for it to stabilise the clusters.

Figure 1: Image used for testing mms.jpg.

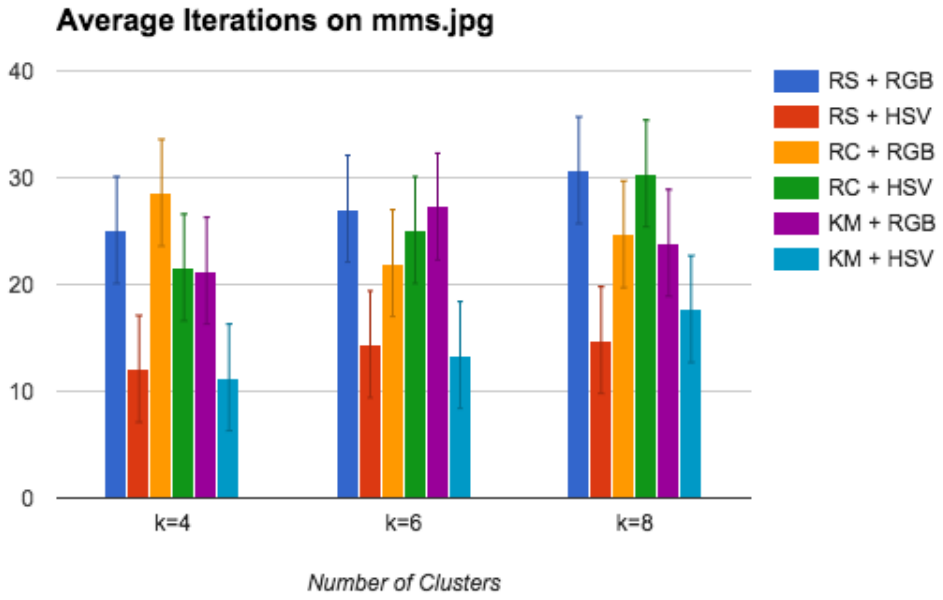

Figure 2: Average iterations on mms.jpg.

3.1.2 lena.png

Figure 4 shows that when k is set to 4, all the combinations seem to perform around the same level. Moving onto a higher k value however, random clustering and HSV start to take longer than the rest to stabilise. k-Means++ with HSV outperform the rest. I believe that this is due to k-means++ initialising more unique clusters from the image, which allows for the clusters to

Figure 3: Image used for testing lena.png.

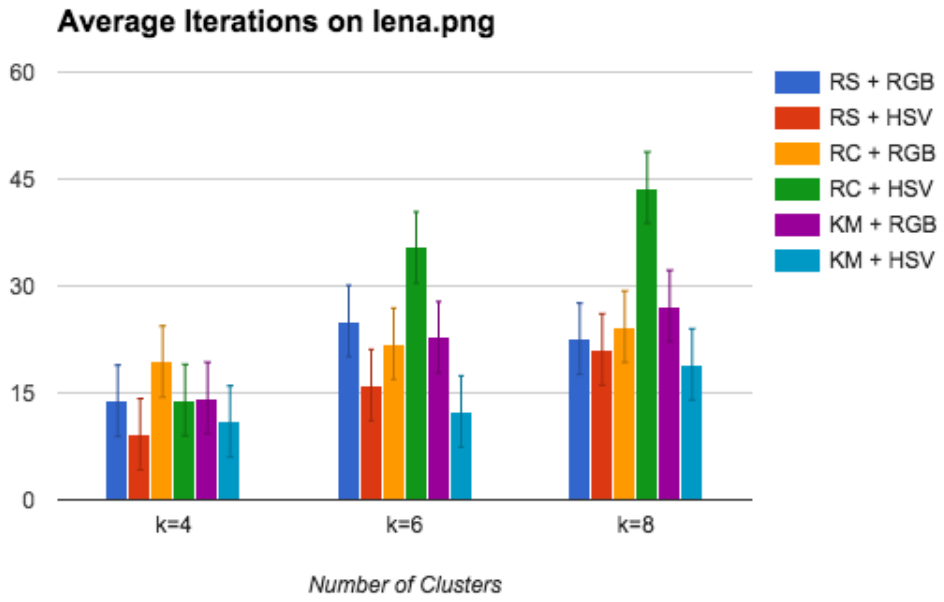

Figure 4: Average iterations on lena.png.

stabilise faster.

Figure 5: Image used for testing penguin.jpeg.

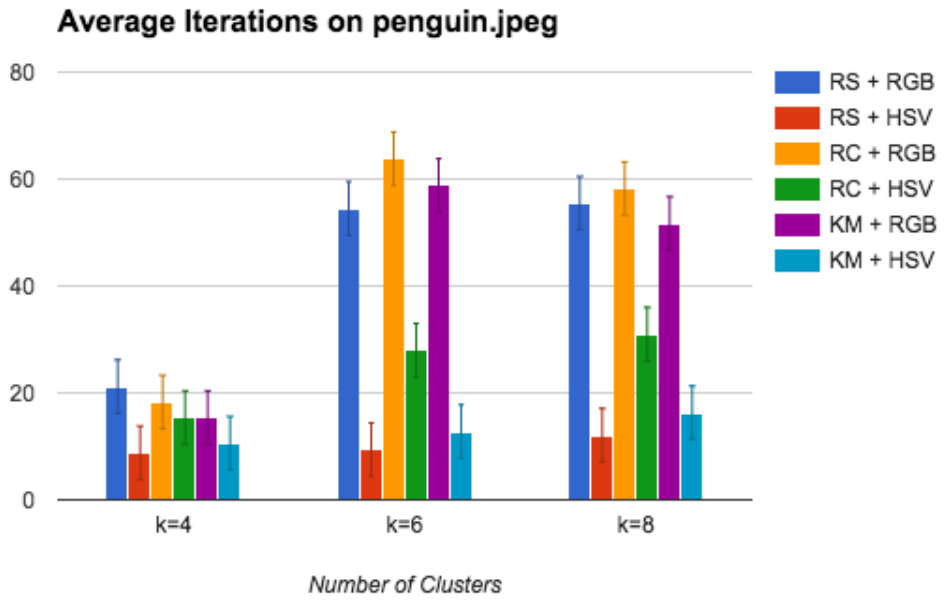

Figure 6: Average iterations on penguin.jpeg.

3.1.3 penguin.jpeg

With Figure 6 when k is set to 4, all combinations seem to perform around the same. When k is increased, all RGB combinations perform horribly. I believe this is caused due to all the clusters

trying to stabilise but all the white in the image is causing this to fail. Due to HSV ignoring colour and only focusing on hue and saturation, it outperforms the RGB version in regards to timing.

3.2 RGB vs HSV

I have taken a look at the resulting images from RGB and HSV distance metrics. I have used k-Means++ with the change in clusters as the termination case. I am looking for which distance metrics segments the image in the best possible way.

3.2.1 Bear k = 8

Figure 7: Bear Original, HSV and RGB. k = 8

For Figure 7 I choose to have k as 8 for this, as HSV failed to properly segment the image with any k lower than that. However in this example, RGB still segments the image better than the HSV version. RGB finds a lot of the colour from the bear, and does not blend it with the green background, unlike the HSV version. The HSV version seems to be picking up on incorrect value on the bear, and clusters some of it with the green background. This could be caused by the bears brown fur having a similar hue/saturation as the gree background on the top left of the image. The RGB version is better segmented as it is more easily distinguished from the rest of the image.

Figure 8: Kitten Original, HSV and RGB. k = 5

3.2.2 Kitten k = 5

Figure 8 HSV segments the image better than the RGB version. The HSV version has segmented the kitten corretly by getting the white and orange fur from it. The white fur was segmented with the stone that the kitten is standing on, and the eyes were segmented with the green background. The RGB version has been segmented into a ghost, as the left side of the image green was merged with the kitten. The orange fur also failed to get segmented out from the test of the image.

3.2.3 Sunset Tree k = 5

Figure 9: Sunset Tree Original, HSV and RGB. k = 5

Figure 9 the HSV version of the segmentation performs way better than the RGB version. HSV properly segmented the tre from the sun and the entire is in one cluster. The only issue with the HSV tree is that it made a random branch. However the tree that is in front of the sun was not merged with the sky or sun. On the other hand the RGB tree was segmented into three parts,

with the darker part of the tree merging with the ground cluster, one part its own colour and the third part being in the same cluster as the sky.

4 Strengths and Weaknesses

k-Means++ is the best initialisation method, followed by random selection, and finally random clustering. In terms of time it takes to stabilise the clusters, k-means++ takes the least iterations during the execution of the program. Random Clustering takes the most time to stabilise, which I believe is caused by the fact the clusters are averages across the entire image, so when the stabilising begins, it has to move around its clusters a lot more than the others. As the other two initialisation methods have selected more diverse clusters.

In my implementation of RGB and HSV distance metric, RGB tends to perform better than HSV, especially when there is a lot of colour in an image. The resulting segmented images that RGB returns have a better image quality than HSV. However, on images that are made up of predominantly one colour, HSV correctly segments the image very well, compared to RGB. HSV also tends to segment images faster than RGB, I believe is because it does not take colour into consideration.

The different termination criterion do not have a huge impact on the resulting images. The cluster termination tends to finish before the label termination. I think this is how I have implemented the averaging at the end of the k-means algorithm, as the clusters are stabailsed before the labels are updated.