

A. Introduction

This program is designed for image segmentation in the color space with iterating K-mean algorithm and built on the environment of Mac OS X Mountain Lion with the tool of Xcode. Since Mac OS X is an Unix-like OS, the program is compatible with the OS like Unix and Linux if using the same compilers. The next section will provide you with the description of how to compile and execute this program. Moreover, in the last section it will give you the idea how the program deals with the process of image segmentation.

B. How to compile and run the program?

This program is compiled by [GNU compiler collection \(GCC\)](#) of the latest version 4.7.2, which is suitable for operating systems, such as Unix, Linux, and Mac OS X. Once you install the compiler, you will be able to compile and run this program on your computer with those operating systems listed above. Here is the instruction how you can compile the program. (Assume you installed the compiler on your computer already)

Step 1: unzip the file.

Step 2: change your current directory into the directory containing the files from Step 1.

Step 3: to compile the source code, type the command as below.

g++ main.cpp -o [the name of executable file]

Step 4: to execute the program, type the name of executable file with prefix ". / " with several parameters as below.

./[the name of executable file] [parameters.....]

Usage:

./executable_file filepath r c [k-start] [k-end]

filepath	a file path including image file name
r	the height of the input image
c	the width of the input image
[k-start]	a value showing how many clusters you want, always smaller than k-end
[k-end]	a value showing how many clusters you want, always greater than k-end

After step 4, the program will start to execute until all the process is completed, and for test image it will produce a text file and several raw files depending on how you specify the K value. For example, if k-start equal to 3 without k-end, the program will output one text file and one raw file for K equal to 3. If you specify K in a range from k-start to k-end like 2 to 5 for example, you will get one text file and four raw files for K equal to 2, 3, 4, and 5 separately.

C. How is the structure of this program?

In the folder of the project, there are one folder named "Results" and four files in total shown as follows. The folder of "Results" includes all results pre-computed from this program for three test images. In addition, the header file is named Config.h, which is included by the other three files because it defines several libraries and definitions of structs that the program needs and it can be commonly used in the other three files. The structure of the folder is shown as below.

Project folder

```
|-- Results
|   |-- Rock-Stream
|   |-- Tiger1
|   |-- Data13
|-- Config.h
|-- Main.cpp
|-- FileController.cpp
|-- Cluster.cpp
```

The program is composed of three parts: main function and two classes. What the task of main function here is to be an entry of the program, which will show you a high-level flow of how the program executes. In addition, two classes would dominate how this program works specifically. One is the class [FileController](#), which is in charge of the process of loading input data and outputting a text file and several raw files for each test image. The other is the class [Cluster](#), which is responsible for performing iterating K-mean algorithm and computing a least square error against a specific K value.

D. Design of the Experiment for Image Segmentation

To do image segmentation, we implemented *iterating K-mean Algorithm* to perform segmentation in RGB color space with different K values ranging from 2 to 10 on three test images. Also, we computed *a least square error measure for each group* when the iterations of the algorithm stop in order to figure out which K value would be the best one. Finally, we will discuss the results from our experiments and conclude final decision about K values we chose for each test image.

Iterating K-mean Algorithm

In this experiment, the algorithm will begin with initial k values as the mean values of k groups. Here is the formula showing how we decide those initial k values.

$$x = \frac{256}{[K + 1]} * j; \quad j = 1 \text{ to } K$$

where x equals the r, g, or b variable, and K equals the number of clusters. Note that each time when it produce a value of x, it will represent a mean value like (x, x, x) = (r, g, b) in RGB color space. Moreover, this algorithm would encounter [the problem of empty clusters](#) when one iteration ends, causing a fail of dividing by zero in the step of updating mean values for groups. To solve the problem, we assume that let mean values remain the same as those of previous iterations if empty clusters appear.

Least Square Error (LSE) measure

Since the experiment was performed in RGB color space, each mean value would be represented by (r, g, b). To compute LSE for each k value, the formula would be as follows:

$$D = \sum_{k=1}^K \sum_{j \in RGB} \sum_{x_i \in C_k} \|x_{ji} - m_{jk}\|^2$$

However, each group may have different number of pixels, causing it possible that the more pixels a group has, the bigger LSE it has or the bigger K value it is, the larger LSE it has. To solve the problem, we also compute a normalized LSE when using a specific K value in the experiment. What we did is using the number of pixels in a group and K value as two normalized factors. The following formula shows how we compute the normalized LSE.

$$D = \frac{\sum_{k=1}^K \frac{\sum_{j \in RGB} \sum_{x_i \in C_k} \|x_{ji} - m_{jk}\|^2}{|C_k|}}{K}$$

The program will produce an output text file when it ends and this file would come with several lines, each of which has three values: K value, LSE, and normalized LSE. Therefore, we could use those data to draw a chart to see how LSE or normalized LSE grow with different K value.

E. Results of Image Segmentation

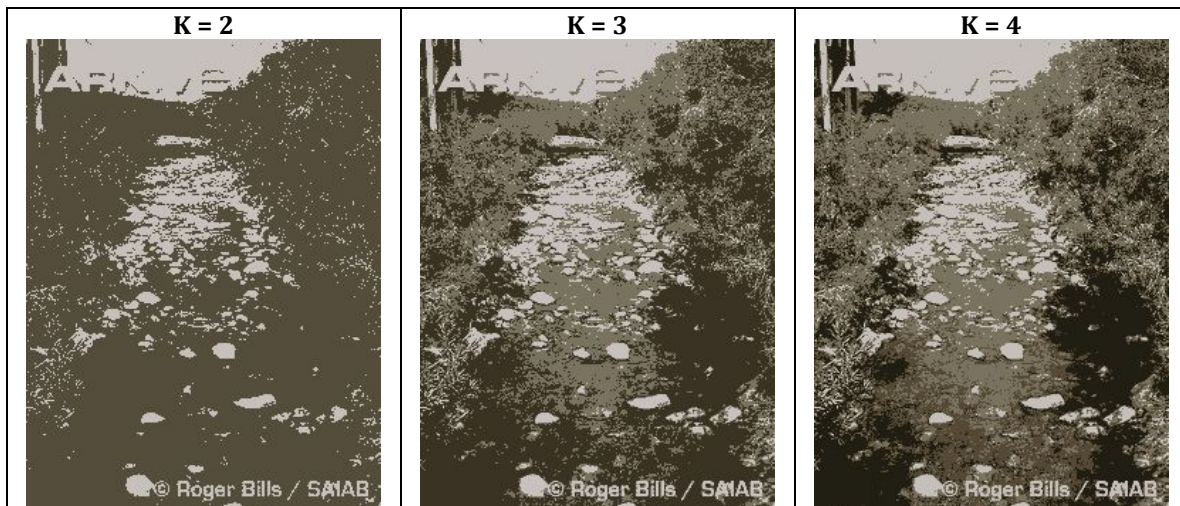
There are three test images used in this project. All of them were clustered on RGB color space with iterating K-mean clustering. While clustering, we specified different K value ranging from 2 to 10 to see which K value is suitable for use to get better results. Also, we computed a value of least square error measure for each group, allowing us to figure out how close data are to their assigned cluster means. The following would show you the results for those three test images separately.

1. Test Image: Rock-Stream.raw



The first image is of size $r \times c = 333 \times 250$. In the first look at the original image, we guessed the number 8 or 9 would be a better K value for this test image. After experiment, **the results also seem to tell us the number 8 and 9 are better K value** when we look at the result images in Table 1. However, from the analysis of LSE in Fig. 1., **K equal to 9 would be the best choice**, while in Fig. 2. it shows **number 10 would be the best choice**. Those analyses give us a conclusion that a specific K value with the lowest LSE or normalized LSE is not always the best choice if we narrow K value between 2 to 10.

Table 1. Result for each K value



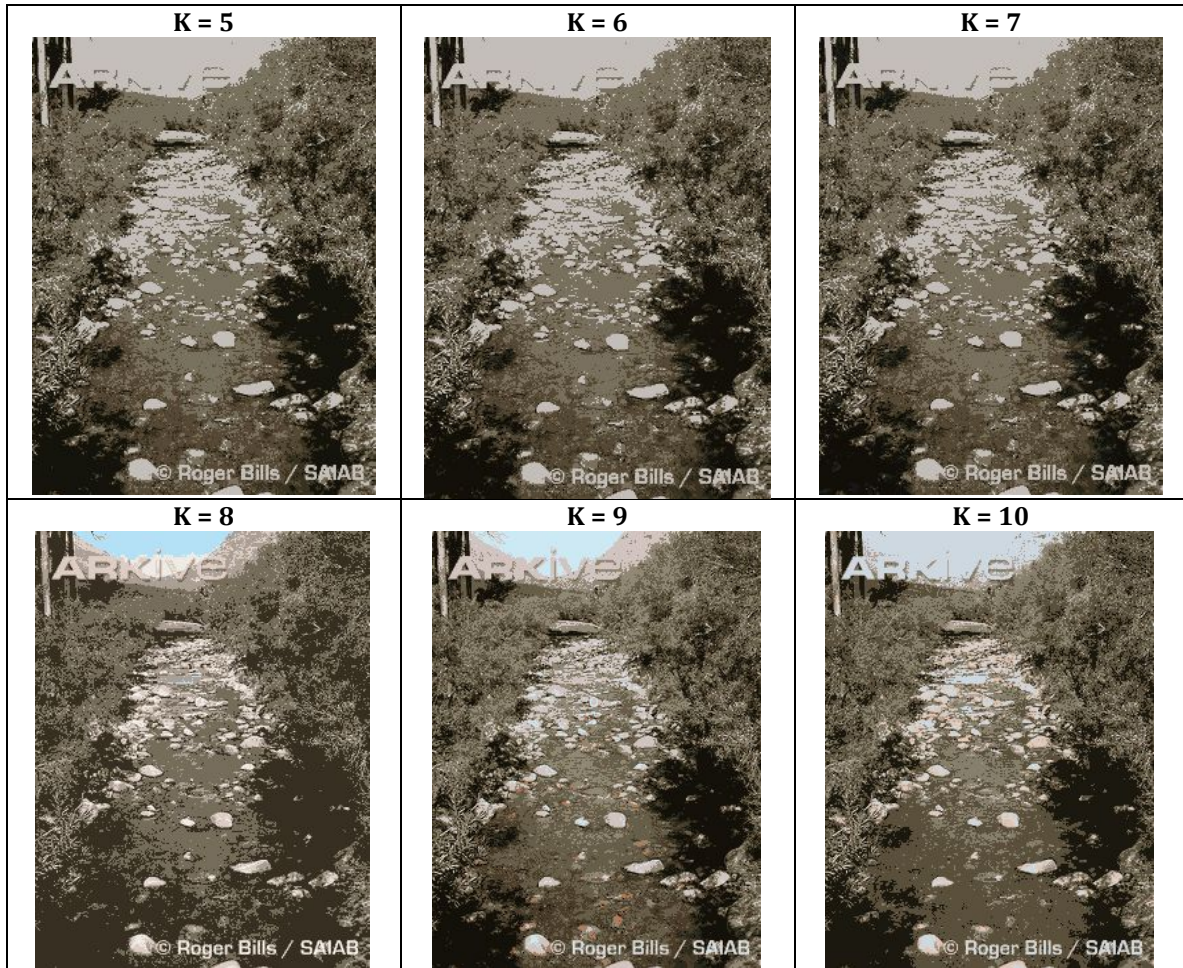


Table 2. Results of D values against K values for the first image

K	D-value	Normalized D-value
2	388652902	4295
3	189948458	2490
4	133616496	1765
5	123830078	1419
6	123113271	1212
7	122777657	1058
8	144826335	905
9	86579415	849
10	105016883	703

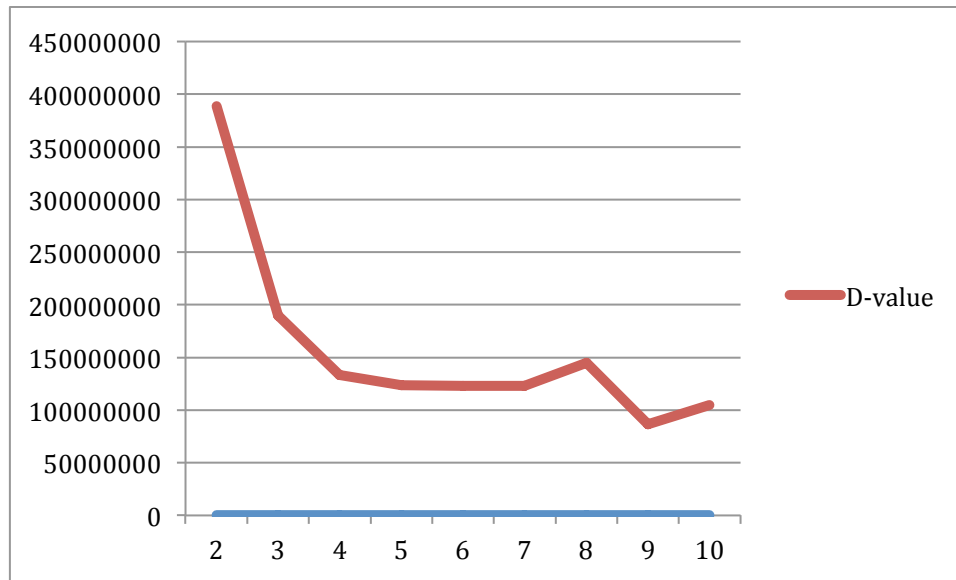


Fig. 1. LSE against K values for the first test image

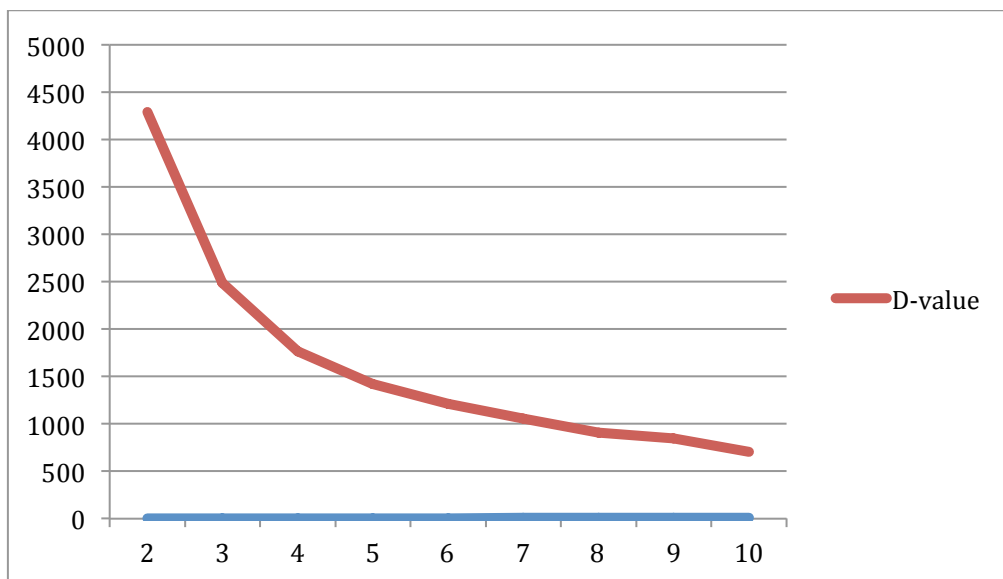


Fig. 2. Normalized LSE against K values for the first test image

2. Test Image: Tiger1.raw



The second image is of size $r \times c = 461 \times 690$. Again, in the first look at the original image, maybe the number 7 would be a better K value for this test image. After experiment, when we look at the result images in Table 3, **the results seem to show us the number 7 and 10 are better K value.** Nevertheless, in the analysis of LSE in Fig. 3. and Normalized LSE in Fig. 4., **K equal to 10 would be the best choice** because it comes with the lowest LSE and normalized

LSE at the same time. Compared with the result of previous test image, the measure of LSE seems able to help us find out the best K value.

Table 3. Result for each K value


K = 2 	K = 3 	K = 4 
K = 5 	K = 6 	K = 7 
K = 8 	K = 9 	K = 10 

Table 4. Results of D values against K values for the second image

K	D-value	Normalized D-value
2	1662978802	5249
3	915877941	3196
4	713635894	2396
5	660426622	1970
6	654330369	1672
7	451835585	1336
8	471011019	1127
9	602971933	1205
10	452716167	903

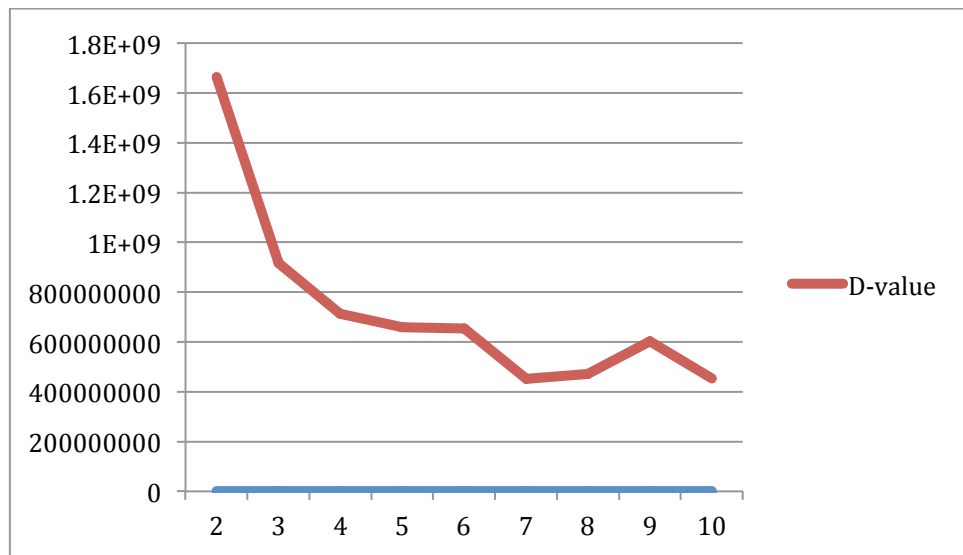


Fig. 3. LSE against K values for the second test image

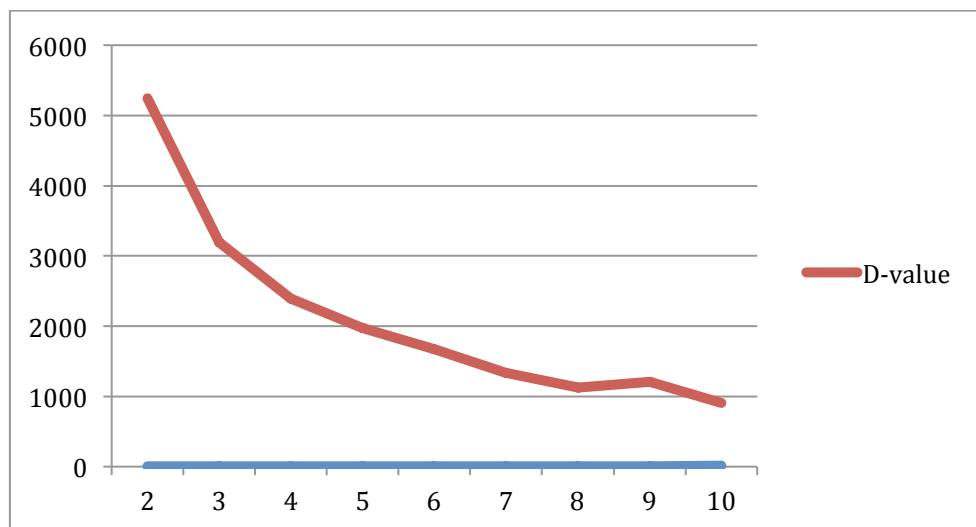


Fig. 4. Normalized LSE against K values for the second test image

3. Test Image: Data13.raw



The third image is of size $r \times c = 240 \times 320$. Similarly, in the first look at the original image, maybe the number 9 or 10 would be a better K value for this test image. From result images in Table 5, **they present the number 10 is the best K value**. Nevertheless, in the analysis of LSE in Fig. 5. and normalized LSE in Fig. 6., hopefully **we have 10 to be the best K value**. Again, in this case we conclude that LSE or normalized LSE may be able to be a good measure to help us find out the

best K value between 2 to 10.

Table 5. Result for each K value

<p>K = 2</p>	<p>K = 3</p>	<p>K = 4</p>
<p>K = 5</p>	<p>K = 6</p>	<p>K = 7</p>
<p>K = 8</p>	<p>K = 9</p>	<p>K = 10</p>

Table 6. Results of D values against K values for the third image

K	D-value	Normalized D-value
2	185355854	6665
3	120386323	4720
4	103040569	2675
5	108156141	2133
6	117574051	1828
7	95176583	1135
8	87594635	1196
9	86644192	1071
10	26392480	799

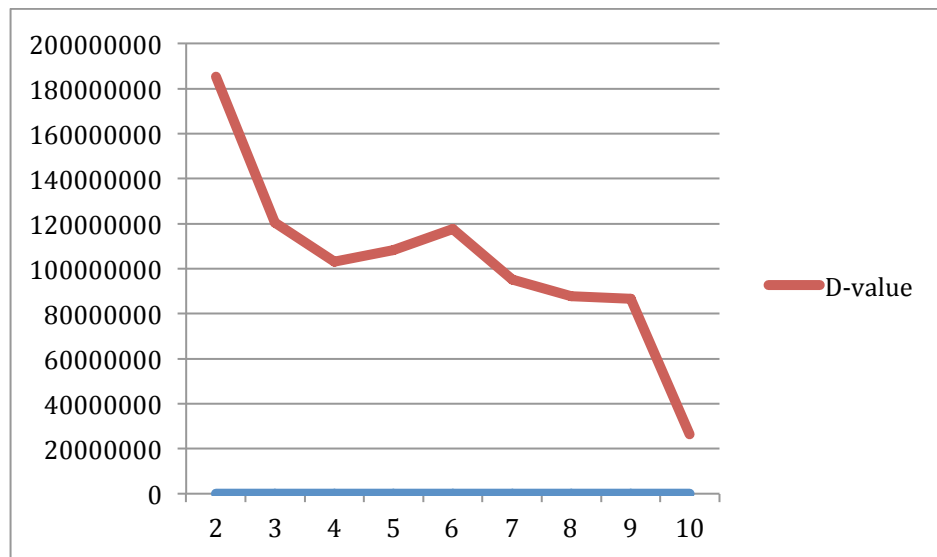


Fig. 5. LSE against K values for the third test image

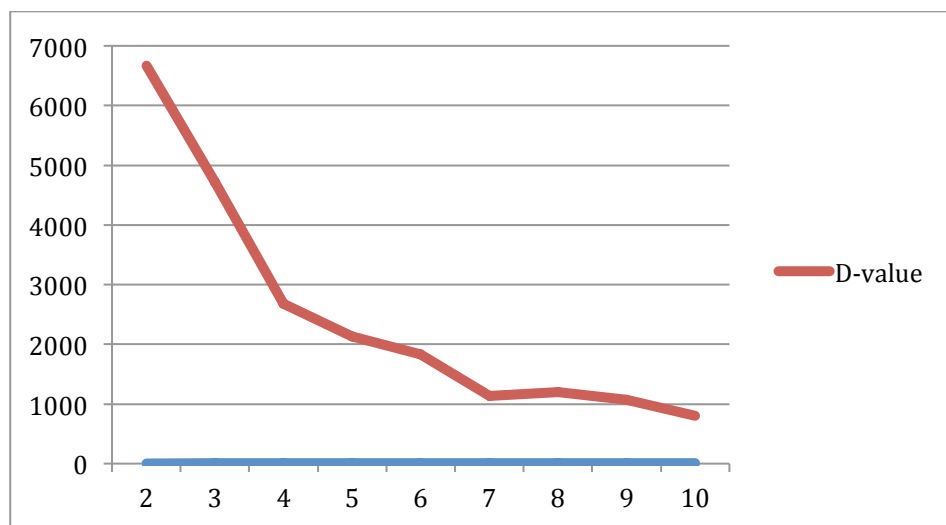


Fig. 6. Normalized LSE against K values for the second test image

F. Conclusion

From the experiment, we tested three images with the K values among 2 to 10. However, we still need to look at the results manually for purpose of choosing the best K value for each images, since LSE or normalized LSE sometimes cannot exactly tell us which K value is better. If we try to think about LSE further, there are some reasons why it did not work well. First, empty clusters would affect the computation of LSE, especially when calculating normalized LSE. Second, another reason would be the number of colors in a test image much greater than K value we used in this experiment, since we only used a K value between 2 to 10. Finally, initial K values would be one of factors affecting the results of LSE.