Report for Computer GraphicII, HW2
The Application of K-means Clustering

Jiang Xinhong 2020533075

October 25, 2022

Acknowledgements:

Deadline: 2022-10-25 23:59:59

You should answer the questions in **English**

You can choose C++ or Python, and no restrictions on programming framework. You can freely use frameworks such as openGL.

The **report** submits as a PDF file to gradscope, the programming part should package all the files include code, input files, executable file, readme.txt, and report. The **package** name is **your_student_name+student_id.zip**.

You will get Zero if the code not passing the plagiarism check.

1 The Application of K-means Clustering

Explore the application of K-means clustering in image superpixel segmentation and mesh simplification. **Give details of the algorithms, experimental results, and analysis**.

1.1 image superpixel segmentation

1.1.1 Description of naive K-means Clustering algorithm

Given a set of observations (x1, x2, ..., xn), where each observation is a d-dimensional real vector, k-means clustering aims to partition the n observations into k ($\leq$n) sets $\mathbf{S} = \{S_1, S_2, ..., S_k\}$ so as to minimize the within-cluster sum of squares (WCSS) (i.e. variance). Formally, the objective is to find:

$$\underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \underset{\mathbf{S}}{\arg\min} \sum_{i=1}^{k} |S_i| \operatorname{Var} S_i$$

where μ_i is the mean of points in S_i. This is equivalent to minimizing the pairwise squared deviations of points in the same cluster:

$$|S_i| \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} \left\| \mathbf{x} - \mathbf{y} \right\|^2$$

The equivalence can be deduced from identity $|S_i| \sum_{\mathbf{x} \in S_i} \left\| \mathbf{x} - \boldsymbol{\mu}_i \right\|^2 = \sum_{\mathbf{x} \neq \mathbf{y} \in S_i} \left\| \mathbf{x} - \mathbf{y} \right\|^2$. Since the total variance is constant, this is equivalent to maximizing the sum of squared deviations between points in different clusters (between-cluster sum of squares, BCSS), This deterministic relationship is also related to the law of total variance in probability theory.

1.1.2 Algorithm steps of K-means

The algorithm steps of K-means are:

1. Select the initial k samples as the initial cluster center;
2. For each sample in the dataset, calculate its distance to k cluster centers and classify it into the class corresponding to the cluster center with the smallest distance;
3. For each class, recalculate its cluster center (ie, the centroid of all samples belonging to that class);
4. Repeat steps 2 and 3 above until a certain termination condition (number of iterations, minimum error change, etc.) is reached.

```
while cluster changed do
    for all data dots do
        for all centroids do
            record the index of closet centroids
        end
        for all clusters do
            find Centroid
        end
    end
    determine whether the Centroid is changed
end
```

For which the pseudo-code is:

In 2 dimension, for m dots, n centroid, the complexity of one iterate is $O(mn)$. However, the times of iterate is unpredictable, it is hard to known the complexity of a complete algorithm.

1.1.3 Implementation

For a test set as follows, we can see that it is roughly divided into three clusters by naked eye observation

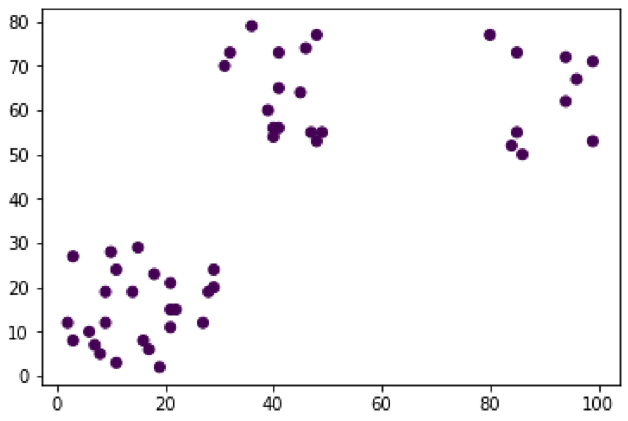

Figure 1: Test set

We manually set the K value to 3, and randomly select three initial centroids, and get the following results

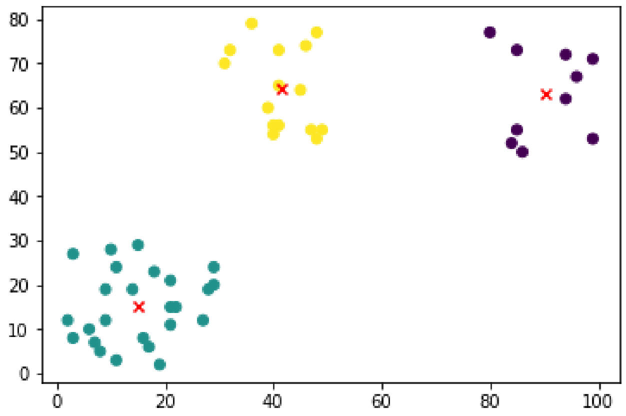

Figure 2: $k = 3$

1.1.4 Discussion

A key limitation of k-means is its cluster model. The concept is based on spherical clusters that are separable so that the mean converges towards the cluster center. The clusters are expected to be of similar size, so that the assignment to the nearest cluster center is the correct assignment.

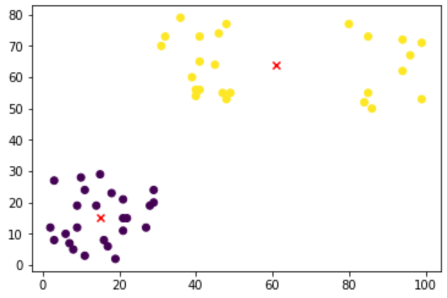

Figure 3: $k = 2$

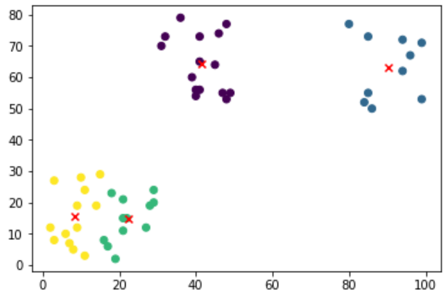

Figure 4: $k = 4$

When for example applying k-means with a value of $k = 2$ onto the data set, the result often fails to separate the three groups contained in the data set. With $k = 2$, the two visible clusters (one containing two species) will be discovered, whereas with k=4 one of the clusters will be split into two even parts. In fact, k=3 is more appropriate for this data set, for the data set's containing 3 classes. As with any other clustering algorithm, the k-means result makes assumptions that the data satisfy certain criteria. It works well on some data sets, and fails on others.

1.1.5 K-means apply to image

Now, we replace Euclidean distance with a new distance function, both spatial distance and color distance are considered:

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b)i)^2}$$

$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

$$D = \sqrt{(d_c)^2 + (d_s/S)^2 m^2}$$

There are 5 dimensions we considered: R,G,B and 2-d euclidean distance, which can cluster similar pixiels. With this distance function we can apply the K-means algorithm to image clustering which is a simple linear iterativeclustering algorithm. To reduce complexity, SLIC only computes the distance from each cluster center to a pixel within a 2S×2S region (desired superpixel size is only S×S)

1.1.6 Implement and Discussion

We choose cartoon pictures for the primary test, because cartoon pictures are mostly composed of color blocks, which is in line with the working principle of superpixiel
For example, We applied it to a Winnie the Pooh cartoon:

The effect of different iterations
We set it to 16 segemnts, because this value can better segment the graph (intuitively), and then perform three iterations of 2, 5, and 15 results.

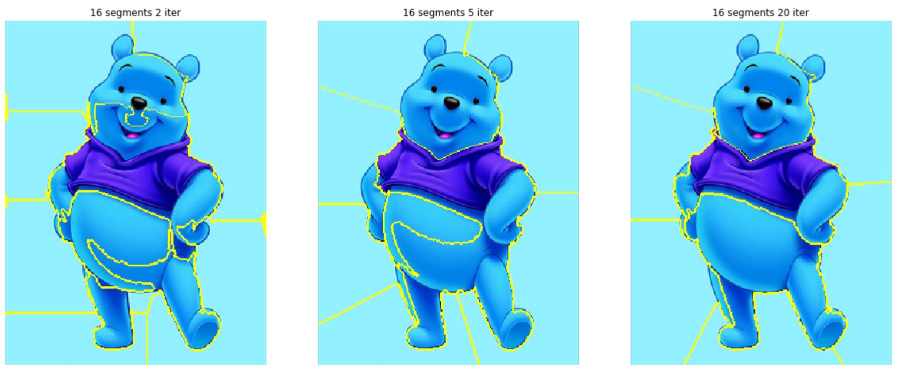

Figure 5: different iterations

It can be seen from the results of many experiments that when the number of iterations is very small, due to the randomness of the initial point selection, we often cannot obtain an ideal result. However, the effect will converge quickly, and the results are already very good after 5 iterations, and the quality difference is not much compared with 15 iterations.

The effect of different segemnts

For each image below, we take 15 iterations (which have been shown to give stable results in previous experiments) and take 4, 16, and 32 segemnts for each image, respectively

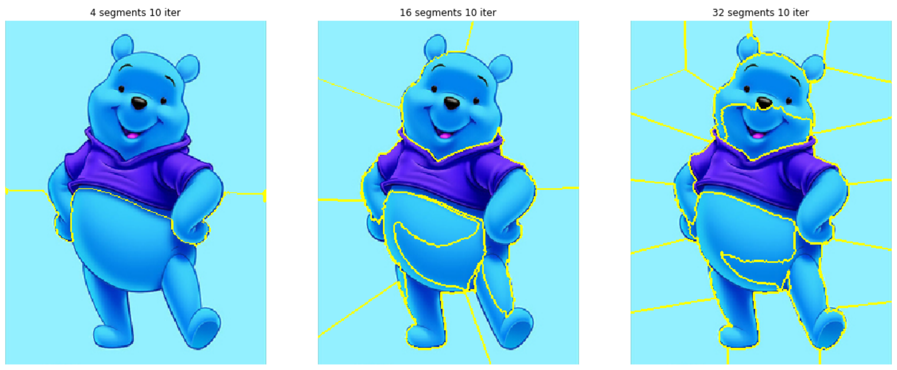

Figure 6: different segemnts

As can be seen in the results, as more and more segemnents are added, the finer degree of image segmentation is also improving.
But it's disappointing that for this Winnie the Pooh picture, we would like to preserve the facial expressions (eyes, nose, mouth, etc.) when simplifying, but these details are difficult to preserve even if we use very large segemnts, Instead, backgrounds of similar colors are divided into multiple blocks, which is what we don not need.
In fact, due to the principle of the algorithm, it is almost impossible for us to separate these details from the picture in subsequent iterations unless they are within the expression details when the centroids were selected for the initial sampling. Since the details of the expressions occupy a small area of the screen, it is almost impossible for us to present the expressions of Winnie the Pooh in our simplified results through the superpixel algorithm.

Apply to more complex images

As can be seen from the above results, if we want to apply superpixeil to complex images (such as real photos), we obviously need more segments. So we will use 1000 segenment for this photo of train platform.

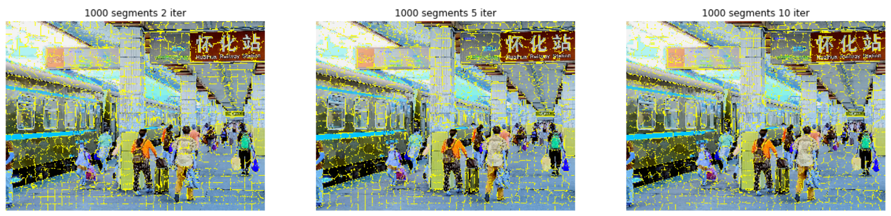

Figure 7: Train station(Photographed by the author)

In this case, our initial sampling points are already dense enough, and the processing object is also very complex, and a small number of iterations can get good results. As shown in the following results, the difference between several cases is not large.
A fun easter egg is, the algorithm can well identify the Chinese "**Huaihua Zhan**" of the platform sign, which shows that k-means clustering is very good for the identification of the same color block, as long as the initial point is selected properly.

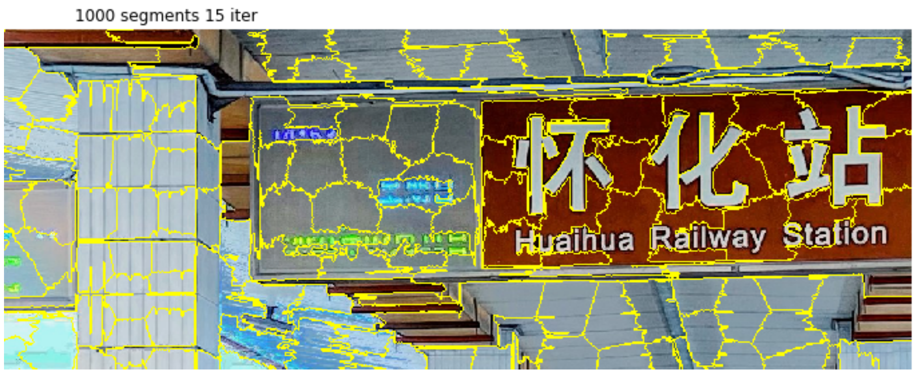

Figure 8: A good recognition of Chinese characters

1.1.7 Summary of Superpixels

For simple linear iterativeclusteringthe advantage is obvious.
The generated superpixels are as compact and neat as cells, and the neighborhood features are easier to express. Such pixel-based methods can be easily transformed into superpixel-based methods. Not only color images can be segmented, but also grayscale images can be segmented compatible.
There are very few parameters to set, by default only a number of pre-segmented superpixels need to be set.
Compared with other superpixel segmentation methods, SLIC is ideal in terms of running speed, compactness of generating superpixels, and contour preservation.

However, for some uneven images, such as Winnie the Pooh and its background, uniform sampling does not work well. It's easy to overlook details (even if it's visually obvious, like the mouth and nose), which makes the processing results unsatisfactory.

1.2 mesh simplification

1.2.1 Description of VSA

Variational Shape Approximation is an algorithm for mesh simplification based on k-means extension.

```
Input: partition($R = \{R_1, \dots, R_k\}, P = \{P_1, \dots, P_k\}$)
1  for i to k do
2      select the trangle $t \in R_i$ that minimizes $E(t, P_i)$
        $R_i = \{t\}$
        set t to conquered
        for all neighbors r of t do
3           insert $(r, P_i)$ into queue

4      end
5  end
6  while the queue is not empty do
7      get $(t, P_i)$ from the queue that minimizes $E(t, P_i)$
        if t is not conquered then
8           set t to conquered
             $R_i = R_i\{t\}$
             for all neighbors r of t do
9               r is not conquered insert $(r, P_i)$ into queue

10          end
11     end
12 end
```

.

1.2.2 Implement

Apply VSA on Stanford bunny
We use meshlab to implement VSA on Stanford bunny mesh.

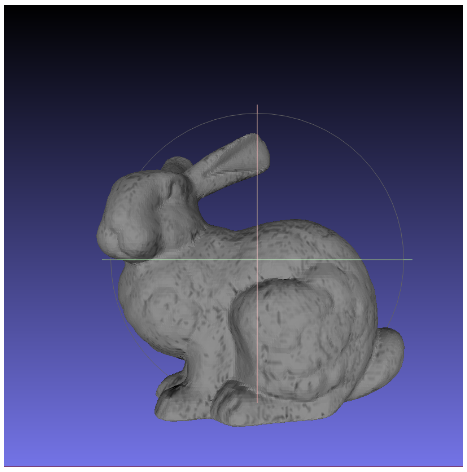

Figure 9: origin

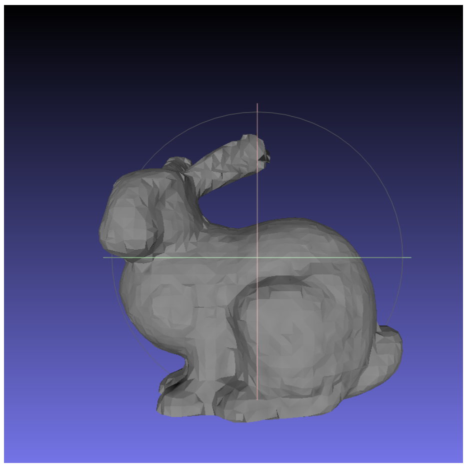

Figure 10: lightly simplified

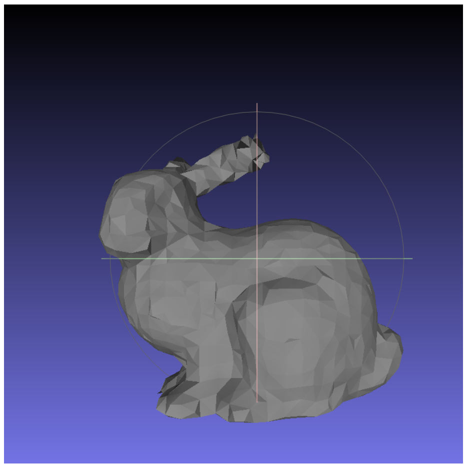

Figure 11: simplified

VSA is easy to fall into the dilemma of local optimal solution or even infinite loop. Therefore, in practical use, we often set the maximum number of iterations, so that the mesh we get will be an approximate solution.

Compare to QEM

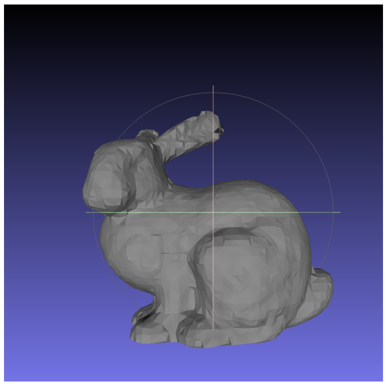

Figure 12: VSA

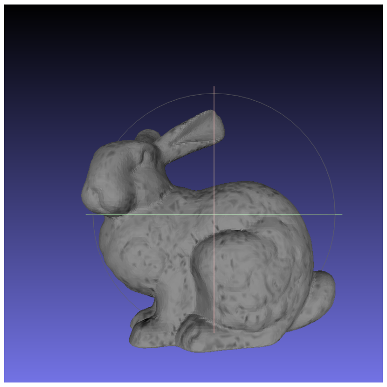

Figure 13: QEM

1.2.3 Discussion

Disadvantages

(1) It is easy to fall into the dilemma of local optimal solutions or even infinite loops

(2) It is good for objects that are not smooth, and it is poorly optimized for objects with a lot of noise

(3) Compared with QEM (quadratic error measure), the speed is slow