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

Dai ZiJia 2022233158

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**.

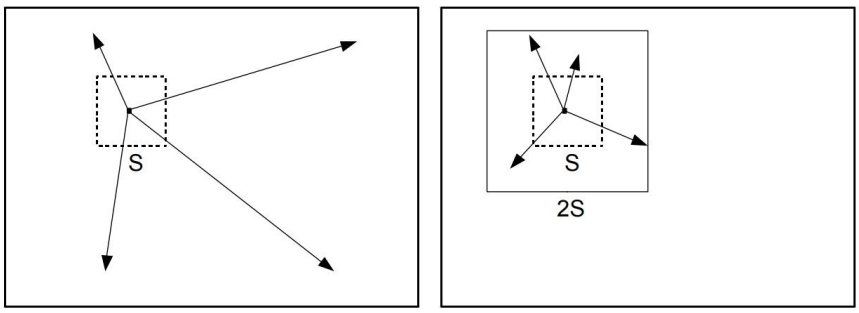
1.1 image superpixel segmentation (50 points)

Details

In this experiment, SLIC(simple linear iterative cluster) is used to do superpixel segmentation. It mainly converts the image from RGB color space to CIE-Lab color space, and each pixel form a 5-dimensional vector $V[l, a, b, x, y]$. The similarity of two pixels can be measured by their vector distance. The similarity becomes small when the vector distance get large.

It is inspired by K-means clustering algorithm. Firstly, it generates k seed points and classify the nearest pixels to each seed point into the same class. Then caculate the average vector value of all pixels in k superpixels to obtain k clustering centers, and then search the most similar pixels around the k centers until all pixels are classified. After that, obtain k superpixels again, and update the clustering centers. Iterate over this, and so on until convergence.

The innovation of SLIC is that it reducing the superpixel search regions. In the conventional k-means algorithm, distances are computed from each cluster center to every pixel in the image. However, SLIC only computes distances from each cluster center to pixels within a $2S \times 2S$ region, that makes the complexity of SLIC is linear in the number of pixels in the image $O(N)$.



(a) standard k-means searches the entire image

(b) SLIC searches a limited region

Figure 1: The innovation of SLIC

The algorithm mainly consist of six steps:

1) Initialize seed points (cluster centers) : Distribute seed points in the image according to the set number of superpixels. Assuming that the image has N pixels in total and is presegmented into K superpixels of the same size.

So the size of each superpixel is N/K, and the distance (step size) between adjacent seed points is approximately $S = \sqrt{N/K}$.

2) Reselect the seed point in the $n * n$ neighborhood of the seed point. First calculate the gradient value of all pixels in the neighborhood, and move the seed point to the place with the minimum gradient in the neighborhood. The purpose of this is to avoid the seed points falling on the contour boundary with large gradient, so as not to affect the subsequent clustering effect.

3) Assign a label to each pixel in the neighborhood around each seed point which cluster center it belongs to.

4) Caculate the distance. It includes color distance and spatial distance. For each searched pixel, calculate the distance between it and the seed point. Since each pixel will be searched by multiple seed points, take the seed point corresponding to the minimum value as the cluster center of the pixel.

5) Iterate and optimize. In theory, the above steps are iterated until the error converges that means the clustering center of each pixel no longer changes.

6) Post-processing. Some "orphaned" pixels that do not belong to the same connected component as their cluster center may remain. It is solved by creating a label table with all elements of -1. According to the 'Z' direction, the discontinuous and small superpixels are reassigned to the adjacent superpixels, and the traversed pixels are assigned to the corresponding labels until all the traversed points are finished.

Results

I mainly use two library skimage and opencv to realize it. They have different result. After that, compare the performance of the algorithm under different iterations.

Analysis

From the result of the algorithm, we could learn some conclusions.

- When the number of iterations becomes larger, the algorithm tends to be stable and the partition of superpixels becomes reasonable. There are almost no change when generally more than 7 times.

- If k is small, the size of each superpixel is big, this will lead to keep border becomes worse. However, if k is too large and each pixel size is small. There will be a similar phenomenon of "overfitting", the shape of superpixels becomea very irregular and neighborhood relationship is very difficult to maintain.

- The number of segmentation k is related to specific application scenarios. For example, if the body part of the figure is segmented, k can be set smaller, but if the detail parts of the hair and eyes of the figure are segmented, the segmentation accuracy is not high compared with the body part at this time. If there are higher requirements, a larger k value is

Figure 2: The result of SLIC by skimage

needed. Since the picture is a painting, but real scene will be much more complex, and the k also needs to be selected according to different occasions. So it will be more convenient if the algorithm can adaptively adjust the k value.

After reading, it is found that SLICO (improved SLIC) completely solves the parameter k selection problem. One no longer needs to set the closeness parameter or try different values. SLICO adaptively selects the closeness parameter for each superpixel. This results in regular shaped superpixels in both textured and non-textured regions. The improvement don't affects the computational efficiency which means SLICO is as fast as SLIC.

Figure 3: The result of SLIC by opencv

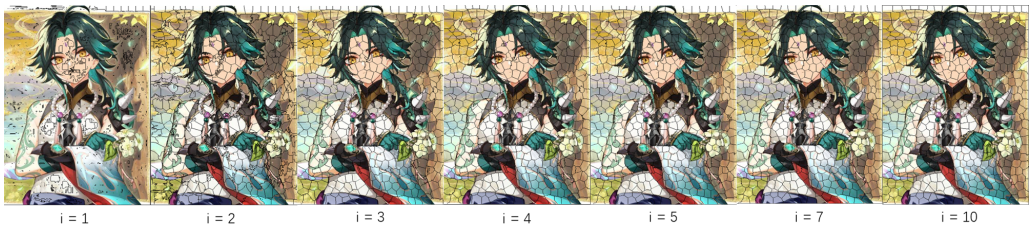


Figure 4: The result of different iteration

1.2 mesh simplification (50 points)

Details

Here I use VSA to do mesh simplification. Variational Shape Approximation(VSA) is an algorithm based on k-means extension for mesh simplification. VSA produces meshes that are anisotropic and of good quality. Moreover, VSA does not require global parameterization information and local differential components of the input model.

The algorithm choose halfedge data structure, a way to get three triangles near a triangle. The basic elements of the half-edge data structure are vertices, faces, and half-edges (directed edges). Each vertex stores an outgoing half-edge (that is, the point at which the half-edge starts). Each face stores a boundary half. Each half contains the following Pointers: 1. the end point; 2. the face that belongs to; 3. the next half of the surface (counter-clockwise); 4. opposite half; 5. (optional) the upper half of the face.

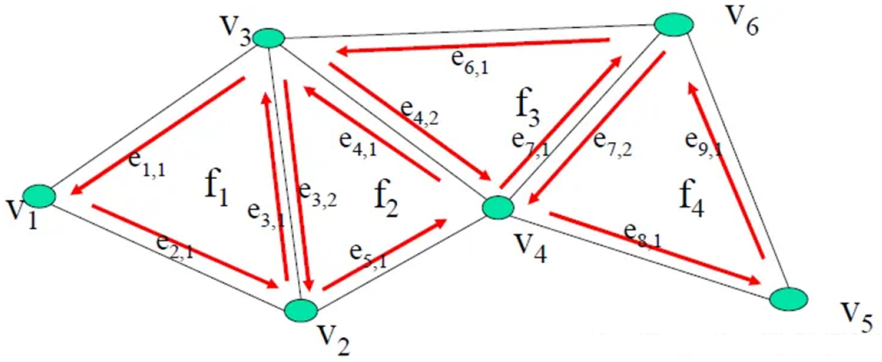


Figure 5: The halfedge data structure.

The algorithm mainly consist of following steps:

1) After constructing the mesh topology, caculate the area, normal vector and center of gravity of each triangle mesh.

2) Initialization. First randomly select seed triangle mesh. Then for each seed triangle mesh, use $L^{2,1}$ to calculate the error of the three triangles around and use a priority queue to store nearby three triangles(and their error,tag). By repeatedly finding the adjacent triangular mesh and calculating the error metric to put into the priority queue, traverse all the remaining triangular mesh.

3) Generate more suitable proxy existing partitions. Calculate the normal vector N and center coordinate B of the partition and use N and B to present proxy.

4) Calculate the global error E. The error of each triangular mesh is $L^{2.1}$

distance between its normal n and the proxy's N(Another way is by L^2). The global error is the sum of error of each triangular mesh.

5) 3 and 4 steps are executed as one iteration, which is repeated until the global error E converge.

6) Add anchor vertices. Create an anchor vertex on every original vertex where three or more partitions meet, it must be guaranteed that every region has at least three anchor points. For each anchored vertex average anchored vertex projects values on the corresponding partition.

7) Add edges that anchors the vertices. If the Angle between the normal of two proxies is too large, a new anchor vertex is added between the two anchor vertices by recursive chord length subdivision algorithm.

8) Use the CDT algorithm to perform triangulation with anchored vertices and edges.

Results

I use bunny, buddha and so on as input. And for 0.5, 0.2, 0.01 simplified rate.

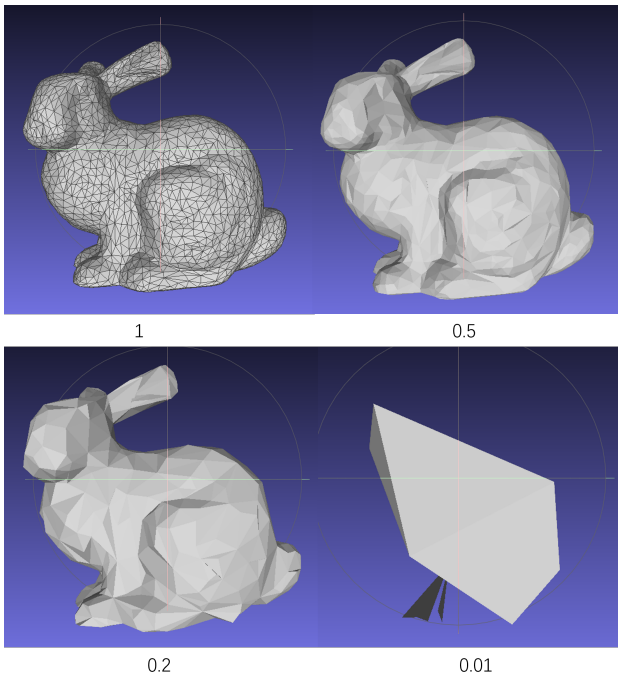


Figure 6: The result of mesh simplification-Bunny.

Analysis

Figure 7: The result of mesh simplification-Buddha.

When the need of simplification becomes higher and vertex becomes more, it takes a lot of time to simplificate the mesh.

In general, compared with other simplification algorithms, VSA controls the simplification degree on the whole, produces less deformation, and has the optimal solution in the flat place. VSA produces anisotropic meshes with good quality, and VSA does not require global parameterization information and local differential components of the input model. The algorithm as a whole has fewer parameters.

VSA also has some disadvantages, the shape of the generated mesh is not very good, but can it be optimized by combining with other schemes. It is easy to fall into the dilemma of local optimal solution or even infinite loop. They do well for objects that are not smooth but poorly optimize for objects with a lot of noise. The last one is It runs slow compare with others.

After reading, I found Greedy Shape Approximation is an improved form of the above algorithm. Compared with the former, it has the following two advantages:

- The algorithm can naturally generate a series of meshes with different approximation degrees.
- The output mesh avoids Fold-Over and Degenerate faces.

However, since the improved algorithm is based on the greedy, it may be a local optimal solution.

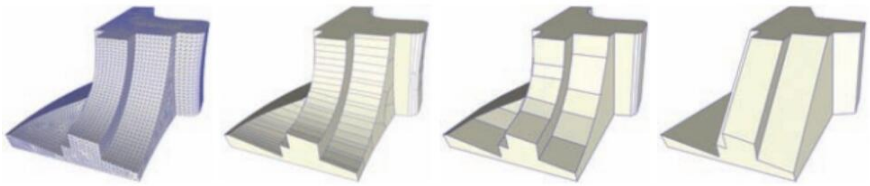


Figure 8: The Greedy Shape Approximation algorithm.