Assignment 2:
The application of K-Means Clustering

NAME: CHENQIHE
STUDENT NUMBER: 2020533088
EMAIL: CHENQH@SHANGHAITECH.EDU.CN

1 INTRODUCTION

In this assignment, we implement two algorithms both for 2D and 3D cases based on the K-Means algorithm.

For 2D case, we implement efficient 2D superpixels generation algorithm with reference to SLIC Superpixels Compared to State-of-the-Art Superpixel Methods.

For 3D case, we implement 3D mesh simplification algorithm with refecence to variational shape approximation.

2 RELATED WORK

K-Means K-Means is a widely used clustering algorithm in field of data mining across different disciplines in the past sixty years. It is also widely used in the field of computer vision including tasks like segmentation. Meanwhile, superpixels are becoming increasingly popular for use in computer vision applications. Intuitively, we can adapt K-Means clustering to generate superpixels.

Superpixel Introduced by Ren and Malik in 2003, superpixels group pixels similar in color and other low-level properties. In this respect, superpixels address two problems inherent to the processing of digital images: firstly, pixels are merely a result of discretization; and secondly, the high number of pixels in large images prevents many algorithms from being computationally feasible.

In general, most authors agree on the following requirements for superpixels:

- Partition. Superpixels should define a partition of the image, i.e. superpixels should be disjoint and assign a label to every pixel, which can be achieved by K-Means clustering.
- Connvectivity. Superpixels are expected to represent connected sets of every pixel, which can be achieved by a post-processing method(flooding) that we will talk about later.
- Boundary Adherence. Superpixels should preserve image boundaries. In this assignment, we don't take it into consideration.
- Compactness, Regularity and Smoothness. IN the absence of image boundaries, superpixels should be compact, placed regularly and exhibit smooth boundaries.
- Efficiency. Superpixels should be generated efficiently. In this assignment, we implement $O(n)$ time complexity algorithm, which is really efficient.
- Controllable Number of Superpixels. The number of generated superpixels should be controllable. In this assignment, we support variable number of superpixels for generation task.

Mesh simplification Mesh simplification problem tries to use the least number of triangle pieces to represent the main geometry structure of original mesh, which can be further used in downstream tasks. Using the concept of geometric proxies, VSA algorithm drive the distortion error down through repeated K-Means clustering of faces into best fitting regions. VSA(variational shape approximation) approach is entirely discrete and error-driven, and does not require parameterization or local estimations of differential quantities.

3 2D SUPERPIXEL GENERATION

3.1 IO and Image pre-processing

Our program applies K-means clustering in image superpixel segmentation one image by one image. And the type of image should be 'jpeg'. For reading a image, we use the third library 'Bitmap'. And we use standard color space transform function to transform the original ARGB image to XYZ to LAB.

The reasons why we deal with the image in LAB color space are considered in two points of view. First, image patches in LAB space are more reliable than in RGB space due to wider range. Second, the differences of image pixels are larger in LAB space.

3.1.1 RGB2XYZ. Assume original pixel in RGB color spaces has three channel (r, g, b), each channel takes value in $[0, 255]$.

First, we apply gamma correction:

$$\begin{cases} R &= \text{gamma}(\frac{r}{255.0}) \\ G &= \text{gamma}(\frac{g}{255.0}) \\ B &= \text{gamma}(\frac{b}{255.0}) \end{cases}$$

where, the gamma function maps rgb to RGB,

$$\text{gamma}(x) = \begin{cases} (\frac{x+0.055}{1.055})^{2.4} & (x > 0.04045) \\ \frac{x}{12.92} & (others) \end{cases}$$

Then, we can get XYZ simply by multiplying a matrix M.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.4124 & 0.3576 & 0.1805 \\ 0.2126 & 0.7152 & 0.0722 \\ 0.0193 & 0.1192 & 0.9505 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

3.1.2 XYZ2LAB. Then we get the image in XYZ color space, which is applied gamma function as non-linear transfer function. By applying another non-linear transfer function, we convert the image in XYZ color space into LAB color space.

$$L^* = 116f(Y/Y_n) - 16$$
$$a^* = 500[f(X/X_n) - f(Y/Y_n)]$$
$$b^* = 200[f(Y/Y_n) - f(Z/Z_n)]$$

where $f(t)$ is another non-linear transfer function and usually $X_n = 95.047$, $Y_n = 100.0$, $Z_n = 108.883$.

$$f(t) = \begin{cases} t^{1/3} & \text{if } t > (\frac{6}{29})^3 \\ \frac{1}{3}(\frac{29}{6})^2 t + \frac{4}{29} & \text{otherwise} \end{cases}$$

where, L^*, a^*, b^* are the final three channels of LAB color space.

3.2 Pick seeds for superpixels

By defauly, the only parameter of our algorithm is k, the desired number of approximately equally-sized superpixels. For color images in the LAB color space, the clustering procedure begins with an initialization step where k initial cluster centers $C_i = [l_i\ a_i\ b_i\ x_i\ y_i]^T$ are sampled on a regular grid spaced S pixels apart.

To produce roughly equally sized superpixels, the grid interval is $S = \sqrt{N/k}$.

3.3 Main algorithm: generating superpixels

In the assignment step, each pixel i is associated with the nearest cluster center whose search region overlaps its location, which is the key to speeding up our algorithm based on K-Means because limiting the size of the search region significantly reduces the number of distance calculations, and results in a significant speed advantage over conventional K-Means clustering where each pixel must be compared with all cluster centers. This is only possible through the introduction of a distance measure D, which determines the nearest cluster center of each pixel, as discussed in later section.

Since the expected spatial extend of a superpixel is a region of approximate size $S \times S$, the search for similar pixels is done in a region $2S \times 2S$ around the superpixel center.

Once each pixel has been associated to the nearest cluster center, an update step adjusts the cluster centers to be the mean $[l\ a\ b\ x\ y]^T$ vector of all pixels belonging to the cluster. The L_2 norm is used to compute a residual error E between the new cluster cneter locations and previous cluster center locations. The assignment and update steps can be repeated iteratively until the error converges. Finally, a post-processing step enforces connectivity by re-assigning disjoint pixels to nearby superpixels.

3.3.1 Distance measure. Our superpixels correspond to clusters in the labxy color-image plane space. It present a problem in defining the distance measure D, which may not be immediately obvious. D computes the distance between a pixel k and cluster center C_i in Algorithm 1. A pixel's color is represented in the LAB color space $[l\ a\ b]^T$, whose range of possible values is known. The pixel's position $[x\ y]^T$, on the other hand, may take a range of values that varies according to the size of the image.

Simply defining D to be five-dimensional Euclidean distance in labxy space will cause inconsistencies in clustering behavior for different superpixel sizes. For large superpixels(small superpixel nums), spatial distances outweight color proximity, giving more relative importance to spatial proximity than color. This produces compact superpixels that do not adhere well to image boundaries. For smaller superpixels, the converse is true.

Algorithm 1 superpixel generation

```
/* Initialization */
Initialize cluster centers C_i = [l_i a_i b_i x_i y_i]^T by sampling pixels
at regular grid steps S
Initialize label(k) and distance(k) for each pixel k
repeat
  /* Assignment */
  for each cluster center C_i do
    for each pixel k in 2S × 2S region around C_i do
      Compute the distance D between C_i and k
      if D < d(k) then
        distance(k) ← D
        label(k) ← i
      end if
    end for
  end for
  /* Update */
  /* Compute new cluster centers */
  Initialize new cluster centers C'_i = [0 0 0 0 0]^T, and cnt(i) for
  each new cluster center i
  for each pixel k in whole image do
    C'_label(k) += k
    cnt(label(k)) += 1
  end for
  for each new cluster center C'_i do
    C'_i /= cnt(i)
    C_i ← C'_i
  end for
  /* Compute residual error E */
  E ← ||C' − C||_2^2
until E ≤threshold or EPOCH ≥ 10
```

To combine the two distances into a single measure, it is neccessary to normalize color proximity and spatial proximity by their respective maximum distances within a cluster, N_s and N_c. Doing so, D' is written as

$$d_c = \sqrt{(l_j - l_i)^2 + (a_j - a_i)^2 + (b_j - b_i)^2}$$
$$d_s = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$
$$D' = \sqrt{(\frac{d_c}{N_c})^2 + (\frac{d_s}{N_s})^2}$$

The maximum spatial distance expected within a given cluster should correspond to the sampling interval, $N_s = S = \sqrt{(N/K)}$. Determining the maximum color distance N_c is not so straightforward, as color distances can vary significantly from cluster to cluster and image to image. This problem can be avoided by fixing N_c to a constant m so that D' becomes:

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

which simplifies to the distance measure we use in practice

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

By definiing D in this manner, m also allows us to weigh the relative importance between color similarity and spatial proximity. When m is large, spatial proximity is more important and the resulting superpixels are more compact(i.e. they have a lower area to permeter ratio). When m is small, the resulting superpixels adhere more tightly to image boundaries, but have less regular size and shape. When using the LAB color space, m can be in the range $[1, 40]$.

3.4 Post-processing

Superpixels generation process does not explicitly enforce connectivity. At the end of the clustering procedure, some "orphaned" pixels that do not belong to the same connected components as their cluster center may remain and be small. To correct for this, such pixels are assigned the label of the nearest cluster center using a connected components algorithm.

3.5 Complexity

By localizing the search in the clustering procedure, our algorithm avoids performing thousands of redundant distance calculations. In practice, a pixel falls in the neighborhood of less than eight cluster centers, meaning that the algorithm is $O(N)$ complex. In contrast, the trivial upper bound for the classical K-Means algorithm is (k^N), abd the practical time complexity is $O(NkI)$, where I is the number of iterations required for convergence. Finally, unlike most superpixel methods, the complexity of our algorithm is linear in the number of pixels, irrespective of k.

3.6 Analysis

Our image superpixel generation algorithm has extremely good speed on real-time running. Although the results look good, there are several limitation for this kind of superpixel generation.

The first limit is that the size of every superpixel is almost same, which is due to our algorithm design and follows the definition of superpixels. Thus, for further simplification, we can perform a more generalized K-Means on these superpixels, which results in different size of superpixels.

The second limit is that superpixels' meaning only depends on the distance measure. Thus, we can add more semantic information to our superpixels by handling distance measure over some semantic information.

4 3D MESH SIMPLIFICATION

To extend K-Means algorithm from 2D image clustering to 3D mesh simplification, there are two problems to be solved. The first problem is how we can traverse the triangle pieces with their neighbor triangles. The second problem is how we can measure the distortion with different clusters and the belonging faces with clustering regions.

The first traversal problem can be solved by recording the adjacent faces for each face. The recording process takes two steps. In first step, we traverse all edges and record which faces each edge

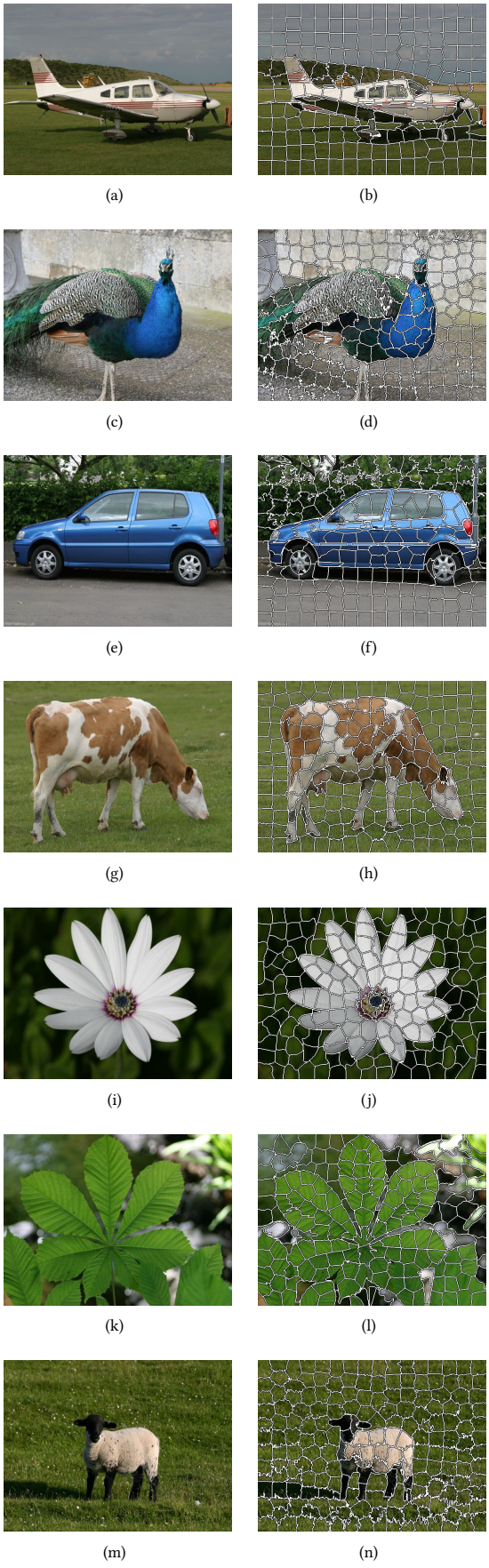


Fig. 1. Results: 2D image superpixel generation(200 superpixels per image)

belongs to. Here, each edge belongs to at most two faces and thus no memory leak problems. In second step, we traverse all faces and merge three edges belonging information for each face. Then, we record the adjacent faces for each face in $O(N)$ time.

The second measure problem can be solved by measuring the geometric relevance of a proxy set for a given surface: new definitions of error metrics are thus presented next. It will allow us to "score" a partition in terms of how well it approximates a surface.

4.1 Proxy

To solve second problem mentioned above, here we introduce a new concept proxy. Proxy identifies the properties of a region or a cluster. One proxy stores the index of the region, the indices of belonging faces and the area-weighted normal.

Then, we need to talk about what the area-weighted normal means and the reason why proxy needs to record it.

For ith proxy P_i, it has a set of belonging faces $\{F_i^j\}$ and a set of face normals $\{N_i^j\}$, we can get the area of each belonging faces by cross-product $\{\text{area}_i^j\}$, then we can calculate the area-weighted normal for ith proxy $\{\hat{N}_i\}$

$$\hat{N}_i = \frac{\sum_j \text{area}_i^j * N_i^j}{\sum_j \text{area}_i^j}$$

4.2 L^2 Metric for Proxies

We can extend the notion of L^2 distance to our framework. Given a region R_i, and its associated proxy $P_i = (X_i, N_i)$, we denote a L^2 measure of the normal field:

$$L^2(R_i, P_i) = \iint_{x \in R_i} \|\hat{N}_i - N_i(x)\|_2^2 dx$$

This measure is valid since finding the best normal proxy is as simple as averaging the nromals over the associated region. We do not have to compute a covariance matrix, and thus, we save a significant amout of computations compared to other L^2 measure in normal field.

4.3 Optimal Shape Proxies

We now have everything we need to define what we mean by an optimal partitioning of an arbitrary surface: Given an error metrix E, a desired number k of proxies, and an input surface S, we call optimal shape proxies a set P of proxies P_i associated to the regions R_i of a partition R of S that minimizes the total distortion:

$$E(R, P) = \sum_{i=1,\ldots,k} E(R_i, P_i)$$

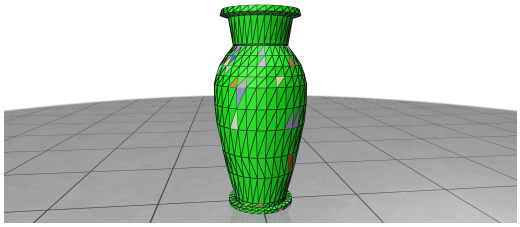
In other words, the set of proxies is optimal with respect to an error metric if it minimizes the total approximation error over the possible sets of proxies of same cardinality. Of course, in practice we cannot hope to find the global minimum in a reasonable time. However, we set up our shape approximation as a discrete, variational partitioning of the initial faces such taht we can now apply simple and powerful discrete clustering algorithms that achieve very good and on some simple cases near-optimal results.

4.4 3D Mesh Simplification algorithm

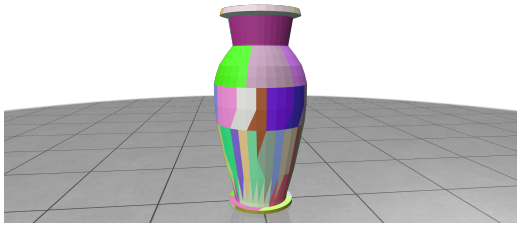
As a iterative algorithm, the quality of results usually depends on the iteration times of the algorithm. In this assignment, we choose the iteration times to be 10. Also, the number of final simplified faces influence the quality of results and we choose the number of simplified faces to be 100.

The main process of the 3D mesh simplification algorithm is given below:

(1) calculate region's area-weighted normals
(2) calculate L^2 distance in normal field between all faces and all regions
(3) put updated proxy into priority queue
(4) repeat popping the top element from the queue until all faces are assigned to a region
(5) split the worst region into two regions (add one region)
(6) update adjacent regions information
(7) merge two best regions into one region (sub one region)
(8) repeat (1)-(7) n times

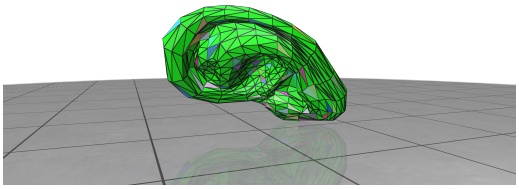


(a)

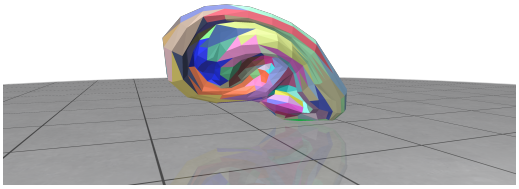


(b)

Fig. 2. Results: vase faces(1504->100)

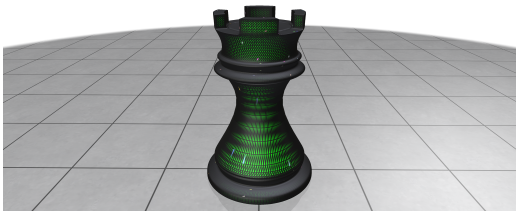


(a)

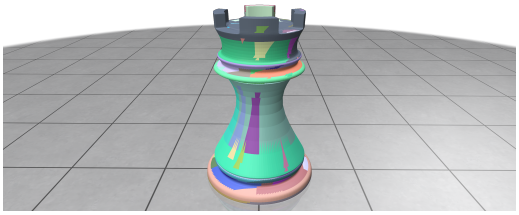


(b)

Fig. 3. Results: ear faces(710->100)

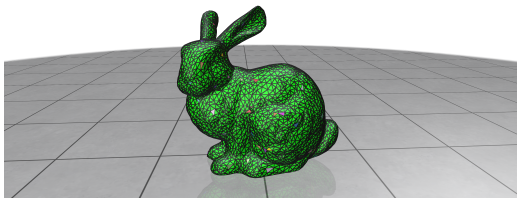


(a)

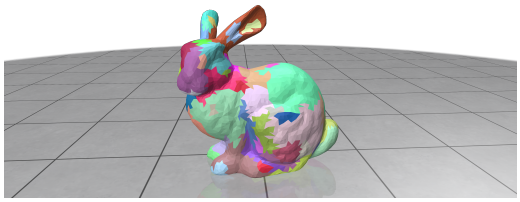


(b)

Fig. 4. Results: chess piece faces(42684->100)

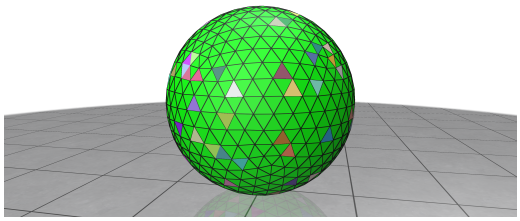


(a)

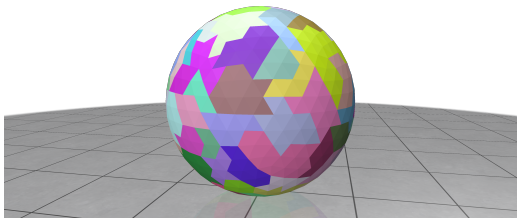


(b)

Fig. 5. Results: bunny faces(7361->100)



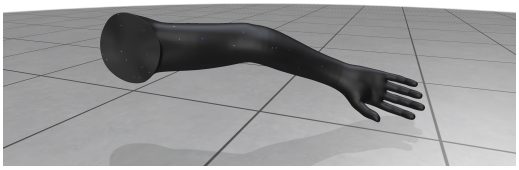
(a)



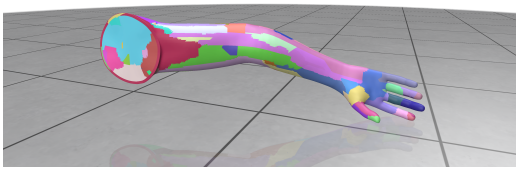
(b)

Fig. 6. Results: bubble faces(1280->100)

student number: 2020533088
email: chenqh@shanghaitech.edu.cn



(a)



(b)

Fig. 7. Results: arm faces(98624->100)