Multimedia Retrieval

Diego Renders (5740894) and Claudiu Gheorghica

Oktober 2019

1 Introduction

Problem statement.

2 Part 1

2.1 File reading

The files from the Princeton Shape Benchmark (PSB) were all provided in the OFF file format and an OFF reader was initially considered as a starting point. However, as a PLY format reader was likely to be necessary in the long-term, an OFF-to-PLY converter was written in order to enforce PLY as the default format for the application. This converter simply creates a PLY header filled with data from the OFF file, then copies the vertex and face data over. While not a scalable approach for more complex OFF files, the converter was able to successfully process the entire PSB database.

2.2 Database evaluation

The database contains around 1800 models in OFF file format. All of the models contain only triangles as faces. The distribution of faces is shown in figure ?. The average number of vertices is 4221, the minimum 10, and the maximum 160940. The histogram, along with the average, shows that most of the models have a vertex count of smaller than twenty thousand. In order to successfully perform the upcoming steps of feature extraction the models need to have approximately the same number of vertices. Two actions can be undertaken to achieve this, supersampling and subsampling. Supersample the models with a low vertex count, and subsample the models with a large vertex count. A target number of vertices has to be chosen for this. One solution could be to choose the average. However because several of the meshes have a very high resolution (a hundred thousand vertices), subsampling this to reach the four thousand required vertices would ruin the original shape. Therefore the target vertices is chosen to be higher than the average, so that the higher resolution models retain their characteristics. The only downside for choosing a higher target number of vertices is an increase in computation time. The target number of vertices is set at forty thousand, and the methodology for supersampling and subsampling is explained in the following two sections.

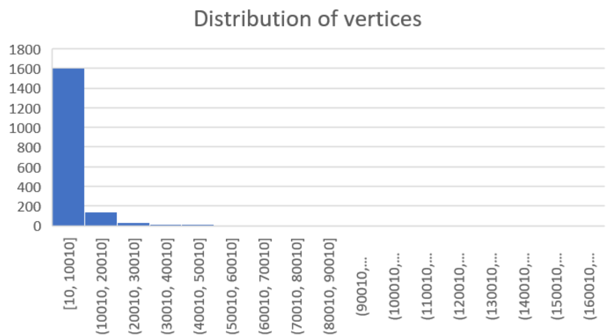


Figure 1: Histogram showing the distribution of vertices in the PSB database

2.3 Supersampling

Supersampling is done in this implementation by splitting a larger triangle into 4 smaller ones via adding the midpoints of the original triangle's edges as vertices and using them to form new triangles with the original vertices. An example is shown below:

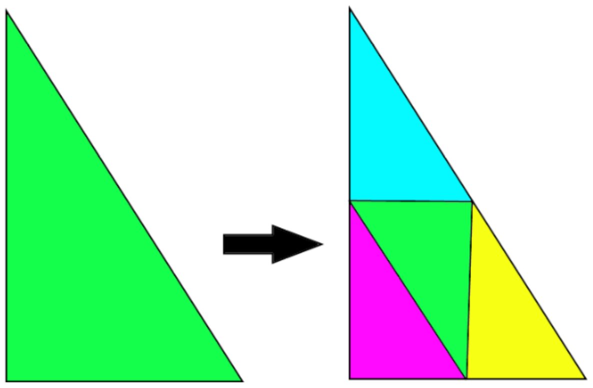


Figure 2: Supersampling Example

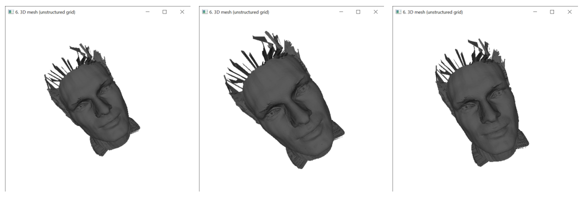
To ensure the triangles obtained like this are not outliers in terms of their size, the cells in the model were first sorted by their area inside a vector, from smallest to largest. The area of each cell was calculated using the following formula:

$$S = \frac{|AB||AC|sin(\theta)}{2} \tag{1}$$

Thus, at each step, only the triangle with the largest surface is split up into smaller triangles. The original triangle is removed from the vector, and the smaller triangles are inserted at positions that do not disturb the ordered property of the data structure.

2.4 Subsampling

For the task of subsampling the Polygon Mesh Processing (pmp) library is used. The SurfaceSimplification class is used to perform mesh decimation, taking as input the mesh and a target number of vertices. To demonstrate that this algorithm works correctly for our models a pre and post visualisation is shown in figure 2. This is the largest model in the dataset with 316.498 faces and 160.940 vertices. The post decimation model contains 78.473 faces and 40.000 vertices in the middle picture and 20.000 vertices in the right picture. Even though the model only has a fourth of its original vertices in the second case, it clearly retains its characteristics. And even when the vertex count is brought down to 20.000 vertices it is hard to distinguish them. Therefore we can conclude that the algorithm is a suitable tool for this project.



(a) 160k vertices (b) 40k vertices (c) 20k vertices

Figure 3: Results of mesh decimation for model m303

2.5 Four step normalization

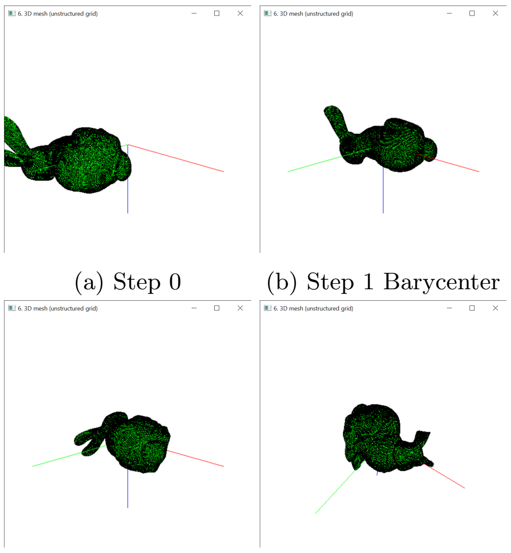
Next each mesh will go through the four step normalization pipeline so that it is ready to be used in upcoming tasks. Figure 1 shows a visualization of what each step does, the red, green, and blue represent the x, y, and z axises respectively from zero to one. The first step is to center on the Barycenter, than in step 2 PCA is done to orient the object in an intuitive way. Step 3 performs a fliptest which result in the majority of the mass in the object to be located in the negative side of the axis. Finally step 4 normalizes the model, which is excluded in the figure because the model was already normalized and thus would show no difference.

2.6 Step 1. Center on the barycenter

For the first step the barycenter b is calculated, which is the average x, y, and z coordinates of all vertices in the mesh. Next, each vertex v is translated, by subtracting these averages from each of its points resulting in the new vertex position u.

$$\begin{bmatrix} u_x \\ u_y \\ u_z \end{bmatrix} = \begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix} - \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix}$$

The result can be verified by calculating the average coordinates again, which should be 0 after the normalization.



(a) Step 0 (b) Step 1 Barycenter

(c) Step 2 PCA (d) Step 3 Fliptest

Figure 4: Visualization of the Four step normalization pipeline.

2.7 Step 2. PCA

Prinicipal component analysis is done using the arglib library. The three eigenvectors e_1, e_2, and e_3 are returned by the algorithm. A translation matrix M is than created using the vectors x, y, and z, that represent the axises of the normal coordinate system.

$$x = \begin{bmatrix}1\\0\\0\end{bmatrix} \; y = \begin{bmatrix}0\\1\\0\end{bmatrix} \; z = \begin{bmatrix}0\\0\\1\end{bmatrix} \; M = \begin{bmatrix}x \cdot e_1 & x \cdot e_2 & x \cdot e_3\\ y \cdot e_1 & y \cdot e_2 & y \cdot e_3\\ z \cdot e_1 & z \cdot e_2 & z \cdot e_3\end{bmatrix}$$

The new vertex u is obtained by multiplying this matrix with the old vertex v.

$$\begin{bmatrix}u_x\\u_y\\u_z\end{bmatrix} = M \cdot \begin{bmatrix}v_x\\v_y\\v_z\end{bmatrix}$$

The result of the translation can easily be verified. Doing PCA again on the translated mesh returns three eigenvectors that now correspond with the x, y, and z axis.

2.8 Step 3. Fliptest

The eigenvectors used for the translation in the previous step are unoriented and thus give no information about to which side the model should be directed. Using the fliptest it is ensured that the majority of the mass resides in the negative half-space. Mass in this case is not indicated by the number of the vertices but also takes momentum into consideration, i.e. vertices farther away from the origin have a higher weight. Three variables are introduced: w_x, w_y, and w_z that indicate the total weight of each of three coordinates.

$$w_i = \sum sign(C_{t,i})(C_{t,i})^2 \qquad (2)$$

where $C_{t,i}$ is the i^{th} coordinate of triangle t ($i \in x, y, z$). The latter part of the summation gives coordinates far away from the origin an higher weight, while the former part gives it either a negative or positive weight. These values are than used for a new translation matrix M. Which flips the coordinates if necessary, in case the mesh is already properly oriented M will simply be the identity matrix.

$$M = \begin{bmatrix}sign(w_x) & 0 & 0\\ 0 & sign(w_y) & 0\\ 0 & 0 & sign(w_z)\end{bmatrix}$$

2.9 Step 4. Normalization

The last step scales the model in the unit volume, i.e. it can fit in a unit cube. First the min and max of the x,y, and z coordinates of the axis-aligned bounding box is found. Next the largest distance δ between the min and max of these coordinates is used for the scaling factor $s = \frac{1}{\delta}$. Finally each vertex v is multiplied with this factor to obtain the new vertex u.

$$\begin{bmatrix}u_x\\u_y\\u_z\end{bmatrix} = \begin{bmatrix}v_x\\v_y\\v_z\end{bmatrix} \cdot s$$

A visualization is excluded, however the effectiveness of this step can easily be verified by looking at the resulting bounding box of the model.