COMP-6311 Project
Hossein Pejman
Hossein.pejman-tavallaee.1@ens.etsmtl.ca
Fall 2023

Compression of Mesh Animation Sequences

Introduction

In this project the following paper related to mesh animation compression was implemented:

- *Bailin Yang, Luhong Zhang, Frederick W. B. Li, Xiaoheng Jiang, Zhigang Deng, Meng Wang, and Mingliang Xu. 2019. Motion-Aware Compression and Transmission of Mesh Animation Sequences. ACM Trans. Intell. Syst. Technol. 10, 3, Article 25 (May 2019), 21 pages. https://doi.org/10.1145/3300198*

Mesh Compression Algorithm

The paper proposed a method for vertex compression of mesh animation sequences. The authors supposed that the connectivity of vertices (faces information) for all frames is the same (faces information in ".obj" files of all frames are the same). The main idea of the paper is to find the motion similarity of vertices of frames by grouping the frames with similar vertex **motion strength** (Temporal clustering) and finding motion strength similarity of vertices inside of each frame (spatial clustering).

Motion strength of vertices is calculated based on the **Torsion** and **Curvature** of curves related to trajectory of vertices over frames. Authors showed the vertices with similar trajectories have similar motion strength. The proposed compression approach has the following steps:

1. *Calculating torsion and curvature of vertices*: in three dimensions, the curvature and torsion of a curve show how sharply it is twisting in space. Suppose that each vertex V moves over frames. As a result, the positions of each vertex in two consecutive frames t and $t+1$ make the given vector r. The torsion and curvature for each vector r can be computed as follows:

$$K = \frac{\left|(\mathbf{r}' \times \mathbf{r}'')\right|}{\|\mathbf{r}'\|^3} \qquad \tau = \frac{(\mathbf{r}' \times \mathbf{r}'') \cdot \mathbf{r}'''}{\|\mathbf{r}' \times \mathbf{r}''\|^2}$$

   Therefore, we need to compute first, second, and third derivatives of all r vectors. In other words, we finally have $F \times N$ curvature values and $F \times N$ torsion values**,** where F is the number of frames and N is the number of vertices in each frame.

2. *Temporal clustering:* first a matrix including curvature and torsion of vertices is formed, where i and j show the where i and j show the indices of vertices and frames, respectively. Then, the rows of this matrix are clustered using the k-mean algorithm. As a result, frames are clustered in S group based on similarities of curvature and torsion of their vertices.

$$\mathbf{T} = \begin{pmatrix} k_{11} & \tau_{11} & \dots & k_{i1} & \tau_{i1} \\ k_{12} & \tau_{12} & \dots & k_{i2} & \tau_{i2} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ k_{1j} & \tau_{1j} & \dots & k_{ij} & \tau_{ij} \end{pmatrix}$$

3. *Spatial segmentation*: like temporal clustering, for spatial segmentation, the matrix of curvatures and torsions of vertices of all frames for each cluster is formed:

$$\mathbf{R} = \begin{pmatrix} \alpha\widehat{x_1} & \alpha\widehat{y_1} & \alpha\widehat{z_1} & \frac{1-\alpha}{2}k_{11} & \frac{1-\alpha}{2}\tau_{11} & \dots & \frac{1-\alpha}{2}k_{1S} & \frac{1-\alpha}{2}\tau_{1S} \\ \alpha\widehat{x_2} & \alpha\widehat{y_2} & \alpha\widehat{z_2} & \frac{1-\alpha}{2}k_{21} & \frac{1-\alpha}{2}\tau_{21} & \dots & \frac{1-\alpha}{2}k_{1S} & \frac{1-\alpha}{2}\tau_{2S} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \alpha\widehat{x_N} & \alpha\widehat{y_N} & \alpha\widehat{z_N} & \frac{1-\alpha}{2}k_{N1} & \frac{1-\alpha}{2}\tau_{N1} & \dots & \frac{1-\alpha}{2}k_{1S} & \frac{1-\alpha}{2}\tau_{NS} \end{pmatrix}$$

The first three columns denote the average coordinates of the vertex i^{th} within a given cluster of frames. Each row shows the torsion and curvature values of all vertices (N) for all frames of a given cluster. The α value is a weight that determines the importance of each component ($\alpha = 0.5$ is used in the project). Like temporal clustering, the rows of this matrix are grouped using the k-mean algorithm. In fact, in this case vertices with similar motion strength inside each frame of the cluster are grouped (segmented).

4. Compute Graph Fourier transform (GFT): GFT is a mathematical transform which decomposes the *Laplacian* matrix of a graph into eigenvalues and eigenvectors. The Laplacian matrix of a graph is computed as follows:

$$L = D - A$$

   where D is the degree matrix and A is the adjacency matrix of the graph and L is the Laplacian matrix. To compute the GFT coefficients of vertices, we first compute the matrix L for the vertices of each segment. If a segment has N vertices, the matrix of eigenvectors of L are $N \times N$. As a result, the vectors of GFT coefficient for axes X, Y, and Z values are calculated for each segment as follows:

$$\mathrm{V_{\omega x}} = \mathrm{U^T.V_x}, \quad \mathrm{V_{\omega y}} = \mathrm{U^T.V_y}, \quad \mathrm{V_{\omega z}} = \mathrm{U^T.V_z}$$

   Where $\mathrm{U^T}$ is the transpose of the eigenvectors matrix, V_x, V_y, and V_z are the vectors of elements of each axis (x, y and z) for all vertices of a segment. Then, the graph can be

compressed by zeroing out the k coefficients of vectors with the smallest magnitudes values from the vector set $V_{\omega i} = (X_{\omega i}, Y_{\omega i}, Z_{\omega i})$, $i = 1..N$.
Finally, by doing inverse transform of GFT, the reconstructed (compressed) vertices are computed:

$$\hat{X} = U.(\hat{X}_\omega),\ \hat{Y} = U.(\hat{Y}_\omega),\ \hat{Z} = U.(\hat{Z}_\omega)$$

Using GFT is like using the Fast Fourier Transform (FFT) for image compression where some frequencies after the FFT transform are removed and then, the inverse FFT is used for making the lossy (compressed) image.

5. In the final step, the compression error between the original and reconstructed vertices are calculated using the Karni-Gotsman error[1] method.

Mesh Animation Compression Software

In this section the implementation of the mesh animation compression algorithm was explained. It was implemented in C++ (Visual studio 2019). Libraries used in the project are as follows:

- Eigen for matrix computations
- GLEW library
- Imgui and nfd for creating controls (Button, TextBox, …) in openGL program.

The main modules of the software are:

- *MotionStrenge.cpp (class)*: calculates the Torsion and curvature of vertices.
- *Temporal clustering.cpp (class)*: temporal clustering is implemented in this class.
- *SpatialSegmentation.cpp (class)*: implements the spatial segmentation algorithm.
- *GFT.cpp (class)*: implement GFT coefficients by computing the eigenvectors of Laplacian matrices. Also, make the reconstructed vertices by doing inverse GFT.
- *KGerror.cpp* (class): computes the compression error between the original and reconstructed vertices.

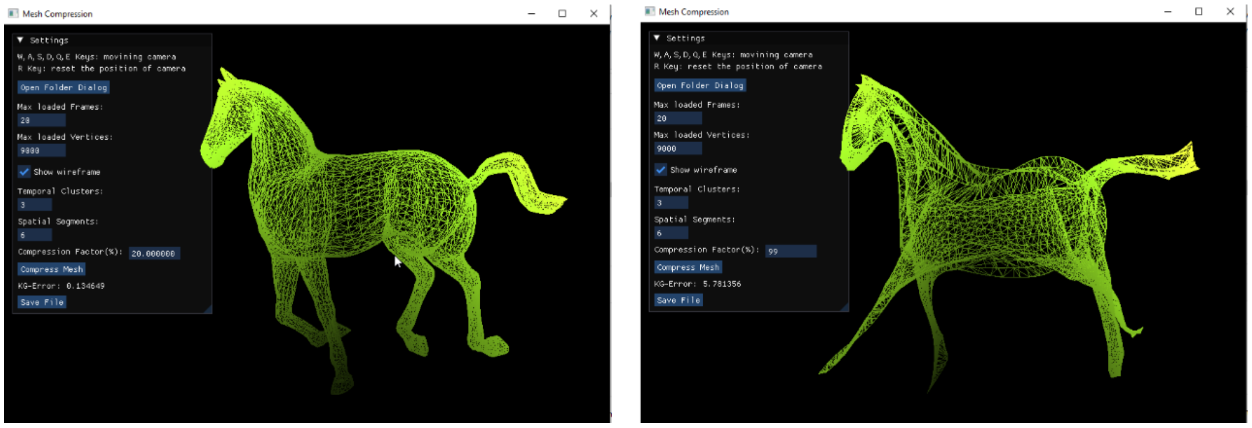
The software has the following features:

- Selecting the number of temporal clusters and spatial segments.
- Loading the .obj file of a mesh animation with an optional number of frames and vertices.
- Reconstructing the mesh animation with an optional compression factor.
- Computing and showing the KG-error after reconstruction.

[1] Zachi Karni and Craig Gotsman. 2004. Compression of soft-body animation sequences. Computers & Graphics 28, 1, https://doi.org/10.1016/j.cag.2003.10.002

- Save the reconstructed frames in an optional folder.

Figure.1 shows an example of reconstructed mesh animation with compression factor of 20 (left) and 99% (right). As can be seen the shape of the mesh with compression factor of 99% is obviously changed.

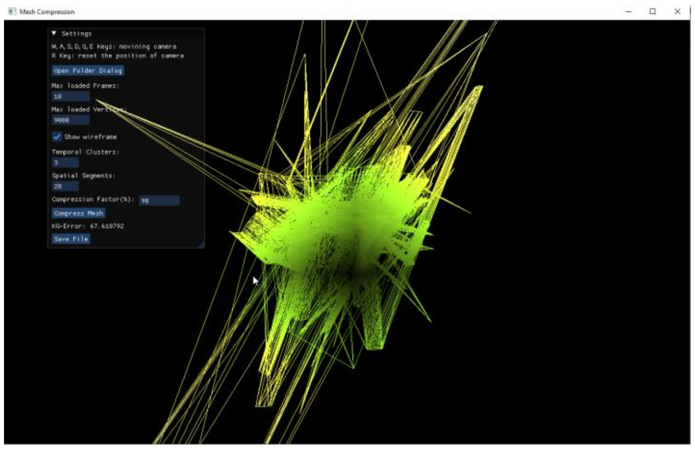


```
▼ Settings
W, A, S, D, Q, E Keys: movining camera
R Key: reset the position of camera
Max loaded Frames:
20
Max loaded Vertices:
9000
✓ Show wireframe
Temporal Clusters:
3
Spatial Segments:
6
Compression Factor(%): 20.000000
KG-Error: 0.134649
```

```
▼ Settings
W, A, S, D, Q, E Keys: movining camera
R Key: reset the position of camera
Max loaded Frames:
20
Max loaded Vertices:
9000
✓ Show wireframe
Temporal Clusters:
3
Spatial Segments:
6
Compression Factor(%): 99
KG-Error: 5.781356
```

Figure 1

One of the drawbacks of this method that was observed during evaluation is that the quality of reconstructed mesh is substantially dependent on the numbers of temporal clusters and spatial segments. For example, as can be seen in Figure.2 choosing 20 segments completely destroys the shape of the object. Therefore, the number of temporal clusters and spatial segments must be carefully selected.



```
▼ Settings
W, A, S, D, Q, E Keys: movining camera
R Key: reset the position of camera
Max loaded Frames:
10
Max loaded Vertices:
9000
✓ Show wireframe
Temporal Clusters:
3
Spatial Segments:
20
Compression Factor(%): 98
KG-Error: 67.610792
```

Figure 2

Conclusion

In this project a software for a proposed mesh animation compression was implemented. The authors of the proposed method used temporal clustering of frames. Then the vertices into each frame are grouped with similar motion strength. This method is similar to the main approach used in the video compression, where temporal and spatial redundancies are removed using inter-prediction and intra-prediction, respectively.