Shape Interpolation in 2D

CM50245 - Computer Animation and Games II

Unit Leader: Dr. Yongliang Yang

Yongqiang Zhu (MSc Digital Entertainment)

April 18, 2019

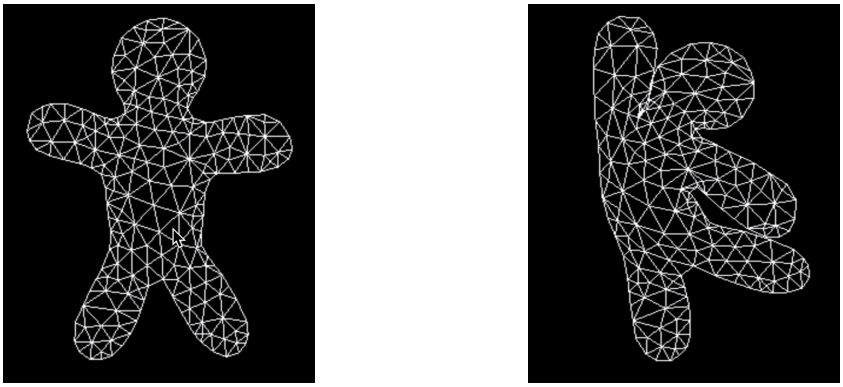
1 Introduction

Interpolation of two objects is a useful technique. For example, given two images of the same place, taken at different time of a day, and by interpolation it can create many new shots at different light level. It can also be used on interpolating shapes. In computer animation and games, there are many situations where an object changes it shape completely, and the entire transformation is animated. This can be achieved by artists manually putting frames together, but interpolation techniques can help doing it, making the process a lot more efficient and lower cost. This report is going to explore different interpolation methods, to see how they perform and how the results look like comparing to each other.

The rest of the report is structured as follow: section 2 explains the shape files and how to read them, section 3 explains the method of linear interpolation. Section 4 shows another method called 'As-Rigid-As-Possible Shape Interpolation' and result comparisons have been shown between it and linear interpolation. Then section 5 is the guidance of the program, and section 6 acknowledges the third party C++ library used in the implementation. In the end section 7 is the conclusion of this report.

2 Mesh Loading

There is a triangle mesh file stored in 'man.obj'. It contains the coordinates of all vertex points, as well as for each triangle/face the corresponding vertex indices are also store in the file. Each line of vertex coordinates has the following format: 'v' x-coordinate y-coordinate z-coordinate, and since in this project we are focusing on shape interpolation in 2D then all the z-coordinates will be 0. As for the faces, they are stored as follow: 'f' vertex-1-index vertex-2-index vertex-3-index. So any other lines which do not begin with 'v' or 'f' are either comments or blank line, these will be ignored. Apart from that, there is another file called 'keyframe.txt', it stores the vertex coordinates of the end pose. Each line contains the x-coordinate and the y-coordinate of a vertex point, and the order of points is the same as the triangle mesh. The begin pose is shown in Figure 1a and the end pose Figure 1b.



(a) Beginning mesh (b) End pose

Figure 1: Loading the mesh and poses file

3 Linear Interpolation

The idea of linear interpolation is rather simple. Since the begin pose has all vertex one-to-one matches with the end pose, the transformation vectors (only translation in this interpolation) can be easily calculate, simply for every matching vertex subtract the end value by the begin value. Then the amount of translation is related to time. This idea is further extracted

into equations 1 and 2. The linear interpolation vectors can be calculated by subtracting all the begin vertices $\boldsymbol{P}$ from the end vertices $\boldsymbol{Q}$. Then the amount of interpolation depends on time, so at time t (between 0 and 1) the interpolation value is the product of t and the interpolation vector, and the interpolated vertex coordinates can be calculated by adding the interpolated value to the begin vertices. The middle steps of interpolation can be visualised in Figure 2a.

$$\boldsymbol{InterpolationVector} = \boldsymbol{Q} - \boldsymbol{P} \tag{1}$$

$$\boldsymbol{V}(t) = \boldsymbol{P} + \boldsymbol{InterpolationVector} * t \tag{2}$$

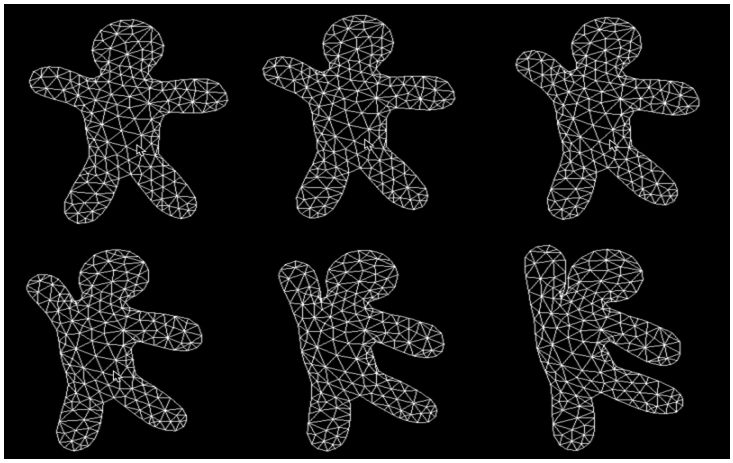
4 As-Rigid-As-Possible Shape Interpolation

Apart from the trivial linear interpolation method, there are also other interesting methods for interpolating shapes. The concept of doing as-rigid-as-possible shape interpolation comes from a paper written by Alexa et al. (2000).

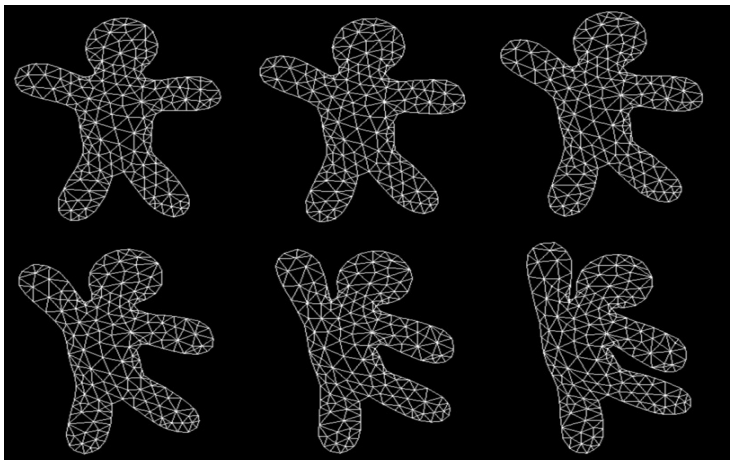
4.1 Affine Transformation

The method begins with computing the ideal affine transformation between a pair of matching triangle face (3 pairs of vertices). An affine transformation matrix can be calculated using at least 3 pairs of matching vertices. This is suitable for the project since the mesh is triangle based. Equation 3 shows the affine transformation from the begin vertex P to the end vertex Q. Since the affine transformation matrix requires 3 pairs of matching vertices, it can be represent in the form of $\boldsymbol{Ax} = \boldsymbol{b}$ like equation 4, and it can be obtained by $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. Here a11, a12, a21 and a22 are the four values of a rotation matrix, whereas a13 and a23 are for translation. Translation will be ignored according to the paper (Alexa, Cohen-Or, & Levin, 2000) since it does not help interpolating the shape.

$$\begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} \begin{bmatrix} a11 & a12 & a13 \\ a21 & a22 & a23 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} q_x \\ q_y \\ 1 \end{bmatrix} \tag{3}$$



(a) Middle steps of linear interpolation



(b) Middle steps of ARAP shape interpolation

Figure 2: Comparing the result of linear interpolation (a) and ARAP shape interpolation(b)

$$\begin{bmatrix} p_{x1} & p_{y1} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{x1} & p_{y1} & 1 \\ p_{x2} & p_{y2} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{x2} & p_{y2} & 1 \\ p_{x3} & p_{y3} & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & p_{x3} & p_{y3} & 1 \end{bmatrix} \begin{bmatrix} a11 \\ a12 \\ a13 \\ a21 \\ a22 \\ a23 \end{bmatrix} = \begin{bmatrix} q_{x1} \\ q_{y1} \\ q_{x2} \\ q_{y2} \\ q_{x3} \\ q_{y3} \end{bmatrix} \tag{4}$$

4.2 Rotation Interpolation

Alexa et al. (2000) have suggested that the transformation matrix can be decomposed into a product of a single rotation R matrix and a symmetric matrix Sym using singular value decomposition, as explained in equation 5.

$$A = USV^T = U(V^TV)SV^T = (UV^T)(VSV^T) = R * Sym \tag{5}$$

It is important to say that while Sym is consistent for each transformation matrix, R is not, it is dependent on time t. Therefore rotation matrix R with respected to time t can be calculated by rotation interpolation. This is similar to linear interpolation in section 3, it is just not interpolating the translation vector, but the rotation angle θ.

$$\theta_t = (1-t)\theta_p + t\theta_q \tag{6}$$

Equation 6 has explained the process of interpolating the rotation angle. Since rotation matrix from P to itself will be the identity matrix, θ_p will therefore be 0, so equation 6 can be simplified to equation 7.

$$\theta_t = t\theta_q \tag{7}$$

4.3 Transformation Matrix

The ideal transformation matrix of each triangle (3 vertex points) is $A(t)$ such that $V(t) = A(t)P$. This matrix can be calculated using the decomposed matrices from section 4.2, using equation 8.

$$A(t) = R(t) * ((1-t)I + t * Sym) \tag{8}$$

However problems occur when putting all triangles together. Since most of the vertices in the mesh are shared between multiple triangles, by using the ideal transformation $A(t)$ they would have inconsistent results. Therefore it is important to construct a different transformation $B(t)$ such that for all vertices they can all have consistent results. Since $B(t)$ is different from $A(t)$ the result will not be ideal individually, so this $B(t)$ should have minimal

difference to $A(t)$ while reserving consistent results. This can be achieved by using least squares optimisation between $A(t)$ and $B(t)$ as shown in equation 9.

$$Min(Error(t)) = Min(\sum ||A(t) - B(t)||^2) \tag{9}$$

4.4 Other Notes

Looking at the calculation of least squares optimisation between $A(t)$ and $B(t)$, it appears that they have only constrained the rotation aspect, while leaving the translation aspect free. If translation is not constrained then the interpolated results will not be corrected. One solution could be constraining the translation by fixing one pair of matching vertices, using linear interpolation. When doing so the translation is fixed, and the rotation can be interpolated so the overall affine transformation can be performed. Since rotation is interpolated based on the fixed translation, if different pair of matching vertices is selected then the overall outcome will be slightly different.

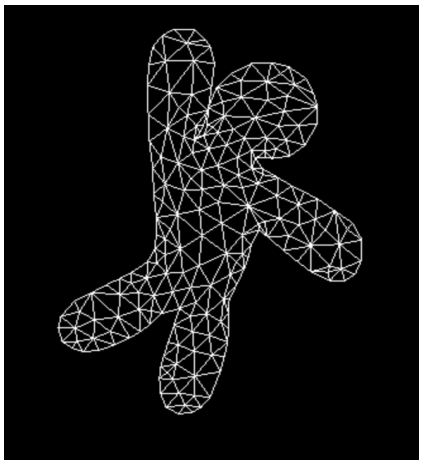


Figure 3: Incorrect angle interpolation

During the implementation of rotation interpolation, one problem has occurred in the first attempt. Like shown in Figure 3 the bottom part of the mesh is different from the expected result (Figure 1b). It can be observed

that the bottom part has rotated clockwise when it is supposed to rotate anti-clockwise. The reason is soon found that while calculating the angle θ_q, trigonometric function 'arccos' has been used. The result of 'arccos' would always return a positive value. After switching the function to using 'arcsin', the problem is solved and the result looks as expected.

4.5 Comparison of Interpolation Methods

While linear interpolation is trivial and very efficient, for complex shapes it would generally have distortions in shape during the interpolation process. That is why some other interpolation methods are introduced to tackle those difficulties. Figure 2 has shown the step comparisons between the results of linear interpolation and ARAP shape interpolation. The begin pose is a natural position of the gingerbread man, and the end pose is the gingerbread man with its left arm raising, right arm moving down, and both legs moving toward the right arm. Since the overall shapes for both begin and end pose remain roughly the same, the right arm and both legs do not tell the difference between these 2 interpolation methods. The left arm on the other hand, reveals the difference. During the gingerbread man being interpolated using linear interpolation (Figure 2a), it can be observed the left arm has become shorter in the middle steps, and got back to normal length at the end. This is a common distortion issue with linear interpolation. Whereas ARAP shape interpolation has shown a different result (Figure 2b), the left arm remains roughly the same length during the entire interpolation process, making the whole transformation more natural.

5 User Manual

- Z: set the interpolation mode to linear interpolation and reset the current time to 0;
- X: set the interpolation mode to ARAP shape interpolation and reset the current time to 0;
- R: reset the current time to 0;
- Q: exit the program.

6 Acknowledgement

A C++ library 'Eigen' is used (https://bitbucket.org/eigen/eigen/), and this library helps to use matrices and do matrix operations.

7 Conclusion

In this report we have been focusing on interpolating shapes between two 2D meshes, using simple technique linear interpolation, and more advanced technique As-Rigid-As-Possible shape interpolation. The results have shown that linear interpolation is simple to implement, but it suffers from shape distortion during interpolation. On the other hand, ARAP shape interpolation requires more work and more computation, but it could provide a more natural outcome during interpolation. Linear interpolation is suitable for simple interpolation tasks such as blending two images with similar content, whereas ARAP shape interpolation can handle more complex shape interpolation, without suffering too much from shape distortion.

References

Alexa, M., Cohen-Or, D., & Levin, D. (2000). As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on computer graphics and interactive techniques* (pp. 157–164).