Task 2. Mesh Animation

Garoe Dorta-Perez
CM50245: Computer Animation and Games II

March 16, 2015

1 Introduction

Generating a realistic mesh which is a morphing of two given meshes can be a challenging task. We present two approaches to produce a intermediate 3D mesh from conforming source and target 3D meshes.

2 Linear interpolation

Linear interpolation is a simple and intuitive first approach. The new position of a vertex $\mathbf{v}_{it} = [v_{ix}, v_{iy}, v_{iz}]^T$ in an interval $t = \{0, \ldots, 1\}$ is given by

$$\mathbf{v}_{it} = (1-t)\mathbf{p}_i + t\mathbf{q}_i \quad i \in \{1, \ldots, n\},$$

where n is the number of vertices, $\mathbf{p}_i = [p_x, p_y, p_z]^T$ and $\mathbf{q}_i = [q_x, q_y, q_z]^T$ are corresponding vertices in the source and target meshes respectively. Some meshes generated with this method are shown in Figure 1.

3 As rigid as possible interpolation

The previous method produces distorted meshes due to its non-rigid nature. One way to produce better results is to use transform-based techniques [1]. Lets define an affine transformation from $\mathbf{p}_i$ to $\mathbf{q}_i$ such that

$$A\mathbf{p}_i + \mathbf{l} = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \begin{pmatrix} p_{ix} \\ p_{iy} \\ p_{iz} \end{pmatrix} + \begin{pmatrix} l_x \\ l_y \\ l_z \end{pmatrix} = \mathbf{q}_i.$$

We have a matrix A_j and a translation vector $\mathbf{l}_j$ for each triangle, since there are 12 unknowns and each triangle only gives 9 equations, we add an extra point to be able to compute A_j. This new point is computed as $\mathbf{p}_{new} = \mathbf{c}_{tr} + w\mathbf{n}_{tr}$, where $\mathbf{c}_{tr}$ is the triangle centroid, w is the average triangle edge length and $\mathbf{n}_{tr}$ is the triangle normal. Mesh vertices are shared among different triangles, so we cannot apply A_j directly. Instead lets define B_j as the matrix that would transform the vertices consistently. Where A_j and B_j are related by

$$E(t) = \sum_{j}^{J} \| A_j(t) - B_j(t) \|_F^2 ,$$

where $\|\cdot\|_F$ is the matrix Frobenius norm, and the objective is to minimize $E(t)$. If we ignore all the $\mathbf{l}_j$ terms, rename $\mathbf{q}_i$ to treat each vertex in individual vertex in the mesh as $\mathbf{v}_i$ the mesh, and rearrange B such that

$$PB_{col} = \begin{pmatrix} \mathbf{p}_1^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_1^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{p}_1^T \\ \mathbf{p}_2^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_2^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{p}_2^T \\ \mathbf{p}_3^T & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{p}_3^T & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{p}_3^T \end{pmatrix} \begin{pmatrix} b_{11} \\ b_{12} \\ b_{13} \\ b_{21} \\ b_{22} \\ b_{23} \\ b_{31} \\ b_{32} \\ b_{33} \end{pmatrix} = \begin{pmatrix} v_{1x} \\ v_{1y} \\ v_{1z} \\ v_{2x} \\ v_{2y} \\ v_{2z} \\ v_{3x} \\ v_{3y} \\ v_{3z} \end{pmatrix} \text{ and } P^{-1} = \begin{pmatrix} \alpha_1 & 0 & 0 & \delta_1 & 0 & 0 & \lambda_1 & 0 & 0 \\ \alpha_2 & 0 & 0 & \delta_2 & 0 & 0 & \lambda_2 & 0 & 0 \\ \alpha_3 & 0 & 0 & \delta_3 & 0 & 0 & \lambda_3 & 0 & 0 \\ 0 & \beta_1 & 0 & 0 & \epsilon_1 & 0 & 0 & \mu_1 & 0 \\ 0 & \beta_2 & 0 & 0 & \epsilon_2 & 0 & 0 & \mu_2 & 0 \\ 0 & \beta_3 & 0 & 0 & \epsilon_3 & 0 & 0 & \mu_3 & 0 \\ 0 & 0 & \gamma_1 & 0 & 0 & \kappa_1 & 0 & 0 & \pi_1 \\ 0 & 0 & \gamma_2 & 0 & 0 & \kappa_2 & 0 & 0 & \pi_2 \\ 0 & 0 & \gamma_3 & 0 & 0 & \kappa_3 & 0 & 0 & \pi_3 \end{pmatrix},$$

then we have that $B = P^{-1}V$. We would have the above system for each triangle in the mesh, with its own P and B matrices. Writing explicitly the Frobenius terms in $E(t)$, and expressing B as a linear combination of $\mathbf{v}_i$, we get

$$\begin{aligned} E(t) = \{ & a_{11}^2 + a_{12}^2 + \ldots + a_{33}^2 + \alpha_1^2 v_{1x}^2 + \ldots + \pi^2 v_{3z}^2 - 2a_{11}\alpha_1 v_{1x} \\ & - 2a_{11}\delta_1 v_{2x} - 2a_{11}\lambda_1 v_{3x} - \ldots - 2a_{33}\pi_3 v_{3z} + 2\alpha_1 v_{1x}\delta_1 v_{2x} \\ & + 2\alpha_1 v_{1x}\lambda_1 v_{3x} + \ldots + 2\kappa_3 v_{2z}\pi_3 v_{3z} \}_{j=1} + \ldots + \{\cdots\}_{j=J}, \end{aligned}$$

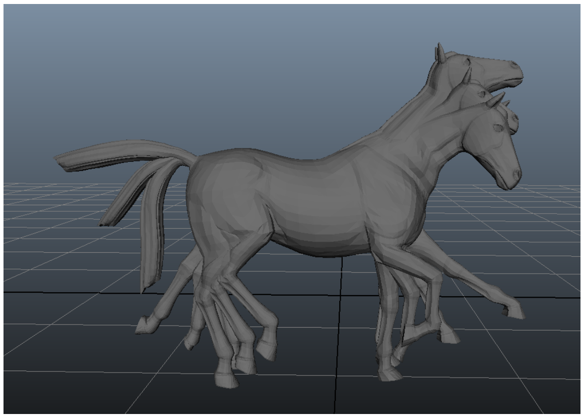
where we have a $\{\cdots\}_j$ term for each triangle. If we set the transformation of $\mathbf{v_1}$ by linear interpolation, and defining $\mathbf{u}(t)^T = \big(1, v_{2x}, v_{2y}, v_{2z}, \ldots, v_{mx}, v_{my}, v_{mz}\big)$, where m is the number of vertices, we can stack the terms in a quadratic matrix form

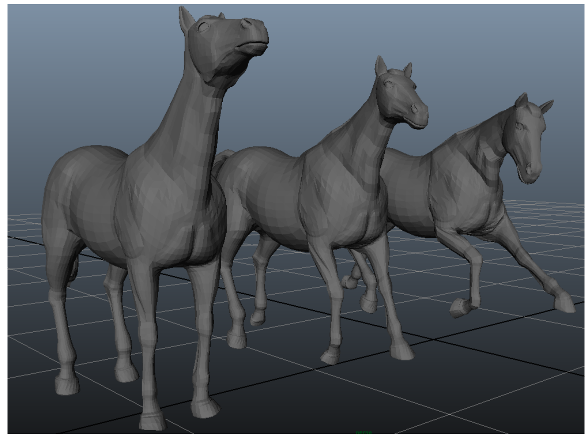
$$E(t) = \mathbf{u}^T \begin{pmatrix} c & \mathbf{g}(t)^T \\ \mathbf{g}(t) & H \end{pmatrix} \mathbf{u},$$

where c are the constant coefficients, $\mathbf{g}$ is a vector with the linear terms and H is matrix with the quadratic and mixed terms. Note that, since $\mathbf{v}_1$ is known, the terms were it appears, become constants or linear in $\mathbf{v}_j$ for $j \neq 1$. Setting the gradient to zero,

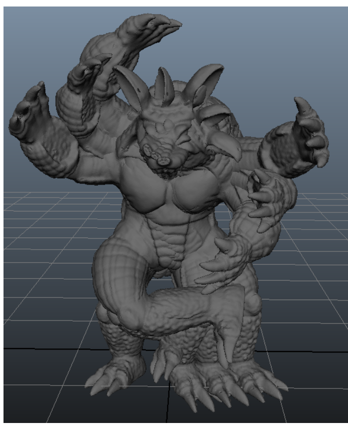
$$H\mathbf{u}(t) = -\mathbf{g}(t) \;\rightarrow\; \mathbf{u}(t) = -H^{-1}\mathbf{g}(t).$$

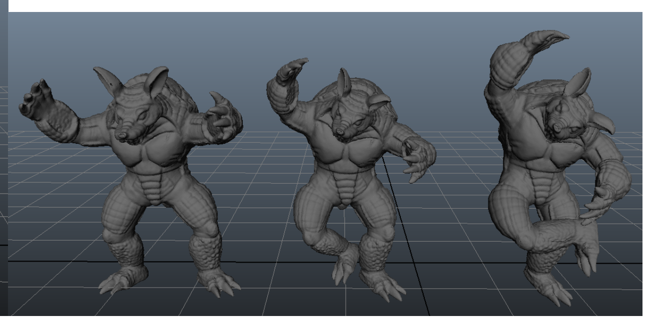
Since H is independent on t, for each iteration we only need to compute the linear interpolation of $\mathbf{v}_1$ and the new $\mathbf{g}(t)$ to solve the system.





(a) fig 3 (b) fig 4





(c) fig 1 (d) fig 2

Figure 1: Horse and armadillo meshes, morphed mesh is in the middle.

4 Results and conclusions

Figure 1 shows that applying a vertex based linear interpolation, rescales the interpolated mesh in unnatural positions. For example, the horse head and tail shapes in Figure 1a are smaller than their counterparts in the source and target shape. Applying the as-rigid-as-possible interpolation produces better results, because it interpolates the transformation matrices instead of the vertex positions.

References

[1] Marc Alexa, Daniel Cohen-Or, and David Levin. As-rigid-as-possible shape interpolation. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00*, pages 157–164, 2000.