Zoo World: Skeleton-Based ARAP Deformation On Animal

Wei-Lin Pai ge62vaz@mytum.de
Wen-Shuo Hsu ge48dur@mytum.de
You-Shin Tsai ge84las@mytum.de
Yu-Heng Lo ge48cul@mytum.de

Abstract

This project implements the As-Rigid-As-Possible (ARAP) deformation algorithm, aided by pseudo skeletons, on custom animal meshes. A pseudo skeleton is manually generated to an animal mesh as a deformation proxy, surrounding it with pseudo-meshes. The skeleton is attached to the mesh through Linear Blend Skinning. The ARAP method is combined with regularization energy and flip-flop optimization. The whole process is tested on a bear model, providing vivid results with surface mesh and real-time deformation in the pseudo mesh.

1. Introduction

In our childhood, we cherished the memories of playing a web game where we could acquire animals and watch them express themselves in a virtual zoo. We were so impressed by the accomplishment of having so many animals in our own zoo. However, the animation of animals is only 2D and not vivid enough, which makes us not really enjoy the game. Therefore, we aspire to one day make our own masterpiece that goes beyond the virtual zoo to create a more realistic, perfect zoo in 3D.

In 3D scenes, shape deformation refers to the process of altering the shape of an object or surface while preserving its overall structure and topology. This technique is commonly used in computer graphics and game fields to create realistic animations and deformations on characters. There are some commonly used methods, such as linear blend skinning (LBS) and dual quaternion skinning (DQS). However, these traditional methods often produce undesirable artefacts like volume loss, collapsing regions, or shearing effects. These drawbacks motivate the development of ARAP-based approaches, which aim to find a balance between rigidity and conformality.

We aim to implement As-Rigid-As-Possible (ARAP) on animal models to accomplish shape deformation and editing, which allow us to generate vivid animal animation.

2. Related Work

As-Rigid-As-Possible (ARAP) is a powerful technique in computer graphics and geometric modelling, used to achieve smooth and natural deformations of 3D surfaces. ARAP focuses on preserving the local rigidity of the surface during deformations. By minimizing distortion, ARAP ensures smoother and more natural shape transformations, making it highly suitable for various applications, including character animation and shape editing.

Sorkine et al. [5] devise a simple iterative mesh editing scheme, which based on non-linear energy formulation, leads to detail-preserving and intuitive deformations. However, the computation is hard to be finished in real-time when the model has a large amount of vertices and meshes. To eliminate the is limitation, researchers recently apply ARAP on proxy geometric to decouple the complexity of computation from the quality of model geometric. In [2], the authors introduce a hybrid approach that combines surface-based deformation with cage-based deformation to simplify the original mesh. To optimize the computing performance, Zollhöfer et al. [9] present a volumetric lattice-based approach (LARAP) and propose a data-parallel implementation on the GPU. In addition to the computation problem, classical surface-based ARAP may cause undesired artefacts due to a lack of volume-preserving constraints. Zhang et al. [6] proposed a volume-aware deformation method which leverages the skeleton information to keep the volume. In [8], the authors apply ARAP on a pseudo skeleton which has a small number of DOFs and allows the user to stretch and twist.

3. Skeleton and Pseudo Mesh

3.1. Skeleton Generation

To simplify calculation complexity, a skeleton was introduced to our model as a deformation proxy. The skeleton was generated and assigned to proper positions relative to the mesh manually with Blender.

3.2. Pseudo-Meshes

As Zollhöfer et al. [7] have suggested, the skeleton cannot directly be used with ARAP. With skeleton only, the

rotation of the joint of the skeleton is under-determined. A pseudo mesh is therefore to be generated before applying ARAP. The mesh was generated by expanding the skeleton from an edge into a rectangular box with a diagonal length of 2δ.

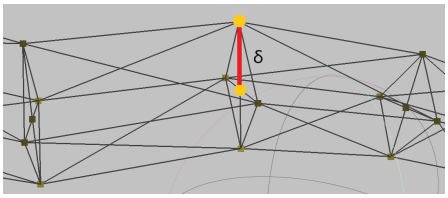


Figure 1. Constitution of Pseudo Mesh

At joints with more than one degree of freedom, for example, the joint between the pelvis and limbs, we found out that without actually connecting the limbs to the spine, the ARAP algorithm will collapse the limbs to achieve lower energy. Therefore, we connected the limbs to the spine and formed a fully watertight mesh.

The connection of the pelvis and limbs also came with a potential drawback. The actual positions of limbs differed from the control point that represents limbs. However, the distance is shorter than δ and, according to our results, does not have a significant effect on the outcome.

3.3. Linear Blend Skinning

Linear Blend Skinning, commonly known as LBS, is a popular and easy method to attach a skeleton, composed of bones, to a mesh. The algorithm assigns weights to each bone based on their influence on specific vertices in the mesh. The deformation of the mesh is calculated by blending the transformation of the bones based on their assigned weight.

In the project, we utilized the code from the 'Pinocchio' [1] to map the skeleton with the mesh. With the given bone transformation, the deformation mesh can be calculated by applying the following formula on each vertex v:

$$v' = \sum_{j \in H} w_j(v) * T_j \begin{pmatrix} v \\ 1 \end{pmatrix} \tag{1}$$

The T_j is the transformation matrix of bone j. The $w_j(v)$ is the influence of bone j on vertex v. The v' will be the new position of v after calculation. The result is in section 6.2.

4. ARAP Deformation

New poses can be adjusted using a handle-based manipulation interface. Each vertex can be selected as a handle point and acts as a constraint to the selected vertex. The skeleton-based deformation proxy provides the users with real-time feedback. Based on the ARAP [5] method, the optimal deformation is cast as a non-linear deformation energy minimization problem:

$$E_{total}(S) = E_{reg}(S) \tag{2}$$

4.1. Regularization Energy E_{reg}

The regularizer in the ARAP method [5] is to minimize the local deviation from rigidity. The local rigidity at the i-th joint of pseudo mesh is measured as:

$$E_{reg}(\hat{n_i}) = \sum_{j \in N(i)} w_{ij} \|(\hat{n_i} - \hat{n_j}) - R_i(n_i - n_j)\|^2 \tag{3}$$

The $\hat{n_i}$ are the unknown joint positions, the n_i are the corresponding positions before deformation and R_i is the unknown rotation matrix that maps undeformed vertices to deformed vertices. $N(i)$ is a set of indices for neighbor vertices to vertex i.

The choice of per-edge weight w_{ij} could measure the weight the neighbours have on the target vertex. The weight should compensate for non-uniformly shaped cells and prevent discretization bias. In this project, we use the cotangent weight formula for w_{ij} in [4]

$$w_{ij} = \frac{1}{2}(\cot \alpha_{i,j} + \cot \beta_{i,j}) \tag{4}$$

where $\alpha_{i,j}$ and $\beta_{i,j}$ are the angles opposite of the mesh edge (i, j) (in the boundary edge, only one angle exists), as figure(2) shows.

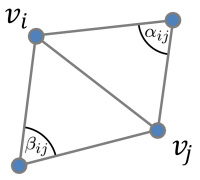


Figure 2. $\alpha_{i,j}$ and $\beta_{i,j}$ in triangular vertex

The global regularization energy is defined as the sum of the local rigidity:

$$E_{reg}(S) = \sum_{i=1}^{|J|} E_{reg}(\hat{n_i}) \tag{5}$$

4.2. Flip-Flop Optimization

The deformation energy E_{total} is non-linear in the unknown rotations R_i and quadratic in the unknown joint positions n_i. To solve this, we utilize a flip-flop optimization

scheme from [5]. For a given fixed set of rigid transformations, we find positions n' that minimize $E(S)$. Then, we optimize the rigid transformation R_i that minimizes $E(S)$ for the given set of positions n'.

4.2.1 Optimize n'_i

In order to compute optimal vertex positions from given rotations, we compute the gradient of $E(S)$ with respect to the positions n'. The partial derivatives w.r.t. n'_i can be simplified as:

$$\frac{\partial E(S)}{\partial n'_i} = \sum_{j \in N(i)} 4w_{ij} \left((n'_i - n'_j) - \frac{1}{2}(R_i + R_j)(n_i - n_j) \right) \tag{6}$$

Let the partial derivative to zero w.r.t each n'_i we arrive the following linear system equation:

$$\sum_{j \in N(i)} w_{ij}(n'_i - n'_j) = \sum_{j \in N(i)} \frac{w_{ij}}{2}(R_i + R_j)(n_i - n_j) \tag{7}$$

The linear combination on the left-hand side is the discrete Laplace-Beltrami operator applied to n'.

The system equation can be written as

$$Ln' = b \tag{8}$$

where b is a vector whose i-th row contains the right-hand side expression from (7). Since L is a symmetric positive definite, the sparse Cholesky LDLT decomposition is an efficient solver. As we have mentioned, the user inputs are assumed as strong constraints. In the simplest form, those can be expressed as some fixed positions:

$$n'_j = c_i, i \in C \tag{9}$$

where C is the set of indices of the constrained vertices. Considering these constraints simply means substituting the corresponding variables, erasing respective rows and columns from L and updating the right-hand side with c_i

4.2.2 Optimize R_i

It can be optimized by solving the maximize equation from [5]:

$$\arg\max_{R_i} Tr(R_i \sum_j w_{ij} e_{ij} e'^T_{ij}) \tag{10}$$

where $e_{ij} = n_i - n_j$. R_i can then be derived from singular value decomposition of $S_i = U_i \sum_i V_i^T$,:

$$R_i = V_i U_i^T \tag{11}$$

where covariance matrix S_i:

$$S_i = \sum_{j=\in N(i)} w_{ij} e_{ij} e'^T_{ij} = P_i D_i P'^T_i \tag{12}$$

Input	Action
Mouse click	Select skeleton point
Mouse move	(a) move camera (b) move point
B	Display surface mesh
C	Add control point
M	Change mode of mouse movement
R	Remove control point
T	Make surface mesh transparent
X, Y, Z	Change control point movement axis

Table 1. Key inputs and their corresponding actions.

D_i is a diagonal matrix containing the weight w_{ij}, P_i is the $3 \times |N(n_i)|$ containing e'_{ij}s as its columns, and similarly for P'_i.

To sum up, the flip-flop optimization proceeds as follows. First the weight w_{ij} are pre-computed. With the initial guess n'_0, the rotations R_i are estimated. New positions n'_1 are obtained by solving (7) with R_i plugged in the right-hand side. Then the optimization is continued by re-computing rotations R_i and using them to define the new right-hand side for the linear system, and so on.

5. Interface

The interface serves as the connection between the user's action and the result deformation. To simplify the process of building the interface, we utilize the libigl [3] library to (a) translate the hardware actions into software actions, and (b) display the mesh and the skeleton, and their deformations.

We define the input actions as Table 1 shows. The interface is designed so that ARAP deformation is conducted simultaneously when the user moves the control point with the mouse, which is also simultaneously displayed in real-time.

The typical process of using the interface is described as follows. When the user starts the program, the interface will display the loaded animal mesh. The user may then wish to select a point in the skeleton as a control point. The user can press B to change the display mode to skeleton mode, which only shows points of the skeleton, or T, which makes the surface mesh transparent. After selecting the control points, the user can press M to switch to ARAP movement mode, where they can move one of the control points to perform ARAP deformation. After that, they can press B again to see the result mesh.

6. Result

Figure 3 shows our method. We first build the skeleton and the pseudo skeleton from the original bear mesh, and bind them all together. When the user moves the joint through our interface, we then apply the ARAP algorithm

to the pseudo skeleton. The skeleton and the mesh update sequentially after the update of the pseudo skeleton.

The code and data for this project are available on GitHub. (1) ARAP[1]: the main program for the user to control (2) LBS[2]: calculate the bone weights for each vertices

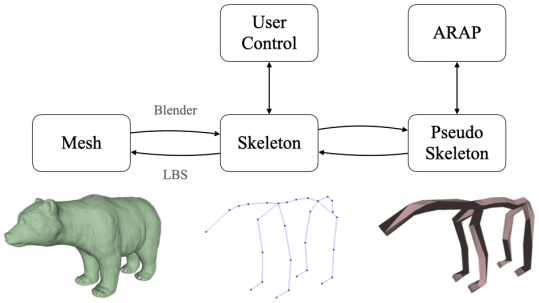
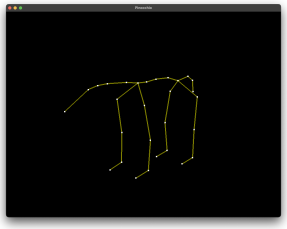
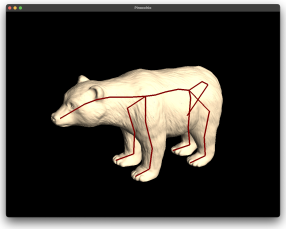


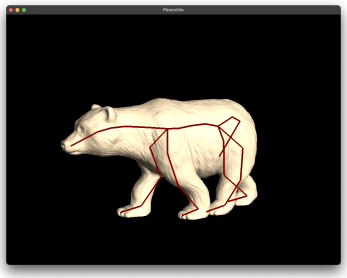
Figure 3. Method of our program



(a) Skeleton of bear



(b) Skeleton attached to bear mesh



(c) Bear mesh with transformed skeleton

Figure 5. LBS Result with Customized Skeleton

6.1. ARAP on customized mesh and skeleton

The above-created bear mesh and skeleton are inputted into the interface as figure 4 shows. Once the user selects the skeleton's control points and moves a vertex, the ARAP deformation is conducted simultaneously.

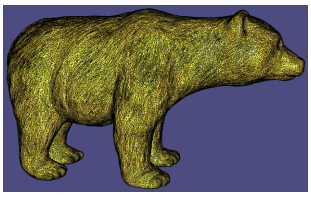
6.2. LBS with customized skeleton

Since the code from [1] only works well on human skeletons, we customized our bear skeleton as figure 5a. The skeleton is defined by the position and connection of joints in file `pinocchino_skeleton.txt`. Each line in the file corresponds to a joint and is of the form of: "`idx jointName x y x prev_jointName`".

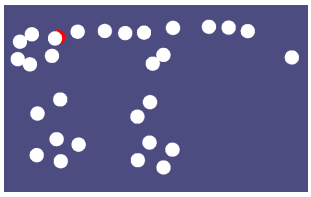
Next, we attached the customized skeleton to the bear mesh as figure 5b shows. After the code runs, it outputs the bone weights to file `attachment.out` and skeleton to file `skeleton.out`. Each line in `attachment.out` represents a vertex in the mesh, and it includes the bone weights for all the bones according to the sequence of bone formation. Each line in `skeleton.out` represents a bone in the skeleton, including the index of a joint, the position of the joint, and another joint index the bone connected to. For example, there are a total of 29 joints and 28 bones in our customized bear skeleton.

To verify the availability of bone weight, we tested it by giving leg bones a new transformation. The original bear mesh deforms reasonably according to the bone weight we obtain. The deformation result shows in figure 5c.

[1]https://github.com/williampai0704/Skeleton_Based_ARAP
[2]https://github.com/tsai-you-shin/Pinocchio



(a) Surface mesh of bear



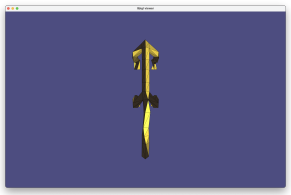
(b) Vertices of bear skeleton

Figure 4. Customized bear mesh

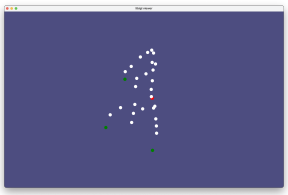
6.3. ARAP Demo

Figure 6 shows the screenshot of the ARAP demo on pseudo skeleton (149 vertices). After launching the interface, the pseudo skeleton will be shown on the screen, as figure 6a shows. User can drag with mouse to adjust the viewing angle. By pressing "B", the pseudo bone will show up, like figure 6b. User can select any bone point as control point by pressing "C". To make the control point move, user needs to press "M" to switch to movement mode. The point movement can switch between xyz axis by pressing "X","Y", and "Z". The deformation result is as figure 6c. ARAP have real-time performance with only pseudo skeleton.

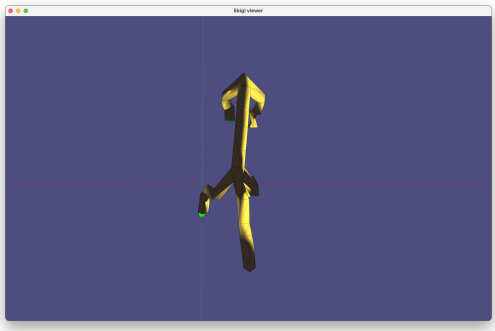
Figure 7 shows the screenshot of the ARAP demo on pseudo skeleton (149 vertices) and bear surface mesh (49998 vertices). Figure 7a shows the surface mesh of bear. By pressing "T", the surface mesh will become transparent and one can see the pseudo skeleton inside, like figure 7b. Figure 7c and 7d show the result after selecting head as control point. The bear head moves and the other part of mesh deform smoothly, showing that the ARAP algorithm works



(a) Pseudo skeleton of bear

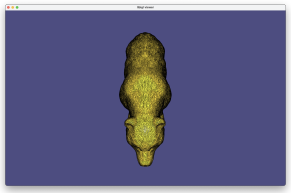


(b) Pseudo bone points of bear

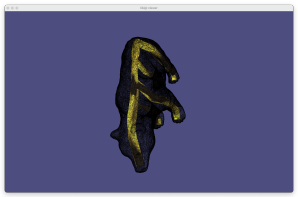


(c) Deformation result on pseudo skeleton

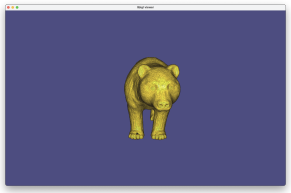
Figure 6. ARAP demo on pseudo skeleton



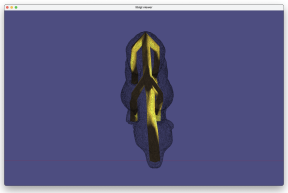
(a) Pseudo surface mesh of bear



(b) Transparent surface with pseudo skeleton



(c) ARAP deformation result (with head move)



(d) Result with surface mesh and pseudo skeleton

Figure 7. ARAP demo on bear mesh

well. However, the deformation does not perform real-time with surface mesh. The possible reason will be discussed in analysis section.

7. Analysis

The result above is tested on a MacBook Pro running an Apple M2 Pro chip. It allows a real-time deformation on pseudo mesh, which contains 149 vertices and 348 faces, but the deformation process becomes slow when deforming the surface mesh, which contains 49,998 vertices and 99,992 faces. The reason might be that the high amount of vertices processing is a heavy workload on our devices.

8. Conclusion and Future Work

In this project, we accomplished the goal of animating an animal through pseudo-skeleton-based ARAP deformation. The process is separated into pseudo-skeleton generation and ARAP deformation, connected by Linear Blend Skinning. We tested this process on a bear model, creating our own bear skeleton and performing ARAP, providing good visual results. We are confident that this project can be applied to other animal meshes, in the end forming a virtual Zoo World. Also, by specifying the skeleton transformation, the animations for animal can be generated easily.

So far the pseudo skeleton is generated manually since there isn't a automatic process for generating pseudo bones and pseudo joints on different 3d meshes. We hope that in the future the pseudo skeleton can automatically generate so that the whole skeleton-based ARAP can be performed without manual interrupt.

References

[1] Ilya Baran and Jovan Popović. Automatic rigging and animation of 3d characters. *ACM Transactions on graphics (TOG)*, 26(3):72–es, 2007. 2, 4

[2] Péter Borosán, Reid Howard, Shaoting Zhang, and Andrew Nealen. Hybrid Mesh Editing. In H. P. A. Lensch and S. Seipel, editors, *Eurographics 2010 - Short Papers*. The Eurographics Association, 2010. 1

[3] Alec Jacobson, Daniele Panozzo, et al. libigl: A simple C++ geometry processing library, 2018. https://libigl.github.io/. 3

[4] Ulrich Pinkall and Konrad Polthier. Computing discrete minimal surfaces and their conjugates. *Experimental mathematics*, 2(1):15–36, 1993. 2

[5] Olga Sorkine and Marc Alexa. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, volume 4, pages 109–116. Citeseer, 2007. 1, 2, 3

[6] Shaoting Zhang, Andrew Nealen, and Dimitris Metaxas. Skeleton Based As-Rigid-As-Possible Volume Modeling. In H. P. A. Lensch and S. Seipel, editors, *Eurographics 2010 - Short Papers*. The Eurographics Association, 2010. 1

[7] M Zollhöfer, A Vieweg, J Süßmuth, and G Greiner. Pseudo-skeleton based arap mesh deformation. 1

[8] Michael Zollhöfer, A. Vieweg, Jochen Süßmuth, and Günther Greiner. Pseudo-skeleton based arap mesh deformation. 2013. 1

[9] Michael Zollhöfer, Ezgi Sert, Günther Greiner, and Jochen Süßmuth. GPU based ARAP Deformation using Volumetric Lattices. In Carlos Andujar and Enrico Puppo, editors, *Eurographics 2012 - Short Papers*. The Eurographics Association, 2012. 1