CPSC 524 - Geometric Modeling - Course Project Report
Subspace Mesh Deformation with Constrained Nonlinear Least Squares Energy

Minchen Li*
University of British Columbia

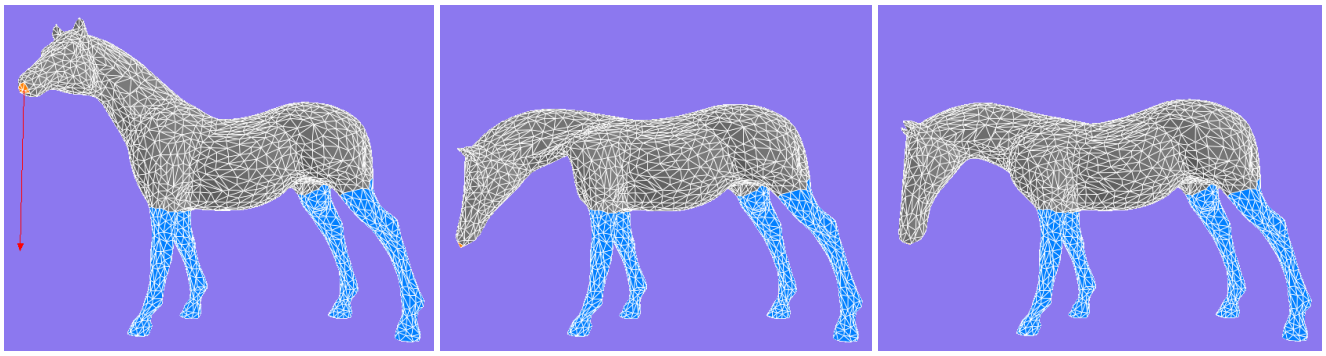


Figure 1: *Mesh deformation using ARAP (middle) and our subspace mesh deformation method with constrained nonlinear least squares energy (right). The blue vertices are fixed, and the orange vertices are moving through the red arrow downwards and backwards as shown in the left picture. It's obvious that ARAP generates unnatural neck shape in this extreme deformation case, while our method gives a better solution although the head is slightly elongated.*

Abstract

Our project aims at implementing a basic version of subspace mesh deformation method with constrained nonlinear least squares energy following [Huang et al. 2006]. This report will cover the key points in [Huang et al. 2006] with many untouched mathematical derivations and implementation details, which could be served as a supplemental material to the original paper for potential readers. Besides, some experiments will be conducted and the results with comparison and analysis will be demonstrated.

1 Specification

The input meshes will be a refined closed-manifold mesh with a relativly large number, n, of vertices and a closed-manifold coarse control mesh[1] with a much smaller number, m, of vertices. It is required that the refined mesh needs to lie exactly inside the interior of the coarse control mesh.

Given the input meshes, a mapping $W \in R^{3n\times 3m}$ between the coarse control mesh and the refined mesh will be constructed using the mean value interpolation method [Ju et al. 2005; Floater 2003]. Then W will be used to map the energy function defined on the refined mesh to the subspace of the coarse control mesh.

The energy function works on preserving surface details, mesh volume, and some objective vertex positions, where the objective vertex positions are defined via user interactions. Note that here this scheme is different from free-form deformation (FFD) [Sederberg and Parry 1986] where energy terms are directly defined on the coarse control mesh. They have different mathematical forms.

*e-mail:minchenl@cs.ubc.ca

[1]The coarse control mesh could be constructed by applying the progressive convex hull construction algorithm in [Sander et al. 2000] on the refined mesh, or by using a modeling software. This will not be a part of the implementation in this project

Finally, to solve for vertex coordinates for the deformed mesh, the energy function will be minimized in the subspace, which will provide better numerical stability, faster convergence, and less runtime and memory cost than that in the original space.

Through this project, we can learn the space reduction idea, which is very useful in many engineering scenarios. Since the mesh deformation problem here is formulated as a nonlinear least squares problem, getting experience on applying Gauss-Newton method with backtracking line search is also a big benefit.

The rest of the report is organized as follows: §2 formulate the mesh deformation problem as a constrained nonlinear least squares problem (§2.1), which will be solved by Gauss-Newton iterations (§2.2) with backtracking line search (§2.3). Then the subspace solve will be discussed in §2.4, so does the space reduction idea (§2.5). In order to provide more useful details on implementation, the computation and verification of derivatives is presented in (§2.6) at length. The experiments and results will be demonstrated in §3 where our subspace solve will be compared with the full space solve (§3.1) and also the ARAP [Sorkine and Alexa 2007] (§3.2, §3.3). Finally, we will conclude our work and provide possible future directions in §4.

2 Formulation and Implementation

2.1 A Constrained Nonlinear Least Squares Problem

With the rotation invariant nonlinear Laplacian constraint $\mathcal{L}X = \hat{\delta}(X)$ and position constraint $\Phi X = \hat{V}$ being the soft constraints, and with volume constraint $\psi(X) = \hat{v}$ being the hard constraint, we can formulate our mesh deformation problem as a constrained nonlinear least squares problem [Huang et al. 2006]

$$\min_X \frac{1}{2}||f(X)||^2 \text{ subject to } g(X) = 0 \qquad (1)$$

where

$$f(X) = \begin{bmatrix} \mathcal{L} \\ \Phi \end{bmatrix} X - \begin{bmatrix} \hat{\delta}(X) \\ \hat{V} \end{bmatrix}$$
$$g(X) = \psi(X) - \hat{v} \tag{2}$$

Here $X \in R^{3n}$ is the vertex coordinates of the refined meshes, $\mathcal{L}$ is the laplacian matrix constructed using the cotangent weights [Desbrun et al. 1999] of the undeformed refined mesh, $\hat{\delta}(X)$ is the rotation invariant expression of the refined meshes, and $\psi(X) = \frac{1}{6}\sum_{T_{ijk}}(x_i \times x_j) \cdot x_k$ is the volume of the refined meshes. $\hat{\delta}(X)$ and $\psi(X)$ are nonlinear functions of X. The position constraint will be defined by user interaction, and the solved X which gives the minimal objective function value will be taken as the vertex coordinates of the deformed refined mesh.

2.2 Gauss-Newton Iterations

To solve the problem, Gauss-Newton iterations [Wedderburn 1974] are applied. We linearly approximate $f(X+h) \approx l(h) \equiv f(X) + J_f(X)h$ at each iteration and solve

$$\min_h \frac{1}{2}||l(h)||^2 \text{ subject to } g(X+h) = 0 \tag{3}$$

By locally linearizing $g(X+h) \approx g(X) + J_g(X)h$ and applying Lagrange multipliers [Bellman 1956] with Newton's method [Kelley 2003], we can express the local update that minimizes Eq.3 as:

$$h = -(J_f^T J_f)^{-1}(J_f^T f + J_g^T \lambda)$$
$$\lambda = -(J_g(J_f^T J_f)^{-1} J_g^T)^{-1}(g - J_g(J_f^T J_f)^{-1} J_f^T f) \tag{4}$$

where $J_f \equiv J_f(X) = \begin{bmatrix} \mathcal{L} - J_{\hat{\delta}}(X) \\ \Phi \end{bmatrix}$ and $J_g \equiv J_g(X) = \nabla_X \psi(X)$. Thus starting from an initial X_0, we can solve Eq.1 iteratively by computing the update h_k from Eq.4 (assuming $X = X_{k-1}$) and then setting $X_k = X_{k-1} + \alpha h_k$, where α is a small constant that can be found by line search.

2.3 Backtracking Line Search with Armijo Condition

In each iteration k after we solve for h_k, we set $p_k = h_k/||h_k||$, $\alpha_k = ||h_k||$ and test whether α_k satisfy

$$\frac{1}{2}||f(X_s)||^2 + \lambda g(X_s) \leq \frac{1}{2}||f(X_{k-1})||^2 + \lambda g(X_{k-1}) + c_1 m \tag{5}$$

where

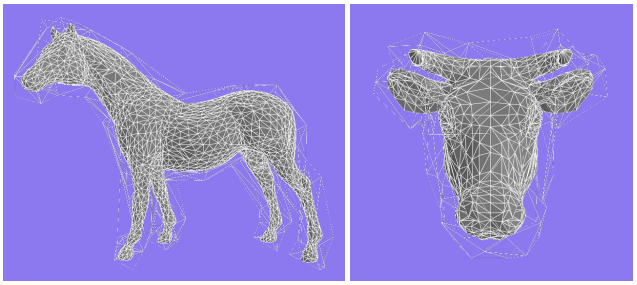
$$X_s = X_{k-1} + \alpha_k p_k$$
$$m = \alpha_k p_k^T((J_f(X_{k-1}))^T f(X_{k-1}) + \lambda(J_g(X_{k-1}))^T) \tag{6}$$

If the above Armijo-Goldstein condition [Armijo 1966] is not satisfied, we set α_k to $c_2\alpha_k$, and test the condition again iteratively until it is satisfied or $\alpha_k p_k$ becomes too tiny with respect to the bounding volume size of X_{k-1}. Here c_1 and c_2 are two constants usually set to $c_1 = c_2 = 0.5$ when Armijo first publish it, or $c_1 = 1.0 \times 10^{-4}$, $c_2 = 0.9$ in Wolfe condition [Wolfe 1969].

Note that in Eq.5, $m = \alpha_k p_k^T(\partial(\frac{1}{2}||f(X)||^2 + \lambda g(X))/\partial X)$. This backtracking line search method ensures that in each iteration, the Lagrange function value decreases sufficiently, which is necessary in most of the Newton-like methods since the linearly approximated local minimum may even not be a descent update of the original nonlinear objective.

2.4 Solve in Subspace

Solving Eq.1 with iterative methods will run into serious problems with slow convergence and numerical instability. Often the stability problem is so severe that the iterations do not converge. The two dominating causes for the instability are the large condition number $\kappa(J_f^T J_f)$ of the matrix $J_f^T J_f$ and the nonlinearity of $\hat{\delta}(X)$. The subspace deformation technique is designed to address these issues.



(a) *Horse* **(b)** *Cow head*

Figure 2: *The coarse control meshes used in this report constructed using 3ds Max.*

With a coarse control mesh around the original mesh (e.g., Fig.2), the deformation energy and the hard constraints are then projected onto the control mesh vertices using mean value interpolation [Ju et al. 2005; Floater 2003]. Let the control mesh vertices $P \in R^{3m}$ be related to original mesh vertices $X \in R^{3n}$ through $X = WP$. After projection we perform energy minimization in the control mesh subspace as follows:

$$\min_P \frac{1}{2}||f(WP)||^2 \text{ subject to } g(WP) = 0 \tag{7}$$

which will change the Gauss-Newton iteration update to

$$h_P = -(W^T J_f^T J_f W)^{-1}(W^T J_f^T f + W^T J_g^T \lambda)$$
$$\lambda_P = -(J_g W(W^T J_f^T J_f W)^{-1} W^T J_g^T)^{-1}\eta \tag{8}$$
$$\eta = (g - J_g W(W^T J_f^T J_f W)^{-1} W^T J_f^T f)$$

This can just be derived by substituting J_f and J_g with their subspace projections $J_f W$ and $J_g W$ into the full space update Eq.4. The line search will also need to be conducted in the subspace.

In fact, this subspace solve is just equivalent to adding one more hard equality constraint $X = WP$ to the original problem, so that the degree of freedom decreases from $|X|$ to $|P|$. Since $|P|$ is much smaller than $|X|$, the linear systems to be solved at each iteration are relatively small. Furthermore, using the smoothness of the mean value coordinates we can show that, for a properly constructed control mesh, $\kappa(W^T J_f^T J_f W)$ has magnitudes smaller than $\kappa(J_f^T J_f)$ and the nonlinearity of $\hat{\delta}(WP)$ with respect to P is significantly reduced from that of $\hat{\delta}(X)$ with respect to X.

Most importantly, this technique does not simply apply constraints and solve the deformation on the coarse mesh P and interpolate back the results to the original mesh X; this naive approach would certainly not preserve mesh properties on the original mesh.

One thing worth noticing is that the time and memory complexity reduction of subspace solve compared to the original solve depends on a more strict dimensionality condition, just $|P| < |X|$ is not enough since W is dense and so does all the subspace systems, while in original space the systems are sparse.

2.5 The Space Reduction Idea

The design of the subspace deformation solver is based on two observations: (1) the key in gradient domain deformation is to deform the low frequency coarse shape while maximally preserving the high frequency features such as surface and skeleton details. Thus, in the view of spectral analysis via singular value decomposition, the deformation is mostly performed in a subspace defined by low frequency features. (2) If the subspace deformation formulation is robust and only involves a small number of variables, then the (inexact) Gauss-Newton method can converge rapidly and hence we can meet the interactive deformation requirements.

To determine a quality subspace and its parameterization, ideally, one can use spectral analysis to capture the subspace of low frequency features: Consider a deformation $X = X_0 + D$, where X_0 denotes the original mesh position and D is the desired displacement of the deformation. Let $\mathcal{L}$ be the Laplacian matrix. Then the changes in the differential coordinates is $\mathcal{L}X - \mathcal{L}X_0 = \mathcal{L}D$. Let $\mathcal{L} = USV^T$ be the SVD of $\mathcal{L}$, and $D = \sum_j d_j V_j$ be an expansion using the singular vectors V_j in V. Then $||\mathcal{L}D||^2 = \sum_j (d_j s_j)^2$, where s_j are the j-th singular values. In order to preserve the high frequency surface details, D should lie in a subspace formed by the set of singular vectors with small singular values. So one can form a reduced subspace in which energy minimization is performed in the subspace formed by the singular vectors in V with small singular values.

In practice, it could be expensive to compute the SVD of $\mathcal{L}$ for large meshes. So the alternatives would be to apply mesh simplification techniques [Sander et al. 2000] or to use modeling software.

2.6 Computation and Verification of Derivatives

Computing $J_{\hat{\delta}}(X)$: We will evaluate $J_{\hat{\delta}}(X)$ analytically by applying chain rule:

$$J_{\hat{\delta}}(X) = \frac{\partial \hat{\delta}(X)}{\partial d(X)} \frac{\partial d(X)}{\partial X}$$

Since $\hat{\delta}_{:i}(X) = (\hat{\gamma}_i / ||d_{:i}(X)||_2) d_{:i}(X)$, we have

$$\frac{\partial \hat{\delta}(X)}{\partial d(X)} = blockDiag(\frac{\partial \hat{\delta}_{:i}(X)}{\partial d_{:i}(X)}) \, , i = 1, 2, ..., n$$

(Since notation X_k has been used to refer to vector X in k-th Gauss-Newton iteration, we use $X_{:i}$ in this report to refer to the i-th element/dimension of a vector.)

Let $d_{:i}(X) \equiv (x, y, z)$, we have

$$\frac{\partial \hat{\delta}_{:i}(X)}{\partial d_{:i}(X)} = \frac{\hat{\gamma}_i}{(x^2+y^2+z^2)^{\frac{3}{2}}} \begin{bmatrix} y^2+z^2 & -xy & -xz \\ -xy & x^2+z^2 & -yz \\ -xz & -yz & x^2+y^2 \end{bmatrix}$$

For $\partial d(X)/\partial X$, we also consider per vertex $\partial d_{:i}(X)/\partial X$:

$$\frac{\partial d_{:i}(X)}{\partial X} = \sum_j^{\mathcal{N}(i)} \mu_{ij} \frac{\partial((x_{j-1} - x_i) \times (x_j - x_i))}{\partial X} \in R^{3 \times 3n}$$

(In this report, we assume gradients to be row vectors, while all the other vectors to be column vectors.) and we can construct it as:

$$\frac{\partial d(X)}{\partial X} = [(\frac{\partial d_{:1}(X)}{\partial X})^T, (\frac{\partial d_{:2}(X)}{\partial X})^T, ..., (\frac{\partial d_{:n}(X)}{\partial X})^T]^T$$

As we know, cross product can be written in matrix-vector product form:

$$a \times b = [a]_\times b = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} b$$

thus we can compute the derivative in a more concise form:

$$\begin{aligned} &\frac{\partial((x_{j-1} - x_i) \times (x_j - x_i))}{\partial X} \\ =&\frac{\partial(x_{j-1} \times x_j + x_i \times x_{j-1} + x_j \times x_i)}{\partial X} \\ =&[..., \mathbf{0}_{3,3}, [x_j]_\times - [x_{j-1}]_\times, \mathbf{0}_{3,3}, \\ &..., \mathbf{0}_{3,3}, [x_i]_\times - [x_j]_\times, \mathbf{0}_{3,3}, \\ &..., \mathbf{0}_{3,3}, [x_{j-1}]_\times - [x_i]_\times, \mathbf{0}_{3,3}, ...] \end{aligned} \tag{9}$$

Computing $J_g(X)$: For $J_g(X)$, we have:

$$J_g(X) = \nabla_X \psi(X) = \frac{1}{6} \sum_{T_{ijk}} \frac{\partial (x_i \times x_j) \cdot x_k}{\partial X}$$

where we first calculate the gradient per triangle as follows and add them up:

$$\begin{aligned} \frac{\partial (x_i \times x_j) \cdot x_k}{\partial X} =&[..., 0, x_k^T [-x_j]_\times, 0, \\ &..., 0, x_k^T [x_i]_\times, 0, \\ &..., 0, (x_i \times x_j)^T, 0, ...] \end{aligned} \tag{10}$$

Verifying $J_{\hat{\delta}}(X)$ and $J_g(X)$: Since the above analytic derivations bear certain complexity, it's necessary to ensure that they are coded correctly. We verify the computation by comparing them to their corresponding finite difference results.

According to finite difference methods [Ascher and Greif 2011], we can compute $J_{\hat{\delta}}(X)$ and $\nabla_X \psi(X)$ as

$$\begin{aligned} J_{\hat{\delta}}(X) &= [(\nabla_{X:0} \hat{\delta}(X))^T, (\nabla_{X:1} \hat{\delta}(X))^T, ..., (\nabla_{X:3n} \hat{\delta}(X))^T]^T \\ \nabla_X \psi(X) &= [\frac{\partial \psi(X)}{\partial X_{:0}}, \frac{\partial \psi(X)}{\partial X_{:1}}, ..., \frac{\partial \psi(X)}{\partial X_{:3n}}] \end{aligned} \tag{11}$$

where

$$\begin{aligned} (\nabla_{X:t} \hat{\delta}(X))^T &= \frac{\hat{\delta}(X + \epsilon \mathbf{e}_{:t}) - \hat{\delta}(X)}{\epsilon} \\ \frac{\partial \psi(X)}{\partial X_{:t}} &= \frac{\psi(X + \epsilon \mathbf{e}_{:t}) - \psi(X)}{\epsilon} \end{aligned} \tag{12}$$

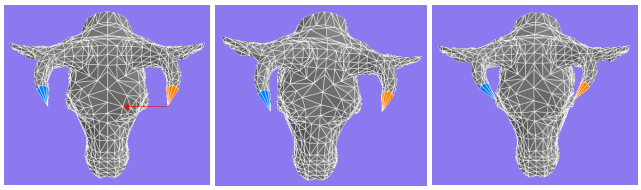
and we use $\epsilon = 1.0 \times 10^{-5}$ in our program.

Although finite difference method is much more straight forward and easy to implement, it usually costs more running time than analytic computation. Besides, the precision of the results given by finite difference is not easy to control sometimes due to the trade off between truncation error and round-off error, which is also dependent on the properties of the objective function. Thus, we use analytic computation in our Gauss-Newton solver.

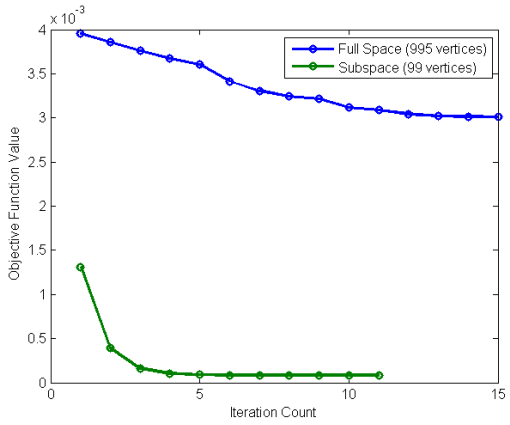
3 Experiments and Results

3.1 Full space v.s. Subspace

We first compare the full space solve and the subspace solve of our nonlinear least squares based mesh deformation method.



(a) *User interaction* **(b)** *Full space solve* **(c)** *Subspace solve*



(d) *Objective function values in each Gauss-Newton iteration.*

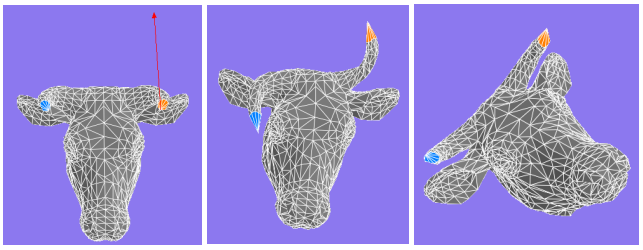
Figure 3: *Comparison of full space solve and subspace solve. The position constraint is given by fixing the left horn singularity and displace the right horn singularity of the cow head model towards left as shown in (a). From (b) we can see that full space solve is easily trapped in bad local minimum, while from (c) we see that subspace solve gives a much better solution. (d) shows that subspace solve is converging faster and better than full space solve.*

Fig.3 shows the results and convergence curve of a very simple experiment, where there are only two vertices whose positions are constrained. With full degree of freedom in the shape space, the full space solve is just trapped in a local minimum where the position constraints are not well satisfied. However, the subspace solve provides a very meaningful result with much smaller objective function value. Both of the two iterations stopped when there are no significant changes of X allowed during the line search. In this sense, we can say that solving in subspace provide better search directions for the nonlinear minimization problem. This is because the search space of the subspace solve only contains low frequency features, i.e. the coarse structure, of the mesh, where high frequency features won't interfere the search, thus directing the search to those local minimums that are more likely to be expected.

3.2 Subspace v.s. ARAP: Large Rotations

Next, we compare our subspace solve with as-rigid-as-possible (ARAP) mesh deformation solved by a local-global scheme [Sorkine and Alexa 2007]. ARAP is also a kind of iterative method, it conquers the local rotation variance obstacle of the basic laplacian coordinates by approximating better pre-scripted rotation iteratively via a local stage applying SVD to the deformation gradient at each vertex, and then a global stage solves a laplacian surface editing problem [Sorkine et al. 2004] with the newly pre-scripted rotations applied on the original laplacian coordinates.

However, if the given position constraint moves some vertices far away from their original position, which means large rotations are expected in the final deformed shape, it is also not easy for ARAP to approximate the rotations. Although it will converge, a local minimum with bad pre-scripted rotations could be encountered, where Fig.4b is just an example. The user interaction shown in Fig.4a fixed the left horn singularity and drags the right one upwards and backwards. Since only two vertex positions are constrained, large global rotations could be involved to preserve surface details better, and this is what our subspace solve produces in Fig.4c.



(a) *User interaction* **(b)** *ARAP iter100* **(c)** *Subspace solve iter10*

Figure 4: *Comparison of ARAP and our subspace solve. The position constraint is given by fixing the left horn singularity and displace the right horn singularity of the cow head model upwards and backwards as shown in (a). From (b) we can see that although ARAP is able to introduce local rotations to preserve surface details, it is still not able to introduce large global rotations to reach better surface preservation. But from (c) we see that our subspace solve gives a solution with global rotation and a wider pair of horn singularities.*

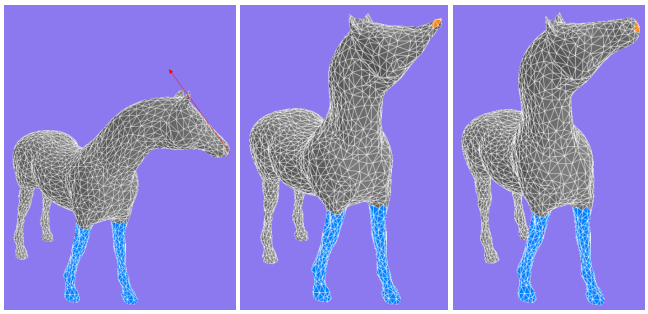
In fact, while implementing ARAP, people usually slices the displacement of the constrained vertices via mouse drag into tiny pieces and conduct the local-global solve between each mouse drag event with the pre-scripted rotations of the last solve being the initial guess for the current one. This can be viewed as an outer iterative framework for the deformation problem. If the displacement of the constrained vertices is small, or if the initial guess for the rotations is just close to the expected optimum, ARAP is able to give results containing large and global rotations, which is why slicing works. Another way to provide initial guess on pre-scripted rotations for ARAP is to use bounded bi-harmonic weights [Jacobson et al. 2011] based on user defined rotations on a small number of handles.

3.3 Subspace v.s. ARAP: Volume Preservation

Finally, we show the volume preservation property of our subspace solve via another mesh deformation case where ARAP fails.

As shown in Fig.5, if we fix the two front legs of the horse and lift the mouse up and left, the ARAP solve is failed to preserve the shape of the horse mouse, while our subspace solve can. This is how our volume perserving nonlinear constraint is superior to ARAP. ARAP tried to preserve volume via rotation approximated surface laplacian constraint, which only works for non-extreme cases. For an extreme case like Fig.5, when the position constraint is strong enough, ARAP may even produce self-intersected output meshes.

However, our volume preserving constraint is not perfect, because it only tries to preserve the global volume, which means a situation might probably occur in some extreme cases where one part of the mesh volume shrinks while another part of the mesh volume increases, e.g., in Fig.1-right, the root of the neck is shrinked while the mouse is elongated a little bit. Besides, once self-intersection happens during the iteration, when the interior of the mesh couldn't be well defined, the solve will proceed to unpredictable solutions.



(a) *User interaction* **(b)** *ARAP iter100* **(c)** *Subspace solve iter13*

Figure 5: *Comparison of ARAP and our subspace solve. The position constraint is given by fixing the two front legs and displace one point on the horse mouse up and left as shown in (a). From (b) we can see that although ARAP is able to lift the head up, it fails to preserve the volume of the mouse. But from (c) we can see that our subspace solve gives a solution with the head lifted and the shape of the mouse preserved.*

4 Conclusions and Future Works

To conclude, we implemented a basic version of subspace mesh deformation method with a constrained nonlinear least squares energy. The problem is solved by applying Gauss-Newton iteration with Lagrange multiplier and backtracking line search. With the search space reduced to the expansion of low features via a mapping between the original mesh and the corresponding coarse control mesh, the subspace solve converges faster with better solution than that in the full space. Compared to ARAP, our subspace solve is able to preserve global volume exactly and estimate the local rotations well even if the displacement of the constrained vertices is large.

For future works, it is very interesting on what will happen to the deformed mesh when we apply SVD on the Laplacian matrix of the original mesh to generate a subspace that is truly with only low frequency features theoretically and then solve the problem in this subspace. Since SVD for large matrices is extremely expensive, we can start with some fast approximation of SVD. Even with the approximated SVDs, it is possible to get better results and insights since constructing coarse control mesh by modeling or mesh simplification is hard to relate the mesh quality to the space reduction theory. Besides, we can construct the mapping matrix by applying positive mean value coordinates (PMVC) [Lipman et al. 2007], which is an improved version of mean value interpolation ensuring that the weights are all non-negative. The negative weights are caused by the concavity of the shape when the interpolating vertices are not visible by the point to be interpolated. PMVC tackle this problem by introducing visibility of the interpolating vertices into the weight computation. Moreover, it is also very promising to enable local volume preservation to our energy function by segmenting the mesh into semantic parts and define volume preserving constraints on those segments.

Acknowledgments

First, a big thank to Alla's inspiring lectures on geometric modeling and basic geometry processing techniques in the whole winter term. Also thank all the classmates for question discussions and paper presentations. Through the course, we all get a better understanding on this research area, combined with a wider horizon on those related mathematical topics. Thank Alla again for trusting and supporting me on conducting this not very easy course project. The experience on coding Gauss-Newton iteration with backtracking line search is just amazing! Last but not least, thank Enrique for teaching me to use 3ds Max to construct the control meshes.

References

ARMIJO, L. 1966. Minimization of functions having lipschitz continuous first partial derivatives. *Pacific Journal of mathematics 16*, 1, 1–3.

ASCHER, U. M., AND GREIF, C. 2011. *A First Course on Numerical Methods*, vol. 7. Siam.

BELLMAN, R. 1956. Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences 42*, 10, 767–769.

DESBRUN, M., MEYER, M., SCHRÖDER, P., AND BARR, A. H. 1999. Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 317–324.

FLOATER, M. S. 2003. Mean value coordinates. *Computer aided geometric design 20*, 1, 19–27.

HUANG, J., SHI, X., LIU, X., ZHOU, K., WEI, L.-Y., TENG, S.-H., BAO, H., GUO, B., AND SHUM, H.-Y. 2006. Subspace gradient domain mesh deformation. In *ACM Transactions on Graphics (TOG)*, vol. 25, ACM, 1126–1134.

JACOBSON, A., BARAN, I., POPOVIC, J., AND SORKINE, O. 2011. Bounded biharmonic weights for real-time deformation. *ACM Trans. Graph. 30*, 4, 78.

JU, T., SCHAEFER, S., AND WARREN, J. 2005. Mean value coordinates for closed triangular meshes. In *ACM Transactions on Graphics (TOG)*, vol. 24, ACM, 561–566.

KELLEY, C. T. 2003. *Solving nonlinear equations with Newton's method*, vol. 1. Siam.

LIPMAN, Y., KOPF, J., COHEN-OR, D., AND LEVIN, D. 2007. Gpu-assisted positive mean value coordinates for mesh deformations. In *Symposium on geometry processing*.

SANDER, P. V., GU, X., GORTLER, S. J., HOPPE, H., AND SNYDER, J. 2000. Silhouette clipping. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 327–334.

SEDERBERG, T. W., AND PARRY, S. R. 1986. Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics 20*, 4, 151–160.

SORKINE, O., AND ALEXA, M. 2007. As-rigid-as-possible surface modeling. In *Symposium on Geometry processing*, vol. 4.

SORKINE, O., COHEN-OR, D., LIPMAN, Y., ALEXA, M., RÖSSL, C., AND SEIDEL, H.-P. 2004. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, ACM, 175–184.

WEDDERBURN, R. W. 1974. Quasi-likelihood functions, generalized linear models, and the gaussnewton method. *Biometrika 61*, 3, 439–447.

WOLFE, P. 1969. Convergence conditions for ascent methods. *SIAM review 11*, 2, 226–235.