CGI2015 manuscript No.
(will be inserted by the editor)

Volumetric Surface Deformation with Auxiliary Voxel Grids

Abstract We propose two space deformation methods based on voxels. Our methods, approximately preserving the volume of a mesh object, are simple to implement and have a high performance due to resorting to voxelization. Volumetric deformations are achieved by either dual quaternion skinning or a nonlinear rigidity energy. The mesh surface embedded within the voxel grid is updated by means of interpolation. Three interpolation methods are implemented, targeting speed and smoothness, respectively. Both visual and quantitative evaluations on a variety of large polygon meshes are provided, demonstrating the effectiveness of the methods.

Keywords Volume Preservation · Voxel Deformation · Blend Skinning · Space Deformation

1 Introduction

Volume preservation has long been a point of focus in deformation in order to enhance the visual plausibility. A widely applied technique is dual quaternion skinning (DQS) [12], which is a closed-form solution approximately preserving the volume. Deformation cast as a surface-based energy minimization problem [3, 9] better preserves the shape details, but for volume preservation, complicated volume constraints need to be defined. Moreover, the resulting deformations might not be as natural as a full volumetric discretization with physically plausible deformation energies [4]. The implementations of such volumetric energies, however, often are difficult to achieve. Physics-based simulations of incompressible solids [7, 10] are difficult to control due to the presence of many parameters [5].

In this paper, we present two volumetric methods that approximately preserve the volume of an object throughout the deformation. Our methods are easy to implement and run fast on a modern laptop. One method is to apply DQS to voxels that represent the volume domain enclosed by the mesh surface, and consequently, deform the surface by the efficient trilinear embedding, which is also extended as blending of transformations if the surrounding voxel nodes of a vertex store rotations. For large polygonal meshes, this strategy generally accelerates the computations by at least an order of magnitude. The other method is to cast the voxel deformation as a rigidity energy minimization problem based on as-rigid-as-possible (ARAP) surface modeling, preserving volumetric properties and producing reasonably natural poses. To maintain the shape smoothness, embedding based on radial basis functions (RBFs) is implemented. To keep user interactions to a minimum in this method, the voxel deformations are controlled by moving point handles in the spirit of linear blend skinning (LBS) instead of by directly manipulating voxels. Moreover, a deformed configuration resulting from LBS assists to speed up the iterative minimization and robustly obtain large deformations. In the context of embedding, our two methods are suitable for a wide range of mesh representations.

The benefit of using voxels is two-fold. On the one hand, the voxelization avoids coping with geometric issues such as self intersections and nonmanifoldness that usually need to be handled in tetrahedral meshing. On the other hand, the voxel grid facilitates the construction of the Laplacian matrix used to build the rigidity energy, as it provides an easily accessible neighborhood structure.

Both visual and quantitative evaluations on a variety of 3D models are provided; the results show the effectiveness of our methods.

2 Volumetric deformation

The volume domain of a mesh denoted by $\mathcal{S}$ is discretized by a hardware accelerated voxelization method [13], resulting in a voxel grid denoted by $\mathcal{V}$. In our case, voxels are isotropic, and the ones across the mesh surface are retained for embedding introduced in Section 3. Voxels are structured in a grid so that they are easily accessible.

2.1 Volumetric DQS

Given a skinned model consisting of $\mathcal{H}$ bones and $\mathcal{P}$ mesh vertices, for each vertex p, associated with influence weights $\mathcal{W}_p = (w_i, ..., w_H)$, DQS parameterizes the transformation of $\mathcal{H}_i$ using dual quaternion denoted by $\dot{q}_i, i \in \{1, ..., \mathcal{H}\}$, and transforms a vertex from rest pose to the deformed position by the final dual quaternion as $\dot{q} = (\sum_{i=1}^{\mathcal{H}} w_i \dot{q}_i) / \parallel \sum_{i=1}^{\mathcal{H}} w_i \dot{q}_i \parallel$.

For each vertex, the runtime cost mainly lies in the interpolations of handle rotations, which are nonlinear calculations. To decrease the cost, a straight way is reducing the number of vertices involving interpolations.

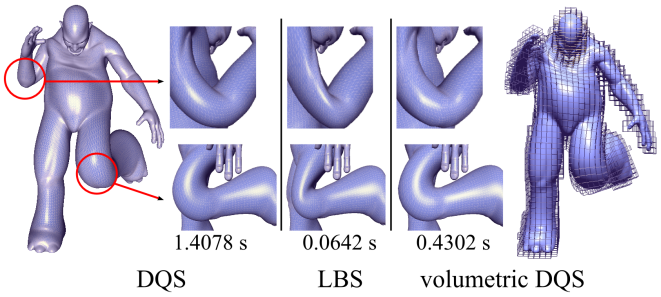

Fig. 1 An animation still of a skinned character consisting of 84,888 vertices. Volumetric DQS (5,542 voxels) results in volumetric deformation fairly similar to DQS while it is at least an order of magnitude faster. Numbers below are average time per frame in seconds.

In most cases, a voxel grid as valid object representation is coarser than embedded polygonal mesh, in particular for high resolution meshes. In the context of space deformation, furthermore only the voxels on the grid hull will be deformed by DQS. As a result, applying DQS to voxel grid together with trilinear embedding largely decreases the cost. Figure 1 shows an illustration.

2.2 Nonlinear volumetric ARAP

The surface-based ARAP energy formulation of [14] is adopted to define our volumetric rigidity energy, as it is conceptually easy to understand and easy to implement. This ARAP formula minimizes the changes of edge lengths and addresses the nonlinearity of transformations of each discrete cell.

In our definition, each discrete, volumetric cell $\mathcal{C}_i$ consists of a voxel and its at most six immediate neighbors. The volumetric ARAP approximately preserves the edge lengths between voxel nodes by iteratively minimizing the cell energy as

$$E(\mathcal{C}_i^{'}) = \sum_{j \in \mathcal{N}(i)} \omega_{ij} \parallel (n_i^{'} - n_j^{'}) - R_i(n_i - n_j) \parallel^2 . \qquad (1)$$

Edges between a voxel node n_i and its immediate neighbors $n_j, j \in \mathcal{N}(i)$ are weighed by ω_{ij} correspondingly. $n_i^{'}$ is the deformed position at an iteration. The presence of rotation R_i of n_i enforces that cell $\mathcal{C}_i$ deforms as-rigidly-as-possible. In detail, R_i is derived from the singular value decomposition (SVD) of the covariance matrix $S_i = \sum_{j \in \mathcal{N}(i)} (\omega_{ij} e_{ij} e_{ij}^{'T}) = U_i \sum_i V_i^T$, and $R_i = V_i U_i^T$ [14], where $e_{ij} = n_i - n_j$ and $e_{ij}^{'}$ is the edge after deformation.

The total energy is the sum of all cell energies and can be compactly written as a matrix form of equations [14] subject to positional constraints. Weights ω_{ij} are assembled into a matrix called Laplace-Beltrami operator denoted by $\mathbf{L}$. In ARAP, cotangent weights [6] instead of uniform weights are computed to avoid deformation artifacts [14], but at the risk of the emergence of degeneration, as they are only defined for 2D manifolds. In our volumetric ARAP, because the graph Laplacian was easily established beforehand based on the neighborhood structures, the Laplace-Beltrami operator is easily defined based on uniform weights as

$$\mathbf{L}_{\mathcal{V}} = \begin{cases} \sum_{j \in \mathcal{N}(i)} 1.0, & i = j, \\ -1.0, & j \in \mathcal{N}(i), \\ 0, & \text{otherwise.} \end{cases} \qquad (2)$$

To keep user interactions to a minimum, points are placed within the volume and their influence weights are computed by the method of [11], as it yields smooth weights. Through LBS, voxel deformations are controlled by moving points instead of by directly manipulating the voxels.

Since LBS is efficient for large deformations, the minimization is accelerated and stabilized for large rotations. First, rotations from rest pose to current configuration resulting from LBS are provided as an initial guess. Second, new positions are computed by minimizing Equation 1. The two procedures are alternately performed to achieve the global unique minimum.

3 Space deformation

We construct space deformation $\mathbf{d} : \mathbb{R}^3 \rightarrow \mathbb{R}^3$ based on spatial interpolation. Trilinear interpolation is the fastest way to embed a mesh within voxels. If the surrounding eight nodes of a mesh vertex store rotations that are from the rest pose to the deformed positions,

Model	# Voxels	# Vertices	Volumetric DQS		Volumetric ARAP	
			Volume (%)	Time (s)	Volume (%)	Time (s)
Long cube	4,800	1,002	99.4188	0.047	95.3269	0.423796
Fat man	5,542	84,888	97.308	0.364397	98.3852	2.74804
Warrior	1,108	40,197	86.3242	0.01398	93.0	0.0631
Head	18,296	48,624	99.422	0.453636	94.1977	23.9236
Arm	13,263	49,874	98.5649	0.227855	99.3745	7.32747
Name card	10,849	130,340	110.425	0.418142	99.8083	6.39093
Mushroom	16,408	131,304	99.9798	4.56489	98.5793	26.13
Tree	3,261	11,645	107.26	0.306284	105.032	1.99311

Table 1 Model statistics: Timings are the overall time of reaching the target pose, and percentages are relative volume also indicating whether an increasing or decreasing volume. ARAP runs with 2 iterations. Volumetric ARAP has relative volume similar to volumetric DQS, while it produces more natural poses. Note that the number of voxels is much smaller than the number of vertices in cases of large meshes.

Fig. 2 Surfaces are rendered under the same point light. Trilinear interpolation and blending of transformations give rise to a pattern of strips on the surface. RBFs interpolation is smooth enough to interpolate the surface.

Model	Trilinear	Linear blending	RBFs
Long cube	0.0008	0.0016	0.1154
Fat man	0.0581489	0.0882	9.20568
Warrior	0.0305	0.0526	1.30276
Head	0.0291048	0.0547445	9.52889
Arm	0.030507	0.056674	8.46751
Name card	0.0686187	0.132592	23.0921
Mushroom	0.0739739	0.135898	82.674
Tree	0.004	0.0094	1.14412

Table 2 Space deformation timings: Trilinear and linear blending are sufficient for real-time requirements. Time is in seconds.

the interpolation is easily extended as blending of either transformation matrices. However, these two methods belong to Shepard's scheme [2] that is not smooth enough to interpolate surfaces.

To address the limitations, smooth interpolation of displacements based on radial basis functions (RBFs) is developed. In particular, we use RBF with compact support to yield a sparse matrix

$$k(x,c) = e^{(\frac{-\|x-c\|^2}{\sigma^2})} + a^T \cdot x + b, \quad (3)$$

where $c \in \mathbb{R}^3$ is the RBF center which is the center of a hull voxel in our case, and $x \in \mathbb{R}^3$ is the evaluation point. The presence of the linear term helps to reproduce global behavior of the function. The resulting sparse and symmetric matrix can be solved by LU decomposition. σ is the average distance between RBF centers following the guideline of ALGLIB [1] which is a popular numerical analysis library, guaranteeing that there will be several centers at distance σ around each evaluation point. This RBFs interpolation gives rise to smooth result regarding the two methods mentioned above, see Figure 2.

4 Results

Our methods were implemented in C++ on a laptop with an Intel Core i7 2.4Ghz processor and 6 GB memory. The LU decomposition, sparse Cholesky solver and two-sided Jacobi SVD are provided by the Eigen library [8].

Model statistics and quantitative comparisons are reported in Table 1, and statistics on space deformation are presented in Table 2.

Both volumetric DQS and volumetric ARAP approximately preserve the volume with respect to LBS even in cases of complex motions such as bending and twisting, see Figure 3 and 4, respectively. Volumetric DQS yields volume preservation fairly similar to DQS, while it often runs much faster (see Section 2.1). Though volumetric ARAP is more computationally expensive than volumetric DQS (see Table 1), it produces more natural poses. This is because essentially it preserves the locality, see Figure 5. Targeting smoothness of the surface in addition to the pose, our volumetric ARAP can be combined with RBFs interpolation to preserve both shape details and volumetric properties simultaneously.

5 Conclusion

In this paper, we have presented two space deformation methods based on voxelizaition. The two methods, named volumetric DQS and volumetric ARAP, both preserve the object's volume to a degree while the latter method produces more natural poses. In most cases,

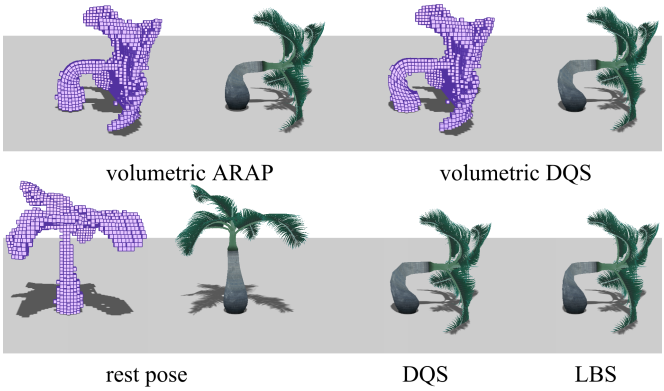

Fig. 3 Volumetric ARAP and volumetric DQS both are able to preserve the volume to a degree with respect to LBS in large bending. Volumetric ARAP, however, produces more natural poses than DQS and volumetric DQS. Note that a surface-based deformation energy might be unsuitable for this tree model, due to the presence of degenerated triangles in the leaf parts.

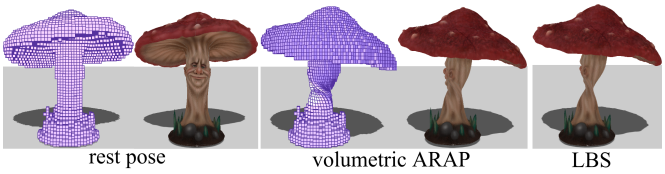

Fig. 4 The mushroom model twists by 135 degrees. Volumetric ARAP preserves 98.5793% of the volume, while LBS gives rise to the well-known shape collapse artifact.

Fig. 5 Volumetric ARAP prevents shape distortion. The letters in the name card model deform as-rigidly-as-possible while they distort by using volumetric DQS, see the close-ups.

speedups are obtained for volumetric DQS with respect to DQS. A variety of results of large polygonal meshes have demonstrated the effectiveness of the methods.

For real-time applications such as computer games where volume preservation is desired, volumetric DQS with trilinear embedding or its extension as blending of transformations gives the solution. In feature films production where shape smoothness, natural posture and volume preservation are desired for better realism, volumetric ARAP combined with RBFs can be a solution.

The volumetric ARAP method holds promise for skeletal deformation. Currently, it cannot always achieve realistic character postures and skin deformations, as additional constraints should be introduced, notably constraints that keep bone lengths constant. Moreover, physically the character should stand throughout the animation, so the center of mass of the body has to be optimized at run time. Since the volumetric ARAP energy aims to conserve edge lengths, hence it may result in a locking problem that can be solved by the method of [10]. Our future work is to figure out such issues, making the method more practical in character animation.

References

1. Alglib: a cross-platform numerical analysis and data processing library. available online at `http://www.alglib.net/`. Accessed: 2015-03-03
2. Barnhill, R., Dube, R., Little, F.: Properties of shepard's surfaces. Journal of Mathematics **13**(2) (1983)
3. Ben-Chen, M., Weber, O., Gotsman, C.: Variational harmonic maps for space deformation. ACM Trans. Graph. **28**(3), 34:1–34:11 (2009)
4. Botsch, M., Pauly, M., Wicke, M., Gross, M.: Adaptive space deformations based on rigid cells. Computer Graphics Forum **26**(3), 339–347 (2007)
5. Bridson, R., Batty, C.: Computational physics in film. Science **330**(6012), 1756–1757 (2010)
6. Desbrun, M., Meyer, M., Schröder, P., Barr, A.H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In: Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH '99, pp. 317–324 (1999)
7. Diziol, R., Bender, J., Bayer, D.: Robust real-time deformation of incompressible surface meshes. In: Proceedings of the 2011 ACM SIGGRAPH/Eurographics Symposium on Computer Animation, SCA '11, pp. 237–246 (2011)
8. Guennebaud, G.: Eigen: a c++ linear algebra library,version 3.0. available online at `http://eigen.tuxfamily.org/`. Accessed: 2014-11-21
9. Huang, J., Shi, X., Liu, X., Zhou, K., Wei, L.Y., Teng, S.H., Bao, H., Guo, B., Shum, H.Y.: Subspace gradient domain mesh deformation. ACM Trans. Graph. **25**(3), 1126–1134 (2006)
10. Irving, G., Schroeder, C., Fedkiw, R.: Volume conserving finite element simulations of deformable models. ACM Trans. Graph. **26**(3) (2007)
11. Jacobson, A., Baran, I., Popović, J., Sorkine, O.: Bounded biharmonic weights for real-time deformation. ACM Trans. Graph. **30**(4), 78:1–78:8 (2011)
12. Kavan, L., Collins, S., Žára, J., O'Sullivan, C.: Geometric skinning with approximate dual quaternion blending. ACM Trans. Graph. **27**(4), 105:1–105:23 (2008)
13. Lefebvre, S., Hornus, S., Lasram, A.: Ha-buffer: Coherent hashing for single-pass a-buffer. Research Report RR-8282, INRIA (2013). URL `https://hal.inria.fr/hal-00811585`
14. Sorkine, O., Alexa, M.: As-rigid-as-possible surface modeling. In: Proceedings of the Fifth Eurographics Symposium on Geometry Processing, SGP '07, pp. 109–116 (2007)