# Hybrid Mesh Editing

Péter Borosán, Reid Howard, Shaoting Zhang and Andrew Nealen

Department of Computer Science, Rutgers University

**Abstract**

*Surface-based deformation and cage-based deformation are two popular shape editing paradigms. Surface-based methods are easy to use and produce high-quality results by preserving differential properties of the surface mesh, but are limited by their computational requirements. Cage-based methods produce results quickly but at the expense of usability and realism, and typically require manual construction of suitable cages. We introduce a hybrid approach that combines the two methods. The user can perform edits on an automatically-generated simplified version of an input shape using As-rigid-as-possible surface modeling, and the edit is propagated to the original shape by a precomputed space deformation based on Mean value coordinates. We analyze deformation quality and running time for a variety of cage sizes. High-quality results are obtained for meshes on the order of 100K vertices at interactive rates by using cages with $\sim 5\%$ of the vertices of the original shape.*

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—geometric algorithms, languages, and systems

## 1. Introduction

Much of the recent work in 3D polygonal mesh editing can be classified as being either surface-based or space-based. Surface-based approaches allow the user to fix one region of a mesh while moving a small "handle" region. The deformation applied to the handle is propagated and distributed across the remaining free vertices in a way that minimizes a predefined energy functional dependent on the geometry of the mesh to provide intuitive deformations [Bot07]. The energy minimization is usually achieved by solving a sparse linear system. Rotations however cannot be linearly parameterized without artifacts. Methods involving non-linear approaches achieve natural-looking results, but as non-linearity comes at a high computational cost, interactive manipulation of high resolution models is not possible using these schemes. To handle many vertices, multiresolution is usually applied [GSS99, ZSS97, HSL*06], which is close to our approach: a low resolution version of the original shape obtained by removing the surface details is deformed first and then details are added back to this deformed surface.

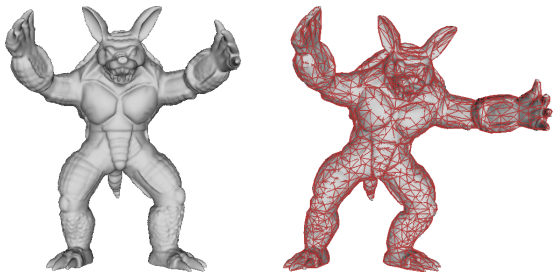

**Figure 1:** *Deformation of the armadillo model. The left is the original shape, the right is the deformed version showing the control mesh*



Space-based approaches allow users to edit a shape indirectly via a control structure that exerts a prescribed influence on the embedding space when deformed [GB08, SP86, BK05, SSP07]. For one particular class of space deformations, which we refer to as *cage deformations*, the control structure is a coarse closed polygonal mesh constructed around the shape, known as a *cage*. When the cage is deformed, the enclosed shape follows accordingly. Mean value coordinates [JSW05] and Harmonic coordinates [JMD*07] determine the space deformation by computing generalized barycentric coordinates with respect to the cage vertices. Green coordinates [LLCO08] extend this concept by incorporating normals of cage faces into the computation to induce quasi-conformal mappings. Cage-based methods allow for fast deformations after the coordinates are precomputed, but the cage must be constructed beforehand, and achiev-

ing a desired deformation on the mesh may require tedious placement of cage vertices and guesswork.

Alternative schemes combine the two approaches. In [SSP07] the core deformation is space-based, but the control structure can be manipulated using the handle paradigm of surface-based methods. We believe a hybrid approach, also advocated in [CO09], has great promise and has not been sufficiently explored; in particular, we take the view that there is a continuum between purely surface-based and purely *cage*-based methods, and we propose a scheme to explore it by applying surface deformations to a simplified version of an input shape mesh. The original shape is updated with a space deformation induced by this simplified control mesh. This allows us to exploit the speed of the space-based method and retain the ease of use and natural results of surface-based methods, while requiring no extra footwork from the user and without loss of surface detail.

Our contribution is to combine surface-based and space-based deformation approaches to produce high-quality deformations of large meshes at interactive rates. We explore the aforementioned "continuum", parametrized by complexity of the control mesh, and provide a preliminary analysis of the tradeoffs in speed, robustness, and quality as we move along this continuum.

## 2. Algorithms

### 2.1. Surface Deformation

As-rigid-as-possible surface modeling, proposed in [SA07], is a powerful deformation technique that produces natural looking results with guaranteed convergence. The principle used is that small parts of the shape should change as rigidly as possible and smoothly. The method builds on a non-linear energy formulation and minimizes it through iterations. For this the shape is covered by cells centered at each of the vertices and covering the incident faces. Using $\{p_1 \dots p_n\}$ for the vertex positions of the original mesh and $\{p'_1 \dots p'_n\}$ for those of the deformed one, the error term for a cell $\mathcal{C}_i$ is defined to be

$$E(\mathcal{C}_i, \mathcal{C}'_i) = \sum_{j \in \mathcal{N}(i)} w_{ij} ||(p'_i - p'_j) - R_i(p_i - p_j)||^2 \qquad (1)$$

where $\mathcal{N}(i)$ is the one-ring neighborhood of vertex $i$, $w_{ij}$ is the weight of edge $(i, j)$ and $R_i$ is the rotation that best approximates the transformation that takes cell $\mathcal{C}_i$ to $\mathcal{C}'_i$. The total error between shape $\mathcal{S}'$ and the reference $\mathcal{S}$ is the sum of the per cell errors.

Based on an initial guess on vertex positions, the algorithm first finds the optimal rotation $R_i$ for each of the cells independently. If we use the notation $e_{ij} = p_i - p_j$ and $U_i \Sigma_i V_i^T = \sum_{j \in \mathcal{N}(i)} w_{ij} e_{ij} e'^T_{ij}$ is the singular value decomposition of the covariance matrix then the rotation that minimizes Eq. 1 given the vertex positions is

$$R_i = V_i U_i^T \qquad (2)$$

Now given the rotations for each cell the next step is solving for the vertex positions that minimize the global error term. The minimum is achieved at the positions where the gradient of the error term is zero. This yields the following system of equations:

$$\sum_{j \in \mathcal{N}(i)} w_{ij}(p'_i - p'_j) = \sum_{j \in \mathcal{N}(i)} \frac{w_{ij}}{2}(R_i + R_j)(p_i - p_j) \qquad (3)$$

which can be rewritten in matrix form as $Lp' = b$ where $p'$ is the vector containing the unknown vertex position and $L$ is the submatrix of the discrete Laplace-Beltrami operator corresponding to these vertices. $L$ is a symmetric positive definite sparse matrix that furthermore stays fixed and therefore can be efficiently factorized in advance while only back substitutions will be required during the editing process. Now given the positions we can again compute the optimal rotations and continue the process until we get close enough to the desired shape. The speed of the convergence depends on the condition number of $L$ which is proportional to the mesh size [CCOST05]. So for large models, both the time per iteration and the number of iterations necessary make this method unsuitable for interactive editing, but for a few thousand vertices 2-3 iterations usually suffice and interactive rates can be achieved.

### 2.2. Cage-Based Deformation

Cage deformation works in the following way. Let $\mathcal{Y}$ be a polygonal mesh with vertices $y_j \in \mathbb{R}^3$, called the *cage*. For each $y_j$, we specify a weight function $\varphi_j(x)$ that is defined in some region of $\mathbb{R}^3$. The *coordinates* of a point $x \in \mathbb{R}^3$ with respect to these points are given by

$$c_j(x) = \frac{\varphi_j(x)}{\sum_k \varphi_k(x)} \qquad (4)$$

For an input mesh $\mathcal{X}$ with vertices $x_i \in \mathbb{R}^3$ (called the *shape*), the values $\{c_j(x_i)\}_{i,j}$ are precomputed and stored only once, with the cage in its rest state (a process known as *binding*). Deformations are applied by manipulating the cage vertices directly, and the shape is updated with

$$x'_i = \sum_j c_j(x_i) y'_j \qquad (5)$$

So the update step for a given shape vertex is linear in the complexity of the cage. In order to produce smooth deformations, the $c_j$ should be smooth functions with $c_j(y_k) = \delta_{jk}$ (Kronecker delta), i.e., the deformation to the embedding space should interpolate the locations of control vertices. Furthermore, for any $x_i$, we should have $\sum_j c_j(x_i) y_j = x_i$, i.e., plugging the $y_i$s into Eq. 5 should not cause a shape vertex to "pop" out of place (this is known as *linear precision*). Equivalently, one might say that the $c_j(x)$ should generalize barycentric coordinates. As mentioned in the introduction, Mean value coordinates [JSW05] and Harmonic coor-

dinates [JMD*07] are two proposed sets of coordinates that do this.

## 3. Hybrid Space/Surface Deformation

The core of our method is to construct a cage that approximates an input shape and apply the surface-based deformation to it, updating the original shape using a cage deformation. Mean value coordinates and Harmonic coordinates have been applied to deformation by manually constructing a very *coarse* cage to enclose the shape of interest. For our method, we are interested in using cages of thousands of vertices, and we would like a cage to fit the original shape very snugly so that the user experience matches that of using purely surface-based methods. Construction of suitable cages is not the contribution of this work, but our task is made much simpler thanks to the observation by Ju et al [JSW05] that Mean value coordinates vary smoothly across cage faces and are in fact well-defined everywhere in $\mathbb{R}^3$. This allows us to relax the requirement that cages fully enclose the shape, and we can obtain cages of arbitrary complexity by simply applying a mesh simplification method [GH97] to the input shape, and use Mean value coordinates for our space deformation.

In our system, the user loads a shape mesh they wish to edit and specifies a cage size. Once the cage is generated, the shape is bound to it using the Mean value coordinates and insignificant coordinates are discarded (explained below). The user may then perform edits on the cage by specifying a region of interest and manipulating a handle. The new locations of the free vertices of the cage are computed with the As-rigid-as-possible method described in Section 2.1, using $w_{ij} = 1$ in the system in Eq. 1 and 3. Finally, the new positions of the shape vertices are computed by applying Eq. 5.

Note that without any further modification to the cage, its vertices are a subset of the input shape vertices. For these vertices of the shape, the only nonzero coordinate will be that of the coinciding cage vertex. Furthermore, on cage faces, Mean value coordinates reduce to 2D barycentric coordinates. Since our cages approximate the shape so closely, shape vertices tend to lie very near to cage faces, so most of the contribution to the computed coordinates will come from the three vertices of the nearest face. One might then view the set of coordinates as 2D barycentric coordinates with an additional displacement term to preserve detail. If a fine enough cage is used, this displacement term typically depends primarily on a small set of other nearby vertices. It is also desirable that the influence of distant cage vertices be minimized to avoid unwanted artifacts. Therefore, to greatly speed up the deformation and localize the cage influence, we can sort the coordinates by magnitude and discard the smallest contributors up to a suitably chosen threshold. In our experiments, we have found that by making this threshold 3% of the total magnitude, we discard all but a constant number of coordinates (on average) for each vertex, and incur

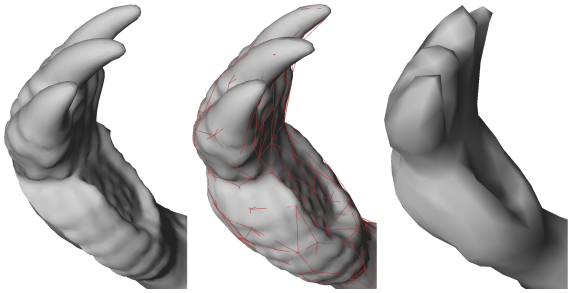

**Figure 2:** *The armadillo's forearm after a deformation. From left to right: Applying ARAP method directly to the 40K model (1.7s), using a 2K cage as shown (116 ms) and applying ARAP directly to a 3K model (115ms)*

no visible loss of surface detail (see Figure. 2 middle). This means that the update step is linear in shape complexity and depends very little on cage complexity. (In fact, since larger cages better approximate the shape, this constant decreases slightly with cage complexity.) This optimization has been applied for all of our tests and images.

## 4. Results

We ran our C++ implementation using the Cholmod sparse Cholesky solver [DH09] on an Intel Core2 Duo T7300 Mobile CPU with 2GB RAM and Linux 2.6.31.

As a result of the formulation of the error term (Eq. 1) the As-rigid-as-possible method preserves edge length to the extent allowed by the modeling constraints; if no stretching is imposed, the optimization converges to a state with small edge length error. Because of this it is reasonable to measure the cost of the lower computation time we gain in terms of this metric of rigidity, using deformation with no cage as a benchmark.

We applied the same deformation (see Figure 1) on two different resolution models of the armadillo, with 21622 (20K) and 43244 (40K) vertices using various cage sizes. Table 1 shows the running times and the root mean squared edge length errors for using 2 As-rigid-as-possible iterations. As expected, the running time scales linearly, while the error decreases with finer cages (Figure 3). Our tests on the Armadillo model indicate that a cage containing 5-10% of the original vertices produces an acceptable amount of error (see Figure 2) and that further increasing the cage complexity does not yield a significant additional benefit. However, the optimal cage size for a given shape is a function of both its local and global complexity and not simply the number of vertices. We note that using a cage that is too coarse limits the effectiveness of the optimization mentioned in Section 3 and may also cause self-intersection under extreme deformations.

We believe that combining surface-based and space-based deformation produces compelling results with the advan-

| Cage | RRMS-E | Time |
|---|---|---|
| 21622 vertices model | | |
| 1000 (4.62%) | 7.39% | 60 ms |
| 1500 (6.94%) | 5.17% | 81 ms |
| 2000 (9.25%) | 4.51% | 91 ms |
| 4000 (18.5%) | 3.66% | 154 ms |
| no cage (100%) | 1.92% | 792 ms |
| 43244 vertices model | | |
| 1500 (3.47%) | 6.69% | 92 ms |
| 2000 (4.62%) | 5.43% | 116 ms |
| 5000 (11.56%) | 3.87% | 213 ms |
| no cage (100%) | 1.72% | 1669 ms |

**Table 1:** *Relative root mean squared errors and running times for different base and control mesh sizes and for applying the ARAP method directly to the mesh*

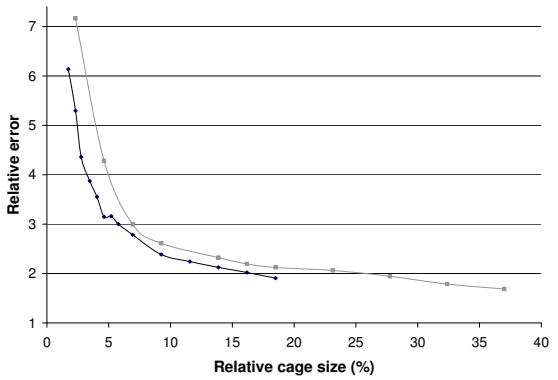


**Figure 3:** *Relative error compared to that of the no-cage benchmark versus relative cage. Black: 40K input shape; Grey: 20K*

tages of both. Avenues for future work include automatic generation of cages that tightly *enclose* a mesh so that e.g. Harmonic coordinates may be employed as a space deformation method and make our framework useful for highly concave meshes. We hope our work is an insightful first step toward fully exploring the continuum between these two successful deformation paradigms.

## 5. Acknowledgements

This work was supported in part by NSF grant No. IIS-0916845. We would like to thank the anonymous reviewers for their valuable comments and suggestions.

## References

[BK05] BOTSCH M., KOBBELT L.: Real-time shape editing using radial basis functions. In *Computer Graphics Forum* (2005), pp. 611–621.

[Bot07] BOTSCH M.: On linear variational surface deformation methods. *IEEE Transaction on Visualization on Computer Graphics 14* (2007), 213–230.

[CCOST05] CHEN D., COHEN-OR D., SORKINE O., TOLEDO S.: Algebraic analysis of high-pass quantization. *ACM Trans. Graph. 24*, 4 (2005), 1259–1282.

[CO09] COHEN-OR D.: Space deformations, surface deformations and the opportunities in-between. *J. Comput. Sci. Technol. 24*, 1 (2009), 2–5.

[DH09] DAVIS T. A., HAGER W. W.: Dynamic supernodes in sparse cholesky update/downdate and triangular solves. *ACM Trans. Math. Softw. 35*, 4 (2009), 1–23.

[GB08] GAIN J., BECHMANN D.: A survey of spatial deformation from a user-centered perspective. *ACM Trans. Graph. 27*, 4 (2008), 1–21.

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. In *SIGGRAPH '97: Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1997), ACM Press/Addison-Wesley Publishing Co., pp. 209–216.

[GSS99] GUSKOV I., SWELDENS W., SCHRÖDER P.: Multiresolution signal processing for meshes. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1999), ACM Press/Addison-Wesley Publishing Co., pp. 325–334.

[HSL*06] 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 (2006), 1126–1134.

[JMD*07] JOSHI P., MEYER M., DEROSE T., GREEN B., SANOCKI T.: Harmonic coordinates for character articulation. In *SIGGRAPH '07: ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), ACM, p. 71.

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers* (New York, NY, USA, 2005), ACM, pp. 561–566.

[LLCO08] LIPMAN Y., LEVIN D., COHEN-OR D.: Green coordinates. In *SIGGRAPH '08: ACM SIGGRAPH 2008 papers* (New York, NY, USA, 2008), ACM, pp. 1–10.

[SA07] SORKINE O., ALEXA M.: As-rigid-as-possible surface modeling. In *SGP '07* (2007), pp. 109–116.

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. *SIGGRAPH Comput. Graph. 20*, 4 (1986), 151–160.

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. *ACM Trans. Graph. 26*, 3 (2007), 80.

[ZSS97] ZORIN D., SCHRÖDER P., SWELDENS W.: Interactive multiresolution mesh editing. In *SIGGRAPH '97* (1997), pp. 259–268.