3D Deformation Using Moving Least Squares

Yuanchen Zhu and Steven J. Gortler

TR-10-07

Computer Science Group
Harvard University
Cambridge, Massachusetts

3D Deformation Using Moving Least Squares

Yuanchen Zhu*
Harvard University

Steven J. Gortler†
Harvard University

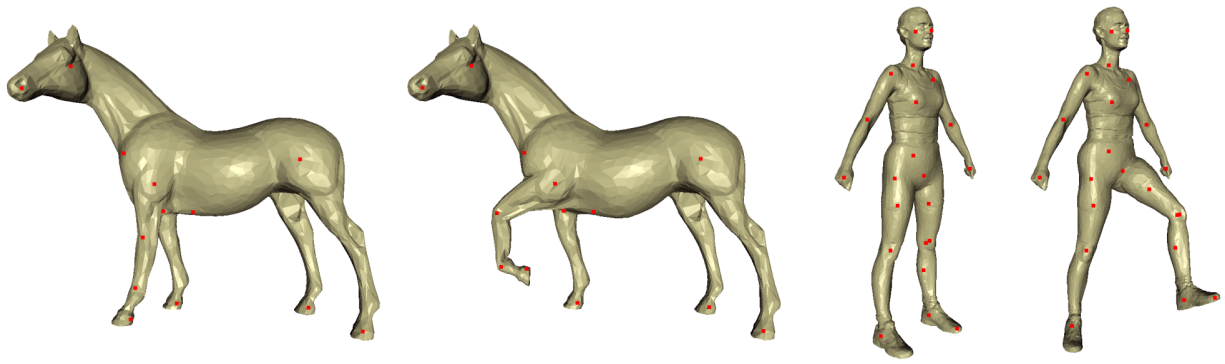

Figure 1: *Deformation results using our method. Point handles are inside the mesh and marked as red dots. (Models courtesy of Cyberware.)*

Abstract

We present a 3d deformation method based on Moving Least Squares that extends the work by Schaefer et al. [Schaefer et al. 2006] to the 3d setting. The user controls the deformation by manipulating a set of point handles. Locally, the deformation takes the form of either a rigid transformation or optionally a similarity transformation, and tends to preserve local features. Our derivation of the closed-form solution is based on singular value decomposition, and is applicable to deformation in arbitrary dimensions, as opposed to the planar case in [Schaefer et al. 2006]. Our prototype implementation allows interactive deformation of meshes of over 100k vertices.

For the application of 3d mesh deformation, we further introduce a weighting scheme that determines the influence of point handles on vertices based on approximate mesh geodesics. In practice, the new scheme gives much better deformation results for limbed character models, compared with simple Euclidean distance based weighting. The new weighting scheme can be of use to the traditional skinny based deformation technique as well.

1 Introduction

3d deformation is useful for various modeling and animation tasks such as surface sculpting and character animation. There have recently been a number of works on *gradient-domain mesh* based deformation, e.g., [Lipman et al. 2005; Huang et al. 2006; Botsch and Kobbelt]. Such methods are capable of preserving local surface details during editing, and is usually "global" in the sense that the algorithms explicitly scatters the desired deformation, often specified on a set of handle vertices, across the entire mesh, usually through least-square solving a large sparse linear system connecting every pair of neighboring mesh elements. Huang et al. [Huang et al. 2006] further introduces a non-linear optimizing framework, allowing the incorporation of various non-linear constraints such as rigidity constraint and volume constrain.

*yzhu@fas.harvard.edu
†sjg@cs.harvard.edu

In our work, we explore an alternative "local" approach towards feature preserving deformation. Our work mainly builds upon the 2d image deformation technique introduced by Schaefer et al. [Schaefer et al. 2006], which solves for the optimal rigid transformation that maps a set of weighted point handles to their deformed positions, at each image domain point. The image domain point is than transformed by the optimal rigid transformation at that point. Because an optimal transformation is calculated for every point to be deformed, it is called Moving Least Square.

The work-flow of deforming a model using our method is as following. The user first defines a set of point handles around the model that he wants to deform. He can then drag the point handles around, and the model will deform in a smooth and realistic manner such that the region close to a handle will move to the new position of the handle.

The amount of influence a handle has on a point in space depends on the "distance" between them, as measured by a distance function d. For the purpose of realistically deforming a mesh containing various parts, simple euclidean distance often has the effect of making a handle influence parts of the mesh that are geometrically close, but logically shouldn't be under the handle's influence. For example, in a human model, the parts corresponding to the left and right feet are close geometrically. If we place a handle one each of the feet, however, the handle on the left feet should not influence the right feet, in spite of the geometric closeness. To amend such problem, we introduce a distance function, the value of which equals the length of the shortest polyline that first connects the handle to a vertex on the mesh, and then travels from that vertex to the vertex in question via mesh edges. In practice, this distance function seems to provide excellent results.

2 Related Work

Lipman et al.[Lipman et al. 2005] represents mesh geometry as *local frames* and *discrete forms*, which are local properties that encode the mesh in a rotation and translation invariant way. They then allow manual editing some of the local frames, and solve a large, sparse linear system to find the optimal mesh that fits the given local frames. In contrast, our method performs singular value

decomposition on a small 3x3 matrix for every vertex. As can be seen from Figure 2, even though our method does not explicitly apply any inter-vertex constraint, the quality of the resulting deformation looks as good. An added advantage is that the interface to our animation system is simpler, the user just defines and drags point handles. In the case [Lipman et al. 2005], the user needs to explictly apply rotational transformation on the selection.

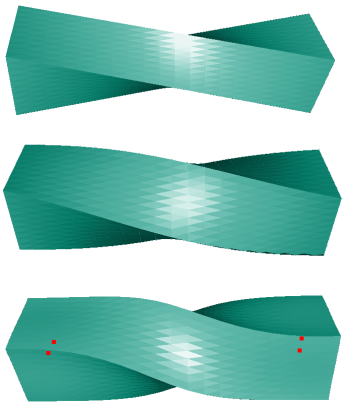

Figure 2: *Twisting a cube using different methods. The top one is the classic skeleton-subspace deformation, the middle one is [Lipman et al. 2005]. The bottom one is our method.*

Lewis et al. [Lewis et al. 2000] provides a summary of more traditional deformation techniques, including the skeleton subspace deformation (SSD), also known as *skinning*. SSD transforms a given point using a linear combination of several rigid transformations corresponding to the influencing bones. This approach interpolate rotation in an unnatural way, whereas Moving Least Square doesn't exhibit this problem. Figure 2 shows twisting a cube. In the case of SSD, at the middle of the cube, the side length of the square cross section is $1/\sqrt{2}$ of the side length of the cross section at the two ends, i.e., the cube shrinks in the middle, which would be undesirable for the purpose of animation.

The concept of *as-rigid-as-possible* transformation is first introduced by Alexa et al.[Alexa et al. 2000] in the context of image morphing. They dissect source and target objects into isomorphic simplicial complexes. Then for every pair of source and target simplex, they factor the transformation between them into an optimal rotation and scale-shear parts, and interpolate the two individually. Igarashi et al.[Igarashi et al. 2005] apply the idea in the context of 2d shape manipulation. They triangulate the input image and solve a linear system whose size is equal to the number of vertices in the triangulation. Because of the need to solve a global system, they report that their method somewhat slow when there are 300 vertices on a 1 GHz machine. In contrast, Schaefer et al.[Schaefer et al. 2006], whose work we build upon, performs image deformation by solving a small 2x2 linear system locally at each grid point.

3 Deformation

We now describe the deformation process. The mathematical formulation is based on [Schaefer et al. 2006] with minor modification. However, extending everything from $\mathbb{R}^2$ to $\mathbb{R}^3$ requires different methods for solving the minimization problem in question.

The user controls the deformation using a set of N control points. Through out this paper we will use bold lowercase letters to denote elements in $\mathbb{R}^3$, represented as a column vector. Let $\mathbf{p}_i$ be the original position of the point handle i ($1 \le i \le N$) and $\mathbf{q}_i$ its deformed position. Given any point $\mathbf{x}$ in space, a function $d(\mathbf{x}, \mathbf{p}_i)$ is defined that measures the distance between $\mathbf{x}$ and $\mathbf{p}_i$. The simple Euclidean distance function can be used, although a better alternative exists for the purpose of deforming a 3d mesh (Section 4).

To define the deformation at point $\mathbf{x}$, we solve for the best rigid or similarity transformation $T_\mathbf{x}$ that minimizes

$$E = \sum_i w_i(\mathbf{x}) |T_\mathbf{x}(\mathbf{p}_i) - \mathbf{q}_i|^2 \tag{1}$$

where the weights $w_i(\cdot)$ is of the form

$$w_i(\mathbf{x}) = d(\mathbf{p}_i, \mathbf{x})^{-2\alpha},$$

with α being a *fall-off* parameter controlling how strongly the deformation at $\mathbf{x}$ is influenced by far away (as measured by d) point handles.

The deformation at $\mathbf{x}$ is simply defined to map $\mathbf{x}$ to $T_\mathbf{x}(\mathbf{x})$. Note that if $\mathbf{x} = \mathbf{p}_i$ for some i, then $w_i(\mathbf{x}) = \infty$, so $T_\mathbf{x}(\mathbf{p}_i) = \mathbf{q}_i$ (assuming the $\mathbf{p}_i$'s are distinct). In other words, the deformation is *interpolating* [Schaefer et al. 2006].

3.1 Rigid Transformation

Now we describe how to solve for the optimal transformation $T_\mathbf{x}$ at $\mathbf{x}$. We first concentrate on the case when $T_\mathbf{x}$ is completely rigid.

Schaefer et al. [Schaefer et al. 2006] finds a closed form expression for $T_\mathbf{x}$ in 2d by first deriving a closed form expression for the optimal *similarity* transformation, and then normalizing the similarity transformation. They prove that the optimal similarity transformation and the optimal rigid transformation for a point in space is related by a uniform scaling, i.e., their rotational components are the same. However, we were not able to easily adapt their derivation to the 3d case.

Fortunately, it turns out that finding the optimal rigid transformation minimizing E (1) is a long solved problem. Eggert et al. [Eggert et al. 1997] give a survey on the topic. The problem has been known as the "Procrustes problem" and dates back to the 1960s [Schönemann 1996]. For completeness, we give a derivation of its solution using singular value decomposition. More recently, Müller et al.[Müller et al. 2005] uses the same formulation to perform rigid shape matching in their physics simulation system, although they use a different, physically based weighting scheme.

Fix $\mathbf{x}$. $T_\mathbf{x}$, being a rigid transformation, has the form $T_\mathbf{x}(\mathbf{y}) = M\mathbf{y} + \mathbf{r}$, where M is a 3 by 3 orthogonal matrix, and $\mathbf{r}$ is the translational component. Let $\mathbf{p}_*$ and $\mathbf{q}_*$ be the weighted centroids of $\mathbf{p}_i$'s and $\mathbf{q}_i$'s:

$$\mathbf{p}_* = \frac{\sum_i w_i(\mathbf{x})\mathbf{p}_i}{\sum_i w_i(\mathbf{x})}, \quad \mathbf{q}_* = \frac{\sum_i w_i(\mathbf{x})\mathbf{q}_i}{\sum_i w_i(\mathbf{x})}.$$

The translational component of $T_\mathbf{x}$ must map $\mathbf{p}_*$ to $\mathbf{q}_*$, i.e., $\mathbf{r} = \mathbf{q}_* - \mathbf{p}_*$. This leaves only M to be determined. Let $\alpha_i = w_i(\mathbf{x})^{\frac{1}{2}}$, $\tilde{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{p}_*$, $\tilde{\mathbf{q}}_i = \mathbf{q}_i - \mathbf{q}_*$, $P = \alpha_i(\tilde{\mathbf{p}}_1 \cdots \tilde{\mathbf{p}}_N)$, $Q = \alpha_i(\tilde{\mathbf{q}}_1 \cdots \tilde{\mathbf{q}}_N)$ (P and Q are 3 by N matrices). Then

$$\begin{aligned} E &= \sum_i w_i(\mathbf{x}) |M(\tilde{\mathbf{p}}_i) - \tilde{\mathbf{q}}_i|^2 = \sum_i |M(\alpha_i\tilde{\mathbf{p}}_i) - \alpha_i\tilde{\mathbf{q}}_i|^2 \\ &= \|MP - Q\|_F^2 = \mathrm{tr}((MP-Q)^t(MP-Q)) \\ &= \mathrm{tr}(P^tP) + \mathrm{tr}(Q^tQ) - 2\mathrm{tr}(Q^tMP), \end{aligned}$$

where $\|\cdot\|_F$ is the Frobenius norm. Since P and Q are constant, minimizing E corresponds to maximizing $\zeta = \mathrm{tr}(Q^t M P) = \mathrm{tr}(MPQ^t)$. Note that PQ^t has a singular value decomposition $PQ^t = U\Lambda V^t$ such that $\Lambda = \mathrm{diag}(\lambda_1, \lambda_2, \lambda_3)$ is diagonal with non-negative entries, and U, V are orthogonal. Thus

$$\zeta = \mathrm{tr}(MPQ^t) = \mathrm{tr}(MU\Lambda V^t) = \mathrm{tr}(U^t M^t V \Lambda).$$

Write $N = U^t M^t V$, then N is orthogonal since U, M, and V are orthogonal. It follows that $|N_{i,j}| \leq 1$, and

$$\zeta = \mathrm{tr}(N\Lambda) = \sum_{i=1}^{3} N_{i,i}\lambda_i \leq \sum_{i=1}^{3} \lambda_i.$$

Hence, ζ is maximized when $N = \mathbf{1} \Leftrightarrow M = VU^t$.

In summary, to calculate $T_{\mathbf{x}}(\mathbf{x})$. First calculate $\mathbf{p}_*$ and $\mathbf{q}_*$. Then perform singular value decomposition on $PQ^t = \sum_i w_i(\mathbf{x})(\mathbf{p}_i - \mathbf{p}_*)(\mathbf{q}_i - \mathbf{q}_*)^t = U\Lambda V^t$. Finally let $T_{\mathbf{x}}(\mathbf{x}) = VU^t(\mathbf{x} - \mathbf{p}_*) + \mathbf{q}_*$.

3.2 Similarity Transformation

Now we extend the above derivation to the case where $T_{\mathbf{x}}$ is allowed to be a similarity transformation. As in the previous section, $T_{\mathbf{x}}(\mathbf{y}) = \tilde{M}\mathbf{y} + \mathbf{r}$, whereas $\tilde{M} = \rho M$ for some orthogonal 3 by 3 matrix M and a positive scaling factor $\rho > 0$. Then

$$E = \mathrm{tr}(P^t P)\rho^2 + \mathrm{tr}(Q^t Q) - 2\mathrm{tr}(MPQ^t)\rho,$$

so minimizing E is the same as minimizing $\eta = \mathrm{tr}(P^t P)\rho^2 - 2\mathrm{tr}(MPQ^t)\rho$. Again substitute PQ^t by its singular value decomposition $U\Lambda V^t$ and we have

$$\eta \geq \mathrm{tr}(P^t P)\rho^2 - 2\mathrm{tr}(\Lambda)\rho,$$

where the equality is reached when $M = VU^t$. Then η is quadratic in ρ and is minimized when $\rho = \mathrm{tr}(\Lambda)/\mathrm{tr}(P^t P)$.

In our prototype implementation, we provides a user specified parameter s that controls the maximally allowed scaling. At each point ρ is clamped to $[1 - s, 1/(1 - s)]$, so $s = 0$ corresponds to completely rigid transformation, whereas $s = 1$ corresponds to similarity transformation with arbitrary uniform scaling.

3.3 Numerical Considerations

The mapping from the set of possible handle positions to the optimal transformation at a particular point is actually not a continuous function. Consider three point handles a, b, c arranged in the shape of the letter L, as shown in Figure 3. As we move c to the position of c', the optimal transformations for points around b changes from close to identity to a 180° rotation along the line connection a and b. By making b and c really close to each other, a movement of c by some tiny amount can cause the optimal transformation to jump from identity to a large amount of rotation.

We find that this discontinuity can cause artifacts in practice. Again consider the above L shaped configuration and suppose the handle a is placed quite far away from b and c. Moreover, suppose the fall-off exponential α is set quite high. Then a small movement of c might cause a huge rotation for points around b. However, because of the large fall-off and floating point round-off errors, for points around a, it would be as if b and c haven't changed their relative positions. This results in visible discontinuity across the surface as shown in Figure 3.

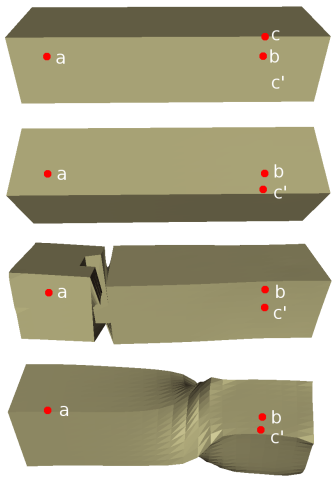

Figure 3: *Effects of fall-off exponential and damping. From top to bottom are: (1) before deformation; (2) after deformation, small fall-off, and no damping; (3) after deformation, big fall-off, and no damping; (4) after deformation, big fall-off, and with damping.*

One approach to alleviate such a problem is to introduce a *damping* factor which prevents the optimal transformation to change too fast. More specifically, we add an additional term to our minimizing goal:

$$E' = E + \delta\|M - M_p\|_F^2 = \|MP' - Q'\|_F^2$$

where M_p is the matrix M from the last frame, δ is a scalar controlling how much damping is performed, $P' = (P \quad \delta^{\frac{1}{2}} I)$ is the previous P with a 3 by 3 identity matrix scaled by $\delta^{\frac{1}{2}}$ juxtaposed on the right, and $Q' = (Q \quad \delta^{\frac{1}{2}} M_p)$ is the previous Q with M_p scaled by $\delta^{\frac{1}{2}}$ juxtaposed on the right. The damping will prevent sudden rotational jump from frame to frame, but also makes the deformation look a little laggy. This kind of strategy is also used in inverse kinematics algorithms to prevent jumpy solutions near singularities [Buss and Kim 2005].

We note, however, that in practice we rarely encounter such discontinuities when there are more than four or five point handles. Thus δ is rarely set to a non-zero value in real-world applications.

4 Distance Calculation

In Section 3, we use the function d to measure the distance between a handle and a point in space. This distance, together with the fall-off exponential α, determines how much influence the handle has on the point.

When we are deforming a model represented as a polygonal mesh, it is typical for the model to have distinct parts. For example, the model of a horse would have legs connected to the body. If we place handles on the four legs of the horse, although a handle can be geometrically very close to a leg it is not placed on, the handle should have minimal influence on that leg.

Ling and Jacobs [Ling and Jacobs 2007] propose the definition of *inner distance* for 2d shapes, defined as the "length of the shortest path between landmark points within the shape silhouette". This seems to capture what we intuitively would like our distance function to be. In our horse example, the distance between a handle and a vertex on a leg that the handle is not placed on would then be the

length of a polyline going first from the handle into the body, and then from the body to the other leg.

Ling and Jacobs's method works as following: They create a graph containing all vertices from the shape silhouettes. An edge connecting two vertices exist in the graph if these two vertices are "visible" from each other, i.e., the line segment connecting them is not intersected by any edge of the shape silhouette. Then to find the inner distance between two vertices, they just find the shortest path between them within the defined graph.

A direct adaption of Ling and Jacob's method to 3d would be prohibitively expensive. For every pair of mesh vertices, the line segment connecting them would have to be checked against every face in the mesh, a $O(n^3)$ process, where n is the number of vertices. Also, the resulting shortest distance computed will still only be an approximation of the actual inner distance, since the shortest polyline connecting two points in 3d can actually cut through a face or an edge.

We propose a different kind of distance function that is similar in spirit to "inner distance" in that it tries to stay within the mesh. It is computed as following: To find the distance between a point handle and a mesh vertex, first create a graph consisting of all the edges and vertices in the mesh, together with a new node representing the point handle. Next for every mesh vertex, create an edge between the point handle and the mesh vertex if they're "visible" to each other, i.e., the line segment connecting them is not intersected by any mesh face. Finally, the distance between the point handle and any mesh vertex is define to be the length of their shortest path in the graph. This results in an $O(n^2)$ algorithm and is practical for meshes containing thousands of vertices. The new distance function works surprisingly well in practice, as demonstrated in Figure 4.

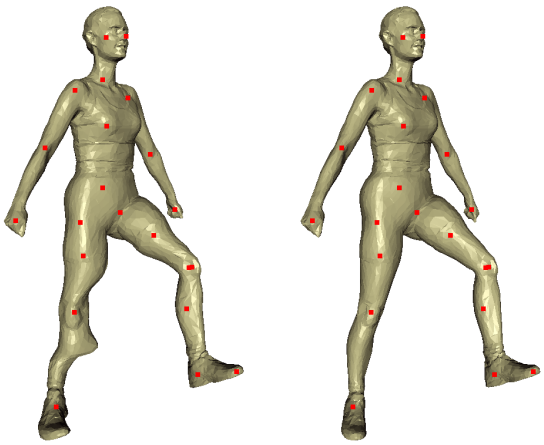

Figure 4: *Comparison of deformation based on Euclidean distance function (left) and our distance function (right).*

Finally, we note that calculating the visibility of mesh vertices from a point is actually the classic *visible surface determination* problem. One way to get approximate visibility information without any precalculation is to just use a z-buffer algorithm. This would reduce the running time of the graph construction phase of the algorithms from $O(n^2)$ down to $O(Cn)$, where the constant C corresponds to the average time needed to rasterization a triangle. This would make the method practical for meshes containing hundreds of thousands of vertices.

5 Results and Discussions

In our prototype implementation, we use Jacobian iterations[Golb and Loan 1991] to calculate the singular value decomposition of the 3 by 3 matrix for each point we want to deform. On a Pentium-M 1.8GHz laptop, we were able to interactively deform models of up to 100k vertices.

We have also implemented the improved distance function, but without the z-buffer optimization. In that case, prepossessing of several minutes is required to calculate the distance from each mesh vertex to each point handle for a mesh of 6k vertices, although we expect the performance to improve once we implement the z-buffer optimization, reducing running time from $O(n^3)$ to $O(n^2)$ where n is the size of the mesh.

Using our method, we were able to realistically deform complex articulate models such as a horse and a human, as shown in Figure 1 within a matter of minutes (excluding the time needed to precalculate the distance function).

One problem that we noticed with our method is that the implied least-square minimization sometimes causes unexpected deformation, often in the form a weird looking limb. In such cases, adding a second point handle allows us to fix the situation and fine tune the local deformation.

6 Conclusions and Future Work

We have presented a method for deforming 3d models using point handles. The deformation is based on Moving Least Square: each point in space is transformed by a rigid or similarity transformation that tries to map the point handles to their deformed positions. Such a transformation is optimal in the sense of weighted least-square. We demonstrate how to solve the optimization problem using singular value decomposition, thus making the technique available to deformation in all dimensions. For the purpose of mesh deformation, a new weighting scheme is introduced that determines the influence of point handles on vertices based on approximate mesh geodesics. In practice, this scheme gives visually pleasing result and would be of use to the traditional skeleton-subspace deformation method.

In the future, we plan to work on better user interfaces for manipulating the point handles. Currently one inconvenience of our method is that the user often needs to place a pair of handles at a joint to fine tune the local deformation. Managing the pair of handles (e.g., keeping them at constant distance from each other and properly oriented) then becomes quite a hassle. A smart user interface should be able to take that burden off the user. More generally, it might be useful to maintain some kind of skeleton-type organization among the point handles. Right now if the user drags a handle defined on the hand of a character, the handle for the elbow will not move along. If the handles are connected through an explicit skeleton, inverse kinematics algorithms can then be used to move the elbow handle as well.

References

ALEXA, M., COHEN-OR, D., AND LEVIN, D. 2000. As-rigid-as-possible shape interpolation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 157–164.

BOTSCH, M., AND KOBBELT, L. Real-time shape editing using radial basis functions. *Eurographics 2005*.

BUSS, S. R., AND KIM, J.-S. 2005. Selectively damped least squares for inverse kinematics. *Journal of Graphics Tools 10*, 3, 37–49.

EGGERT, D. W., LORUSSO, A., AND FISHER, R. B. 1997. Estimating 3-d rigid body transformations: a comparison of four major algorithms. *Mach. Vision Appl. 9*, 5-6, 272–290.

GOLB, G. H., AND LOAN, C. F. V. 1991. *Matrix Computation*, 3 ed.

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. *ACM Trans. Graph. 25*, 3, 1126–1134.

IGARASHI, T., MOSCOVICH, T., AND HUGHES, J. F. 2005. As-rigid-as-possible shape manipulation. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 1134–1141.

LEWIS, J. P., CORDNER, M., AND FONG, N. 2000. Pose space deformation: a unified approach to shape interpolation and skeleton-driven deformation. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 165–172.

LING, H., AND JACOBS, D. W. 2007. Shape classification using the inner-distance. *IEEE Trans. Pattern Anal. Mach. Intell. 29*, 2, 286–299.

LIPMAN, Y., SORKINE, O., LEVIN, D., AND COHEN-OR, D. 2005. Linear rotation-invariant coordinates for meshes. *ACM Trans. Graph. 24*, 3, 479–487.

MACCRACKEN, R., AND JOY, K. I. 1996. Free-form deformations with lattices of arbitrary topology. In *SIGGRAPH '96: Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, ACM Press, New York, NY, USA, 181–188.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. In *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, ACM Press, New York, NY, USA, 471–478.

SCHAEFER, S., MCPHAIL, T., AND WARREN, J. 2006. Image deformation using moving least squares. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Papers*, ACM Press, New York, NY, USA, 533–540.

SCHÖNEMANN, P. H. 1996. A generalized solution of the orthogonal procrustes problem. *Psychometrika 31*, 1–10.