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

3D As-Rigid-As-Possible Deformations Using MLS

Alvaro Cuno, Claudio Esperança, Antonio Oliveira, Paulo Roma Cavalcanti

Systems Engineering and Computer Science Program - COPPE
Federal University of Rio de Janeiro
Tel.: +55-21-25628572
Fax: +55-21-25628676
e-mail: {alvaro,esperanc,oliveira,roma}@lcg.ufrj.br

Abstract We present a formulation for achieving "as rigid as possible" deformations of 3D models using a Moving Least Squares (MLS) approach. This research was inspired by the work of Schaefer et al.[25] which describes an approach solving a 2D version of the same problem. Our main contribution is showing how the problem may be efficiently handled in 3D for both point and line segment constraints. Our prototype implementation is capable of performing close to real-time deformations of models with a few thousand vertices.

Key words Mesh deformation Moving Least Squares (MLS) Rigid transformations Animation

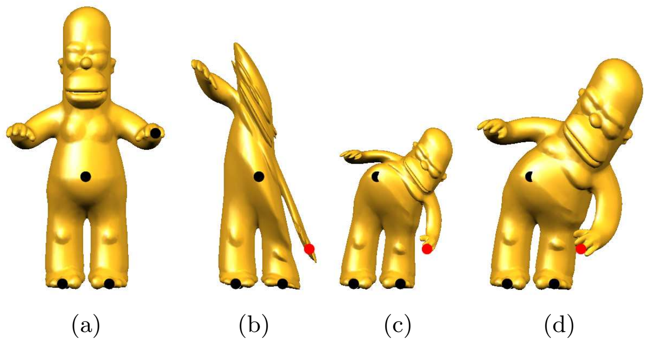

Fig. 1 Deformations using MLS. (a) Original model[1] and control points. Deformation given by (a) general affine transformations, (c) similarity transformations, and (d) rigid transformations.

1 Introduction

The problem of deforming 3D models has undergone close scrutiny in recent years. Operations of this kind find application in modeling, animation, shape interpolation, illustrative visualization, surgical simulation and many others. In general, the problem consists of altering the shape of a given model in a smooth way using some deformation paradigm. This modification should yield predictable results and, in many cases, conform to certain restrictions such as maintaining the overall volume of the model, preserving details or avoiding self-intersections. Additionally, it is generally required that the deformation process is performed at interactive rates.

In this work, we investigate the use of deformation for shape *posing*, that is, trying to obtain several variations of the same model without overly affecting its recognizable features. Among the many approaches proposed in the last few years, those which try to obtain so-called *as rigid as possible* deformations are of special interest, since they tend to produce more physically plausible results by avoiding unnatural shearing and non-uniform scaling of the model. In particular, by making use of a Moving Least Squares approach, Schaefer et al. [25] have recently presented a way of computing such deformations for 2D images. The deformation is split into two components: a translation and a rotation transformation for which the authors derive closed-form equations. These depend only on a set of point to point or line segment to line segment constraints which are used to model the overall deformation.

The extension of this technique to 3D is fairly simple for the translation component, but solving the rotation transformation leads to an eigenvector problem. Actually, the deformation problem can be seen as the registration problem between two corresponding 3D point sets, for which several solutions have been proposed, either iterative [24] and closed-form [10]. We are interested, however, in a simple closed-form solution that can be easily adapted for use in interactive deformations of medium size models. In this paper we show how this can be achieved by deriving a closed-form solution for the problem. The problem is studied as an optimization problem, that minimizes a sum of weighted squared errors, with solutions defined by a rotation axis and angular parameters. These are found by computing the

[1] Thanks to the AIM@SHAPE shape repository.

largest real root of a depressed quartic equation. Our approach has been implemented and we show a few examples of deformations obtained with our prototype.

2 Related work

Since the 80s, many researchers have investigated algorithms and techniques for applying meaningful geometric deformations to 3D models [4,6]. A recent survey [1] classifies deformation techniques into freeform methods, which are mostly aimed at producing global smooth deformations, and detail-preserving methods. The former class is further divided into surface-based and space-based methods, while the latter class includes, among others, multiresolution techniques and methods based on differential coordinates.

Freeform surface-based techniques try to obtain a displacement function of the form $f : \mathcal{S} \mapsto \Re^3$, which maps surface $\mathcal{S}$ into a deformed version $\mathcal{S}' = \{\mathbf{p} + f(\mathbf{p}) \mid \mathbf{p} \in \mathcal{S}\}$. Usually, f is modelled by means of some energy minimization process defined on the surface subject to a set of conditions on its border. The deformation is controlled by deformation handles, most commonly a set of points $\mathbf{p}_i$ such that $f(\mathbf{p}_i) = \mathbf{q}_i$, where $\mathbf{q}_i$ refer to the new positions of $\mathbf{p}_i$ [7,17,23,29].

While surface-based methods are quite flexible and support different smoothness criteria, their computational complexity and numerical stability are strongly related to the size and quality of the input mesh. In contrast, space-based techniques deform the 3D space as a whole, thus affecting the shape of models contained therein indirectly [8,14,26]. Such methods employ a deformation function of the form $f : \Re^3 \mapsto \Re^3$ to transform all points of the original surface $\mathcal{S}$ to a new, deformed surface $\mathcal{S}' = \{f(\mathbf{p}) \mid \mathbf{p} \in \mathcal{S}\}$.

Differential methods are characterized by applying modifications to differential rather than spatial coordinates of the models. Once the new values for handle positions or normals are specified, a deformed model is reconstructed by considering the desired differential properties and minimizing the model distortion. Commonly used differential representations are: gradient fields [30, 31], Laplacian coordinates [19,27,30–32], and first/second fundamental forms of a surface [18,20].

Another important category of deformation techniques are the so-called multiresolution methods [17,21,22]. Here, the key idea is to use mesh-based signal processing techniques in order to apply the deformation to a low-frequency version of the model and later reconstructing the high-frequency details.

Many of the deformation techniques proposed in the past rely on some kind of energy minimization process, where the energy function is defined in such a way as to measure the distortion introduced by the deformation. A related concept characterizes the so-called *as rigid as possible* approaches, in which the deformation function is defined as a smoothly varying affine transformation applied on the model in such a way as to minimize local scaling and shearing. This idea has recently been applied to the problem of deforming images and meshes in 2D [2,15,25].

In this paper, we focus on obtaining an interactive solution for producing as-rigid-as-possible deformations in 3D by adapting the Moving Least Squares approach of Schaefer et al. [25]. Thus, the main problem is that of obtaining a rigid transformation which best approximates a mapping in $\Re^3$ for a given set of points. This problem is also known as point registration, or, more specifically, the Absolute Orientation Problem. Analytical solutions have been proposed, for instance, based on SVD (*Singular Value Decomposition*) [3], quaternions [12,16], orthonormal matrices [13] and dual quaternions [28]. Another related group of techniques includes those based on iterative schemes such as the ICP (*Iterative Closest Point*) algorithm [5] and its variants. Differently from these approaches, however, the solution described below directly obtains the axis and angular parameters of the rotation. Our method requires a smaller number of computations than matricial and even quaternion-based approaches. Notice that, in our application, a new rigid transformation must be computed for *every* vertex of the mesh, and thus it must be as efficient as possible.

3 Moving least squares deformation

The Moving Least Squares (MLS) formulation can be thought of as an extension of the traditional Least Squares minimization technique. Rather than finding a global optimum solution for the problem, MLS tries to find continuously varying solutions for all points of the domain. Let us define the deformation operation as a transformation which maps a set of points $\{\mathbf{p}_i\}$ of the domain onto new positions $\{\mathbf{q}_i\}$. Thus, solving the problem for a given point $\mathbf{v} = [x\,y\,z]$ of the domain can be reduced to finding the best transformation $l_\mathbf{v}(\mathbf{x})$ that minimizes

$$\sum_i w_i |l_\mathbf{v}(\mathbf{p}_i) - \mathbf{q}_i|^2, \qquad (1)$$

where w_i are weights of the form $w_i = |\mathbf{p}_i - \mathbf{v}|^{-u}$ for some integer constant $u > 0$.

Let us define the deformation function f as $f(\mathbf{v}) = l_\mathbf{v}(\mathbf{v})$. We observe that when $\mathbf{v}$ is close to some constraint $\mathbf{p}_i$, then w_i tends to infinity, which means that f is interpolating with respect to the constraint points, i.e., $f(\mathbf{p}_i) = \mathbf{q}_i$. Further, if $\mathbf{q}_i = \mathbf{p}_i$, then $f(\mathbf{v}) = \mathbf{v}$, for all $\mathbf{v}$, meaning that, in this case, f is the identity function. Finally, it can be shown that f is smooth everywhere for $u \geq 2$. This defines the Moving Least Squares minimization in which the sought transformation $l_\mathbf{v}$ depends on the point of evaluation $\mathbf{v}$.

4 As rigid as possible MLS

By imposing different additional requirements on the form of $l_{\mathbf{v}}$, we may obtain different results. We may require, for instance, that $l_{\mathbf{v}}$ is a general affine transformation, in which case the classical normal equations solution can be applied directly [25]. For obtaining deformations which are *as rigid as possible*, $l_{\mathbf{v}}$ must be constrained to be a rigid transformation, i.e., $l_{\mathbf{v}}$ must be of the form: $l_{\mathbf{v}}(\mathbf{x}) = \mathbf{x}\mathrm{R} + \mathrm{T}$, where R is a rotation matrix and T is a translation vector. Solving for T yields $\mathrm{T} = \mathbf{q}* - \mathbf{p}*\mathrm{R}$, where $\mathbf{q}*$ and $\mathbf{p}*$ are the weighted centroids of $\{\mathbf{q}_i\}$ and $\{\mathbf{p}_i\}$ respectively:

$$\mathbf{p}* = \frac{\sum_i w_i\, \mathbf{p}_i}{\sum_i w_i}, \quad \mathbf{q}* = \frac{\sum_i w_i\, \mathbf{q}_i}{\sum_i w_i}. \tag{2}$$

This permits us to factor out the translation from (1) by rewriting it as

$$\sum_i w_i |\hat{\mathbf{p}}_i \mathrm{R} - \hat{\mathbf{q}}_i|^2, \tag{3}$$

where $\hat{\mathbf{q}}_i = \mathbf{q}_i - \mathbf{q}*$ and $\hat{\mathbf{p}}_i = \mathbf{p}_i - \mathbf{p}*$. Expanding (3) then yields

$$-2\sum_i w_i \hat{\mathbf{q}}_i \mathrm{R} \hat{\mathbf{p}}_i^{\mathsf{T}} + \sum_i w_i \|\hat{\mathbf{q}}_i\|^2 + \sum_i w_i \|\hat{\mathbf{p}}_i\|^2.$$

Since the last 2 terms are constant, we infer that R minimizes (3) if and only if it maximizes

$$\sum_i w_i \hat{\mathbf{q}}_i \mathrm{R} \hat{\mathbf{p}}_i^{\mathsf{T}}. \tag{4}$$

4.1 3D rigid transformations

In 3D space, R may be defined as a rotation of an angle α around an axis $\mathbf{e}$. Applying such a rotation on a vector $\mathbf{v}$ yields:

$$\begin{aligned} \mathrm{R}_{\mathbf{e},\alpha}(\mathbf{v}^{\mathsf{T}}) = &\, \mathbf{e}^{\mathsf{T}}\mathbf{e}\,\mathbf{v}^{\mathsf{T}} + \\ &\cos(\alpha)(\mathrm{I} - \mathbf{e}^{\mathsf{T}}\mathbf{e})\mathbf{v}^{\mathsf{T}} + \\ &\sin(\alpha)\begin{pmatrix} 0 & -\mathbf{e}_z & \mathbf{e}_y \\ \mathbf{e}_z & 0 & -\mathbf{e}_x \\ -\mathbf{e}_y & \mathbf{e}_x & 0 \end{pmatrix} \mathbf{v}^{\mathsf{T}}. \end{aligned} \tag{5}$$

By replacing this definition of R in (4) we obtain

$$\sum_i w_i \hat{\mathbf{q}}_i \mathrm{R} \hat{\mathbf{p}}_i^{\mathsf{T}} = \mathbf{e}\,\mathrm{M}\,\mathbf{e}^{\mathsf{T}} + \cos(\alpha)(\mathrm{E} - \mathbf{e}\,\mathrm{M}\,\mathbf{e}^{\mathsf{T}}) + \sin(\alpha)\mathrm{V}\,\mathbf{e}^{\mathsf{T}},$$

where

$$\begin{aligned} \mathrm{M} = &\textstyle\sum_i w_i \hat{\mathbf{q}}_i^{\mathsf{T}} \hat{\mathbf{p}}_i = \begin{pmatrix} \sum_i w_i \hat{\mathbf{q}}_{ix}\hat{\mathbf{p}}_{ix} & \sum_i w_i \hat{\mathbf{q}}_{ix}\hat{\mathbf{p}}_{iy} & \sum_i w_i \hat{\mathbf{q}}_{ix}\hat{\mathbf{p}}_{iz} \\ \sum_i w_i \hat{\mathbf{q}}_{iy}\hat{\mathbf{p}}_{ix} & \sum_i w_i \hat{\mathbf{q}}_{iy}\hat{\mathbf{p}}_{iy} & \sum_i w_i \hat{\mathbf{q}}_{iy}\hat{\mathbf{p}}_{iz} \\ \sum_i w_i \hat{\mathbf{q}}_{iz}\hat{\mathbf{p}}_{ix} & \sum_i w_i \hat{\mathbf{q}}_{iz}\hat{\mathbf{p}}_{iy} & \sum_i w_i \hat{\mathbf{q}}_{iz}\hat{\mathbf{p}}_{iz} \end{pmatrix}, \\ \mathrm{E} = &\textstyle\sum_i w_i \hat{\mathbf{q}}_i \cdot \hat{\mathbf{p}}_i = \mathrm{Trace}(\mathrm{M}), \\ \mathrm{V} = &\textstyle\sum_i w_i \hat{\mathbf{p}}_i \times \hat{\mathbf{q}}_i = (\mathrm{M}_{23} - \mathrm{M}_{32} \quad \mathrm{M}_{31} - \mathrm{M}_{13} \quad \mathrm{M}_{12} - \mathrm{M}_{21}). \end{aligned} \tag{6}$$

4.2 Optimization problem

Thus, the optimization problem can be written as

$$\begin{aligned} \max\ \mathrm{F}(\mathbf{e},\alpha) &= \mathbf{e}\,\mathrm{M}\,\mathbf{e}^{\mathsf{T}} + \cos(\alpha)(\mathrm{E} - \mathbf{e}\,\mathrm{M}\,\mathbf{e}^{\mathsf{T}}) + \sin(\alpha)\mathrm{V}\,\mathbf{e}^{\mathsf{T}} \\ \text{s.t.} \quad & \|\mathbf{e}\| = 1, \\ & \cos(\alpha)^2 + \sin(\alpha)^2 = 1. \end{aligned} \tag{7}$$

By considering the optimality conditions (Kuhn-Tucker) for this problem, the solutions must satisfy

$$(1 - \cos(\alpha))\,\mathbf{e}(\mathrm{M} + \mathrm{M}^{\mathsf{T}}) + \sin(\alpha)\mathrm{V} = k_1\,\mathbf{e}, \tag{8}$$

$$\begin{pmatrix} \mathrm{E} - \mathbf{e}\,\mathrm{M}\,\mathbf{e}^{\mathsf{T}} \\ \mathrm{V}\,\mathbf{e}^{\mathsf{T}} \end{pmatrix} = k_2 \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \end{pmatrix}. \tag{9}$$

If these conditions are satisfied with $\alpha = 0$ or $k_2 = 0$ then $\mathrm{F}(\mathbf{e},\alpha) = \mathrm{E}$. While searching for $(\mathbf{e},\alpha)$ such that $\mathrm{F}(\mathbf{e},\alpha) > \mathrm{E}$, we can, therefore, assume that both these conditions do not hold. If that search does not succeed, the null rotation is a solution of (7).

From equation (9), we have $\sin(\alpha) = \mathrm{V}\mathbf{e}^{\mathsf{T}}/k_2$, and, defining $\mathrm{N} = \mathrm{M} + \mathrm{M}^{\mathsf{T}}$, we may rewrite equation (8) as

$$(\mathrm{N} + a\mathrm{V}^{\mathsf{T}}\mathrm{V})\mathbf{e}^{\mathsf{T}} = \lambda\,\mathbf{e}^{\mathsf{T}}, \tag{10}$$

where

$$a = \frac{1}{k_2(1 - \cos(\alpha))} \quad \text{and} \quad \lambda = \frac{k_1}{(1 - \cos(\alpha))}. \tag{11}$$

In summary, the optimal rotation axis will correspond to the eigenvector of matrix $(\mathrm{N} + a\mathrm{V}^{\mathsf{T}}\mathrm{V})$ associated with eigenvalue λ. We notice, however, that the values of a and λ depend on k_1 and k_2, making this a non-standard eigenvalue problem. To reduce the indetermination of the problem, it is possible to show that a and λ are related by

$$a = \frac{1}{\lambda - 2\mathrm{E}}. \tag{12}$$

4.3 Eigenvalue determination

An analysis of equation (10) reveals that since λ is an eigenvalue of $(\mathrm{N} + a\mathrm{V}^{\mathsf{T}}\mathrm{V})$, it must also be a root of the characteristic polynomial

$$\begin{aligned} P(\lambda) = &\lambda^3 - (\mathrm{Trace}(\mathrm{N}) + a\|\mathrm{V}\|^2)\,\lambda^2 \\ &+ \left[(\tfrac{1}{2}(\mathrm{Trace}(\mathrm{N})^2 - \|\mathrm{N}\|^2) + a(\|\mathrm{V}\|^2\mathrm{Trace}(\mathrm{N}) - \mathrm{V}\mathrm{N}\mathrm{V}^{\mathsf{T}})\right]\lambda \\ &- \det(\mathrm{N})(1 + \mathrm{V}\mathrm{N}^{-1}\mathrm{V}^{\mathsf{T}}a). \end{aligned}$$

By substituting equation (12) and noting that $\mathrm{Trace}(\mathrm{N}) = 2\mathrm{Trace}(\mathrm{M}) = 2\mathrm{E}$ and that $2\|\mathrm{M}\|^2 = \frac{1}{2}\|\mathrm{N}\|^2 + \|\mathrm{V}\|^2$, then equation $P(\lambda) = 0$ becomes

$$\begin{aligned} &\lambda^4 - 4\mathrm{E}\,\lambda^3 + (6\mathrm{E}^2 - 2\|\mathrm{M}\|^2)\,\lambda^2 \\ &+ (4(\|\mathrm{M}\|^2 - \mathrm{E}^2)\mathrm{E} - 2\mathrm{V}\mathrm{M}\mathrm{V}^{\mathsf{T}} - \det(\mathrm{N}))\,\lambda \\ &+ \det(\mathrm{N})(2\mathrm{E} - \mathrm{V}\mathrm{N}^{-1}\mathrm{V}^{\mathsf{T}}) = 0. \end{aligned}$$

The third degree term may be eliminated by effecting a variable change of the form $y = \lambda - \mathrm{E}$. Thus, we obtain the following equation in y which allows for a closed-form solution:

$$\begin{gathered} y^4 - 2\|\mathrm{M}\|^2 y^2 - 8\det(\mathrm{M})\, y - \mathrm{E}^4 + 2\|\mathrm{M}\|^2\mathrm{E}^2 \\ -8\det(\mathrm{M})\mathrm{E} + \det(\mathrm{N})(2\mathrm{E} - \mathrm{V}\mathrm{N}^{-1}\mathrm{V}^\mathsf{T}) = 0. \end{gathered} \tag{13}$$

The independent term in (13) is more easily obtained by the expression

$$\|\mathrm{M}\|^4 - 4\sum_i\sum_{j<i}\|\mathrm{M}_i \times \mathrm{M}_j\|^2,$$

where M_k is the k^{th} column of M.

It can be shown that if we use the optimality conditions (8) and (9) in the objective function (7), the latter becomes $\lambda - \mathrm{E} = y$, meaning that a value of $\mathbf{e}$ which satisfies the optimality conditions is given by the value of y itself. Since our ultimate goal is to satisfy the objective function, we must look for a value for y which is the largest real solution of the polynomial (13).

4.4 Solving the depressed quartic equation

If a depressed quartic polynomial $y^4 + ay^2 + by + c$ is factored as $(y^2 + py + q)(y^2 - py + s)$, then p must satisfy $p^6 + 2ap^4 + (a^2 - 4c)p^2 - b^2 = 0$ which is a cubic equation in $p^2 = z$. If the depressed polinomial is (13), this cubic polinomial becomes

$$\begin{gathered} Q(z) = z^3 - 4\|\mathrm{M}\|^2 z^2 + \\ 16(\textstyle\sum_i\sum_{j<i}(\|\mathrm{M}_i\|\,\|\mathrm{M}_j\|)^2 - (\mathrm{M}_i\mathrm{M}_j^\mathsf{T})^2)z - 64\det(\mathrm{M})^2. \end{gathered}$$

Now, let Q^* be the characteristic polynomial of $(\mathrm{M}^\mathsf{T}\mathrm{M})$. Since $Q(z) = 64Q^*(z/4)$, then a root r of Q must be related to some eigenvalue μ of $(\mathrm{M}^\mathsf{T}\mathrm{M})$ by $r = 4\mu$. This implies that $r \geq 0$, since all eigenvalues of the positive semi-definite matrix $(\mathrm{M}^\mathsf{T}\mathrm{M})$ must be non-negative.

Let $p = \sqrt{r}$. Having p, y can then be obtained as the largest value between

$$\begin{gathered} -p/2 + \sqrt{-p^2/4 + \|\mathrm{M}\|^2 - 4\det(\mathrm{M})/p} \text{ and} \\ p/2 + \sqrt{-p^2/4 + \|\mathrm{M}\|^2 + 4\det(\mathrm{M})/p}. \end{gathered}$$

Observe that $Q(z)$ is essencially the same polynomial solved by a matricial method like Horn's in order to find the eigenvalues of $\mathrm{M}^\mathsf{T}\mathrm{M}$. Moreover, if $\det(\mathrm{M}) > 0$, then the value of y obtained by that process is $\mathrm{Trace}((\mathrm{M}^\mathsf{T}\mathrm{M})^{1/2})$. Otherwise, it is $\mathrm{Trace}((\mathrm{M}^\mathsf{T}\mathrm{M})^{1/2}) - 2\sqrt{\mu_{min}}$, where μ_{min} is the smallest eigenvalue of $\mathrm{M}^\mathsf{T}\mathrm{M}$.

From the reasoning above, we can realize that finding λ demands no more effort than obtaining all the eigenvalues of $\mathrm{M}^\mathsf{T}\mathrm{M}$, as required by matricial approaches. From this point on, the method described here is advantageous. While Horn's approach determines a basis of unit eigenvectors of $\mathrm{M}^\mathsf{T}\mathrm{M}$ using it to compute $\mathrm{M}(\mathrm{M}^\mathsf{T}\mathrm{M})^{1/2}$, we need to solve a single linear 3×3 system and, from the solution of that system, easily obtain $\{\mathbf{e}, \cos(\alpha), \sin(\alpha)\}$ as explained below.

4.5 Determining the rotation axis $\mathbf{e}$

Assume, initially that $\mathrm{V}\mathbf{e}^\mathsf{T} \neq 0$. Then, $\phi = -a(\mathrm{V}\mathbf{e}^\mathsf{T})$ is well defined and Equation (10) can be rewritten as $(\mathrm{N} - \lambda\mathrm{I})(\mathbf{e}^\mathsf{T}/\phi) = \mathrm{V}^\mathsf{T}$. Thus, considering that $\pm(\mathbf{e}, \alpha)$ represent the same rotation, we can infer that $\mathbf{e}$ can be found by solving

$$(\mathrm{N} - \lambda\mathrm{I})\, \mathbf{u}^\mathsf{T} = \mathrm{V}^\mathsf{T} \tag{14}$$

and taking the unit vector of $\mathbf{u}$. For small deformations, in special, this approach is more appropriate than directly finding an unit eigenvector of $(\mathrm{N} + a\mathrm{V}\mathrm{V}^\mathsf{T})$ since a may assume very large values.

It remains to consider the case where $\mathrm{V}\mathbf{e}^\mathsf{T} = 0$. Since we assume that $k_2 \neq 0$ and $\alpha \neq 0$, then the three conditions below are equivalent:

(i) $\mathrm{V}\mathbf{e}^\mathsf{T} = 0$;
(ii) (14) is a non-determined system;
(iii) λ is an eigenvalue of N. In this case $\mathbf{e}$ is an unitary vector associated to it.

Thus, if $\mathrm{V}\mathbf{e}^\mathsf{T} = 0$, condition (b) holds and having identified it we can find $\mathbf{e}$ as an unitary eigenvector of N associated to the eigenvalue λ.

4.6 Determining $\sin(\alpha)$ *and* $\cos(\alpha)$

From equations (11) and (12) we infer that $\lambda - 2\mathrm{E} = k_2(1 - \cos(\alpha))$. Additionally, equation (9) asserts that $\mathrm{E} - \mathbf{e}\mathrm{M}\mathbf{e}^\mathsf{T} = k_2\cos(\alpha)$. Adding these two equations, we obtain $k_2 = \lambda - \mathrm{E} - \mathbf{e}\mathrm{M}\mathbf{e}^\mathsf{T}$. This allows the use of equation (9) to obtain expressions for $\cos(\alpha)$ and $\sin(\alpha)$ which do not contain any squared terms:

$$\cos(\alpha) = \frac{\mathrm{E} - \mathbf{e}\mathrm{M}\mathbf{e}^\mathsf{T}}{\lambda - \mathrm{E} - \mathbf{e}\mathrm{M}\mathbf{e}^\mathsf{T}},\ \sin(\alpha) = \frac{\mathrm{V}\mathbf{e}^\mathsf{T}}{\lambda - \mathrm{E} - \mathbf{e}\mathrm{M}\mathbf{e}^\mathsf{T}}.$$

The optimality conditions of (7), and the fact that $\|\mathbf{u}\| = -(1/a)\mathrm{V}\mathbf{e}^\mathsf{T}$ allow us to rewrite these two expressions as

$$\begin{aligned} \cos(\alpha) &= \frac{(\mathrm{V}\mathbf{e}^\mathsf{T})^2 - (\lambda - 2\mathrm{E})^2}{(\mathrm{V}\mathbf{e}^\mathsf{T})^2 + (\lambda - 2\mathrm{E})^2} = \frac{1 - \|\mathbf{u}\|^2}{1 + \|\mathbf{u}\|^2}, \\ \sin(\alpha) &= \frac{2(\mathrm{V}\mathbf{e}^\mathsf{T})}{(\mathrm{V}\mathbf{e}^\mathsf{T})^2 + (\lambda - 2\mathrm{E})^2} = \frac{-2\|\mathbf{u}\|}{1 + \|\mathbf{u}\|^2}, \end{aligned} \tag{15}$$

which require very little computation considering that $\|\mathbf{u}\|^2$ and $\|\mathbf{u}\|$ have already been determined while obtaining $\mathbf{e}$. The equations (15) also indicate that the obtained values are in the range [-1,1].

Notice that up to the point where vector $\mathbf{u}$ is found our approach is computationally equivalent to the quaternion-based technique of Kanatani [16], except that, in that case, an eigenvector of a 4×4 matrix would be determined. However, all it remains now is to find the values $\{\mathbf{e}, \cos(\alpha), \sin(\alpha)\}$ from $\mathbf{u}$ and directly substitute them in equation (5), whereas to effect a quaternion-based transformation, a more lengthy computation is required for every vertex.

Finally, we must observe that the obtained rotation is exactly $M(M^{T}M)^{1/2}$ if $\det(M) >= 0$. Otherwise it is $M(M^{T}M)^{1/2}(I - \mathbf{e}_{min}\mathbf{e}_{min}^{T})$ where $\mathbf{e}_{min}$ is an eigenvector of $M^{T}M$ associated to the eigenvalue μ_{min}. These expressions indicate that the rotation is a continuous function of M while $\det(M) > 0$. A discontinuity can only occur if M is singular or $\det(M) < 0$ and μ_{min} is a multiple eigenvalue of $M^{T}M$. At discontinuity points, we might notice a "twisting" effect while interactively applying a large deformation to a model. It is important to stress, however, that any method for solving (7) will exhibit this behavior.

5 Algorithm

A deformation session uses as input a 3D model in the form of a polygonal mesh with n vertices and a set of control points $\{\mathbf{p}_i\}$, $i = 0 \ldots k-1$, not necessarily on the mesh. The user then establishes new positions for the control points which are stored in $\{\mathbf{q}_i\}$. Finally, Algorithm 1 is invoked to obtain new positions for each vertex $\mathbf{v}$ of the mesh.

Algorithm 1: Compute deformed vertex position

```
Input: Vertex position $\mathbf{v}$
Input: Original and deformed positions of control
       points $\{\mathbf{p}_i\}$ and $\{\mathbf{q}_i\}$
Output: Deformed vertex position $\mathbf{v}'$
1  $\mathbf{p}* \leftarrow$ WeightedCentroid($\mathbf{v}$, $\{\mathbf{p}_i\}$);
2  $\mathbf{q}* \leftarrow$ WeightedCentroid($\mathbf{v}$, $\{\mathbf{q}_i\}$);
3  $M \leftarrow$ CorrelationMatrix($\mathbf{p}*$, $\mathbf{q}*$);
4  $mroot \leftarrow$ MaximumRootOfP(M);
5  $\lambda \leftarrow mroot + \mathrm{Trace}(M)$ ;                    // eigenvalue
6  $\mathbf{e} \leftarrow$ RotationAxis(M, $\lambda$) ;
7  $[\sin(\alpha), \cos(\alpha)] \leftarrow$ SinCos(M, $\mathbf{e}$) ;
8  $\hat{\mathbf{v}} \leftarrow \mathbf{v} - \mathbf{p}*$;
9  $\mathbf{d} \leftarrow (\hat{\mathbf{v}}\,\mathbf{e}^{T})\,\mathbf{e} + \cos(\alpha)(\hat{\mathbf{v}} - (\hat{\mathbf{v}}\,\mathbf{e}^{T})\,\mathbf{e}) + \sin(\alpha)(\hat{\mathbf{v}} \times \mathbf{e})$;
10 $\mathbf{v}' \leftarrow \mathbf{q}* + \mathbf{d}$;
11 return $\mathbf{v}'$
```

The auxiliary routines used in Algorithm 1 merely compute the values of the various equations presented in this Section, as summarized in the Table 1.

Our implementation of this algorithm includes several tweaks to improve the performance. For instance, all values of $\hat{\mathbf{v}}$, $\mathbf{p}*$, w and $\hat{\mathbf{p}}_i$ are precomputed during a deformation session where the values of $\mathbf{q}_i$ are changed interactively. Figures 2 and 3 show sample results of the deformation algorithm as applied to two different models. Table 2 shows frame rates for deformations applied on several polygonal meshes. We notice, as expected, that the performance is directly proportional to the number of vertices of the model and to the number of constraints controlling the deformation.

Table 1 Routines and their equations

Routine	Equation
`WeightedCentroid`	(2)
`CorrelationMatrix`	(6)
`MaximumRootOfP`	(13)
`RotationAxis`	(14)
`SinCos`	(15)

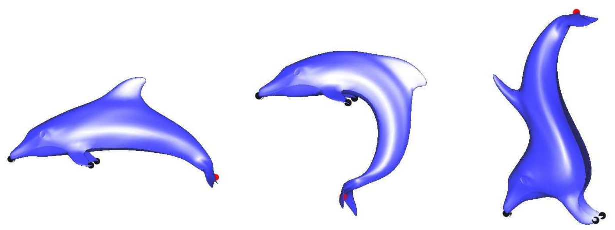
Fig. 2 Deformation of a dolphin using 4 control points.

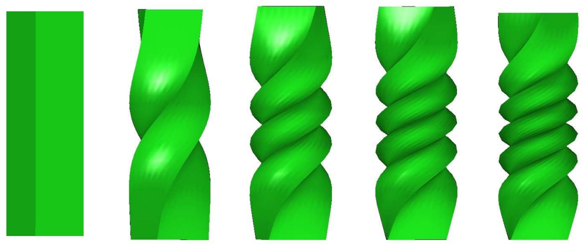
Fig. 3 Twisting deformation of a parallelepiped. The effect is obtained by applying several small-angle torsions in succession.

Table 2 Frame rates for the deformation of several models.

Model	Vertices	Constraints	FPS
Dolphin	2811	4	40
Homer	5103	6	22
Plane	1089	132	19.5
Dino	14050	5	6.5
Bar	6146	256	2
Cylinder	1058	322	9.5

The prototype has been implemented in C++ under Linux. We used the OpenGL API for rendering and all the experiments have been performed setting the OpenGL's canvas resolution to 600×600 pixels. Times have been taken on a PC equipped with a Pentium-IV processor running at 3.2 GHz with 1GB of main memory and a GeForce 7300LE NVidia graphics card.

6 Extensions

6.1 Similarity deformation

If a more general *similarity* – rather than rigid – transformation is required, then an uniform scaling factor $\mu_s \in \Re$ must be introduced in the optimization problem. In this case, equation 3 becomes

$$\sum_i w_i |\mu_s \hat{\mathbf{p}}_i \mathrm{R} - \hat{\mathbf{q}}_i|^2,$$

and to optimize it we must solve

$$\max\ \mu_s \sum_i w_i \hat{\mathbf{q}}_i \mathrm{R} \hat{\mathbf{p}}_i^{\mathsf{T}} - \mu_s^2 \sum_i w_i \|\hat{\mathbf{p}}_i\|^2.$$

This problem generates the same optimality conditions with respect to $\{\mathbf{e}, \cos(\alpha), \sin(\alpha)\}$ as (7). In addition, there is an optimality condition with respect to μ_s given by

$$\sum_i w_i \hat{\mathbf{q}}_i \mathrm{R} \hat{\mathbf{p}}_i^{\mathsf{T}} - \mu_s \sum_i w_i \|\hat{\mathbf{p}}_i\|^2 = 0.$$

Since the optimality conditions of (7) determine that an optimal solution satisfies $\sum_i w_i \hat{\mathbf{q}}_i \mathrm{R} \hat{\mathbf{p}}_i^{\mathsf{T}} = y$, we finally obtain

$$\mu_s = \frac{y}{\sum_i w_i \|\hat{\mathbf{p}}_i\|^2}.$$

Thus, step 10 of Algorithm 1 would read: $\mathbf{v}' \leftarrow \mathbf{q}{*} + \mu_s \mathbf{d}$.

6.2 Deformations using line segments

In many situations, the deformation induced solely by moving control points may produce undesirable distortion or counter-intuitive results, as shown in Figure 4(a). This may be alleviated by increasing the number of control points (Figure 4(b)), but only at a cost in terms of performance and interactivity. One possible solution is to use other geometric primitives such as line segments to control the deformation (Figure 4(c)).

The generalization of MLS-based deformations controlled by line segments may be stated as the minimization of

$$\textstyle\sum_i \int_0^1 w_i |\mathbf{p}_i(t)\mathrm{R} + \mathrm{T} - \mathbf{q}_i(t)|^2,$$

where $\mathbf{p}_i(t)$ is the i^{th} original line segment and $\mathbf{q}_i(t)$ the corresponding deformed line segment. In much the same way as what was done with control points, this can be reduced the maximization of

$$\textstyle\sum_i \int_0^1 w_i \hat{\mathbf{q}}_i(t) \mathrm{R} \hat{\mathbf{p}}_i^{\mathsf{T}}(t),$$

where $\hat{\mathbf{q}}_i(t) = \mathbf{q}_i(t) - \mathbf{q}{*}$ and $\hat{\mathbf{p}}_i(t) = \mathbf{p}_i(t) - \mathbf{p}{*}$. Thus, the optimization problem is also stated by Equation (7), where M is defined as

$$\mathrm{M} {=} \textstyle\sum_i \int_0^1 w_i \hat{\mathbf{q}}_i^{\mathsf{T}}(t) \hat{\mathbf{p}}_i(t) dt.$$

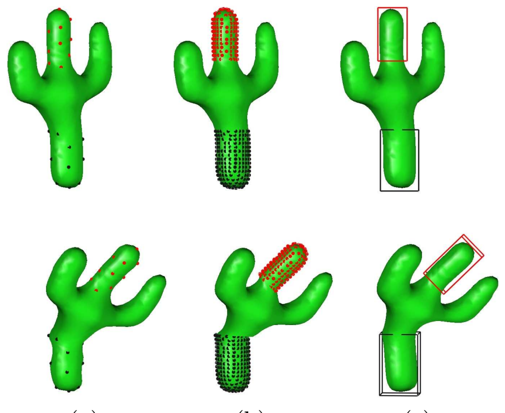

Fig. 4 Deforming a model using: (a) and (b) control points, and (c) line segments.

Line segments $\hat{\mathbf{p}}_i(t)$ and $\hat{\mathbf{q}}_i(t)$ may be represented by the products

$$\hat{\mathbf{p}}_i(t) = (1-t \quad t) \begin{pmatrix} \hat{\mathbf{a}}_i \\ \hat{\mathbf{b}}_i \end{pmatrix} \text{ and } \quad \hat{\mathbf{q}}_i(t) = (1-t \quad t) \begin{pmatrix} \hat{\mathbf{c}}_i \\ \hat{\mathbf{d}}_i \end{pmatrix},$$

where $\hat{\mathbf{a}}_i$, $\hat{\mathbf{b}}_i$ are the endpoints of $\hat{\mathbf{p}}_i(t)$ and $\hat{\mathbf{c}}_i$, $\hat{\mathbf{d}}_i$ are the endpoints of $\hat{\mathbf{q}}_i(t)$. Thus, M may be rewritten as

$$\mathrm{M} = \sum_i (\hat{\mathbf{c}}_i^{\mathsf{T}} \quad \hat{\mathbf{d}}_i^{\mathsf{T}})\, W_i \begin{pmatrix} \hat{\mathbf{a}}_i \\ \hat{\mathbf{b}}_i \end{pmatrix},$$

where

$$W_i = \begin{pmatrix} \int_0^1 w_i(t) t^2 dt & \int_0^1 w_i(t) t(1-t) dt \\ \int_0^1 w_i(t) t(1-t) dt & \int_0^1 w_i(t)(1-t)^2 dt \end{pmatrix}.$$

Depending on the choice of $w_i(t)$, the integrals in W_i may have very complicated explicit formulations, as mentioned by Schaefer, or even require an iterative numerical method to be computed. A simple formulation for W_i may be obtained if we consider that $w_i(t)$ has a constant value given by d_i^{-u} where d_i is the distance between the vertex v being evaluated and the segment $\overline{\mathbf{a}_i \mathbf{b}_i}$, that is,

$$W_i = d_i^{-u} \begin{pmatrix} 1/3 & 1/6 \\ 1/6 & 1/3 \end{pmatrix}.$$

Finally, the values of $\mathbf{p}{*}$ and $\mathbf{q}{*}$ are determined by

$$\mathbf{p}{*} = \frac{\sum_i d_i^{-u} (\mathbf{a}_i + \mathbf{b}_i)/2}{\sum_i d_i^{-u}}, \ \mathbf{q}{*} = \frac{\sum_i d_i^{-u} (\mathbf{c}_i + \mathbf{d}_i)/2}{\sum_i d_i^{-u}}.$$

In our experiments, we did not notice significant distortions caused by this simplification.

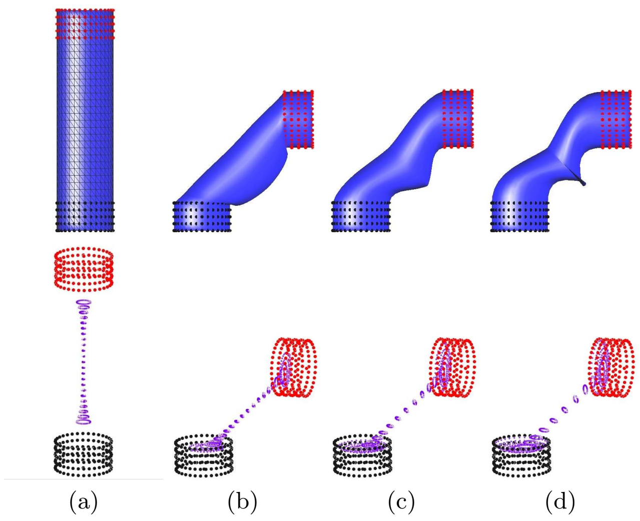

Fig. 5 Large deformation of cylinder: (a) Initial model and control points. Deformation induced by rotating the top group of vertices using different weight functions: (b) $w(d) = d^{-2}$, (c) $w(d) = d^{-3}$ and (d) $w(d) = d^{-4}$. The lower row of pictures show the positions of the weighted centroids $\mathbf{q}_i^*$

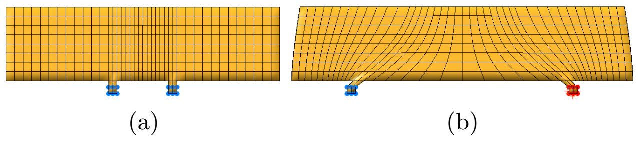

Fig. 6 Deformation caused by translation. (a) Initial model and control points. (b) Deformed model.

7 Discussion

As noted by Botsch et al. [9], deformation techniques may lead to non-intuitive results when control handles are subject to large displacements. In our case, large translations and/or rotations may indeed produce non-smooth results as illustrated in Figures 5, and 8.

The key observation that provides an explanation to this behavior is that the weighted centroids $\mathbf{p}*_i$ and $\mathbf{q}*_i$ are, in fact, computed as convex combinations of control points $\{\mathbf{p}_i\}$ and $\{\mathbf{q}_i\}$. One might be tempted to obtain a smoother distribution of the centroids by modifying the weighing function but this does not lead, in general, to satisfactory results, as illustrated in Figure 5.

Another interesting effect may be observed when the handles are subject only to translations as shown in Figure 6. Notice, in particular, that lines which are parallel to the handle movement are not distorted.

It must be stressed, however, that in many cases, an unexpected result is caused merely by insufficient data input. Consider, for instance, the deformation depicted in Figure 1 where the intended result was clearly to make the character bend at the waist. For this to occur naturally, the placement of a control point near the navel is crucial.

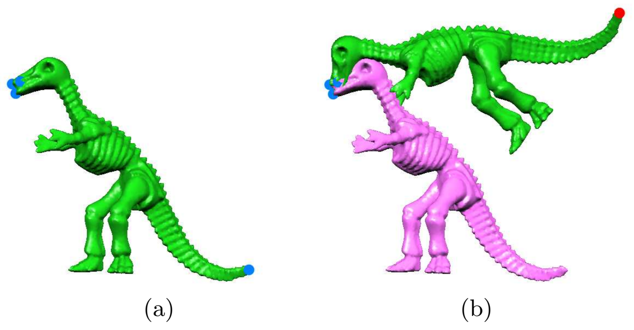

Fig. 7 Deformations with large displacements. (a) Initial model and control points. (b) Deformed model.

On the other hand, it is worth mentioning that MLS-rigid deformations behave nicely in the sense that it tends to preserve details and global characteristics of the model better than, say, RBF-based or differential coordinate-based techniques. Figure 7 supports this claim when we compare it with a similar figure shown in [11] (Figure 7, to be precise).

8 Conclusions and future work

We have presented a practical approach for computing as-rigid-as-possible deformations of 3D models using a Moving Least Squares (MLS) minimization scheme. The rotation solution is defined by a rotation axis and angular parameters $\{\mathbf{e}, \cos(\alpha), \sin(\alpha)\}$. These are found by computing the largest real root of a depressed quartic equation and solving a 3×3 system. Our preliminary experiments show that this approach is slightly more efficient than the classical formulation which computes the rotation matrix R as $\mathrm{M}^{\mathsf{T}}(\mathrm{MM}^{\mathsf{T}})^{-1/2}$ [13]. It also uses fewer operations than the quaternion-based representation of R as suggested in [16].

A more efficient approach to the problem may be devised in the future in the form of an incremental method which approximates the best rotation without the need of computing eigenvalues for every vertex at every frame of the interaction. In fact, we consider our approach a good starting point for deriving such a method.

The expressive power of as-rigid-as-possible deformations is hindered by some non-smooth results for some control point configurations, specially if the handles are subjected to large translations or rotations. Some ideas for reducing these problems are being investigated, such as interpolating the centroids $\mathbf{q}*$ using radial basis functions (RBFs). Some preliminar results are shown in Figure 8.

References

1. Alexa, M.: Interactive shape editing. ACM SIGGRAPH Courses (2006)

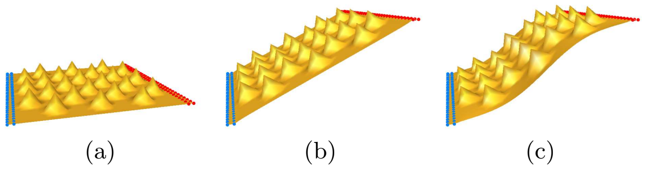

Fig. 8 Mixing MLS with RBFs. Model (a) is deformed by displacing a row of control points giving a non-smooth result (b), whereas in (c) centroids $\mathbf{q}*$ are computed as a function of their original positions $\mathbf{p}*$ by means of a RBF-based interpolation scheme using the control points as restrictions.

2. Alexa, M., Cohen-Or, D., Levin, D.: As-rigid-as-possible shape interpolation. Proceedings of SIGGRAPH 2000 pp. 157–164 (2000). USA, July 23-28
3. Arun, K.S., Huang, T.S., Blostein, S.D.: Least-squares fitting of two 3-d point sets. IEEE Trans. Pattern Anal. Mach. Intell. **9**(5), 698–700 (1987)
4. Barr, A.H.: Global and local deformations of solid primitives. In: Proceedings of the 11th annual conference on Computer graphics and interactive techniques, pp. 21–30. ACM Press, New York, NY, USA (1984)
5. Besl, P.J., McKay, N.D.: A method for registration of 3-D shapes. IEEE Transactions on Pattern Analysis and machine Intelligence **14**(2), 239–258 (1992)
6. Blinn, J.F.: A generalization of algebraic surface drawing. ACM Trans. Graph. **1**(3), 235–256 (1982)
7. Botsch, M., Kobbelt, L.: An intuitive framework for real-time freeform modeling. ACM Trans. Graph. **23**(3), 630–634 (2004)
8. Botsch, M., Kobbelt, L.: Real-time shape editing using radial basis functions. Computer Graphics Forum **24**(3), 611–621 (2005)
9. Botsch, M., Pauly, M., Gross, M., Kobbelt, L.: Primo: Coupled prisms for intuitive surface modeling. In: Eurographics Symposium on Geometry Processing, pp. 11–20. Eurographics (2006)
10. Eggert, D.W., Lorusso, A., Fisher, R.B.: Estimating 3-d rigid body transformations: a comparison of four major algorithms. Mach. Vision Appl. **9**(5-6), 272–290 (1997)
11. Fu, H., Au, O.K.C., Tai, C.L.: Effective derivation of similarity transformations for implicit laplacian mesh editing. Computer Graphics Forum (to appear) (2006)
12. Horn, B.K.P.: Closed form solutions of absolute orientation using unit quaternions. Journal of the Optical Society of America **4**(4), 629–642 (1987)
13. Horn, B.K.P., Hilden, H.M., Negahdaripour, S.: Closed form solutions of absolute orientation using orthonormal matrices. Journal of the Optical Society of America **5**(7), 1127–1135 (1988)
14. Hsu, W.M., Hughes, J.F., Kaufman, H.: Direct manipulation of free-form deformations. In: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 177–184. New York, NY, USA (1992)
15. Igarashi, T., Moscovich, T., Hughes, J.F.: As-rigid-as-possible shape manipulation. ACM Trans. Graph. **24**(3), 1134–1141 (2005)
16. Kanatani, K.: Analysis of 3-d rotation fitting. IEEE Trans. Pattern Anal. Mach. Intell. **16**(5), 543–549 (1994)
17. Kobbelt, L., Campagna, S., Vorsatz, J., Seidel, H.P.: Interactive multi-resolution modeling on arbitrary meshes. In: Proceedings of the 25th annual conference on Computer graphics and interactive techniques, pp. 105–114. ACM Press, New York, NY, USA (1998)
18. Lipman, Y., Cohen-Or, D., Gal, R., Levin, D.: Volume and shape preservation via moving frame manipulation. ACM Trans. Graph. **26**(1), 5 (2007). DOI http://doi.acm.org/10.1145/1189762.1189767
19. Lipman, Y., Sorkine, O., Cohen-Or, D., Levin, D., Rössl, C., Seidel, H.P.: Differential coordinates for interactive mesh editing. In: F. Giannini, A. Pasko (eds.) Shape Modeling International 2004 (SMI 2004), pp. 181–190. IEEE, Genova, Italy (2004)
20. Lipman, Y., Sorkine, O., Levin, D., Cohen-Or, D.: Linear rotation-invariant coordinates for meshes. ACM Trans. Graph. **24**(3), 479–487 (2005)
21. Marinov, M., Botsch, M., Kobbelt, L.: Gpu-based multiresolution deformation using approximate normal field reconstruction. ACM Journal of Graphics Tools, 2007, to appear (2007)
22. Mario Botsch, L.K.: Multiresolution surface representation based on displacement volumes. Computer Graphics Forum **22**(3), 483–491 (2003)
23. Moreton, H.P., Séquin, C.H.: Functional optimization for fair surface design. In: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 167–176. ACM Press, NY, USA (1992)
24. Pottmann, H., Huang, Q.X., Yang, Y.L., Hu, S.M.: Geometry and convergence analysis of algorithms for registration of 3d shapes. Int. J. Comput. Vision **67**(3), 277–296 (2006)
25. Schaefer, S., McPhail, T., Warren, J.: Image deformation using moving least squares. ACM Trans. Graph. **25**(3), 533–540 (2006)
26. Sederberg, T.W., Parry, S.R.: Free-form deformation of solid geometric models. In: Proceedings of the 13th annual conference on Computer graphics and interactive techniques, pp. 151–160. ACM Press, NY, USA (1986)
27. Sorkine, O., Cohen-Or, D., Lipman, Y., Alexa, M., Rössl, C., Seidel, H.P.: Laplacian surface editing. In: Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing, pp. 175–184. ACM Press, New York, NY, USA (2004)
28. Walker, M.W., Shao, L., Volz, R.A.: Estimating 3-D location parameters using dual number quaternions. CVGIP: Image Understanding **54**(3), 358–367 (1991)
29. Welch, W., Witkin, A.: Variational surface modeling. In: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 157–166. ACM Press, New York, NY, USA (1992)
30. Yu, Y., Zhou, K., Xu, D., Shi, X., Bao, H., Guo, B., Shum, H.Y.: Mesh editing with poisson-based gradient field manipulation. ACM Trans. Graph. **23**(3), 644–651 (2004)
31. Zayer, R., Rössl, C., Karni, Z., Seidel, H.P.: Harmonic guidance for surface deformation. Computer Graphics Forum **24**(3), 601–609 (2005)
32. Zhou, K., Huang, J., Snyder, J., Liu, X., Bao, H., Guo, B., Shum, H.Y.: Large mesh deformation using the volumetric graph laplacian. ACM Trans. Graph. **24**(3), 496–503 (2005)