Nearly Rigid Deformation by Linear Optimization

SoHyeon Jeong Chang-Hun Kim

Korea University

Abstract

Fast near-rigid 3D deformation of a shape is achieved by approximating a rigid transformation of each point by a moving least-squares (MLS) optimization with a linear closed-form solution. We can deform tens of thousands points in real time while preserving the rigidity of the original shape as far as possible. This technique inherits the properties of earlier MLS-based deformation techniques, such as independence of geometric representation and smoothness. The user can control the deformations by using control points or line segments and by the adjustment of weight functions.

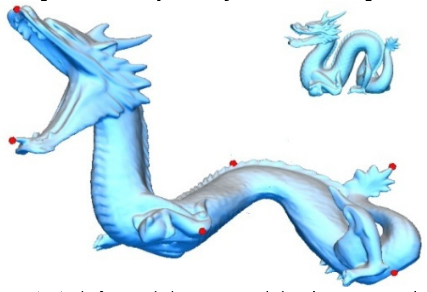

Figure 1: A deformed dragon model using our method.

1. Motivation

Deforming 3D models which consist of large numbers of points in real-time is challenging. The fulfillment of constraints which improve the quality of the deformed shape can be traded off against computation time, but the number of points must be limited to obtain plausible interactivity.

Deformation using Moving Least Squares involves a near-rigid transformation which reduces local distortion [Schaefer et al. 2006]. Since a near-rigid transformation approximates a 2D rotation by means of a linear closed-form solution, it can be computed very fast. However, in 3D it is no longer a linear problem, and the resulting eigen-vector computation takes longer.

2. Approximating Point Rotations

The main problem in performing a near-rigid transformation is to approximate a rigid transformation for each point $\mathbf{v}$ by transforming a set of control points, which we will write as $\{\boldsymbol{p}_i\}$ and $\{\boldsymbol{q}_i\}$ in their undeformed and deformed states respectively. We can also represent these points as sets of control vectors $\{\hat{\boldsymbol{p}}_i\}$ and $\{\hat{\boldsymbol{q}}_i\}$ which are relative to their weighted centroids $\{\boldsymbol{p}_*\}$ and $\{\boldsymbol{q}_*\}$. This eliminates any translation and exposes the rotational relation between the sets of points.

Our idea to extract a rotation from each pair of vectors and then to combine them into on approximate an overall rotation. To express the rotation between two vectors in 3D, we use a rotation vector (a.k.a. axis-angle) $\mathbf{r}$ which is the product of a rotation axis $\mathbf{u}$ and an angle of rotation $\boldsymbol{\theta}$ around $\mathbf{u}$. Unlike 3x3 orthogonal matrices or quaternions, the weighted average of these rotation vectors can be converted to a 3x3 orthogonal matrix. The rotation vector $\mathbf{r}$ that is then applied to each point $\mathbf{v}$ minimizes a weighted error formulated in terms of the set of rotation vectors $\{\boldsymbol{r}_i\}$ of the control points, using the function $w_i = 1/|\mathbf{p}_i - \mathbf{v}|^{2\alpha}$.

$$\sum_i w_i |\hat{\mathbf{p}}_i|^2 |\mathbf{r}_i - \mathbf{r}|^2 \quad (1)$$

A point $\mathbf{v}$ is transformed using an orthonormal matrix $\boldsymbol{R}$ which is converted from the approximated rotation vector $\boldsymbol{r}$.

$$\mathbf{v}^{\text{new}} = (\mathbf{v} - \mathbf{p}_*)\mathbf{R} + \mathbf{q}_* \quad (2)$$

3. Weight Functions

We can control the blending of the control-point transformations by adjusting the parameter α in the weight function. Considering only spatial distance in formulating the weight functions produces unsatisfactory results, as shown in Figure 2(c): control points can affect points which in the shape are not topologically close. We can overcome this problem by building the weight function in terms of the shortest distance over the surface of the object (Figure 2(d)).

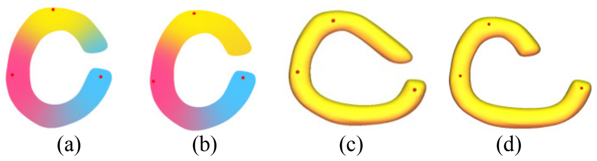

Figure 2: A comparison between spatial and topological weights. (a), (b) influence of control points using spatial weights and topological weights, (c) limitation of spatial weights, (d) a model deformed using topological weights.

4. Control Handles

We provide control points and line segments as handles to deform 3D shapes. Compared with control points, line segments enable more elaborate deformations. Figure 3 shows the twisting and bending of a box using control lines. These line segments can also be used as the skeleton of a model.

These two types of handle can be used concurrently. Algebraically, line segments are as the same as control points. We simply extract the variables w_i, $\hat{\boldsymbol{p}}_i$, $\hat{\boldsymbol{q}}_i$ and $\boldsymbol{r}_i$ that control the approximation in a different way. For instance, for a control line segment $\boldsymbol{p}_i, \boldsymbol{r}_i$ is the rotation vector between the line segment before and after deformation.

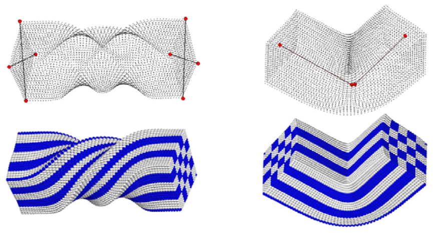

Figure 3: Twisting and bending a box modeled as a point cloud.

5. Results

Our prototype has been implemented using C++, MFC and the STL library. Table 1 shows its performance on an Intel cord 2 Quad CPU 2.40 GHz with 2 GB RAM. In essence, the computation times are linearly proportional to the number of points and the number of handles.

Model	Points(p)	Handles(h)	Times(ms)	Time/p/h
Dragon	35820	5	56.3	3.17e-4
Bunny	35947	4	46.8	3.25e-4
Armadillo	20002	13	64.1	2.46e-4
Box	6006	4	6.2	2.58e-4

Table 1: Computation times of several models.

References

SCHAEFER, S., MCPHAIL, T., and WARREN, J. 2006. Image Deformation using Moving Least Squares. *ACM Transaction on Graphics*, 25, 3, 533-540.