A simple approach to mesh deformation

Shingo Ito 1200045 Supervised by Pierre-Alain Fayolle

Abstract

In this dissertation, we present a simple approach for mesh deformation. Points are sampled from the surface represented by the triangle mesh. The point cloud data is deformed instead of the mesh. A surface reconstruction algorithm is finally used to reconstruct the surface from the deformed point cloud, preventing defects usually appearing in mesh deformation algorithms.

Keywords:

Mesh processing; Mesh deformation; Surface reconstruction

1 Introduction

Mesh processing and mesh deformation is an important research topic in the geometric modeling community, with important practical applications. It allows users with relatively little technical knowledge to model complex freeform shapes through simple and intuitive interfaces. Shape deformation is a challenging topic because it can require complex mathematical formulations to be implemented as efficient (in terms of speed) computer programs. The topic of shape deformation and shape sculpture is a popular topic in geometric modeling. See for example [**?**] for a survey of recent mesh-based deformation approaches. One of the problems with the typical mesh-based deformation approaches is that they may lead to defects in the input triangle mesh, such as self-intersection for example. In order to overcome this difficulty, we propose an algorithm where a sampling of the triangle mesh is deformed and the deformed surface is obtained from applying a surface reconstruction algorithm to the deformed point cloud. Our deformation algorithm works by applying a vector field, corresponding to the deformation, to the vertices of a triangle mesh. In order to prevent defects in the mesh such as self-intersection, we reconstruct a surface from the deformed vertices by using a standard surface reconstruction algorithm.

2 Related works

Surface deformation is an important research topic in shape and geometric modeling. The book by Botsch and colleagues on polygon mesh processing [**?**] has a full chapter on techniques for polygon mesh deformation and provides many related references on the topic. When surfaces are represented implicitly, the main technique consists in computing an approximation of the distance to the zero level-set and deforming this distance field. The dissertation of Bridson [**?**] describes such a technique. More recently the dissertation of Jacobson [**?**] presents various novel techniques for mesh deformation techniques based on variational methods and solving partial differential equations. It also contains several references related to the domain of mesh deformation.

3 Algorithm

Algorithms for surface deformation traditionally work by moving the triangle mesh vertices according to some specified vector field corresponding to the deformation. Depending on how the deformation is computed, this approach can result in defects in the triangle mesh such as for example self-intersecting triangles. In this work we propose to apply the deformation to a sampling of the surface and reconstruct the deformed surface by using a surface reconstruction algorithm. Our approach is summarized in the algorithm 1 below.

Algorithm 1 Overview of the main approach

Sample from the surface
Apply the deformation field to the samples
Apply a surface reconstruction algorithm to the samples

3.1 Surface sampling

The input surface is represented by a triangle mesh read from a file. If the triangle mesh is dense enough, we select as points the mesh vertices. Otherwise we uniformly sample from each triangle according to the following method 2.

Algorithm 2 Uniform sampling from a triangle

function SAMPLE(p_1, p_2, p_3, ξ_1, ξ_2)
▷ p_1, p_2 and p_3 are the triangle vertices, ξ_1 and ξ_2 are samples from a uniform distribution
Compute the barycentric coordinates: $u = 1 - \sqrt{\xi_1}$ and $v = \xi_2 \sqrt{\xi_1}$
Compute the sample: $P = u * p_1 + v * p_2 + (1 - u - v) * p_3$
return P
end function

3.2 Points deformation

There are different possible approaches for deforming points. We have implemented the following simple approach. Given a selected vertex from the input surface, we apply a truncated Gaussian centered on this selected vertex:

$$f(\mathbf{x}) = \begin{cases} \frac{1}{\sqrt{2\pi\sigma}} \exp\left(-\frac{(\mathbf{x}-\mathbf{x}_S)^2}{2\sigma^2}\right) & \text{if } \|\mathbf{x} - \mathbf{x}_S\| < \epsilon \\ 0 & \text{otherwise} \end{cases}$$

In this expression, x_S is the selected vertex, σ controls the width of the Gaussian and ϵ controls the truncation.

After deformation, we need to recompute the normal vector to each point of the point cloud since it is needed for the surface reconstruction algorithms described in the next section. We used the implementation provided by CGAL [**?**] for normal computation and orientation. It implements the algorithm described in [**?**].

3.3 Surface reconstruction

Given the deformed point-cloud, we then apply a surface reconstruction algorithm to reconstruct a surface from it. We have implemented and experimented with three surface reconstruction algorithms in our implementation. All approaches are implicit surface based surface reconstruction algorithms. It means that the surface is obtained as the zero level-set of a function $f : \mathbb{R}^3 \to \mathbb{R}$.

Hermite Radial Basis Functions First we have implemented Hermite Radial Basis Functions surface reconstruction (HRBF) surface reconstruction [**?**]. Given a set of points $P = \mathbf{x}_i$ with normal vector at each point $N = \mathbf{n}_i$, the output of the method is a function: $f : \mathbb{R}^3 \to \mathbb{R}$ defined as follows:

$$f(\mathbf{x}) = \sum_{i=1}^{n} (\alpha_i \phi(\|\mathbf{x} - \mathbf{x}_i\|) - \beta_i \nabla\phi(\|\mathbf{x} - \mathbf{x}_i\|))$$

where ϕ is a radially symmetric functions (compactly supported or not), $\alpha_i \in \mathbb{R}$ and $\beta_i \in \mathbb{R}^3$ are unknown coefficients to be determined. The conditions used to determine the coefficients are:

$$f(\mathbf{x}_i) = 0, \mathbf{x}_i \in \mathbb{R}^3$$

for interpolating points on the surface. And:

$$\nabla f(\mathbf{x}_i) = \mathbf{n}_i, \mathbf{x}_i \in \mathbb{R}^3, \mathbf{n}_i \in \mathbb{R}^3$$

for interpolating the normals on the surface. Taking into account the expression of f, we have:

$$f(\mathbf{x}_i) = \sum_{j=1}^{n} (\alpha_j \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) - \beta_j \nabla\phi(\|\mathbf{x}_i - \mathbf{x}_j\|)) = 0$$

and

$$\nabla f(\mathbf{x}_i) = \sum_{j=1}^{n} (\alpha_j \nabla\phi(\|\mathbf{x}_i - \mathbf{x}_j\|) - H\phi(\|\mathbf{x}_i - \mathbf{x}_j\|)\beta_j) = \mathbf{n}_i$$

where H is the Hessian matrix of ϕ.
The interpolation conditions can be written as:

$$\sum_{j=1}^{n} \begin{bmatrix} \phi(\|\mathbf{x}_i - \mathbf{x}_j\|) & -\nabla\phi(\|\mathbf{x}_i - \mathbf{x}_j\|)^T \\ \nabla\phi(\|\mathbf{x}_i - \mathbf{x}_j\|) & -H\phi(\|\mathbf{x}_i - \mathbf{x}_j\|) \end{bmatrix} \begin{bmatrix} \alpha_j \\ \beta_j \end{bmatrix} + \sum_{l=1}^{m} \lambda_l \begin{bmatrix} p_l(\mathbf{x}_i) \\ \nabla p_l(\mathbf{x}_i) \end{bmatrix} = \begin{bmatrix} 0 \\ \mathbf{n}_i \end{bmatrix} \quad (1)$$

With the additional condition:

$$\sum_{j=1}^{n} [p_k(x_j) \nabla p_k(x_j)^T] \begin{bmatrix} \alpha_j \\ \beta_j \end{bmatrix} = 0 \quad (2)$$

Popular choices of radial basis function are: $\phi(\|x\|) = \|x\|$ or $\phi(\|x\|) = \|x\|^3$ or Wendland's compactly supported functions. In our experiments we have implemented: $\phi(\|x\|) = \|x\|^3$. Gradient and Hessian matrix of ϕ are given by:

$$\nabla\phi(\|x\|) = 3x\|x\|$$

$$H\phi(\|x\|) = \begin{cases} 3/\|x\|(\|x\|^2 I_{3\times3}) + xx^T, & \|x\| \neq 0 \\ 0_{3\times3}, & \|x\| = 0 \end{cases}$$

where $I_{3\times3}$ is the identity matrix and $0_{3\times3}$ is the zero matrix.

The system of linear equations in (1) with the conditions (2) can be rewritten in matrix form as: $\mathbf{Aw} = \mathbf{b}$ where the vector $\mathbf{w}$ contains the unknown coefficients. They are obtained from inverting the matrix $\mathbf{A}$.

Closed form solution for compactly supported functions The use of non-compactly supported splines results in a dense matrix, which makes the approach above not practical for large point sets. In order to handle such large point sets, it is preferrable to use compactly supported basis functions such as the ones proposed by Wendland [**?**]. In a recent work, Liu and Wang [**?**] showed that when Wendland's compactly supported radial basis functions are used in the approach above, it is possible to obtain a good approximation to the solution in closed form. Most notably their approach does not involve solving any linear system. In their approach, the implicit surface is obtained from the function:

$$f(x) = -\sum_{j=1}^{n} < \frac{\rho_j^2}{20 + \eta\rho_j^2}\mathbf{n}_j, \nabla\phi(\|\mathbf{x} - \mathbf{x}_j\|) > \qquad (3)$$

, where ρ_j is the radius of support of the basis function associated to the center j, and ϕ is the compactly supported basis function. In this paper, we use a Wendalandfs CSRBF [**?**] as the kernel function.

$$\phi_\rho(r) = \phi(r/\rho)$$

$$\phi(t) = \begin{cases} (1-t)^4(4t+1), & t \in [0,1] \\ 0, & otherwise \end{cases}$$

r is the Euclidean distance between a query point and the center of a RBF.

Poisson surface reconstruction The last surface reconstruction algorithm that we experimented with is Poisson surface reconstruction [**?**]. We used the implementation provided by CGAL [**?**]. This algorithm works by solving the Poisson equation: $\Delta f = \nabla \cdot V$ by the finite element method on an irregular grid. f is an indicator function for the solid approximated by the point-cloud, and V is an extrapolation of the normal vector field. To address this issue, need to solve for the function f such that the projection of $\nabla \cdot V$ onto the space $S_{\mathbb{O},F}$ is closest to the projection of Δf. Since, in general, the functions F_o do not form an orthonormal basis, solving this problem directly is expensive. However, CGAL can simplify the problem by solving for the function f minimizing:

$$\sum_{o\in\mathbb{O}} \| \langle \Delta f - \nabla \cdot V, F_o \rangle \|^2 = \sum_{o\in\mathbb{O}} \| \langle \Delta f, F_o \rangle - \langle \nabla \cdot V, F_o \rangle \|^2$$

Thus given the $\|\mathbb{O}\|$ -dimensional vector v whose o-th coordinate is $v_o = \langle \nabla \cdot V, F_o \rangle$, the goal is to solve for the function f such that the vector obtained by projecting the Laplacian of f onto each of the F_o is as close to v as possible.

To express this in matrix form, let $f = \sum_o x_o F_o$, so that we are solving for the vector $x \in \mathbb{R}^{|\mathbb{O}|}$. Then, let us define the $|\mathbb{O}| \times |\mathbb{O}|$ matrix L such that Lx returns the dot product of the Laplacian with each of the F_o. Specifically, for all $o, o' \in \mathbb{O}$, the (o, o')-th entry of L is set to:

$$L_{o,o'} \equiv \langle \frac{\partial^2 F_o}{\partial x^2}, F_o' \rangle + \langle \frac{\partial^2 F_o}{\partial y^2}, F_o' \rangle + \langle \frac{\partial^2 F_o}{\partial z^2}, F_o' \rangle$$

Thus, solving for f amounts to finding

$$\min_{x\in\mathbb{R}^{|\mathbb{O}|}} \| Lx - v \|^2$$

3.4 Implicit surface meshing

In order to obtain a deformed surface, we need to compute a triangle mesh implementation of the implicit surface computed as described in the previous section.

Marching Cubes The most common technique for meshing the zero level-set of implicit surface relies on the Marching Cubes algorithm [**?**]. The Marching Cubes algorithm is an algorithm for rendering isosurfaces from volumetric data. The basic idea is to consider a bounding box for the object to be meshed and subdivide it regularly into smaller cells Then the function f is sampled at the eight corners of each cell. If one or more values is less than the user-specified isovalue, and one or more have values is greater than this isovalue, the cell must intersect the isosurface. By determining the edges in the cell that are intersected by the isosurface, and connecting them we can form a linear approximation of the surface in each cell. By connecting the patches from all cells, we get a linear approximation of the isosurface.

Delaunay based implicit surface meshing Since we know that all the points from the deformed point-cloud belong to the surface of the deformed object (or at least are close to it), Marching Cubes based algorithm do not look like the most effective approach. Instead one could compute a Delaunay tetrahedralization of the deformed point-cloud and peel off outside tetrahedra using the fitted function (either from the HRBF or Poisson surface reconstruction approach). One practical implementation is the implicit surface mesh generator implemented by CGAL which is an implementation of the algorithm of Boissonat and Oudot [**?**].

Figure 1: Deformed sphere using the HRBF approach

Figure 2: Deformed sphere using the closed form solution to the HRBF approach

4 Experiments and results

A prototype for the methods described in section 3 was implemented on a laptop with the following specifications: CPU is 1.4GHz Intel Core i5, Memory is 4 GB 1600 MHz DDR3, graphics board is Intel HD Graphics 5000 1536 MB, OS is OS X Yosemite version 10.10.5. It was implemented in C++ using the following libraries: CGAL4.7, OpenGL and Eigen.

4.1 Deformation of a sphere

In the first experiment, we start from a sphere represented by a triangle mesh, made of 174 vertices and 344 triangles. The algorithm 3 is applied to deform the sphere. Figures 1, 2 and 3 show the results obtained with respectively: the HRBF reconstruction approach with the basis function $\|x\|^3$, the closed form approximation (3) and the Poisson surface reconstrion approach.

In our experiments, HRBF with the basis function $\|x\|^3$ is giving the best result but it is also the slowest approach. Additionally, it takes too much memory when non-compactly supported splines are used, because it needs to store a dense matrix in memory. It makes the approach impractical for large triangle meshes. The closed form approximation to HRBF is faster but in our experiments we found it difficult

Figure 3: Deformed sphere using the Poisson surface reconstruction approach

Figure 4: Example to model a simple character face

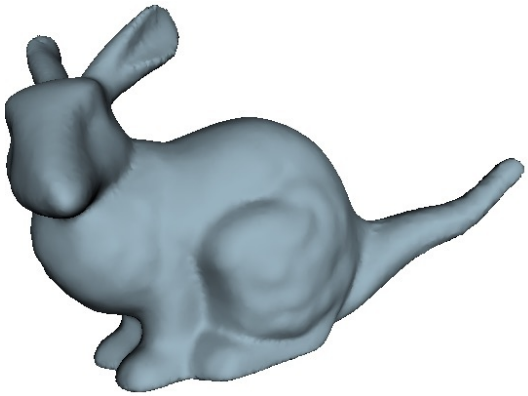

Figure 5: Deformed tail and nose of bunny

to control its behavior. Poisson surface reconstruction is fast enough and results in accurate enough reconstructed surfaces. However, it tends to produce smoother surface. For example observe the difference between fig. 1 and fig. 3.

4.2 Example of sculptures

Starting from a simple sphere, we used our program to model a simple character face. See fig. 4 for the result. By changing the sign of the field used for the deformation, we carved the eyes and the mouth.

4.3 Deformation of more complicated surfaces

The result in fig. 5 uses the Delaunay based approach while fig. Original bunny object have 14072 vertices and 28042 triangles. The number of subdivisions used for generating the mesh of the deformed bunny with the Marching Cubes algorithm is $64 \times 64 \times 64 = 262144$.

4.4 Recommendations

Based on our experiments, we recommend to use the HRBF approach with a non-compact basis function for small mesh (mesh with few vertices), and the Poisson based approach for larger mesh (mesh with a large number of vertices). It seems preferrable to mesh the implicit surface using the Delaunay based approach since it will keep mesh vertices.

5 Conclusion

In this paper, we proposed a simple algorithm for mesh deformation that works by applying a deformation field to the vertices (or a sampling) of a triangle mesh and reconstructs the deformed surface from the deformed point-cloud. Reconstruction from the point-cloud is made by fitting an implicit surface to the point-cloud and meshing it. Different algorithms for implicit surface fitting were experimented. We used our prototype for creating simple deformations of triangle meshes and simple sculptures.

Our algorithm and its implementation can be improved in several ways that are left for future works. Examples of possible improvements include: implementing parts of the algorithm on the graphics card (GPU), using a modified meshing algorithm that tracks the surface, and exploiting the locality of the deformation.