1 Mesh Editing with Laplacian Deform

This work was accepted and completed for the software Blender [1] that is an open source 3D application for modeling, animation, rendering, compositing, video editing and game creation. In the Google Summer of Code 2013 program which was administered for Google Inc.

1.1 Synopsis

The mesh editing is generally done with affine transformations, Blender3D offers some tools to transform the vertices as "proportional editing object mode" with which the transformation of some vertices is interpolated to the other vertices connected with the use of simple distance functions.

This project proposes to implement a method for mesh editing based on sketching lines defines by user and preserving the geometric details of the surface.

This method captures the geometric details using a differential coordinates representations. The differential coordinates captures the local geometric information (curvature and direction) of the vertex based on its neighbors. This method allows you to retrieve the best possible original model after changing the positions of some vertices using the differential coordinates of the original model.

1.2 Benefits to Blender

This project proposes a new tool for blender user that requires preserve the geometric details of the surface during a modeling, transformation, definition of the shape keys of the mesh vertices.

The method will allow novice users to edit any polygon mesh preserving the surface details.

This method allows the user to define new shape keys in a most fast and intuitive way.

1.3 Deliverables

- A new mesh editing tool for Blender.
- Some pages of documentation to be included in the manual
- A technical document for developers to improve the method in the future.
- A tutorial explaining the use of the tool.

1.4 Project Details

The project would divide into several parts:

1. Calculate the differential coordinates.
2. Store the fixed vertices (Hard constraints).
3. Store positions of the edited vertices.
4. Store the more representative vertex for retrieve rotation of every differential coordinate.
5. Solve the initial solution – in least-squares sense.
6. Rotate the differential coordinates with base on initial solution and more representative vertex.
7. Reconstruct the surface – in least-squares sense.
8. Generation of the documentation and tutorials.

1.5 Project Schedule

- 2 Weeks: Calculate the differential coordinates.
- 2 Weeks: Store the fixed vertexes (Hard constraints).
- 2 Weeks: Store positions of the edited vertexes.
- 2 Weeks: Compute initial solution.
- 2 Weeks: Rotate differential coordinates.
- 2 Weeks: Reconstruct the surface – in least-squares sense.
- 1 Weeks: Testing the tool and Define and implement graphical user integration.
- 2 Weeks: Generation of the documentation and tutorials.

1.6 Laplacian Deform

The Laplacian deform allows to pose a mesh while preserving geometric details of the surface. This method allows to defines a set of anchor vertices, and then moves some of them around. The system keeps the rest of the anchor vertices in fixed positions, and calculates the best possible locations of all the remaining vertices to preserve the original geometric details.

This work adapt the method proposed by Sorkine [3] for mesh deformations, delete the use of static vertices and allow application of this system in hybrid meshes composes by triangles and quads with the TQLBO proposed.

This method captures the geometric details using a differential coordinates representations. The differential coordinates captures the local geometric information (curvature and direction) of the vertex based on its neighbors how show in figure **1-1**.

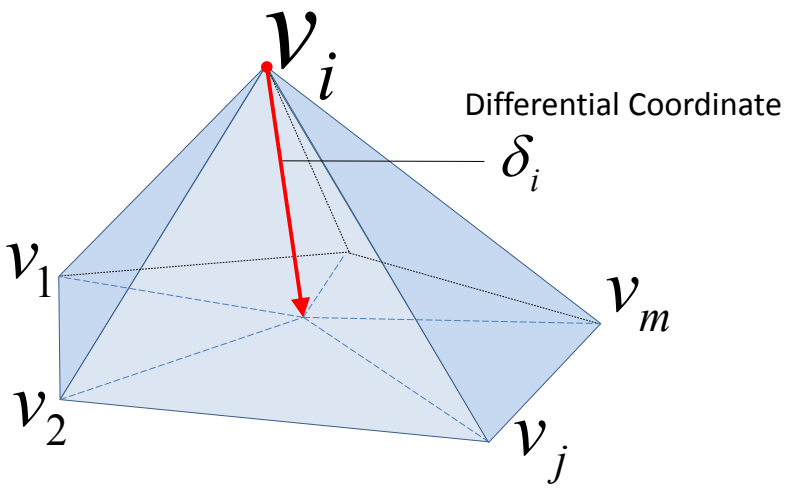

Figure **1-1**: Difference between v_i and the center of mass of its neighbors $v_1, ..., v$.

$$\delta_i = \sum_{j=1}^{m} w_{ij} \left(v_i - v_j\right) \tag{1-1}$$

Where δ_i is the differential coordinate for vertex v_i. The v_j are the immediate neighbors of v_i, and w_{ij} is the weight between vertex v_i and v_j defined in equation **??** that is TQLBO.

Then the linear systems for find the new pose of a mesh is.

$$\begin{bmatrix} w_l L \\ W_c \end{bmatrix} X = \begin{bmatrix} \delta \\ W_c C \end{bmatrix} \tag{1-2}$$

Where w_l is the weight for Laplacian Matrix L, the Laplacian matrix L was defined in equation **??** . Wc is a matrix that contains ones int the indexes of anchors vertices. C

is a vector with coordinates of anchors vertices after several transformations. δ are the differential coordinates defined in equation 1-1.

1.7 Performance Testing of Solvers

For this project we choose a numerical solver to be included in Blender software with base in a performance evaluation of computation of the initial factorization of the Laplacian deform system.

Linear equation system to solve

$$\begin{bmatrix} w_l L \\ W_c \end{bmatrix} X = \begin{bmatrix} \delta \\ W_c C \end{bmatrix}$$

Solving the sparse linear system

$$Ax = b$$

Where:

$$A = \begin{bmatrix} w_l L \\ W_c \end{bmatrix}$$

$$x = V$$

$$b = \begin{bmatrix} \delta \\ W_c C \end{bmatrix}$$

1.7.1 Hardware Specification

- Processor: AMD Quad-Core 2.40 GHz
- RAM: 8.0 GB
- OS: Windows 7 Professional
- Graphics controller: NVIDIA Quadro FX 570

1.7.2 Software Specification

CGAL Computational Geometry Algorithms Library

Graphite Research platform for computer graphics

1.7.3 Numeric Solvers Used

CG: Conjugate gradient method.

BICGSTAB: Biconjugate gradient stabilized method.

GMRES: Generalized minimal residual method.

SUPERLU: Sparse Direct Solver, LU decomposition with partial pivoting.

TAUCS_LDLT: A library of sparse linear solvers with LDLT factorization.

CHOLMOD: Supernodal sparse Cholesky factorization.

LU factorization is a numerical method that works with large, sparse, non-symmetric systems of linear equations [2]. We choose the implementation of LU factorization in a OpenNL-SuperLU library, because this method show the better performance for the computation of a solution for a Laplacian Deform linear system of equations presented in equation 1-2 how see in the table and plot **1-1**, **1-2**. OpenNL SuperLU allow works with the Graphics Unit Processor GPU, for exploit the capacity of GPU to work with parallel structures, more fast that traditional CPU.

Model	Vertices	CG	BICGSTAB	GMRES	SUPERLU	TAUCS	CHOLMOD
Cross	24	0.05	0.05	0.04	0.04	0.05	0.05
King	538	0.83	0.63	0.71	0.61	0.68	0.79
YModel	4770	19.60	16.44	16.93	16.06	16.88	17.95
Man	10002	33.43	27.76	29.91	28.54	29.53	30.80
Neptune	28052	133.97	136.46	136.39	133.21	142.87	142.76
Armadillo	34594	194.48	174.88	175.80	169.92	181.70	183.49

Table **1-1**: Vertices Vs Seconds, Laplacian Deform initial factorization performance.

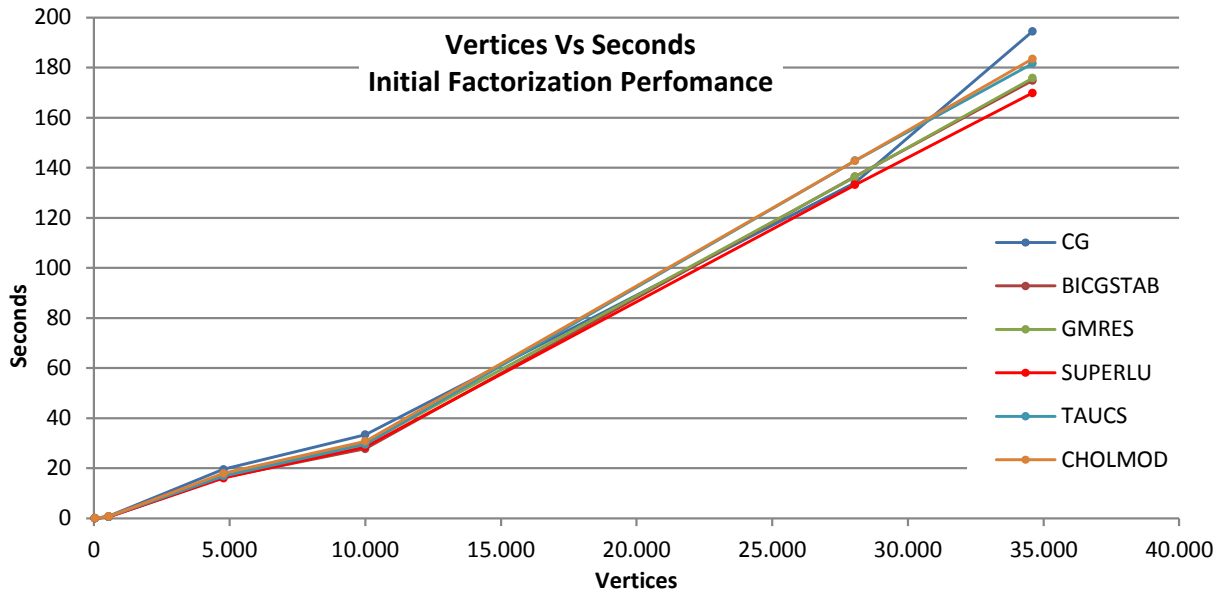

Figure **1-2**: Plot of Vertices Vs Seconds, Initial factorization performance.

1.8 Results

The user interface for software Blender can be seen in figure (**1-3**). In this tool the user define the anchor vertices with the use of vertex groups a feature offered by Blender, the user configure this name in the field *Anchors Vertex Group*. The *Bind* option initiate the system and capture the geometry details in form of differential coordinates and compute the factorization of the linear system, after that the system is ready to repose meshes in real-time user interaction session.

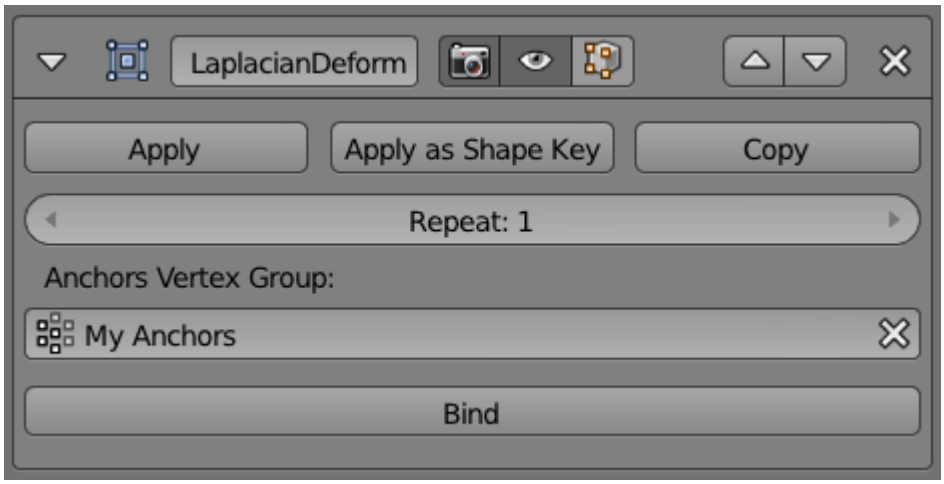

Figure **1-3**: Panel inside blender user interface of the Laplacian Deform modifier tool.

Figure (**1-4**) shows the Laplacian Deform applied in a model with 173K vertices, only anchor vertices was used and are represented in blue color, when the user apply several transformation (location, rotation, scale) over this anchors vertices, the system find a solution and estimate the position of the vertices in yellow color. This method work in real time, to achieve this the matrix $\begin{bmatrix} w_l L \\ W_c \end{bmatrix}$ in the equation (1-2) is decomposed in matrix LU with LU factorization only one time when the system initiate. Once the matrix is factorized the system can solve the unknowns in a fast way in term of milliseconds. For get better results the method permits solve several times the system of equations, and not need factorize matrix LU could be that at every iteration, only the differential coordinates are adjust since the differential coordinates can be rotated at every iteration and get better results. In the figure only four iterations were used, but the system find good solutions with only one iteration when the angle of rotations are less that π.

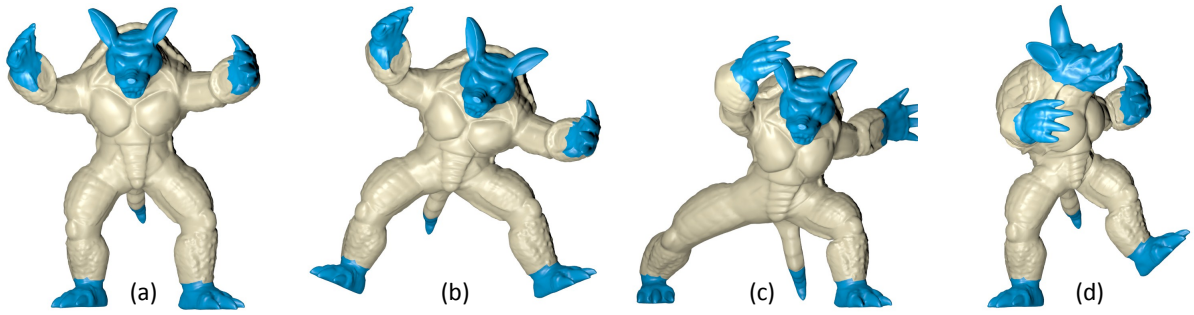

Figure **1-4**: Anchor vertices in blue color. (a) Original Model, (b,c,d) new poses change only anchor vertices, system find positions for vertices in yellow color.

In figure **1-5** show a comparison after made a one simple transformation that consist in rotate 70^{o} to right the parts in blue color. between traditional method to interpolate the changes made in some parts of the mesh to entire model. The results for simple interpolation is show in **1-5**.b and show how the main trunk loss your shape and is rotated. The propagation of changes made with Laplacian deform (figure **1-5**.c) for the same transformation show better results and how the details and shape of the original model are preserve as best as possible.

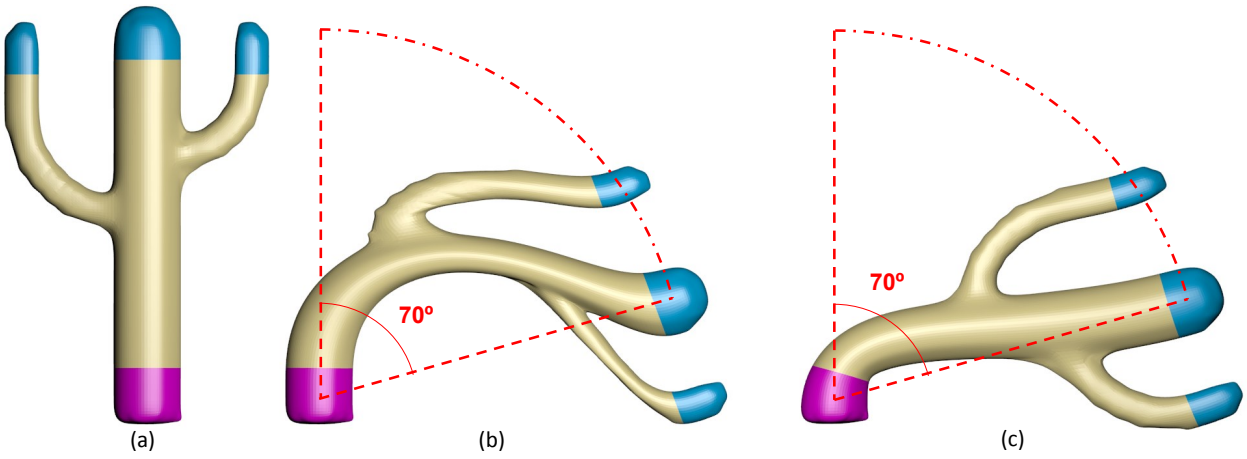

Figure **1-5**: (a) Original cactus model. (b) Rotate 70^{o} to right the blue segments with basic interpolation (c) Rotate 70^{o} to right the blue segments with Laplacian deform tool.

The Laplacian Deform tool allow a user to repose a model while preserve the geometry details in the figure **1-6**.b y **1-6**c you can see the new pose of he horse after five transformations and rotation of the head, in figure **1-6**.b you was use basic interpolation and the shape and details are lost with every change made. In figure **1-6**..c yo can see how the new pose of the horse look more natural, and how the body and neck, of the horse preserve the original shape, this comparison show that method allow to apply any number of transformations without lost details, could be the system try to recover geometry details with base in the original pose of the model.

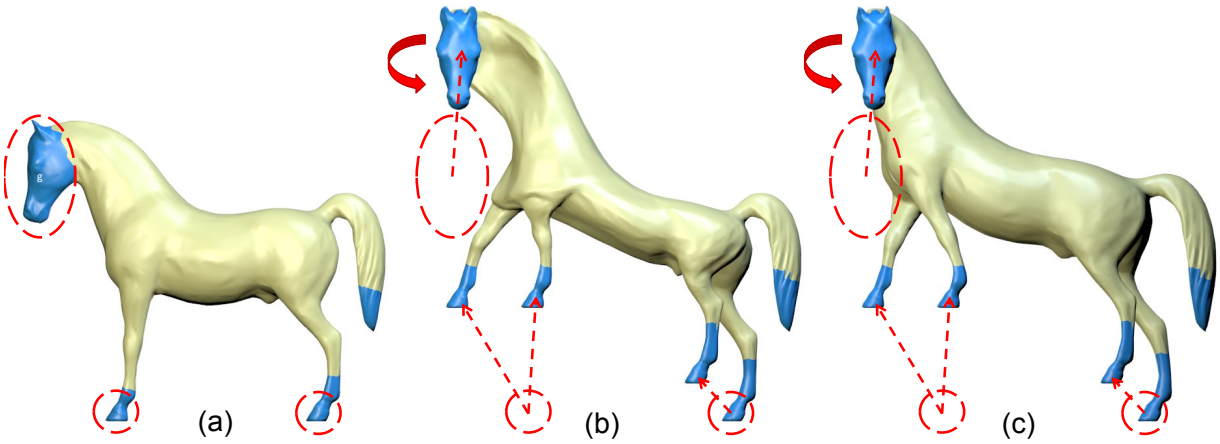

Figure **1-6**: (a) Original Horse model. (b) Translate and rotate the blue segments with basic interpolation (c) Translate and rotate the blue segments with Laplacian deform tool.

Bibliography

[1] Blender-Foundation. Blender open source 3d application for modeling, animation, rendering, compositing, video editing and game creation. http://www.blender.org/, 2012.

[2] Bruno Levy. Numerical methods for digital geometry processing. In *2005 Israel-Korea Bi-national Conference on New Technologies and Visualization Methods for Product Development on Design and Reverse Engineering*, Haifa/Israel, Nov 2005.

[3] O. Sorkine, D. Cohen-Or, Y. Lipman, M. Alexa, C. Rössl, and H.-P. Seidel. Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, SGP '04, pages 175–184, New York, NY, USA, 2004. ACM.