EG UK Theory and Practice of Computer Graphics (2005)
L. Lever, M. McDerby (Editors)

Fast Simulation of Facial Tissue Deformations Using Mass-Spring Chain Algorithm

A. Duysak[1] and J. J. Zhang[2]

[1]Dumlupinar University, Turkey
[2]Bournemouth University, United Kingdom

Abstract
We propose a method to develop a unique head model to be used in craniofacial surgery simulations. This method considers the shape of the head and skull structure and provides a polygonal model, which includes different tissue layers with realistic tissue thickness. We also introduce the use of the new deformation simulation technique called mass-spring chain algorithm in simulation of facial tissue deformations caused by operations on the bone structure. This method produces plausible results and considerably reduces the simulation time.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Physically Based Modelling I.3.7 [Computer Graphics]: Animation and Virtual Reality

1. Introduction

Simulating craniofacial surgery is one of the most challenging applications in deformable object simulation for the following reasons. It is very important that the outcome of the simulation is as accurate as possible. This is a challenge because facial tissue consists of several different tissue layers with different deformation characteristics. It is also very important that the simulation concludes as quickly as possible. Since very high numbers of modelling elements are used in the polygonal model to approximate the facial appearance, simulation takes a long time to perform. Polygonal models must include different tissue layers as well as their connections with the skull. Finding connections may introduce problems because of the skull's unique structure. Finally, the simulation algorithm must carefully choose simulation parameters that capture the tissue characteristics.

Simulation of facial tissue deformations can be considered to consist of two parts; Data acquisition-modeling and simulation phases. The two most common techniques used in acquiring 3D medical images are Computerized Topography (CT) and Magnetic Resonance Imaging (MRI). CT and MRI imaging have become very popular because they create cross-sectional sliced images, which can be stacked to form volume data showing the internal structures as well as outer surface of a body. Volume visualization of the medical image data is necessary for further analysis. At this stage, measurement and manipulation of the data takes place and experts decide the required surgical operations and procedures. Once surgery has been planned, the medical data is processed by a number of algorithms in order to generate a polygonal model suitable for simulation algorithms.

There has been numerous simulation methods proposed that are mainly divided into two categories: non-physically based and physically based. There are some fundamental limitations in non-physical methods such as, the deformation characteristic of the object is not taken into account and the deformation accuracy is based on the user expertise. Physically based models, on the other hand, incorporate the physical properties of the object, thus produce more realistic deformations.

Among physically based methods, finite element modelling (FEM) and mass-spring systems (MSS) have been widely used in a variety of areas from cloth simulations to soft tissue simulations. The finite element method is a common choice if accuracy is the main concern while mass-spring systems may be preferred if speed is essential. But even with the mass-spring system, real-time performance is difficult to achieve, since most real applications involve a large number elements. In addition the mass-spring system is an iterative method, which uses numerical iteration to perform deformation. Therefore, whilst much research effort has been spent on improving such techniques in the area of physical accuracy and performance, other methods, such as the chainmail and mass-spring chain algorithms, have been proposed for real-time interactive frame rates.

1.1. Related work

There are numerous publications dedicated to facial tissue simulations using FEMs. An implementation of FEMs for soft-tissue simulation is given in [RGT*98]. Instead of linear elasticity theory, Roth et al. used higher order polynomial interpolation functions using a Bernstein-Bezeir formulation, therefore aiming for more accurate results and admitting higher computational cost. The main drawback of their work is the lack of global C^1 continuity, which results in lower quality surfaces. An anatomy based 3D finite element tissue model was developed by [KK98]. Their work includes a comprehensive flexibility that allows for any craniofacial operation on the bone structure. They improved their early work by taking into account the individual patient's anatomy [KK99]. They used six node prisms to discretize the face model.

Koch et al. [KGB*96, KRG*02] developed a facial surgery simulation based on volumetric finite element modeling. Their implementation aims for physical accuracy therefore includes geometric and topological detail added interactively to the model, which represents a facial volume by prismatic shape functions. This model provides globally C^1 and internally C^0 continuity. In their work, they registered 3D laser scan data with CT data to achieve photo realistic appearances. Results from their simulations are compared to real surgery images for several different patients. Koch et al. [KGB98] also used a FEM model to implement a facial expression editor. Their generic facial model uses medical data and correct facial anatomy in defining muscle groups.

An MSS is also used in soft-tissue simulations. A real-time muscle deformation using an MSS was studied in [NT98]. Using a new kind of spring type, called an angular spring, the authors simulated a surface-based muscle model. [TW91] and Lee et al. [LTW95] successfully used an MSS for realistic facial modeling and simulation. A detailed facial model represented by a four-layered mass-spring model. In their motion equation, they include volume preservation forces and other constraint forces, such as skull penetration forces. U. Kuhnapfel et al. [KCM00] implemented an MSS in their system for simulating soft tissues. They integrated a mass-spring system simulation module into their surgical training system, which is capable of performing several surgical tasks (such as grasping and cutting). Teschner et al. [TGG99, TGG00] used an optimization approach to improve the physical realism as well as the performance of mass-spring systems. A multi-layer soft tissue model of the head was developed including the skin turgor and sliding effect between soft tissue and bone. They employed a variety of different optimization methods and compared them with regard to the computational cost and the robustness of their results.

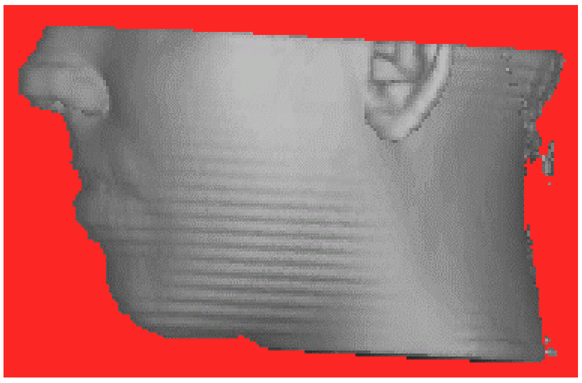
1.2. Our contributions

In this paper we introduce a new method to generate a head model to be used in craniofacial surgery simulation. We use mass-spring chain method for performance reasons. This method works in real time for most applications and has not been applied to simulation of facial tissue deformations before.

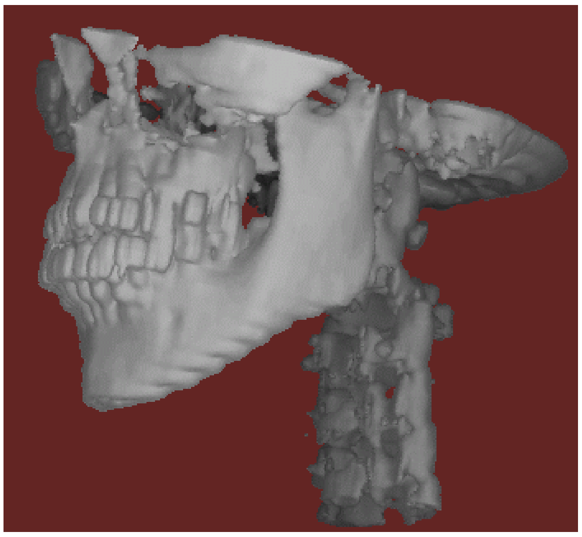
2. Model generation

We use a head image, which is 512*512 pixels in size and consists of 22 slices in the DICOM file format. Slice spacing is 4.91 millimetres and maximum density is 4095.00 while minimum density is 0.00. The pixel size is 0.39 by 0.39 millimetres. The image captures the whole head of a patient. Data is read using the 3DVIEWNIX [3DV] software and then it is visualized. Slice images are segmented using appropriate threshold values and segmentation results for skin and bone surfaces are given in Figure 1 (a) and (b) respectively.

The first step in the reconstruction of 3D geometric models is segmentation, extracting regions or features of interests. The second step is to generate a surface representation of segmented volumes. The geometric models representing the surface of a 3D segmented volume are often described by a set of triangles because of their simple structure. This structure allows fast mathematical manipulations and it is very suitable for simulation algorithms. There are many methods proposed for the isosurface generation. The marching cubes algorithm [LC87] is one of the most popular methods used in generating surface triangulation because of its sub-voxel processing that produce high quality meshes. The bone and face surfaces are generated using the marching cubes algorithm and the results are then decimated ensuring the number of triangles generated is suitable for the simulation algorithm. The resultant face surface consists of 6292 vertices and 10519 triangles and the bone surface has 10056 vertices and 19770 triangles. Once both skin and bone surfaces are generated, tissue layers between them need to be included to the model.



(a)



(b)

Figure 1: Segmenting head image for skin (a) and bone (b) surfaces.

The tissue between the face surface and the bone surface is represented by a number of small prismatic volumes. Given that the skin and bone surfaces are represented by triangles, prismatic elements can be generated using various methods. Each vertex of a triangle of the skin surface is projected on to the bone surface. This is done by finding the intersection points between the normal vectors of each skin vertex and the bone triangles. For better and smooth results the normal vertex is determined by averaging the normals of the triangles meeting at this vertex. Since the face surface contains curved regions, it may be impossible to find an intersection point for every normal vector of a skin vertex. Besides, there are hollow areas on the bone structure preventing an intersection. In addition, some of the intersection points found may be at completely the wrong places. Therefore using the skin vertex normal will not result in a good representation of tissue layers.

In [KK99] a method tracing a ray from each skin vertex to a predetermined point on the bone structure (or a point inside the skull) is used in the generation of the prismatic elements. An average point, called the centre point, is determined for all bone triangles. Each skin vertex is then traced towards this point and intersections with the bone surface are recorded. If there is no intersection, then by interpolating the neighbouring points of intersection a false point of intersection is generated. This method guaranties an intersection point for each skin vertex but may not produce very good prismatic element shapes. In addition some prisms may overlap. This happens because a single centre point can not realistically represent the midpoint of the face, which is not a sphere. Interpolating neighbouring elements in order to assign a connection point may result in an unrealistic approximation as well.

We modified the method mentioned above as follows. One centre point fails to represent the human head accurately, which is not a perfect sphere. Therefore assigning two centre points may better represent the facial model in terms of finding the origin. As shown in Figure 2 we placed two centre points into the bone structure. The lip level at the skin surface determines which skin vertices use a specific centre point. Vertices above the lips level are traced back to the centre point at the upper part of the jaw. Vertices below the lip level use the centre point at the lower jaw. Each skin vertex is then traced back to one of the two centre points and any intersection with the bone triangles is recorded. On this first run we also determine an average thickness based on the intersected rays. On the second run we assign an intersection point to those vertices that did not get a hit in the ray direction on the first run (preventing interpolating the neighbouring points of intersection). The average thickness is used to determine the depth of these intersections. This method also guaranties an intersection for each skin vertex and produces better-shaped prism elements, representing different tissue layers, while minimizing any possible overlaps.

Using ray tracing algorithm, prism elements are obtained. A typical prism element between the skin and bone surfaces is shown in Figure 3. The number of prism elements is equal to the number of skin triangles. It is important to note that none of the skin vertices above the bone level are used in this process. The skin and bone surfaces as well as the bone level can clearly be seen from Figure 2. Therefore the number of skin vertices involved in finding the prism element is 2628 and the number of the skin triangles is 4606.

Once the prism elements representing different tissue layers between the skin and bone surfaces are obtained, surgical operation can take place and consecutively simulation algorithm performs deformation. The simulation algorithm, mass-spring chain is briefly explained. Details of the algorithm and its implementation are given in [DZ2004].

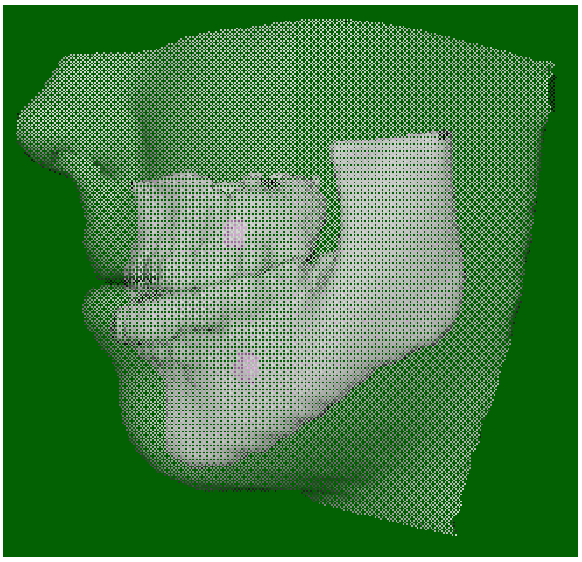


Figure 2: Skin and bone surfaces with two marks representing two centres used in ray tracing algorithm.

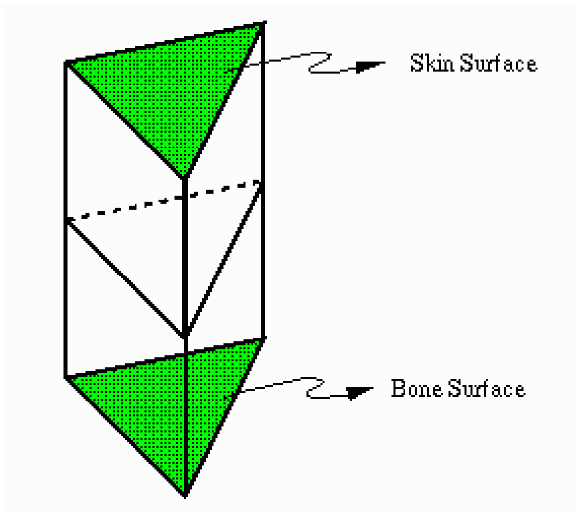


Figure 3: A prismatic element between the skin surface and the bone surface.

3. Mass-spring chain (MSC) algorithm

The Mass-Spring-Chain algorithm models the object in a similar way to the mass-spring systems algorithm in that the object consists of a number of mass-points connected with springs. As in the mass-spring system's algorithm, springs perform a deformation by stretching or compression. The deformation starts from the moved mass-points and propagates through the entire 3D lattice of springs. Spring movement is limited between two extremes; rigid movement and elastic movement. The spring length is also constrained between the allowed maximum compressions and stretching. The deformation algorithm is then responsible for finding the necessary movements and the amounts of deformation of the springs within these set limits. Mass-spring chain algorithm can be viewed in there parts.

3.1. Deformation pattern

The 3D lattice is initially considered to be in a passive state, which implies that all points and springs in the mesh are not under the influence of any external force. Figure 4 (a) illustrates such a lattice where the object is constrained at its two vertices shown as black. When a point in 3D mesh is subject to an external force (by grabbing it and moving it) this particular point becomes an active point or, in other words, a source point for the deformation. An active point is shown as green in Figure 4 (b).

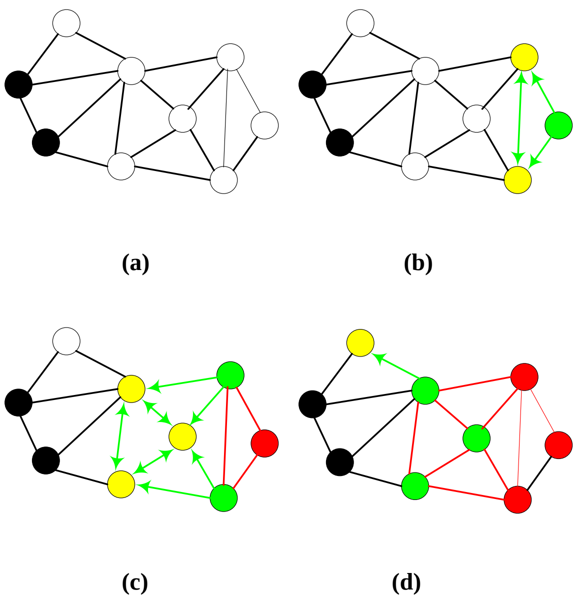


Figure 4: Deformation propagation of the proposed algorithm; (a) initial stage, (b) first, (c) second, and (c) last step.

A deformation starts from an active point and travels through the rest of the lattice in every direction using the springs. The springs connected to the active point are now defined as active springs, because they are subject to movement and deformation. These springs are represented by green lines. Arrows on these lines show direction of the deformation propagation. The other end points of the active springs are called semi-active points, because they will be repositioned (causing the deformation), and will become active points themselves in the next step of the algorithm. Semi-active points are shown in yellow in Figure 4 (b). Springs connecting semi-active points are called semi-active springs and they have arrows at their both ends representing deformation at both ends. Figure 4 (c) shows the second step of the deformation. Previous semi-active points (yellow in Figure 4 (b)) are now active points and shown as green. The last step's active point is now shown by red representing deformed state. The last step of the propagation is given in Figure 4 (d) where only one semi-active point is left. This point is processed and deformation propagation ends here. Constrained vertices are not considered as active or semi-active points and are not repositioned (deformed). Springs connected to constrained vertices, however, are moved and deformed.

3.2. Movement limits and new orientation

We assume that there are two extreme cases possible regarding spring movements. One is defined as a rigid movement without any rotation. When a point is moved, a connected spring moves accordingly. The initial spring and the moved spring are now parallel to each other and the distance between them is equal to the distance traveled by the moving point. An example is given in Figure 5 where initial triangle is represented by $\overset{\Delta}{ACB}$ and vertex A is moved to a new location given by a. The spring given by $\overline{AB}$ makes a rigid movement to a new location given by $\overline{ab}$. A vector called the rigid movement vector, $\vec{u}$, defines this new location and sets the rigid limit or, in other words, the upper limit beyond which there will not be any movement. The opposite situation is known as super elastic movement. It is assumed that the spring offers no resistance to its movement, i.e. while one end is moving the other end stays still. A vector, represented by $\vec{v}$, from the moving end to the stationary end of the spring sets this limit. It is therefore our assumption that the moved and deformed spring will be somewhere between the rigid and elastic movement vectors.

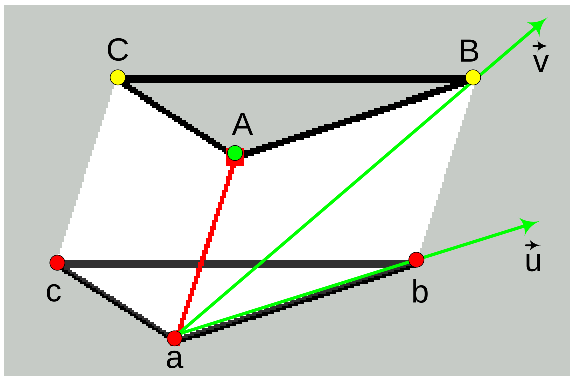


Figure 5: Elastic ($\vec{v}$) and rigid ($\vec{u}$) limit (movement) vectors.

In order to establish the spring location after the vertex movement, we employ a vector called the orientation vector whose purpose is to indicate where the spring lies between the movement vectors $\vec{v}$ and $\vec{u}$. Both movement vectors are located on a surface defined by $S(ABba)$ in Figure 5. Orientation vector and new location for moved spring ($\overline{AB}$) will be on this surface as well. Depending on the material properties, the spring's location will vary between the two movement vectors and this location is indicated by orientation vector. The orientation vector therefore spans the entire surface between $\vec{v}$ and $\vec{u}$. Therefore an equation for the orientation vector can be expressed in terms of the movement vectors as:

$$\vec{w} = \alpha \vec{u} + (1-\alpha) \vec{v}$$

where $\vec{w}$ is orientation vector and α represents the deformation characteristics of the object under consideration. If the object is very elastic in nature, then the orientation vector is expected to be closer to the elastic limit. Alternatively if the springs are defined with a higher stiffness then the orientation vector approaches the rigid limit. Thus, the parameter α can also be considered as a control coefficient for the spring movement. New locations (orientations) for springs in $\overset{\Delta}{ACB}$ triangle are given in Figure 6 for different values of parameter α.

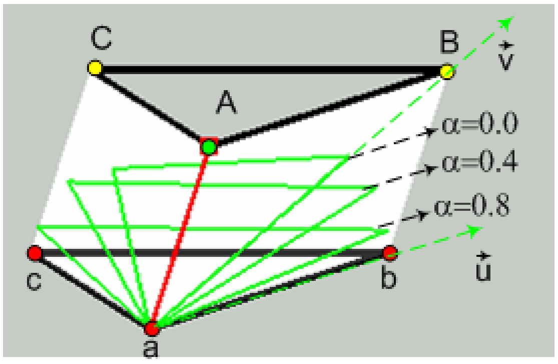


Figure 6: New locations of triangle $\overset{\Delta}{ACB}$ because of movement of vertex A. Control parameter α varies from 0.0 (elastic) to 1.0 (rigid).

3.3. Deformation magnitude and direction

Once new locations for springs are determined deformation algorithm finds necessary deformations and their directions. Springs are allowed to stretch or compress for a certain percentage of their original lengths. The current spring length varies between the maximum compression length and the maximum stretch length. As in the mass-spring system's algorithms, the current spring lengths are found and compared with their initial (pre-set) lengths. There are three possible outcomes from this comparison. If there is no difference between them, there will be no spring deformation. The current spring length may be larger then its rest length. In this case, the spring is being stretched. In opposite case, where the initial spring length is larger than the current spring length, thus the spring is being compressed. In both cases the spring is deformed and the amount of the deformation needs to be determined. The magnitude of the deformation may be calculated using many different formulations. Here, we use the following:

$$def = d_{\max}(1 - e^{-\beta \frac{dr}{r_o}})$$

where $d_{\max}$ is the allowed maximum stretch or compression, dr is the difference between the current and the rest length, r_o is the rest length of the spring and the slope parameter β represents the deformation rate.

In our work, for simplicity we choose the direction of the orientation vector as the deformation direction. If the spring is being compressed, then the deformation direction is opposite to the orientation direction. The amount of deformation is then subtracted from the original spring length, leaving the current spring length shorter than the original length. If the spring is being stretched, then the deformation direction is the same as the orientation direction. In this case the deformation is added to the original spring length elongating the spring. Deforming semi-active springs is slightly more complex than deforming active springs. The strategy is different for semi-active springs because unlike active springs both their endpoints are moving at the same time. Semi-active springs are allowed to move and deform freely during the deformation of active springs. Then the deformation algorithm checks if the semi-active springs violate the spring length criteria. If the maximum stretch and the maximum compression conditions are satisfied, no action is taken. Otherwise, the semi-active springs are deformed to meet the set conditions. Figure 7 shows an animation of a simple geometric figure. The figure is pulled and pushed from one of its vertices while the two most left vertices are constrained.

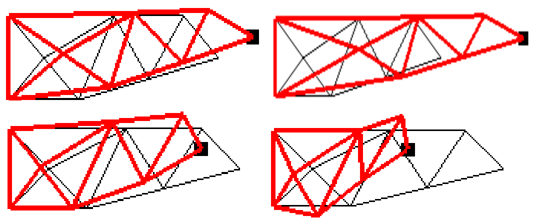
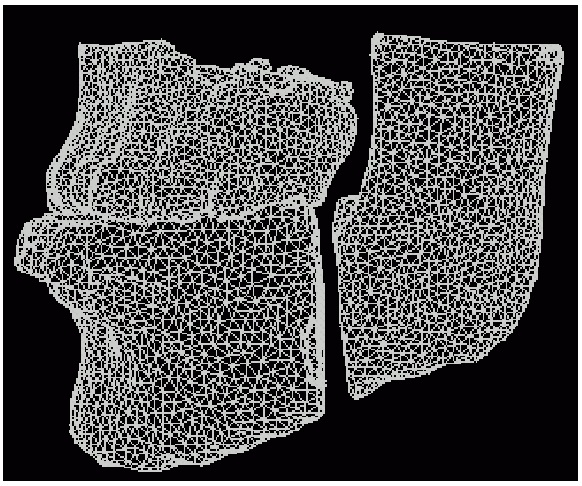


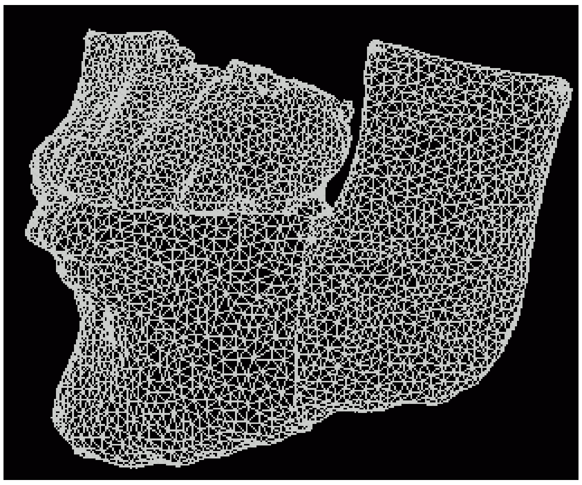
Figure 7: Example of simple figure, which is animated by pulling and pushing one of its vertices.

4. Results and conclusions

Figure 8 (a) and (b) show surgical operations round the jaw area. Lower jaw is cut, Figure 8 (a), and pushed back to align the lower and upper parts, Figure 8 (b). This operation causes deformations on the facial tissues. These deformations are predicted here using mass-spring chain algorithm.



(a)



(b)

Figure 8: Surgical operations are performed on the bone structure.

Deformations are simulated and results are shown in Figure 9. First two figures show pre and after surgery images. Deformations especially around the lips area show the changes on the face surface. The images from pre-surgery and after surgery are given in the last part of the Figure 9, which allows clear comparison between them and shows that simulation algorithm successfully predicts the soft tissue changes due to bone manipulations. Simulation time for this prediction is 0.21 seconds on a 2.40 GHz. Pentium 4 computer. Considering that model consists of 7884 vertices and 19724 springs, mass-spring chain algorithm concludes in a reasonable time.

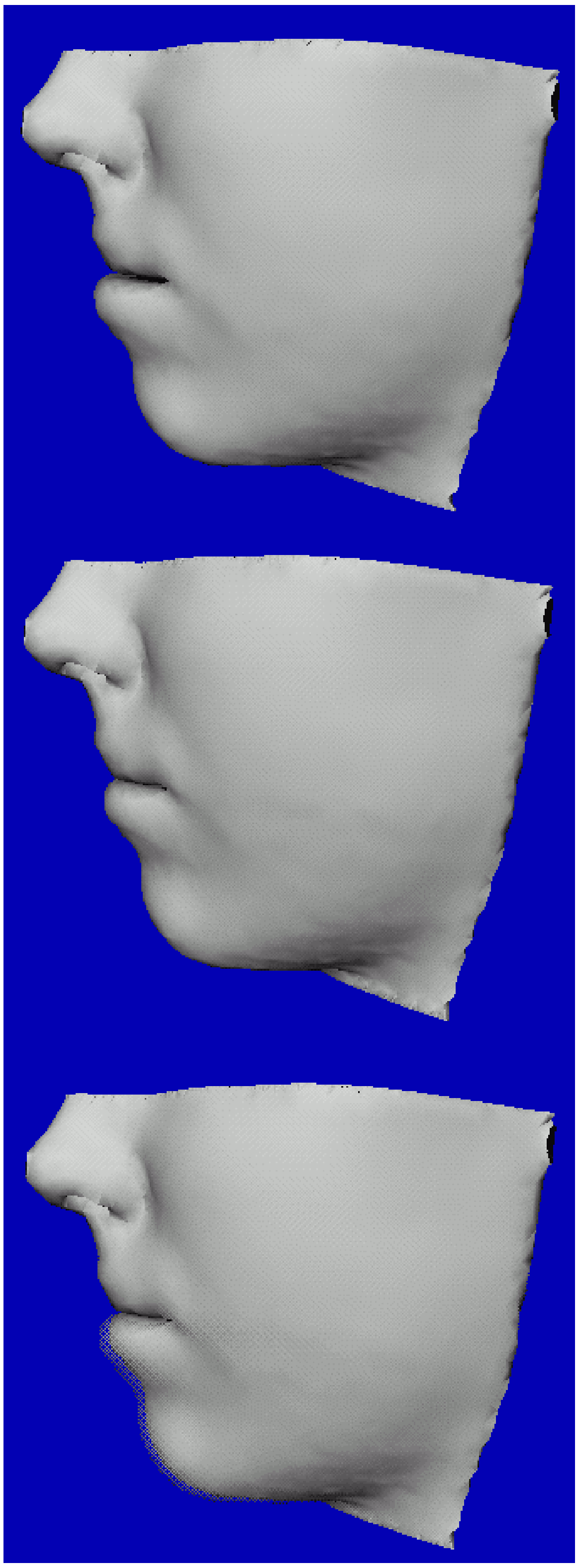


Figure 9: Facial tissue prediction. First image is the face before the surgery, middle one gives tissue deformations and the last image represents both pre and post surgery images superimposed.

An animation of mouth opening due to jaw rotation is also simulated and resultant facial tissue changes are given in Figure 10. The real-time capability of mass-spring chain algorithm is demonstrated in Figure 11, where a moon character, consisting of 3198 vertices and 9505 springs, is animated. Animation time for this example is 0.015 seconds.

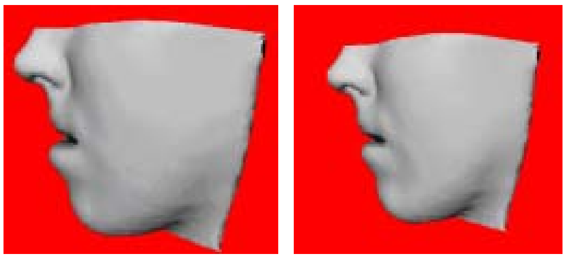


Figure 10: Mouth opening before and after the operation.

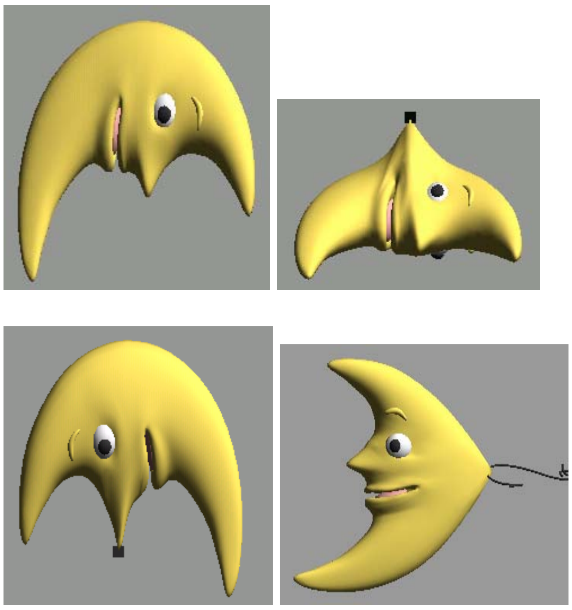


Figure 11: Deformation simulation of a moon character.

Although verification using real data was not done due to the lack of the post-operation images, the results were verified visually by the surgeons. Future works will include comparison of the results with post operation images and with the results form other simulation methods.

References

[DZ04] DUYSAK, A, ZHANG J. J.: Fast Simulation of Deformable Objects, *International symposium on Computer Animation, The 8th International Conference on Information Visualization, IEEE Computer Society, (IV 2004, London)* (2004), pp.422-427.

[KCM00] KUHNOPFEL, U., CAKMAK, H.K., MAAB H.: Endoscapic surgery training using virtual reality and deformable tissue simulation. *Computers&Graphics 24* (2000), pp. 621-632.

[KGB*96] KOCH, R. M., GROSS, M. H., BUREN, D.F., FANKHAUSER, G., PARISH, Y. I. H., CARLS, F.R.: Simulating facial surgery using finite element models. *ACM Computer Graphics SIGGRAPH,(* 1996*)* pp.421-428.

[KGB98] KOCH R. M., GROSS, M. H. BOSSHARD A. A.: Emotion Editing Using Finite Elements, *Proceedings of the Eurographics, Computer Graphics Forum*, (1998) Vol. 17, N0. 3, C295- C302.

[KGK*98] KEEVE, E., GIROD, S., KIKINS, R., GIROD, B.: Deformable modeling of facial tissue for craniofacial surgery simulation. *Invited Paper, Computer aided surgery*, (1998) pp. 1-10.

[KK99] KEEVE, E., KIKINIS, R.: Deformable Modeling of Facial Tissue. *Proceedings of theFirst Joint BMES/EMBS Conference*, (1999) Vol. 1, pp. 502.

[KRG*02] KOCH, R. M., ROTH, S.H.M., GRASS, M.H., ZIMMERMANN, A. P., SOILER, H. F.: A framework for facial surgery simulation.*Proceedings of ACMSCCG (*2002). http://graphics.ethz.ch

[LC87] LORENSEN, W. E., CLINE, H. E.: A High Resolution 3D Surface Construction Algorithm. *ACMComputerGraphics* (1987) Volume 21, No 24, pp. 163-169.

[LTW95] LEE, Y., TERZOPOULOS, D., WATERS, K.: Realistic modeling for facial animation. *ACM Computer Graphics*, (1995) Vol. 29, pp. 55-62, Aug. 6-11.

[NT98] NEDEL, L. P., THALMANN, D.: Real Time Muscle Deformations Using Mass-spring Systems. *Computer Graphics International, Proceedings,(* 1998*)* pp. 156-165.

[RGT*98] ROTH, S. H. M., GROS, M. H., TURELLO, S., CARLS, F. R.: A Bernstein-Bezier Approach to Soft Tissue Simulation. *EUROGRAPHICS* (1998). Volume 17, No 3, pp. C285-29.

[TGG99] TESCHNER, M., GIROD, S., GIROD, B.: Optimization approaches for soft-tissue prediction in craniofocial surgery simulation. *Second Int. Conf. On Medical Image Computing and Computer-Assisted Intervention MICCAI'99*, pp. 1183-1190.

[TGG00] TESCHNER, M., GIROD, S., GIROD, B.: Direct computation of nonlinear soft-tissue deformation. *Vision, Modeling, and Visualization VMV'00*, pp. 383-390.

[TW91] TERZOPOULOS, D., WATERS, K.: Techniques for realistic facial modeling and animating. *Proc. Of Computer Animation'91*, (1991) pp 59-73.

[3DV] 3DVIEWNIX: http://www.mipg.upenn.edu/~Vnews/**.**