Real Time Simulation of Elastic Latex Hand Puppets

Charles A. Wüthrich[†], Jing Augusto[‡†], Sven Banisch[†], Gordon Wetzstein[†], Przemyslaw Musialski[†], Chrystoph Toll[†], Tobias Hofmann[†]

(†) CoGVis/MMC, Faculty of Media

Bauhaus-University Weimar, D-99421 Weimar, Germany

(‡) ABS-CBN Foundation, Mo. Ignacia St.,

Diliman, Quezon City 1100, Philippines

E-Mail: caw@medien.uni-weimar.de

ABSTRACT

Children television productions have been using puppets for a long time. Since the early days of computer animation, computer puppet simulation has been researched intensively. Complex motion capture equipment allows nowadays the real time mapping of movement for virtual puppets (performance animation). However, the costs of capturing equipment are too high and the difference in the workflow make it difficult for small production teams to access and use such technology. This paper presents a system for the real time simulation of elastic latex hand puppets which are used in television productions. After an analysis of the production processes of real puppets and of the materials used for their production, the paper describes the components of the system simulating them. The system connects a high resolution visual mesh to a three-layered 3D mass spring mesh, which is used for the elastic simulation. Polygonal mesh decimation of the puppet surface model is used as a basis for generating the elastic mesh. From the decimated mesh a new metod is proposed for generating the internal layers of the mass-spring mesh. A data handglove is used for transmitting forces to the elastic mesh, indirectly moving the surface of the virtual puppet in real time. Dataglove interaction maps in a natural way the hand movements of a puppeteer to the computer model. The tradeoffs of the implementation on low cost hardware and its efficiency are also discussed.

Keywords

Real Time Animation, Physical Simulation, Interactive Puppet Simulation

1 Introduction

Children television production has been using puppets for a long time. Shows like the ”Muppet Show” are pleasant remembrances of many people’s youth, and puppet characters are particularly well accepted by the young audience because of their natural sympathy and their flair. Children’s attention is more drawn visually to colourful puppet characters on television than to regular adult actors. When watching a puppet, children consider it one of their own, and relate to the character played by the puppeteer in an easy and sympathetic way. Plenty of puppet characters in television became famous, and were a long term companion of many children’s daily life.

WSCG 2006 proceedings, ISBN 80-86943-03-8

WSCG’2006, January 30 - February 3, 2006, Plzen, Czech Republic. Copyright UNION Agency–Science Press

With the introduction of powerful and affordable PC graphics cards with TV output, and the development of more and more algorithms for the photorealistic rendering of different materials, at least theoretically it is possible nowadays to simulate such puppets in real time and feed the resulting TV output through a normal TV mixer, mixing it into a normal TV production. Modern PCs with high performance graphics hardware have become affordable even for institutions with a relatively low budget, and this makes the exploration of the possibility of using PCs for puppeteering attractive, even for institutions not traditionally computer oriented, such as TV production companies. Since the production of real puppets is extremely time and cost intensive, computer modeling and computer simulation are a feasible alternative to handcrafting. All it takes to control the character is a PC, a good graphics card, and an input device. In theory.

The idea of mapping input devices onto a virtual character is not new: starting from the seminal work of Parke on the acquisition and mapping of faces and ex-

pression onto a computer-animated face [31], already in the late eighties and in the beginning of the nineties, tracked data or data stemming from image acquisition systems was mapped onto movement of animated characters [39, 37, 12], often focussing on facial animation, albeit not in real time. The development of real time motion capture devices allowed sensor driven real time motion mapping onto characters: new devices were developed [15]. Parallel to this, researchers explored how to build appropriate interfaces for interactive physically–based animated characters [23]. One of the major problems tackled was how to map captured movement onto different models in real time, a process which is known as retargeting [7, 19, 38]. Often, the underlying models are skeleton based [30]. A relatively recent good overview of the current state of the art in motion capture mapping is done in [33].

Parallel to this, real time elastic simulation methods were being developed in two dimensions for cloth [26] and in three dimensions for virtual surgery [40, 6, 16]. Such models are based on mass-spring systems, or variations thereof, and are capable of solving ordinary differential equations at interactive frame rates. Surgery systems often require expensive parallel hardware, such as linux clusters, to achieve real time frame rates. Cloth simulation is usually done in two dimensions, since fabric cloth can be adequately approximated by a surface. It models interaction with the body of the character wearing the cloth through external collision forces with the cloth itself.

To the authors' knowledge, an attempt to simulate an interactive, user driven three-dimensional elastic mesh for interactive puppets has not yet been done. The problems arising from such a simulation are multiple: such a simulation is not possible in real time if the size of the underlying elastic simulation mesh is too big. For puppet rendering, instead, a high resolution surface simulation is necessary to render the surface detail. The structure of the surface mesh (necessary for the visualization) has therefore to be detached from the structure of the physical simulation, so that elastic computations can be done in real time, and mechanisms have to be found for "attaching" the physical mesh to the high resolution mesh of the puppet surface modeled. Starting from the modeled surface, mesh reduction algorithms have to be applied to the visualization mesh to generate the underlying physical low resolution mesh. The physical mesh has to be at a sufficient low resolution to allow real time simulation, and has to have multiple layers, at least three, to simulate the physics of the puppet materials.

This paper will present the problems and implementation issues of an affordable puppeteering system which is being developed in an international cooperation between the Bauhaus-University Weimar and the ABS-CBN Foundation, an educational television production institution based in Quezon City, Philippines. The requirements on the system are that it should be low cost [1], that it simulates latex (i.e. elastic) puppets in real time, and that it is coupled through a data handglove to a puppeteer. Latex puppets have been used in the puppetry field for quite some time. Their main advantage lies in their elastic properties: through the combination of a harder latex layer on the exterior and of elastic foam in their interior, they map well the expressive characteristics of a real live being.

Section 2 will give a brief description of real latex puppets. In Section 3 we will provide an overview of the system. Section 4 will propose one method for reducing the complexity of the model and generating a three–dimensional elastic mass–spring mesh so that the physical simulation can be run at a low resolution. Section 5 will explain the detail of the simulation of elastic material, while Section 6 will present how forces are applied to the puppet. Finally, we will draw some conclusions and will outline future work issues.

2 Making a real latex puppet

The production of latex puppets has been discovered a long time ago [4, 9]. However, they have been used mostly for step by step animation purposes [10]. Puppet stepwise movement is achieved through the insertion of stiff but bendable materials such as thick iron wire. The animation is then produced by moving the puppet slightly frame by frame. Step by step animation gives the animator excellent control on the results, but requires long production times.

Recently, television stations used latex puppets interactively for real time television production. The operation of the puppets is quite simple. In puppets having a complete body, "loose" parts, such as arms, legs and ears, are manipulated interactively through rods and ropes attached to the puppet. Main body parts instead are moved through a hand inserted in the foam filling the body of the puppet. The hand inserted in the puppet manipulates the head, the mouth, and, more rarely, its ears. Figure 1 shows how the puppeteers manipulate such a puppet. In this paper, we will concentrate on puppets having a hand inserted into them.

The process of production of a puppet is long and tedious. It involves several drying phases, which are very sensitive to the atmospheric conditions of the environment. Due to the fact that such puppets can vary a lot in size, it is not always possible to put the puppets for drying in a controlled environment. Mistakes in the creation process or temperature changes result often into unusable or malformed puppets. As a consequence, very often two or more identical puppets are produced at the same time.

[1] Children television productions have generally a much lower budget than adult television ones.

Figure 1: Puppets are manipulated by inserting the hand in the head. Loose parts are manipulated by a second puppeteer, eventually with rods.

The structure of such elastic puppets is composed of three layers. Externally, a latex rubber layer constitutes the skin of the puppet. This layer can have different consistencies, depending on the number of layers of latex being applied to the surface. Internally, a mattress similar foam is injected in the puppet. The purpose of the foam is to fill in the puppet so as to transmit the movement of the hand of the puppeteer to the surface of the puppet. Finally, a hole is dug in the foam to allow the puppeteer to insert his hand. Mouth movement can be improved by glueing a bent rubber mat into the mouth cavity, so that when the puppeteer moves the hand, the two lips are moved symmetrically. Puppets are painted and hair and cloth accessories are applied to them to enrich visual detail. Figure 2 shows such a puppet.

Figure 2: Froggy, a latex puppet.

The materials used for the puppet give it a quite stiff consistency. The latex surface gets stiffer the more layers are applied, while the foam provides for an even distribution of the forces generated by the hand of the puppeteer onto a large part of the puppet. The puppet is flexible and stiff at the same time, and elastically returns naturally to its resting position when moved.

3 Overview of the system

For puppet simulation in real time, the system should be able to feed the output directly into the TV mixer through the graphics card. For the input, a low cost dataglove in the 100$ price range (Powerglove P5™by Essential Reality) is used. All simulations were done on a low cost 1.7 GHz P4 PC, with an NVidia FX 5700 graphics card with TV output[2].

The virtual simulation of a puppet can be basically subdivided into three parts: input capture, physical simulation of the elastic materials, and mapping of the results of the physical simulation onto the puppet surface. Once the deformed puppet surface is generated, the new positions of the polygon mesh are passed to the graphic card and visualized by it.

Input capture is done through a device independent abstract layer. Data is read through the dataglove libraries, and prepared for the physical simulation.

The physics simulator implements a three-dimensional mass-spring system. On one side, some nodes of the mass-spring system are attached directly and controlled by the finger positions of the dataglove. On the other side, the external nodes of the mesh are attached to the surface of the virtual puppet. The three-dimensional mass-spring mesh is composed of three different layers. Layers generation from the original model is presented in section 4, and the mass spring system real time simulation in section 5.

When the fingertips move, the corresponding nodes to the mass spring mesh apply forces to their neighbours, and movement is transmitted to the surface of the puppet. The mass–spring simulation computes in real time the resulting new positions of the mesh nodes, and deforms the puppet surface display mesh accordingly. Finally, the newly positioned surface geometry is rendered and displayed.

4 Discrete 3D mesh generation

Our puppet models are created using standard digital content creation tools. To preserve the visual quality along with a robust simulation that works at interactive frame rates, we separate the polygonal surface from the simulation mesh. Custom decimation methods are applied to the puppet model generating a three-dimensional tetrahedral mesh from the decimated surface. This mesh is attached to the original surface and updates its shape after each simulation step.

4.1 Decimating the original model

Mesh simplification and multiresolution data structures have many applications in Computer Graphics. They can be used for collision detection, ”level of detail” (LOD) rendering or physical computations. An extensive comparison of different algorithms can be found in [13]. A ’good’ physical mesh should preserve the shape of the original model as well as possi-

[2] This due to budget restrictions of the commissioning institution.

ble, it should have edges with almost equal length and no side flipped faces.

We use a custom combination of three different mesh decimation algorithms. In all cases, a scalar is assigned to each vertex v indicating the decimated surface error Δv. Such error equals the priority of the vertex of being removed after the current iteration. The vertex with the highest error is deleted and the procedure is repeated until a predefined number of vertices remain or the error of removing more vertices becomes too high.

The first decimation method used is the normal flipping mode, which computes the maximum angle of each vertex normal n_i to its neighboring face normals n_{ij}. The error is equal to the angle and can be determined by the scalar product

$$\Delta_i^n = max\left\{cos^{-1}\left(\frac{n_i \cdot n_{ij}}{|n_i|\,|n_{ij}|}\right)\right\} \qquad (1)$$

In case of unit length normals the denominator is 1 and can be discarded. Normal flipping is used to prevent highly curved regions from being decimated and assure side flipping.

The second method preserves equidistant triangles within the decimated mesh. We define the normalized roundness R^* of a triangle as a metric for its equidistance. The roundness R is the ratio between the circumference's radius and the shortest edge's length. The circumference's radius r can be computed as $r = \frac{a}{2sin(\alpha)}$. The length of the edges of an equidistant triangle with unit radius is $\sqrt{1/3}$, thus $R^* = \frac{\sqrt{1/3}}{R}$.

The actual roundness is computed for each face that would be created if the vertex was removed. The priority of vertex i is the minimum of 1-(roundness of each neighboring face j) $\Delta_i^r = \min\left\{1 - R_j^*\right\}$.

Surface simplification using quadric error metrics was introduced by [17] and later extended [18] to correctly decimate colored and textured meshes. It is currently one of the most common mesh decimation techniques used in many 3D–modelers.

Arbitrary vertex pairs are iteratively contracted $(v_1, v_2) \rightarrow \overline{v}$, incident edges are connected to $\overline{v}$, v_2 as well as all edges and faces which have become degenerated are removed. The surface error at vertex $v = [v_x v_y v_z 1]^T$ is expressed by a symmetric 4x4 quadric matrix Q of the quadratic form $\Delta^q\{v\} = v^T Q v$. For a given contraction $(v_1, v_2) \rightarrow \overline{v}$ a new matrix $\overline{Q}$ is derived using the additive rule $\overline{Q} = Q_1 + Q2$. The position of $\overline{v}$ is determined by minimizing $\Delta\{\overline{v}\}$, which can be done by computing the partial derivatives of

$$\begin{aligned} v^T Q v \quad = \quad & q_{11}x^2 + 2q_{12}xy + 2q_{13}xz + 2q_{14}x + \\ & q_{22}y^2 + 2q_{23}yz + 2q_{24}y + q_{33}y^2 + \\ & 2q_{34}z + q_{44}, \end{aligned}$$

thus

$$\overline{v} = \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ 1 & 1 & 1 & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} \qquad (2)$$

Garlands simplification produces the best visual results of these algorithms, while equidistance of decimated triangles is preserved by taking the roundness into account. Normal flipping prevents faces from flipping direction. We compute the error for each vertex as a combination of these three error metrics. Usually normal flipping only indicates an invalid removal if the maximum angle is higher than a certain threshold $\Delta^n\{v\} \geq \tau$ we have an invalid removal. Roundness and quadric error are combined according to weights which are dependent on the puppet model $\Delta\{v\} = \lambda\Delta^r + (1-\lambda)\Delta^q$.

The high resolution visual polygon mesh is deformed after each simulation step according to a connection to the simulation mesh. We use the correspondence between the decimated and the original surface to create a connection map that stores weights and connections from the surface of our tetrahedral layered mesh to the visual structure. Each surface vertex is connected to the decimated vertex with the minimum edge distance. If more that one decimated point is possible all the vertices are connected. Each connection to vertex v_i from the direct connectors $\{v_j\}$ gets a weight ω_i that is relative to the fraction of the distance to the connected vertex and the sum of all distances to vertices that are connected to v_i:

$$\omega(v_i, v_j) = \frac{\delta(v_i, v_j)}{\sum_j \delta(v_i, \{v_j\})} \qquad (3)$$

Updating v_i according to the simulation is done by translating v_i according to the translation T_j of its connectors with respect to the assigned weights ω_{ij}:

$$T(v_i) = \sum \omega_{ij} T(v_j). \qquad (4)$$

4.2 Internal layer generation

Taking the resulting decimated surface as initial source, we introduce a method to generate a full and closed layer of tetrahedrons underneath it.

The reason of creating layers instead of filling the model with a uniform mesh is that real time animation of mass-spring systems is still a computational expensive task with time complexity of $O(n+e)$, where n is the number of mass-points and e is the number of springs [21]. While simulating structured meshes is more stable and in general faster, such meshes do not fit to complicated geometric shapes well. Since in our case the amount of storage and the preprocessing time does not influence the simulation, we decided to use an

advancing front approach combined with the Delaunay criterion [5] to create the physical structure. The solution we have chosen allows at the same time to access different layers of the mesh and adjust attributes like spring constants and mass values, so that different degrees of stiffness can be reached.

In the first step we use the "biting spheres" approach [24] to provide proper points in the domain. This method is based on the sphere packing: the basic idea is to fill a three-dimensional geometric domain with spheres (bubbles) to generate new points in the interior. Since we want to create only layers of tetrahedrons instead of stuffing the whole object, we combine this method with the advancing front approach as proposed by Li [25].

Starting from the reduced surface mesh we define a sphere on each vertex with the radius of the half of the distance to the next neighbour vertex. On each intersection of at least three spheres we create new points in the interior of the object. Iterating this procedure with the newly created points generates the next layer of steiner points in the geometric domain.

Next, we use the Delaunay criterion to create new tetrahedrons from existing points, which we define as follows: let P be subset of the whole geometric domain Ω where P contains the inner points p_l created by the procedure described above. Assume that none of the faces $\triangle_{p_j p_k p_l}$ containing the points $p_j p_k p_l \in \Omega$ is a surface face. The Delaunay point for every $\triangle_{p_j p_k p_l}$ is a point $p_{delaunay} \in P$ such that:

$$\begin{aligned}(\|p_{delaunay} - p_j\| &< \|p_{delaunay} - p_m\|) \wedge \\ (\|p_{delaunay} - p_k\| &< \|p_{delaunay} - p_m\|) \wedge \\ (\|p_{delaunay} - p_l\| &< \|p_{delaunay} - p_m\|),\end{aligned}$$

where $\forall m \neq j,k,l$ and $p_m \in P$.

In fact, this condition is still not sufficient to find proper points for creating new cells, therefore prior tests such as verifying the positive distance of the points with respect to the faces are needed. This boils down to finding the proper points $p_{delaunay}$, and connecting them with their corresponding faces $\triangle_{p_j p_k p_l}$ creates new tetrahedrons $tet_{p_j p_k p_l p_{delaunay}}$. According to the generated mesh, we create mass-points and springs for the physical system.

To reach best elasticity while preserving shape different stiffness properties of springs and different mass values are needed in the different layers. Since we can separately access springs and mass-points lying on the surface, on the interior, or even in between, changing their attributes ends up in different physical behavior of the simulated material.

During the simulation, forces applied on several points propagate stepwise to the whole structure. To transmit the deformations to the surface, each outer mass-point is tied in position to its reduced surface vertex, and after every simulation step the position of the polygonal surface is updated. After this operation, a separate rendering module can access the data and display the puppet.

5 Real time elastic simulation

The first issue that has to be addressed when implementing latex puppets is the simulation of the physical properties of the materials involved. Since a layered structure of different materials is to be simulated, an adjustable and flexible physical model is needed, which allows that different extents of elasticity, viscosity, plasticity, etc. are present within it. The model must furthermore be able to perform in real time.

Mass–spring systems meet these requirements, and have been frequently used for the simulation of deformable objects, such as hair [32, 1, 2], cloth [27, 14, 21][3] and three dimensional bodies [8, 22]. Terzopoulos was among the first to suggest mass–spring systems in computer graphics [35, 34, 36] for simulating elastic behaviour. In 1988, Miller [28] simulated dynamics of snakes and worms using a chain of masses and springs. With an increase of computation power of computer systems, more complex structures have been animated, and it is nowadays possible to simulate them in real time.

Mass–spring systems involve complex mathematics: they base on Newtons fundamental law $\vec{F} = m\vec{a}$. For a model consisting of n masses a system of n second order differential equations (ODEs)

$$M\ddot{p} = F(t, p, \dot{p}) \tag{5}$$

has to be solved. The n dimensional vector $p = (p_1, p_2, \ldots, p_n)^T$ represents the positions $p_i = (x_i, y_i, z_i)^T$ of all points. The $n \times n$ dimensional matrix M stores the masses of all particles on its diagonal, and F is a function which describes the system's force field at time t. Equation 5 can be reduced to the system of $2n$ ODEs of first order

$$\frac{d}{dt}\begin{pmatrix} p \\ \dot{p} \end{pmatrix} = \frac{d}{dt}\begin{pmatrix} p \\ v \end{pmatrix} = \begin{pmatrix} v \\ M^{-1}F(t) \end{pmatrix} \tag{6}$$

by introducing the velocity $v = (v_1, v_2, \ldots, v_n)^T$ as a separate variable.

There exist several ways to numerically solve such a system of ODEs. Numerical time integration methods range from the simple explicit Euler scheme [28], which is very fast at expense of accuracy and instability, over higher order explicit methods such as Runge Kutta, to implicit predictor–corrector schemes [3, 27], which are considered more stable [11].

[3] See [26] for an up to date overview of deformable models in cloth simulation.

For our system, we use the implicit Euler scheme

$$\begin{aligned} p(t+h) &= p(t) + hv(t+h) \\ v(t+h) &= v(t) + hM^{-1}F(t+h) \end{aligned} \quad (7)$$

to find the positions at the next time step $t+h$. Here, the system's force field $F(t+h)$ has to be approximated through the second order Taylor series

$$F(t+h) = F(t) + J\Delta p(t+h) + H\Delta v(t+h), \quad (8)$$

where J is the Jacobian and H the Hessian matrix of the system's force field with respect to the positions and, respectively, velocities. With $\Delta p(t+h) = hv(t+h)$ and $\Delta v(t+h) = hM^{-1}F(t+h)$ in Equation 7 we derive

$$\Delta p(t+h) = h(v(t) + \Delta v(t+h)), \quad (9)$$

which is a non–linear equation of the form

$$y(x+h) = f(x+h, y(x+h)). \quad (10)$$

Equation 10 can be solved iteratively through

$$\begin{aligned} y(x+h)^0 &= f(x, y(x)) \\ y(x+h)^{\mu+1} &= f(x+h, y(x+h)^\mu), \end{aligned} \quad (11)$$

where μ represents the iteration number. As suggested in [21], we use the result after one iteration which is sufficient approximation of the solution for real time purposes.

In order to imitate the multi–layered material behaviour of real puppets, a three–dimensional mesh has been created as described in Section 4, and layers have been defined. The entire mesh forms a single mass–spring system. The layered structure is achieved by tuning parameters of the mass–spring system according to the desired materials properties of each layer. This results in a mass–spring system in which the layers result from regions having the same parameter values. Material properties can be made to resemble foam, latex or other materials used for real puppets.

The parameters influencing the behaviour of the puppet are:

- The *time step* h, which has a very strong influence on the behaviour of the simulation. It can be set by the user within reasonable limits. A very small time step decreases the system's reactiveness because of increased computations, whereas a too large time step will lead to instability. Once set, the step size usually stays constant over the whole simulation.
- The *mass* M of the physical points, which determines how big the inertia on the system is. If masses are increased, the reactiveness to external and internal forces is reduced. Thus M can be used to achieve different behaviours of the model.
- The *spring constants* C_{spring}, which are used to calculate the Hookean part of the internal forces. A layered structure (e.g. foam and latex) can be modeled by adjusting the spring constants within the layers.
- The *damping coefficients* C_{damp}, which determine how quickly mass points can be moved by spring forces. It is therefore a measure of the viscosity of the deformable body and strongly affects its elastic behaviour.
- The *topology* T of the three–dimensional mesh, which influences the stiffness of the puppet, and is given by the arrangement and density of mesh points within the puppet's body.

Mass–spring systems are often prone to instabilities, especially if an explicit integration scheme is used. The time step h has to be set very small to ensure stability. We use the implicit Euler method because it is accurate while allowing the step size h to be larger [11]. Assuming that the non–linear system of equations is solved properly, the implicit Euler method is unconditionally stable [20] because the systems force field is consistent over time. If the non–linear system of equations is solved iteratively, as in our case, new restrictions on stability arise. A detailed mathematical analysis of the problem is not done here, since the majority of parameter configurations of M, C_{spring}, C_{damp} and T do not cause instabilities when the time step h is within a proper range, i.e., for normal usage of the system.

For puppet simulation real time interaction is required, and the performance of the system is very important. The easiest way to decrease the computational complexity is to reduce the size of the mass–spring system, i.e., to decrease the number of mass points and springs. Table 1 shows the performance of the physical solver. At the required frame rate of a PAL television (25fps) it is possible to simulate in real time a mass–spring system constituted by up to 5000 mass points and 14600 spring links. Through the decimation and layer generation methods presented above, we reduced the models to such sizes.

Masses	Springs	Update time (in s)
3744	1872	0.015
3744	6952	0.025
5664	15056	0.043
9978	25892	0.076

Table 1: Performance of the physical system.

6 Applying forces to the puppet

To generate the shape of the Puppet, a separate modeler is used. Five node regions on the physical mesh are marked by hand, one for each finger of the data handglove. The individual regions must of course consist of adjacent points, and must be disjunct. The marked points are preserved through the mesh reduction process, so that the corresponding nodes do not disappear in the simplification.

When the puppeteer moves his fingers, the movement of the fingertips is converted into forces according to Newton's first law, and fed to the physics engine as external forces acting on the corresponding nodes. The physical simulation then computes the state of the mass–spring system in the next time interval, and updates the position of the mesh nodes. Since puppets are used in live production the system must be tuned for output at TV signal frame rates.

Figure 3 shows the result of opening the dataglove on the face of a computer generated puppet. The accompanying video shows a real time recording of the computer output while operating the puppet.

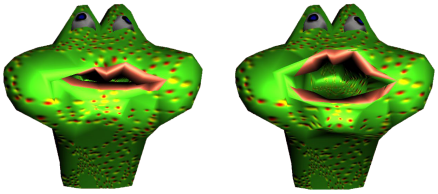

Figure 3: Elastic simulation propagates the movement of the dataglove to the mouth of the puppet

7 Conclusions and future work

We have presented a system for the simulation of hand puppets made of latex. The system is capable of running the simulation in real time on low cost machines, allowing a puppeteer to interactively "operate" the puppet, and allowing real time output feeding in a chroma key device, and consequently the mixing of the simulated puppet with a live television signal.

The system couples a data hand glove to application points of a three-dimensional mass-spring system. When the finger sensors of the data glove are moved, a force is applied to the corresponding points of the mass-spring system, which in turn propagate the movement to the rest of the nodes. The surface of the puppet is attached to the mass-spring system, and therefore is moved indirectly by the hand of the puppeteer. The system is capable of computing in real time (at 25 frames per second) the dynamic simulation for a mass-spring system having more than 5000 mass points and around 14600 springs, and to display the deformed surface of the puppet at the frame rate mentioned above.

Since the resolution of the mass spring system is much lower than the polygon resolution needed to display the detail needed for a broadcast quality puppet, a method for the derivation of the mass-spring mesh from the surface mesh of the puppet has been developed. This method computes first a simplification of the surface mesh, reducing the number of polygons to a predefined number. The centers of the resulting polygons are glued to the surface mesh of the puppet and constitute the first layer of the physical mass spring mesh. From this reduced layer, a new method for the generation of the internal layers of the mass spring system has been developed.

Although the system works well in all tested cases, the resulting mass spring mesh is not uniform. This because polygon simplification algorithms are optimized for visual shape appearance, not for regularity. It is known [29] that uniform tetrahedral meshes behave well as mass-spring systems. Theoretically, this makes the system prone to instability. However, even during several long test session made by children, the mesh topology did not generate unexpected behaviours. We are currently working on mesh simplification algorithms tuned for mesh regularity.

One of the challenging aspects of numerically solving partial differential equations is understanding how parameters like time step or mesh configuration influence the stability of the solvers. Time step size adaption helps greatly the stability of the system at the cost of speed. It would be of benefit to study when and at which costs step adaption can to be applied.

Acknowledgments

We are grateful to the ABS-CBN Foundation, Quezon City, Philippines for allowing Jing Augusto to come to Weimar for the entire project duration. Thanks also to the students that have been involved in the project and implemented parts of the system: Uwe Hahne, Bernhard Bittdorf, Benjamin Schmidt, Annekathrin Linge, Andreas Emmerling, Andreas Kunze, Jonas Schild, Sebastian Knoedel and Marc Pelao.

References

[1] K. Anjyo, Y. Usami, and T. Kurihara. A simple method for extracting the natural beauty of hair. In *Proc. of SIGGRAPH '92*, pages 111–120. ACM Press, 1992.

[2] Y. Bando, T. Nishita, and B. Chen. Animating hair with loosely connected particles. *Comp. Graphics Forum*, 22(3):411–418, 2003.

[3] D. Baraff and A. Witkin. Large steps in cloth simulation. In *Proc. of SIGGRAPH '98*, pages 43–54. ACM Press, 1998.

[4] J. Bell. *Strings, Hands, Shadows: A Modern Puppet History*. Detroit Institute of Arts, Detroit, MI, 2000.

[5] M. Bern and P. Plassmann. Mesh generation. In J. Sack and J. Urrutia, editors, *Handbook of Computational Geometry*, chapter 6. Elsevier, 2000.

[6] D. Bielser, V. Maiwald, and M. Gross. Interactive cuts through 3-dimensional soft tissue. *Comput. Graph. Forum*, 18(3):31–38, 1999.

[7] B. Bodenheimer, C. Rose, S. Rosenthal, and J. Pella. The process of motion capture - dealing with the data. In *Proc. of the Int. Conf. on Computer Simulation and Animation 97, Budapest, Hungary*, pages 3–18. Springer Verlag, Vienna, Austria, 1997.

[8] D. Bourguignon and M.-P. Cani. Controlling anisotropy in mass-spring systems. In *Proc. of Computer Animation and Simulation '00*, pages 113–123, Wien, aug 2000. Springer.

[9] T. Brierton. At last, foam puppet fabrication explained! *Animation World Magazine*, 1(2), 1998.

[10] T. Brierton. *Stop-Motion Puppet Sculpting: A Manual of Foam Injection, Build-Up and Finishing Techniques*. McFarland & Company, Jefferson, N.C., 2004.

[11] I. Bronstein, K. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, Thun, third edition, 1997.

[12] G. Cameron, A. Bustanoby, K. Cope, S. Greenberg, C. Hayes, and O. Ozoux. Motion capture and CG character animation (panel). In *Proc. of SIGGRAPH'97*, pages 442–445, 1997.

[13] P. Cignoni, C. Montani, and R. Scopigno. A comparison of mesh simplification algorithms. *Computers & Graphics*, 22(6):37–54, 1998.

[14] M. Desbrun, P. Schröder, and A. Barr. Interactive animation of structured deformable objects. In *Proc. of Graphics Interface '99*, pages 1–8, Kingston, ON, June 1999.

[15] C. Esposito, W. B. Paley, and J.-C. Ong. Of mice and monkeys: A specialized input device for virtual body animation. In *Simposium of Interactive 3D Graphics*, pages 109–114, 213, Monterey, CA., 1995.

[16] F. Ganovelli, P. Cignoni, C. Montani, and R. Scopigno. A multiresolution model for soft objects supporting interactive cuts and lacerations. *Computer Graphics Forum*, 19(3), 2000.

[17] M. Garland and P. Heckbert. Surface simplification using quadric error metrics. In *Proc. of SIGGRAPH '97*, pages 209–216. ACM Press, 1997.

[18] M. Garland and P. Heckbert. Simplifying surfaces with color and texture using quadric error metrics. In *Proc. of IEEE Visualization 98*, pages 263–269, 1998.

[19] M. Gleicher. Retargeting motion to new characters. In *Proc. of SIGGRAPH 98*, pages 33–42, 1998.

[20] M. Hauth. *Visual Simulation of Deformable Models*. PhD thesis, Univ. of Tübingen, 2004.

[21] Y.-M. Kang and H.-G. Cho. Complex deformable objects in virtual reality. In *VRST '02: Proc. of the ACM symposium on Virtual reality software and technology*, pages 49–56. ACM Press, 2002.

[22] U. G. Kühnapfel, H. K. Çakmak, and H. Maaß. Endoscopic surgery training using virtual reality and deformable tissue simulation. *Computers & Graphics*, 24(5):671–682, 2000.

[23] J. Laszlo, M. van de Panne, and E. Fiume. Interactive control for physically-based animation. In *Proc. of SIGGRAPH 00*, pages 201–208, 2000.

[24] X.-Y. Li, S.-H. Teng, and A. Üngör. Biting spheres in 3d. In *8th Int. Meshing Roundtable*, pages 85–95, 1999.

[25] X.-Y. Li, S.-H. Teng, and A. Üngör. Biting: advancing front meets sphere packing. *Int. Jour. for Numerical Methods in Engg(2000)*, 2000.

[26] N. Magnenat-Thalmann, F. Cordier, M. Keckeisen, S. Kimmerle, R. Klein, and J. Meseth. Simulation of Clothes for Real-time Applications. In *Proc. of Eurographics 2004, Tutorial 1*, 2004.

[27] M. Meyer, G. Debunne, M. Desbrun, and A. Barr. Interactive animation of cloth-like objects in virtual reality. *Journ. of Visualisation and Comp. Anim.*, 2000.

[28] G. S. P. Miller. The motion dynamics of snakes and worms. In *Proc. of SIGGRAPH '88*, pages 169–173. ACM Press, 1988.

[29] N. Molino, R. Bridson, J. Teran, and R. Fedkiw. A crystalline, red green strategy for meshing highly deformable objects with tetrahedra. In *Proc. of the 12th Int. Meshing Roundtable*, Santa Fe, NM, 2003.

[30] S. Oore, D. Terzopoulos, and G. Hinton. Local physical models for interactive character animation. *Comp. Graphics Forum*, 21(3):1–17, Sept. 2002.

[31] F. I. Parke. *A Parametric Model for Human Faces*. PhD thesis, Dept. of Computer Science, Univ. of Utah, 1974.

[32] R. E. Rosenblum, W. E. Carlson, and E. Tripp, III. Simulating the structure and dynamics of human hair: modelling, rendering and animation. *Journ. of Visualisation and Comp. Anim.*, 2(4):141–148, 1991.

[33] H.-J. Shin, J. Lee, S.-Y. Shin, and M. Gleicher. Computer puppetry: An importance-based approach. *ACM Trans. on Graphics*, 20(2):67–94, 2001.

[34] D. Terzopoulos and K. Fleischer. Modeling inelastic deformation: viscolelasticity, plasticity, fracture. In *Proc. of SIGGRAPH '88*, pages 269–278. ACM Press, 1988.

[35] D. Terzopoulos, J. Platt, A. Barr, and K. Fleischer. Elastically deformable models. In *Proc. of SIGGRAPH '87*, pages 205–214. ACM Press, 1987.

[36] D. Terzopoulos, J. Platt, and K. Fleischer. From goop to glop: Heating and melting deformable models. In *Proc. of Graphics Interface '89*, pages 219–226, 1989.

[37] D. Terzopoulos and K. Waters. Analysis and synthesis of facial image sequences using physical and anatomical models. *IEEE Trans. on Patt. Analysis and Mach. Intelligence*, PAMI-15(6):569–579, 1993.

[38] S. Vacchi, G. Civati, D. Marini, and A. Rizzi. Neo euclide: A low-cost system for performance animation and puppetry. In *Proc. of Gesture Workshop '03*, pages 361–368, Wien, 2003. Springer Verlag.

[39] L. Williams. Preformance-driven facial animation. *Proc. of SIGGRAPH '90*, 24(4):235–242, 1990.

[40] R. Yagel, D. Stredney, G. Wiet, P. Schmalbrock, L. B. Rosenberg, D. Sessanna, and Y. Kurzion. Building a virtual environment for endoscopic sinus surgery simulation. *Computers & Graphics*, 20(6):813–823, 1996.