Fast Body-Cloth simulation with moving humanoids

J. Rodriguez-Navarro[1], M. Sainz[2] and A. Susin[1]

[1]Dept. Applied Mathematics, Universitat Politecnica de Catalunya, Barcelona, Spain.
[2] Developer Technology Group. Nvidia Corporation.

Abstract

In this paper we present a very fast method for body-cloth animation. The usual bottle-neck in cloth simulation performance is collision detection, which becomes more difficult to solve when a complex geometry, like a human body, is involved. Recent image based methods, that use depth images to detect collisions, usually relays on CPU for collision correction. In our work we implement a GPU based simulation that takes care both of cloth simulation and body-cloth collisions when the humanoid is moving. Our solution is based on a hierarchic depth map structure. A high frame rate is obtained with both structured and unstructured cloth meshes with thousands of particles.

Categories and Subject Descriptors (according to ACM CCS): I.3.7 [Computer Graphics]: Animation

1. Introduction

Cloth simulation can be considered as a particular case included at the more general framework of deformable models. A deformable model simulation requires the update of the model vertex positions at each time step according to some deformation law. This can be considered like a new object representation at each time step and was not possible to take advantage of the old graphic cards for deformable simulations. Nowadays graphic cards incorporate programable capacity for their GPU (Graphics Processor Unity) and allows to read vertex positions from a texture using Vertex Texture Fetch (VTF) in a vertex program. As we will show, this can be used for achieving a high frame rate simulation on deformable models. Moreover, we have also introduced a new technic, the *quasi-feedback* method, that give us a fast criteria to ask about the status of the system during simulation on the GPU. This was only possible, so far, reading back to CPU with the corresponding decrease in the performance.

More precisely, we are facing the problem of cloth simulation on the GPU and the interaction between cloth and solid objects, essentially a human body. Collision detection is usually one of the bottleneck for cloth simulation because a huge number of computations are involved. When using the graphics card pipeline a new image based tool is available for collision detection: depth map from appropriate point of views allow us to detect when a cloth particle is inside another object and then a collision correction can be activated.

In order to have more versatility we have used both structured rectangular and unstructured triangle meshes for the cloth, when only structured rectangular meshes are used the frame rate can be increased, due to the known neighbor relationship which takes advantage of the internal texture codification. Unstructured triangle meshes are more general but, as we will explain, it will require an extra texture for taking account of vertex connectivity.

Early works are essentially devoted to the modelling aspects. Papers by Terzopoulos et al. [TF88] where the first physically-based ones, they introduced cloth simulation for graphics community as a problem of deformable surfaces and used techniques from mechanical engineering like the finite element method and energy minimization. Other approaches on the dynamic modelling has been, the energy-based models introduced by Baraff and Witkin [BW98], the particle-based one by Breen et al. [BHW94] and the mass-spring one by Provot [Pro95].

The work of Jacobsen [Jac01] for the game industry has been the pioneer in introducing a dynamic model for the cloth together with the numerical integrator, the Verlet method, which is suitable for GPU implementations. One of the first GPU cloth implementation is due to S. Green [Gre03], a structured rectangular mesh cloth and a solid sphere are simulated using Verlet method on the GPU, but only stretch forces has been simulated. For dealing with collisions on the GPU image based methods are introduced by Vassilev et al. [VSC01].

2. Cloth Simulation model

Our cloth simulation model is based on [Jac01] and the GPU implementation presented in [Gre03]. We extend these previous works to a general triangulated mesh solving the difficulties arising when a different number of neighbors has to be considered for each vertex.

The usual Newton's dynamic equations are integrated using a stable Verlet numerical scheme which can be stated for each particle as

$$\mathbf{x}^i_{n+1} = \mathbf{x}^i_n + \varepsilon(\mathbf{x}^i_n - \mathbf{x}^i_{n-1}) + \mathbf{a}^i \Delta t^2. \quad (1)$$

Here $\mathbf{x}^i_n$ stands for the present particle position and $\mathbf{a}^i$ its acceleration (essentially gravity). $0 < \varepsilon \leq 1$ is a drag term usually used to stabilize numerically the system. The time step Δt is the one used for advance in time (0.01 in our simulations). Indeed only external forces are taken into account when computing the acceleration term. Internal forces are included in the simulation as a restriction of the movement.

The cloth model is based on a particle system where each vertex is a mass particle and the plane deformation properties of stretch, shear and bending are simulated as a constraint on the initial length of the edges of the mesh [Pro95]. For unstructured triangular meshes we simulate internal forces using connections between a vertex and all neighbor particles. For that, we have an auxiliary texture, *connectivity texture* used as a look-up table storing both connection indices and rest lengths.

Length constraints are satisfied using a relaxation iterative procedure. At each time step and for each of the mesh edges, if the present length is different than the original (or the rest) one, the two particles on this edge are displaced a percentage (stiffness constant) of this difference in the edge direction. Usually the stiffness constant, k_s, is 0.5. Thus, if we denote by $d_{ij} = \| \mathbf{x}^i_n - \mathbf{x}^j_n \|$ and we assume, for instance, that $d_{ij} > l_{ij}$ being l_{ij} the rest length between particles i and j, then the corrected positions will be

$$\begin{aligned} \mathbf{x}^i_n &= \mathbf{x}^i_n - k_s * (d_{ij} - l_{ij}) * \mathbf{v}_{ij}, \\ \mathbf{x}^j_n &= \mathbf{x}^j_n + k_s * (d_{ij} - l_{ij}) * \mathbf{v}_{ij}. \end{aligned} \quad (2)$$

Here, $\mathbf{v}_{ij}$ stands for the unit vector at the edge direction from particle $\mathbf{x}^j_n$ to particle $\mathbf{x}^i_n$.

After several iterations of the same procedure, the system will converge to the rest length for all edges. The number of iterations were fixed in all previous implementations of such an iterative process in GPU so far. We have implemented a new technic, the *quasi-feedback method*, to stop an iterative process saving a lot of useless iterations. This method consists in a mechanism to obtain a value from the GPU that reports of the system status. In our case we want to stop iterations when the deformation of the cloth is not significant. For that, we render the cloth discarding the points whose displacement are less than a suitable threshold, we call the `ARB_occlusion_query` extension and it gives the number of samples that are less than this threshold. We stop the iterative process when no correction is needed. Finally, the cloth is rendered directly using `NV_vertex_program3` extension what have the possibility to fetch vertex from a texture in the vertex program.

3. Dealing with Collisions

In computer animations, in general, there is always a bottleneck: the collision problem. For dealing with a complex object, like a human figure, an image-based hardware accelerated approach is one of the best choices. We follow a depth map based approach similar to [KLRS04] but we extended it to collision with moving objects. Moreover, using actualized depth information, we are not forced to consider the human body as a set of rigid bodies, but we can consider it as a deformable surface, taken into account the skinning correction at the vertex near the joints and also consider muscle shape deformations.

For moving models, the work of [VSC01] was a first step, but they use only two depth maps, front and back, for the whole figure. Therefore, when some occlusions arises, for instance when the arm passes in front of the chest, no possible information about collisions can be extracted from these depth maps. To overcome this problem we consider several detailed views, in fact, we use a front and back view for each of the anatomical sets of our humanoid. Based on the hierarchic structure of the humanoid, already used for animation purposes, we place two fixed cameras in local coordinates (front and back) for each anatomical set. During motion, the position of the cameras will be always the same with respect to the associated set and the occlusion problem can be solved. In our approach we consider 14 anatomical sets, we choose head, torso, and 3 segments for each arm and leg. This give us a maximum total amount of 28 different depth maps that will be processed two by two as we explain in what follows.

3.1. GPU collision detection

For each cloth particle we should check when there are interpenetration with the human body. Since it is divided in different parts, we associate at the beginning two orthographic camera, front (FC) and back (BC) cameras, to each part. The rotation and translation which are applied to the each part of the body at the present frame is also applied to the FC and BC, so we keep the point of view and handle each part independently to remove possible occlusions with other different body parts. Then, at each time step, to check the interpenetration of a cloth particle into the deformable body, we have to express the particle coordinates in the corresponding local coordinate system for each part of the body. We verify that the particle belongs to the inner part of the corresponding camera volume of viewing. Finally, we detect an interpene-

tration when the depth of the particle is less than the depth of the back and front part.

Figure 1: *Local cameras associated to different body parts. A close-up of the left arm*

To improve the performance we use only one texture RGBA to save the normal map and the depth map associated to each anatomic body set. Furthermore, the normal-depth texture contains both the front and back depth, since the fragment program accepts only 16 textures. Therefore, we use a frame buffer object to render it offscreen.

In principle, we verify each particle collisions with all parts of the human body. If the cloth is really in contact with many body parts, this approach is efficient. However, when only few body parts are in contact with the cloth, this is not suitable. In this case, we only select parts of the body such that the cloth has interpenetrated in theirs oriented bounding box (OBB). We remark that the volume of viewing of the camera associate to each body part fits in the OBB of its part. So when we display the cloth from FC or BC, if there are not interpenetrations between the cloth and the OBB of its associated part, the number of drawn samples are zero. To know the number of drawn samples we call the function `glGetQueryObjectuivARB` of `ARB_occlusion_query` extension. One can improve the performance of this method in following sense: it is not necessary render completely the cloth, we can maintain two list of index positions, one of them, the complete list, including all the positions, and the other, a reduced list, containing an eighth part. The reduction of the list depends on the dimension of the particle system. In our case, we have reduced the list until no gain in latency and no loss accuracy have been found.

3.2. Collision correction

When the front and back checks return penetration a collision correction is needed. Here we found a new problem, the normal direction to the body at the collision point has to be known in order to correct the particle position and avoid penetration. We follow [KLRS04] to solve problem using again the graphics hardware. The idea is to use an implicit model representation of the body skin, which consists essentially in storing together depth and body's normal information as it has been explained above. When we render the corresponding human subset from the point of view of the local cameras, first we have stored a normal for each vertex of the human body at the starting moment and during animation, we recompute every normal according to the corresponding motion matrix and skinning weight, see fig. 2.

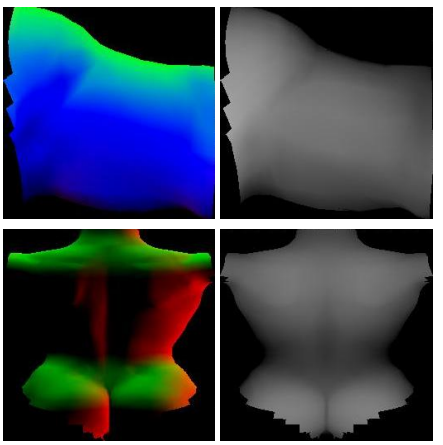

Figure 2: *Two Normal-Depth textures corresponding to Front-Arm view (up) and Back-Torso view (down).*

When a cloth particle p_c is detected inside an anatomical set (which can be though topologically as a cylinder), two points of the corresponding depth maps are associated to this particle, the front p_f and back p_b points. The amount of real penetration RP is computed from the scaled differences

$$RP_f = z_s(B_f - p_f)$$
and
$$RP_b = z_s(B_b - p_b), \quad (3)$$

respectively. $B_f - p_f$ measures depth difference between the body and the cloth point expressed in the front camera system (analogously for the back camera) and z_s is the scale parameter passing from camera distance to real world distance. These quantities are both positives in case of penetration and the stored normal at the closest depth point is used for correcting its position outside of the body.

If the distance between RP_f, RP_b is less than an advisable tolerance an average normal is taken. This, according to the shape of body segments and the small simulation time, can happens only in the boundaries of the depth maps.

4. Experiments and Results

The simulation example consists in a cloth falling on a walking woman, see figure 3. We have chosen this simulation because the cloth is not fixed to any part of the body and occlusions between different body parts are present, for instance, forearm and torso. The used woman mesh is composed of 17523 vertex, 17539 quadrangles and 2789 triangles and motion is read from a standard biovision format

	fixed iter(20/40)	quasi-feedback
64x64	105/86	88-124
128x128	63/43	44-90
256x256	22/14	14-42

Table 1: *Frame rate results for different cloth dimensions considering all body parts*

	fixed iter(20/40)	quasi-feedback
64x64	90-144/77-108	107-200
128x128	54-77/41-50	40-133
256x256	20-24/14-15	14-55

Table 2: *Frame rate results for different cloth dimensions considering only body parts which can collide.*

file. Skinning weights values were calculated only once according to the animation and stored in a skinning file. The maximum number of iterations in the relaxation algorithm is 40, but using the quasi-feedback method, at the beginning when there are a little deformation, only 2-5 iterations are needed.

On table 1 we present the results for different cloth dimensions. The values of the table give the number of frames per second for each render, including simulation. First column represents the simulation times with a fixed number of relaxation iterations, 20/40. And the second one gives a minimum and maximum frame rate when the quasi-feedback method is activated. On table 2 the same results are shown when the oriented bounding box are used as a previous filter to select the possible body parts for collision. In this case, first column show also the minimum and maximum frame rate achieved during animation. These results are obtained using a desktop computer with a Pentium IV 3.0 Ghz processor, and a nVidia Geforce 6800 GT graphics card.

Figure 3: *Two screenshots corresponding to the final simulation of the cloth falling on the walking woman.*

5. Conclusions and Future Work

We have presented a new method for simulating virtual cloth for real-time animations. Our method is running completely on the GPU using the last features of the present graphics cards. Collision correction bottleneck is overcome by means of image based depth-map methods. A high increase of the simulation speed is due to cut off the unnecessary computations using a new technic for getting a feedback from the GPU that allows interrupt an iterative process.

For cloth simulation we use structured meshes in our results, although we can also simulate a cloth defined by an unstructured mesh. In this case, a small decrease of the frame rate is appreciable due to an extra call to the connectivity texture. However, for unstructured meshes the internal deformation forces can not be simulated using only mass-spring connections if you want to avoid too large iterations, and therefore, loosing real-time performance. When a small number of iterations are used for general cloth meshes, an unrealistic elongation of the cloth is produced. We are working in other different models to solve this problem on the GPU. This will allow us to dress a virtual humanoid using any cloth generated previously with a modeler software. With our present model we are limited to garments formed only by rectangle meshes.

6. Acknowledgments

We would like to thank Prof. Renato Pajarola for his help at the beginning of this work at UCI. This work has been partially supported by CCEGP-03/04 and TIN2004-08065-C02-01.

References

[BHW94] BREEN D., HOUSE D., WOZNY M.: Predicting the drape of woven cloth using interacting particles. In *Computer Graphics Proceedings, Annual Conference Series* (1994), vol. 94, pp. 365–372.

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Computer Graphics Proceedings, Annual Conference Series* (1998), vol. 98, pp. 3–54.

[Gre03] GREEN S.: Stupid opengl shader tricks. In *Proc. Game Developers Conference'03* (2003).

[Jac01] JACOBSEN T.: Advanced character physics. In *Proc. Game Developers Conference'01* (2001), pp. 1–10.

[KLRS04] KOLB A., LATTA L., REZK-SALAMA C.: Hardware-based simulation and collision detection for large particle systems. In *Graphics Hardware'04, T. Akenine-Möller, M. McCool Ed.* (2004), pp. 1–9.

[Pro95] PROVOT X.: Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *In Proceedings of Graphics Interface* (1995), vol. 95, pp. 141–155.

[TF88] TERZOPOULOS D., FLEISCHER K.: Deformable models. *Visual Computer 4* (1988), 306–331.

[VSC01] VASSILEV T., SPANLANG B., CHRYSANTHOU Y.: Fast cloth animation on walking avatars. *Computer Graphics Forum 20*, 3 (2001), 1–8.