International Journal of Applied Engineering Research ISSN 0973-4562 Volume 13, Number 8 (2018) pp. 6036-6046

Quantitative Evaluation of Forces and Jacobians for Real-Time Garment Simulation with Semi-Implicit Euler Integration and Hardware Assisted Vector Calculation

Sungmin Kim[1], Chang Kyu Park[2], Woong-Ryeol Yu[3], In Hwan Sul[†]

[1]: Department of Textiles, Merchandising, and Fashion Design, Seoul National University, Seoul 08826, Republic of Korea.

[2]: Department of Textiles and Engineering, Col. of Engineering, Konkuk Unversity, Seoul 05029, Republic of Korea.

[3]: Department of Materials Science and Engineering, Seoul National University, 599 Gwanangno, Gwanak-gu, Seoul 08826, Republic of Korea.

† (corresponding author): Department of Materials Design Engineering, Kumoh National Institute of Technology, 61 Daehak-ro, Gumi, GyeongBuk 39177, Republic of Korea.

Abstract

Real-time cloth simulation is a key factor in 3D garment computer-aided-design (CAD) system. This paper tries to provide a quantitative analysis and optimal selection of forces and Jacobian terms for particle-based garment simulation. Particle based method of Baraff and Witkin (1998) was optimized using the selected force and Jacobian terms. And sub-modules comprising the whole system, including collision detection/response step, linear matrix system solving step, force/Jacobian calculation step, and rendering, were analyzed in detail to find the time bottle neck in realizing real-time simulation. Hardware assisted acceleration techniques, like multi-threading, SSE (Streaming SIMD Extensions), and openMP (Open Multi-Processing) as well as data vectorization, were also tested to find the possibility of real-time cloth simulation in cloud computing environment.

Keywords: Real-time cloth simulation, 3D garment CAD, particle based method, hardware assistance, cloud computing

INTRODUCTION

There has been a massive interest in modeling and simulation of cloth or garment deformation in various fields of engineering. But each academia had different point of view. Mechanical engineering needs high precision of material deformation, while computer graphics engineers are interested in visual reality and computational efficiency. Especially for 3D garment CAD system development, real-time simulation of garment is critical factor, because garments are design by many trial and errors of pattern modification. Our first concern in this paper is to find the optimal combination of forces and their derivatives (Jacobian) to find computationally efficient linear system from various literatures. Another concern is to use parallel computing feature of the current hardware. It is well known that general purpose graphics processor unit (GPGPU) (Mogal and Barde, 2015) and cloud computing (Hashem *et al.*, 2015) are the next generation computing device to succeed conventional CPU. Our final goal is to make a cloud computing based real-time garment simulation platform, which can be used to realize big-data based on-line personalized garment recommendation system (Kim and LaBat, 2013). To make the proposed system, the garment simulation algorithm should be analyzed and optimized to fit the parallel computing environment. Chapter 2 introduces various papers concerned with this topic. The forces, along with the proposed optimal real-time system, is described in Chapter 3. Chapter 4 delivers the results and discussion about the real-time system and related issues. Finally, conclusion will be given in Chapter 5.

THEORY AND LITERATURE REVIEW

There have been enormous works on cloth simulation and a several good review papers have already been published. For example, please refer to (Breen *et al.*, 1994), (Gibson and Mirtich, 1997) , (Witkin, 1997), (House and Breen, 2000), (Nealen *et al.*, 2006) for physics (particle) based cloth modeling or deformable models, (Hu and Teng, 1996) for F.E.M. based cloth simulation, (Teschner *et al.*, 2005) for collision detection among deformable models. The following literature summarizes several selected methods used in our implementation, which were especially suitable for vectorized computation.

GARMENT SIMULATION ALGORITHM

Governing equation for PBM :

Garment is three-dimensional structure, made of two-dimensional flat fabrics using stitching process. So the modeling of fabrics' deformation is briefly discussed first. Throughout the paper, 'cloth', 'fabrics' or 'textile' means the 2D flat raw materials before sewing, and 'garment' means the final product after sewing.

Fabrics are manufacturing by weaving or knitting process. In either way, the fabrics are actually 3D entangled structure of

fibers. As the modeling in fiber scale is practically infeasible, the fabrics are assumed to be a sheet-like homogenous material in general. The conventional method, continuum mechanics with finite difference method (Terzopoulos *et al.*, 1987) or finite element method (O'brien and Hodgins, 1999;Yu *et al.*, 2000) can also be used to modeling of simple fabric deformation. Continuum models can give accurate prediction for various material deformation, except for cloth. The cons of continuum mechanics approach is that it is not suitable for real-time application, as it needs time-consuming re-calculation of mass and stiffness in each step when the material has big deformation like cloth (Gibson and Mirtich, 1997). That is why particle based modeling is more preferred modeling methodology for cloth and garment these days.

Particle based method gives a final linear system which can be numerically easily solved, instead of sacrificing a little exactness in material property. As the simulation is dynamic problem, selection of integration scheme is important to maintain the precision of calculation result. Among various methods, (Baraff and Witkin, 1998)'s semi-implicit integration scheme (equation (1)) is the compromised version between calculation.

$$(\mathbf{M} - h\frac{\partial \mathbf{f}}{\partial \mathbf{v}} - h^2\frac{\partial \mathbf{f}}{\partial \mathbf{p}})\Delta \mathbf{v} = h\left(\mathbf{f}_0 + h\frac{\partial \mathbf{f}}{\partial \mathbf{p}}\mathbf{v}_0\right) \quad (1)$$

, where M is mass matrix of *m*×*m* size with 3×3 sub matrices (m = number of vertices). *h* is the time step, $\mathbf{f}_0$ and v_0 are m x 3 force and velocity vector of current step. To construct the system, the derivatives of forces, also known as Jacobian terms, such as $\frac{\partial \mathbf{f}}{\partial \mathbf{v}}$ and $\frac{\partial \mathbf{f}}{\partial \mathbf{p}}$ should be also derived and accumulated in the left-hand side global matrix just as in the stiffness matrix of F.E.M. Solving equation (1) gives positions of the mesh vertices in the next time step.

Now we get the final linear system from eq. (1) after force and Jacobians for vertices are calculated and collected into the matrix. The left-hand-side of eq. (1) is sparse matrix which is advantageous to use iterative matrix solving strategy like modified preconditioned conjugate gradient (MPCG) method (Baraff and Witkin, 1998), (Gibson and Mirtich, 1997). The selection of forces and Jacobians are critical. If there is any negative term in the forces or Jacobins, the matrix becomes ill-conditioned system and MPCG method cannot converge well (Shewchuk, 1994;Ascher and Boxerman, 2003).

Optimal selection of forces :

Force vector of each vertex *i* is a sum of many force terms such as tension, bending force, shear force, sewing force and so on (equation (2)).

$$\mathbf{f}_{0,i} = \mathbf{f}_{Tension,i} + \mathbf{f}_{Bending,i} + \mathbf{f}_{Shear,i} + \mathbf{f}_{Sewing,i} + \mathbf{f}_{Friction,i} + \mathbf{f}_{Damping,i} + \ldots \quad (2)$$

The simple example for such tensile force and Jacobian between two vertices **x***i* and **x***j* is like equation (2) and (3) (Kang *et al.*, 2000;Nealen *et al.*, 2006).

$$\mathbf{f}^T = k_T(\|\mathbf{x}_{ij}\| - l_0)\frac{\mathbf{x}_{ij}}{\|\mathbf{x}_{ij}\|} \quad (3)$$

$$\frac{\partial \mathbf{f}^T}{\partial \mathbf{x}_j} = k_T\frac{(\|\mathbf{x}_{ij}\| - l_0)}{\|\mathbf{x}_{ij}\|}\mathbf{I}_{33} + k_T\frac{l_0}{\|\mathbf{x}_{ij}\|}\mathbf{I}_{33}\frac{\mathbf{x}_{ij}\cdot\mathbf{x}_{ij}^T}{\|\mathbf{x}_{ij}\|^2}, \frac{\partial \mathbf{f}^T}{\partial \mathbf{v}_j} = 0 \quad (4)$$

This form is very easy to implement, while there is no term to express warp and weft material property. (Baraff and Witkin, 1998) devised a new mapping function to solve this problem.

$$(\mathbf{w}_u\ \mathbf{w}_v) = (\Delta\mathbf{x}_1\ \Delta\mathbf{x}_2)\begin{pmatrix}\Delta u_1 & \Delta u_2\\ \Delta v_1 & \Delta v_2\end{pmatrix}^{-1} \quad (5)$$

, where ($\mathbf{x_i}$,$\mathbf{x_j}$,$\mathbf{x_k}$) is the vertices of the fabric mesh vertices in 3D Cartesian coordinates, and $\Delta u_1, \Delta u_2, \Delta v_1, \Delta v_2$ are the 2D projected coordinates of ($\mathbf{x_i}$,$\mathbf{x_j}$,$\mathbf{x_k}$) on imaginary plane. They expressed tensile force from state function $C^T(x)$ like the following;

$$\mathbf{C}^T(\mathbf{x}) = a\begin{pmatrix}\|w_u(\mathbf{x})\| - 1\\ \|w_v(\mathbf{x})\| - 1\end{pmatrix} \quad (6)$$

and the tension force and its Jacobian becomes

$$\mathrm{f}_i^T = -\frac{\partial E_i^T}{\partial \mathbf{x}_i} = -k\frac{\partial \mathbf{C}_i^T(\mathbf{x})}{\partial \mathbf{x}_i}\mathbf{C}_i^T(\mathbf{x}) \quad (7)$$

$$\frac{\partial \mathrm{f}_i}{\partial \mathbf{x}_j} = -k\left(\frac{\partial \mathrm{C}(\mathbf{x})}{\partial \mathbf{x}_i}\frac{\partial \mathrm{C}(\mathbf{x})}{\partial \mathbf{x}_j}^T + \frac{\partial^2 \mathrm{C}(\mathbf{x})}{\partial \mathbf{x}_i \partial \mathbf{x}_j}\mathrm{C}(\mathbf{x})\right) \quad (8)$$

The advantage of (Baraff and Witkin, 1998)'s mapping function is that shear force can be also simply derived in a similar way.

$$\mathbf{C}_i^{Sh}(\mathbf{x}) = a w_u(\mathbf{x})^T w_v(\mathbf{x}) \quad (9)$$

For bending, the following state function was used.

$$\mathbf{C}_i^B = \theta \quad (10)$$

But derivation of equation (1) leads into very complex formula, even third order tensors. Principle component analysis was used to make a simpler equation for bending force, by minimizing the degree of freedom of for each vertex (Bridson *et al.*, 2003) (Pentland and Williams, 1989).

$$\mathbf{f}^B = k_B\frac{\|E_B\|^2}{\|n_1\| + \|n_2\|}\sin(\theta/2)u_i \quad (11)$$

, where u_i means the displacement vectors of two adjacent triangle vertices, and E_B the bending energy.

Collision detection :

Collision detection is done also on the basis of triangles. Using computational geometry operators, collision between triangles can be checked exactly (Provot, 1997;Lin and Otaduy, 2005;Teschner *et al.*, 2005). But this exact calculation needs a considerable time, so it is better to omit the unnecessary checking. The easiest method to use data structure like voxels, bounding boxes and hierarchical structure together (Klosowski *et al.*, 1998;Zhang and Yuen, 2000). Voxels or bounding boxes are series of stacked boxes which has same or different size, and it is very easy to identify the box ID from the collision detection target's coordinates. If the two triangles reside in different boxes, their collision checking can be omitted. The cons of using bounding box alone is that it needs a memory space as large as the number of voxels. This problem can be solved by using hierarchical structure of bounding boxes. If some number of small boxes make a group repetitively like a tree structure, memory usage can be saved for unnecessary boxes. K-DOP with binary tree structure was used to skip unnecessary collision checking in this paper.

IMPLEMENTATION/PERFORMANCE ISSUES

Linear system solving :

The equation (1) leads to a sparse *3m×3m* size matrix. It would be better to use exact methods like Gauss elimination or LU decomposition(Baraff and Witkin, 1998), if the number *m* is small (roughly not more than 100). But the exact method needs many operations, which cannot be achieved real-time calculation, even if the matrix is sparse. The alternative is to use iterative methods like conjugate gradient method (Ascher and Boxerman, 2003), which sacrifices exactness for numerical efficiency. The PBM is based on vibration of particles and such deficiency of exactness in matrix operation is not critical to the final simulation quality (Nealen *et al.*, 2006).

Mesh generation :

PBM sacrifices a little bit of exactness in material property to get numerical stability. Meanwhile PBM has another advantage to the conventional continuum mechanics and F.E.M. based approach, that it is not sensitive to the mesh element shapes. For example, triangles of F.E.M. have inner angles bigger than 25 degree []. Such a constraint can be a hurdle in complex geometries like garment patterns. PBM is more flexible in this feature. The element shape does not affect the simulation quality at all. Generally triangles or rectangles are favored element shape (Shewchuk, 1997).

If the density of mesh elements are varied, it can upgrade computational efficiency or simulation quality. Multi-grid is a good example for the former case. Applying geometric or algebraic multi-grid method to garment simulation makes the system about 2 times faster (Lee *et al.*, 2010), like the case of conventional 3D CAD system , Clo3D® . Otherwise, the change of mesh configuration can make more realistic simulation result. Narain et al. (Narain *et al.*, 2012) used adaptive re-meshing and Koh et al. (Koh *et al.*, 2015) used view-dependeing adaptive mesh for buckling compensation. The idea is based on that fabric buckling does not always match with the mesh element configuration. But the pro is that the simulation speed is not real-time, because of the re-meshing operation. For example, Wang et al. (Wang *et al.*, 2011) 's data-driven elastic models for cloth used time step of 0.1 ~ 0.07 msec per frame and each frame needed 20~30 seconds of calculation. Volino et al.(Volino *et al.*, 2009) recently succeeded in implementing accurate nonlinear tensile stiffness, yet the speed is 15% slower than the conventional PBS.

Rendering :

Rendering of 3D objects needs a little portion of the whole simulation step, even if the current GPUs' have enormous operation speed. OpenGL® (Shreiner and Group, 2009) is the famous tool kit for 3D rendering in computer graphics along with Microsoft DirectX®. Graphic card (often called as GPU) can be utilized for general purpose linear system operation as shown in the next section, as well as 3D rendering. 3D mesh objects can be saved not only in 3D file CAD format, but also in 2D graphic formats. Moon et al. reported a 2D vector based file format for 3D objects, which can replace the conventional raster graphic file capture format (Moon, 2016).

Hardware assisted calculation :

Most of the engineering problems result into a sparse linear system. Real-time simulation can be defined as a system which can solve such linear system more than 10 times (frames) per each second. But the sparse system itself is not proper for fast matrix operation. For example, our data such as forces or final matrix have scattered location in computer memory. If they can be re-arranged in a single row, theoretical calculation speed can be achieved via the CPU's various vector computing features like multi-threading, SSE and OpenMP. General purpose graphic card processing unit (GPGPU) calculation was also applied to cloth simulation by many groups (Sul, 2009;Vassilev, 2010), but it is beyond our concern now because it needs different algorithm because of graphic card's inherent rectangular texture based data structure.

SSE was feature adopted from Pentium 3 CPU. The SSE feature can do a multiple number of floating point calculations simultaneously using a side bandwidth of CPU (Raman *et al.*, 2000;Satish *et al.*, 2010). To use this feature, data should be prepared in series like the AoS format.

OpenMP is a parallel computing technique for 'loop' codes (Dagum and Menon, 1998;Lario *et al.*, 2001;Mujahid *et al.*, 2004;Gutiérrez *et al.*, 2005). Using OpenMP preheaders in 'for loops' of C/C++ language makes the calculation split into multiple CPU cores. It is similar with multi-thread in that multiple number of cords work together. But the target of parallelization is different. OpenMP uses multi-cores partially

for 'loops' in a source code, while multi-thread strategy loads the whole source code into a specific CPU core.

(Mujahid *et al.*, 2004) applied OpenMP parallel computing and adaptive meshing to table draping simulation, but they use very simple tensile spring force only. (Gutiérrez *et al.*, 2005) considered all the force terms and used implicit integration scheme, but Jacobian terms for bending were not explicitly used. This paper applied hardware assistance techniques for general force, Jacobian terms and collision detection of PBM (eq. (2)~(11)), and analyzed the effect quantitatively.

EXPERIMENTAL

PROCEDURE

Garment simulation system needs a several data structure and algorithms. The typical procedure is shown in Figure 1, which is authors' own implementation of particle based approach for garments. The procedure is composed of two main steps, initialization (chapter 3.1.1) and iteration (3.1.2). In Figure 1, dashed line shows a process done only once in initialization step, while solid lines means a process which is repeated in each time step. The source codes were written using C++ compiler (Embacadero C++ Builder 2007) in IBM-PC with Intel CPU (i5-4670, 3.4GHz). The vectorization schemes used in this paper, SSE and OpenMP, are optimized for Intel CPU, but not limited to. CPU's of other vendors supply use similar functionality like former AMD Core Math Library(AMD, 2012).

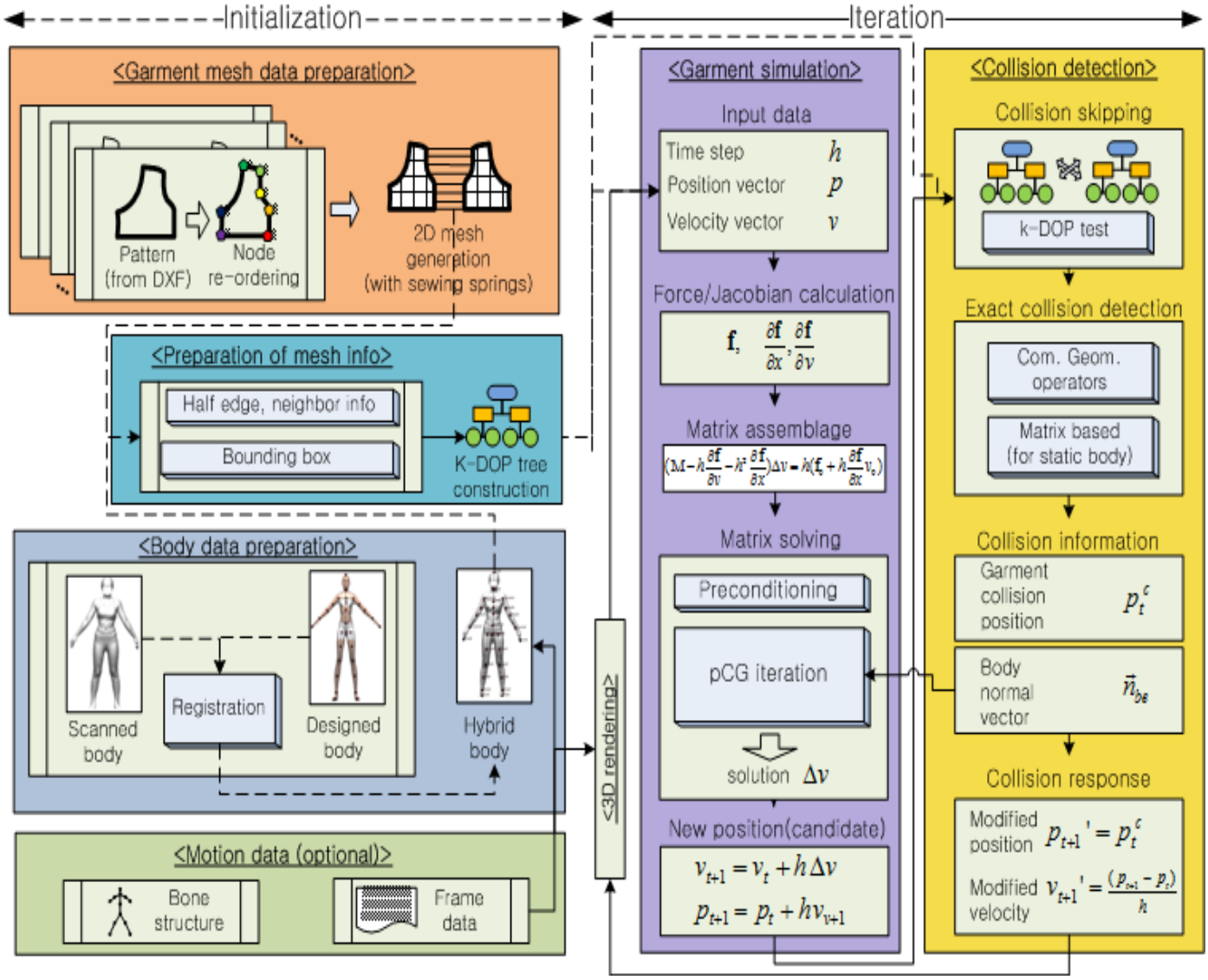


Figure 1. Flowchart of the 3D garment simulation system (garment simulation procedure was borrowed from Baraff and Witkin (1998))

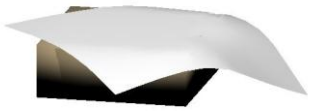
Initialization step :

Garment pattern mesh data

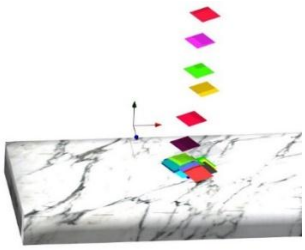
Garment pattern sets with various topology and collision situation (Figure 2) were prepared in DXF file format (Sul, 2010;Sul and Kang, 2010). The boundary nodes were re-ordered clockwise or counter-clockwise depending on their relative position to the bodice. This re-ordering makes it easier to assign sewing lines in mesh generation step. (Shewchuk, 1997)'s open source code was used for triangular element mesh generation. The mesh was regenerated automatically when the mesh density or sewing line assignment was changed. Table I shows the mesh information of the garment patterns. After mesh generation, half edge structure and neighbor element information was retrieved for tension and bending force calculation.

Table I. Specification of garment mesh data

Garment set	# of patterns	# of vertices	# of elements
Stiff table cloth	1	560	1054
Falling handkerchiefs	23	3427	5014
Pleated skirts	2	1604	2966
Shirts on pants	14	3816	6307



(a) Stiff table cloth



(b) Falling handkerchiefs

(c) Pleated skirts

(d) Shirts on pants

Figure 2. Garment sets and body data used in the test

Body data preparation

A hybrid body mesh data of scanned body and designed body was used for garment simulation. The scanned body mesh was acquired using 3D body scanner (Hamamatsu Photonics, C9036-02) and the number of mesh elements were reduced for save storage space. Body textures and motion skin weight of designed by from Maya software using mesh registration (Allen *et al.*, 2003;Sul and Kang, 2010). Body feature points and bone points were detected automatically and used for reference points for the mesh registration. Half edge data structure and neighbor information of mesh were also constructed for collision detection using bounding box and k-DOP trees (Klosowski *et al.*, 1998).

Iteration step :

Garment simulation

(Baraff and Witkin, 1998)'s original semi-integration scheme with (Ascher and Boxerman, 2003)'s MPCG method was used for constructing and solving the governing equation. Collision detection information was used for giving plane constraint in the MPCG iteration, where collided garment vertices are bounded to move only on the tangent plane with the body. The final garment positions after matrix solving ($\mathbf{p}_{t+1}$) were modified using the collision response ($\mathbf{p}_{t+1}$') to avoid garment-to-body penetration artifact.

Collision detection / response

Voxels were prepared for body mesh data to find collision candidates easily. K-DOP binary tree was used garment data, because there can be self-collisions in garment itself. Redundant collisions were skipped using both voxels and k-DOP trees (Sul, 2010). Final collision information was sent to MPCG iteration step for plane constraints and modification of the final garment vertex coordinates.

3D Rendering

OpenGL 2.0 shading language was used for displaying the results via graphic card (nVidia Geforce GTX 760, screen resolution at 1920 x 1080). Rendering conditions were controlled using vertex and fragment shaders of text file format. Additional CPU thread was assigned for fast rendering.

SPEED OPTIMIZATION AND PERFORMANCE EVALUATION

Calculation optimization using algorithm and hardware assistance :

Multi-thread (multi-core) calculation

Source codes of each calculation step (shown in boxes with different background colors) in Figure 1 were encapsulated by an independent thread. With this feature, each step could run in different CPU cores in parallel.

SoA (structure of array) data structure

All the data were arranged to SoA format to make the codes vector-computing-friendly. Table II shows an example for AoS and SoA source codes and their alignment in memory (Héman *et al.*, 2007). And then vector computing including multithreading, SSE and OpenMP were applied.

Table II. Comparison of AoS and SoA data structure

Type	**AoS**	**SoA**

Source example	#define XYZ 3
	
	struct Garment_vertex
	{
	double position[XYZ];
	};
	
	Garment_vertex v[1000];
		#define XYZ 3
		
		struct Garment_positions
		{
		double value[1000];
		};
		
		Garment_positions p[XYZ];

memory layout example	$x_0, y_0, z_0, x_1, y_1, z_1, \ldots, x_{999}, y_{999}, z_{999}$	$x_0, x_1, \ldots x_{999}, y_0, y_1, \ldots y_{999}, z_0, z_1, \ldots z_{999}$

Evaluation of simulation performance :

The simulation steps are not perfectly independent. So the simulation time for each step was measured by turning off each part selectively (Figure 3). To use multi-thread computation feature, of openMP, some iteration parts were ported to Microsoft Visual C++ 2005 dynamic link library (DLL).

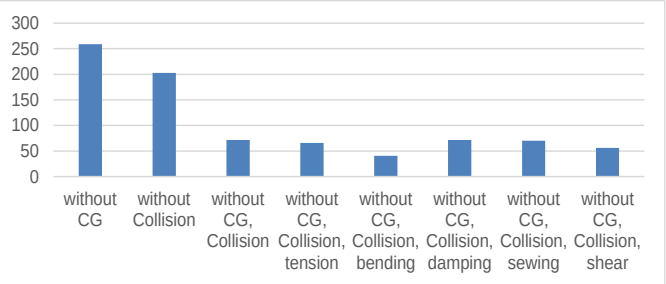


Figure 3. Example of simulation performance evaluation ("Shirts and pants" case, unit: millisecond/frame)

RESULTS AND DISCUSSION

The whole procedure was analyzed step by step first, and then the effect of hardware assistance was verified.

STEPWISE ALGORITHM PERFORMANCE EVALUATION

Overall procedure :

Figure 4 shows an example of calculation time analysis result for iteration steps of Figure 1. Measurement of initialization time was omitted because it did not affect the whole simulation. The calculation time of each frame is dependent on many factors such as mesh density, degree of collision detection precision, number of MPCG iterations, and even rendering. Therefore it is not easy to generalize, but roughly the time consuming steps are in the order of collision detection, MPCG iteration (matrix solving), force and Jacobian calculation, and other steps including rendering. Times for collision detection and force calculation steps are mainly determined by number of mesh vertices and there are not much room for improvement. It is the MPCG step that can be optimized in speed. Also proper choice of force and Jacobian are important, because negative terms can damage the numerical convergence, which needs a very large number of MPCG iteration. There is no reference yet for definition of 'real-time', but generally speed of 30 frames per second for 1,000 mesh vertices are the speed from (Baraff and Witkin, 1998)'s method after several optimization. For comparison, please refer to the open source code ("FreeCloth") which is intuitive implementation of Baraff and Witkin's method without further optimization (Pritchard, 2002).

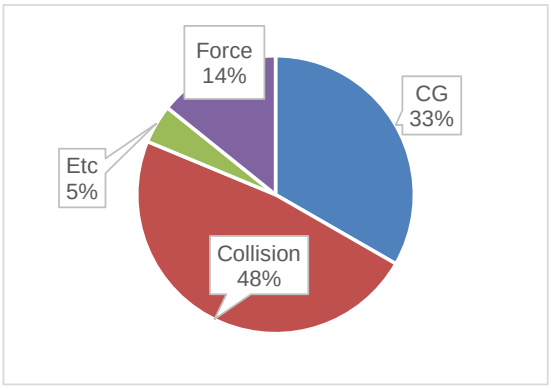


Figure 4. Analysis of calculation speed in garment simulation ("Shirts on pants" case)

Force and Jacobian calculation step :

Figure 5 shows the detailed view of components for force and Jacobian calculation step, using (Baraff and Witkin, 1998)'s force terms (equation (5)-(10)). Definitely most of the time was needed for bending force terms, because of the complex derivatives of the bending angle theta with respect to mesh vertices.

As an alternative, simple tension spring such as equation (3) and (4) can replace the tension, shear and even bending force, which can be reduce the time even by an order. But in this case, the anisotropic material property cannot be easily expressed with a simple spring. Also bending rigidity cannot be expressed freely. A simple table cloth or flag with a small bending rigidity can be covered, but the spring based bending force cannot express a very rigid cloth, because there is no term to distinguish both sides with only bending springs.

Another solution was to use eq. (11), which is an approximation of actual bending force, but generally it worked well even if the time steps are not excessively well. But the time step used is generally 33ms at most, so eq. (11) is a good choice for both quality and speed. The other problem of (Baraff and Witkin, 1998)'s bending force term from eq. (10) is that it is very tricky to calculate. As the differentiations lead to even three order tensor, the numerical round of errors are accumulated and sometimes the MPCG iterations do not converge for a stiff cloth.

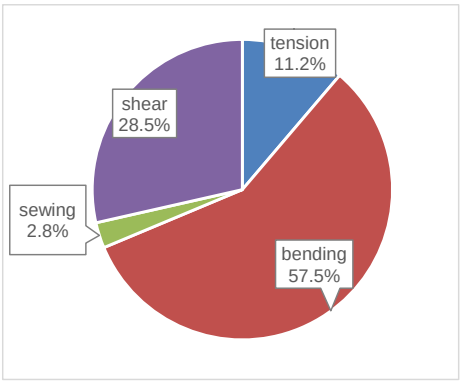


Figure 5. Detailed view of "Force" calculation step in Figure 4

Matrix solving step

Figure 6 shows time measured for MPCG steps. Issues in MPCG steps are;

- Minimizing the number of iteration until convergence, and
- Making the data vector-friendly format.

The latter cases will be shown in the following chapter (3.2). As long as the forces and Jacobians are suitably merged in the matrix, each iteration gives closer solution to the exact one (Shewchuk, 1994;Ascher and Boxerman, 2003). Performance is related with the terminal condition in which the iteration should stop. The iteration can be stopped;

- If the number of loops have reached a predefined value, or
- If error norm between the current and previous solution vector lies under a predefined value.

For better results, the latter case would be a good choice. When speed is more important factor than exactness, the former case would be the better option. Our experiments showed that the number of maximum loops under 10 gave a reliable result in most of the cases. But the size of error norm can be erroneous when there is big external forces or other constraints, therefore using both criteria was the safest method.

Iterative matrix solving using MPCG has another advantage than fast calculation. The Jacobian terms need not be used in matrix form in MPCG (Hauth, 2003). This fact has two effects. First, each rows of matrix can be dealt in a vectorized form, which can make use of the vector-computing power of CPU. Second, some Jacobians can be omitted, because the number of Jacobians in each row equals to zero (Kang and Cho, 2002). The second effect minimizes not only size of calculation, but also possible round of errors from redundant Jacobian calculation. The effect of hardware vectorization would be shown in chapter 4.2 again.

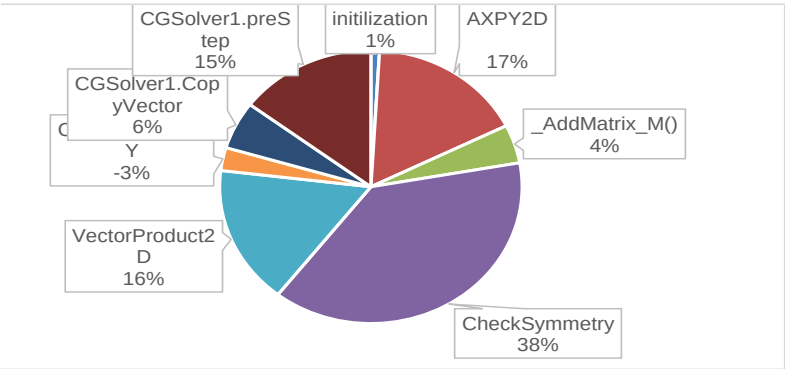


Figure 6. Detailed view of "CG" step in Figure 4

Collision detection step :

There are two kinds of collisions in garment simulation, i.e. Garment-to-body collision and garment-to-garment (self) collision. Under static body simulation, which is the common

case in 3D CAD system, the data structure for body can be prepared only at the initialization stage. (Volino and Magnenat-Thalmann, 2012) made a matrix-based simplified collision detection algorithm using this fact for static body garment simulation. But the other object, i.e. garment, should be updated in each frame. So the update time was the bottleneck in collision detection step as shown in Figure 7. Most of unnecessary collision candidate mesh element pairs can be skipped by using bounding boxes and hierarchical data structure. But actual collision detections are inevitable in each time, as the garment and body are always in touch with each other. Thus the time for collision detection cannot be optimized dramatically in garment CAD system. By the way, the collision response can help MPCG steps converge more easily. One of (Baraff and Witkin, 1998)'s pioneering works is that the MPCG step is given constraint. One example is plane constraint, i.e. when a garment mesh vertex is on the surface of body mesh element, its movement is confined only on the tangent plane. So each collision decreases the number of degree of freedom by one. The more collisions occur, the MPCG iteration becomes easier to converge.

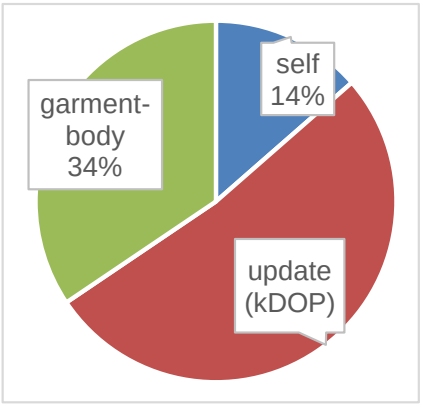


Figure 7. Detailed view of "Collision" detection step in Figure 4

HARDWARE ASSISTANCE EFFECT

Figure 8 is the result of multi-thread calculation for 'shirt' data with two different mesh density. The acceleration effect was more vivid for more number of vertices. As the figure shows, number of thread did not show linear relationship with speed improvement. The reason would be that threads could not proceed perfectly independently, as they shared same garment data simultaneously. The effect of multi-core strategy is advantageous for continuous and vectorizable data.

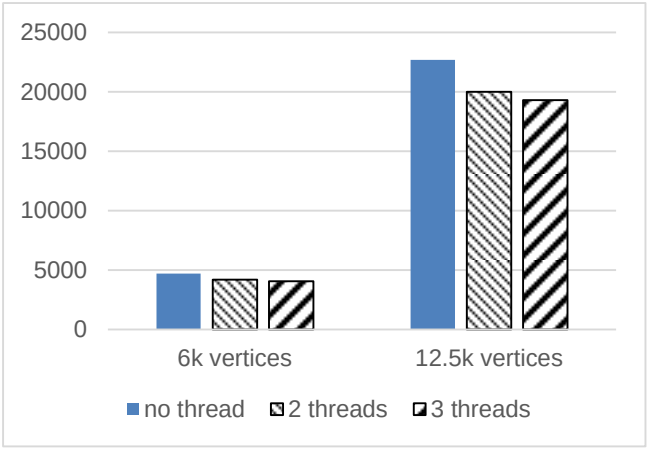


Figure 8. Effect of multi-thread calculation for different mesh resolution ("Shirts on pants" case, unit: millisecond)

The fact is more evident in Figure 9, which shows effect of multi-thread for MPCG and collision detection steps. Matrix for MPCG step cannot easily be split, so the threads did not help. Meanwhile use of threads gave a significant improvement for k-DOP collision detection step, as collision detection data can be easily partitioned into different threads.

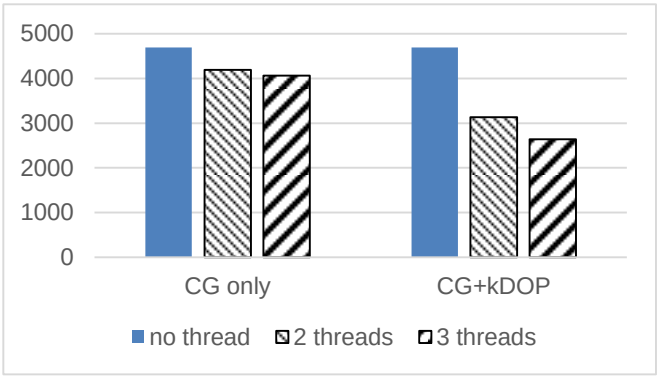


Figure 9. Effect of multi-thread in each calculation step ("Falling handkerchiefs" case, unit: millisecond/frame)

Figure 10 is effect of AoS and SoA data structure in tension (eq. (3)) and bending force (eq. (11)) calculation for 'tablecloth'. The current C++ compilers try so many optimization that only making the data in SoA format could reduce calculation time by six times. Adapting SSE feature gave a more advanced result. The time was reduced by about three times again. Memory prefetching was also tried, but it gave fluctuating result, only 10% improvement at most. It seems that the compiler already used SSE feature in its optimization process.

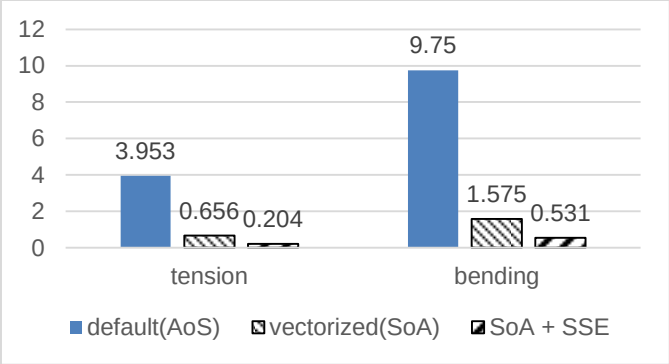


Figure 10. Effect of data structure change and SSE usage in tension and bending force calculation ("Stiff table cloth" case, unit: millisecond/frame)

Figure 11 shows the comparison between the build configuration of compiler, such as debug build or release build in SpMV (Sparse matrix and vector multiplication) operation of 17k mesh vertices. As the debug build contains many redundant source for debugging, it showed less optimized result. Effect of OpenMP in MPCG step was also tested. Because the garment simulation is composed of many 'loop' operations, the use of MPCG gave about 5% improvement in release build. The SoA and AoS data type did not give significant difference in optimization using OpenMP.

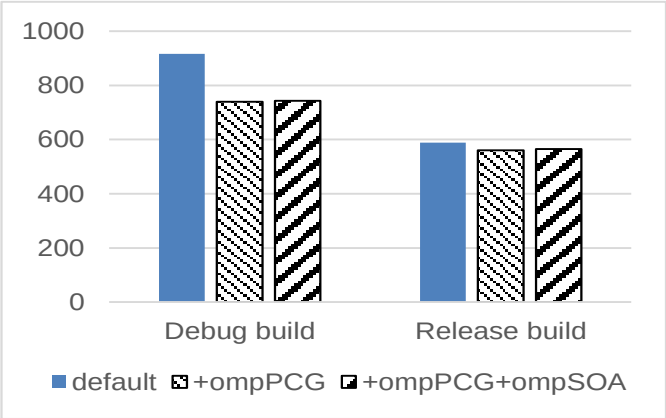


Figure 11. Effect of C++ compiler build configuration and openMP usage ("Pleated skirt" case, unit: millisecond/frame)

CONCLUSION

Modeling of fabric deformation is an interesting theme in many engineering fields. Especially for 3D garment CAD system development, fast calculation speed as well as exact expression of cloth material property are the primary concern. This paper chose the most efficient cloth modeling method, particle based simulation with semi-implicit integration scheme and reviewed some key references needed for real-time garment simulation system. Simulation with no optimization showed unpractical performance. Naive implementation of the famous (Baraff and Witkin, 1998)'s semi-implicit Euler integration method showed simulation speed of less than 1 fps for 1,000 mesh vertices. The authors analyzed the simulation steps and categorized issues to consider for simulation speed optimization.

Algorithmic issues include;

- Use of (Baraff and Witkin, 1998)'s bending formula needs complex differentiation and may lead to round off errors.
- Instead, (Bridson *et al.*, 2003)'s principal component analysis based bending term (eq.(11)) was more satisfactory both in speed and simulation quality.
- In MPCG iteration, using fixed number of iteration was more advantageous in calculation time.
- In collision detection, updating k-DOP structure took the most of time.
- Using (Baraff and Witkin, 1998)'s plane constraint decreased the degree of freedom and made the linear system converge easier.
- Hardware issues include;
- Using multi-thread helped only when the data could be partitioned, such as collision detection step.
- Aligning the data as SoA format decreased the time by about six times.
- Also SoA format was beneficial for adopting vectorized computing features such as SSE or openMP, although their simultaneous use did not overlap.

Hardware assisted optimization showed obvious speed enhancement, but there seems to be no much room for further improvement. The better way to enhance the numerical convergence would be using algorithmic alteration. Therefore it is our further work to find the better forces and Jacobians that is positive definite for fast convergence and suitable for vector computing.

ACKNOWLEDGEMENT

This paper was supported by a research fund, Kumoh National Institute of Technology (2015-104-166).

REFERENCES

[1] Allen, B., Curless, B. and Popović, Z. (2003), "The space of human body shapes: reconstruction and parameterization from range scans", in *ACM transactions on graphics (TOG)*, *New York*, ACM, pp. 587-594.

[2] Amd, A. (2012), "Core Math Library (ACML)", *URL http://developer.amd.com/tools-and-sdks/archive/compute/amd-core-math-library-acml/acml-product-features/*, Vol. No., pp. 25.

[3] Ascher, U. M. and Boxerman, E. (2003), "On the modified conjugate gradient method in cloth simulation", *The Visual Computer,* Vol. 19 No. 7, pp. 526-531.

[4] Baraff, D. and Witkin, A. (1998), "Large steps in cloth simulation", in *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, pp. 43-54.

[5] Breen, D. E., House, D. H. and Wozny, M. J. (1994), "A particle-based model for simulating the draping behavior of woven cloth", *Textile Research Journal,* Vol. 64 No. 11, pp. 663-685.

[6] Bridson, R., Marino, S. and Fedkiw, R. (2003), "Simulation of clothing with folds and wrinkles", in *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, Eurographics Association, pp. 28-36.

[7] Dagum, L. and Menon, R. (1998), "OpenMP: an industry standard API for shared-memory programming", *IEEE computational science and engineering,* Vol. 5 No. 1, pp. 46-55.

[8] Gibson, S. F. and Mirtich, B. 1997. A survey of deformable modeling in computer graphics. Citeseer.

[9] Gutiérrez, E., Romero, S., Romero, L. F., Plata, O. and Zapata, E. L. (2005), "Parallel techniques in irregular codes: cloth simulation as case of study", *Journal of Parallel and Distributed Computing,* Vol. 65 No. 4, pp. 424-436.

[10] Héman, S., Nes, N., Zukowski, M. and Boncz, P. (2007), "Vectorized data processing on the cell broadband engine", in *Proceedings of the 3rd international workshop on Data management on new hardware*, ACM, pp. 4.

[11] Hashem, I. a. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A. and Khan, S. U. (2015), "The rise of "big data" on cloud computing: Review and open research issues", *Information Systems,* Vol. 47 No., pp. 98-115.

[12] Hauth, M. 2003. Numerical techniques for cloth simulation. SIGGRAPH 2003 Course #29: Clothing Simulation and Animation: October.

[13] House, D. H. and Breen, D. E. (2000), *Cloth modeling and animation,* AK Peters, Ltd., Natrick, MA.

[14] Hu, J. and Teng, J. (1996), "Computational fabric mechanics: Present status and future trends", *Finite Elements in Analysis and Design,* Vol. 21 No. 4, pp. 225-237.

[15] Kang, Y.-M. and Cho, H.-G. (2002), "Complex deformable objects in virtual reality", in *Proceedings of the ACM symposium on Virtual reality software and technology*, ACM, pp. 49-56.

[16] Kang, Y.-M., Choi, J.-H. and Cho, H.-G. (2000), "Real-time animation technique for flexible and thin objects", Vol. No.

[17] Kim, D.-E. and Labat, K. (2013), "Consumer experience in using 3D virtual garment simulation technology", *Journal of the Textile Institute,* Vol. 104 No. 8, pp. 819-829.

[18] Klosowski, J. T., Held, M., Mitchell, J. S., Sowizral, H. and Zikan, K. (1998), "Efficient collision detection using bounding volume hierarchies of k-DOPs", *IEEE transactions on Visualization and Computer Graphics,* Vol. 4 No. 1, pp. 21-36.

[19] Koh, W., Narain, R. and O'brien, J. F. (2015), "View-dependent adaptive cloth simulation with buckling compensation", *IEEE transactions on visualization and computer graphics,* Vol. 21 No. 10, pp. 1138-1145.

[20] Lario, R., Garcia, C., Prieto, M. and Tirado, F. (2001), "Rapid parallelization of a multilevel cloth simulator using OpenMP", in *Third European Workshop on OpenMP*, pp. 40.

[21] Lee, Y., Yoon, S. E., Oh, S., Kim, D. and Choi, S. (2010), "Multi-Resolution Cloth Simulation", in *Computer Graphics Forum*, Wiley Online Library, pp. 2225-2232.

[22] Lin, M. C. and Otaduy, M. (2005), "Recent advances in haptic rendering & applications", in *International Conference on Computer Graphics and Interactive Techniques: ACM SIGGRAPH 2005 Courses: Los Angeles, California*, pp.

[23] Mogal, C. and Barde, C. 2015. Review Paper on Optimized and accelerated Parallel Graph Algorithm on GPGPU. IJCSMC.

[24] Moon, S. H. K., Sungmin; Sul, in Hwan (2016), "2D vectorized storage technique of 3D digital garment data using depth-buffer test and pseudo-Bernstein polynomial coefficient correction", *Textile Science and Engieering* Vol. 53 No. 5, pp. 366-371.

[25] Mujahid, A., Kakusho, K., Minoh, M., Nakashima, Y., Mori, S.-I. and Tomita, S. (2004), "Simulating realistic force and shape of virtual cloth with adaptive meshes and its parallel implementation in OpenMP", in *Parallel and Distributed Computing and Networks*, pp. 386-391.

[26] Narain, R., Samii, A. and O'brien, J. F. (2012), "Adaptive anisotropic remeshing for cloth simulation", *ACM transactions on graphics (TOG),* Vol. 31 No. 6, pp. 152.

[27] Nealen, A., Müller, M., Keiser, R., Boxerman, E. and Carlson, M. (2006), "Physically based deformable models in computer graphics", in *Computer graphics forum*, Wiley Online Library, pp. 809-836.

[28] O'brien, J. F. and Hodgins, J. K. (1999), "Graphical modeling and animation of brittle fracture", in *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., pp. 137-146.

[29] Pentland, A. and Williams, J. (1989), *Good vibrations: Modal dynamics for graphics and animation,* ACM.

[30] Pritchard, D. (2002), "Freecloth project", available at: http://davidpritchard.org/freecloth/ (accessed 02 Feb 2018).

[31] Provot, X. 1997. Collision and self-collision handling in cloth model dedicated to design garments. *Computer Animation and Simulation'97.* Springer.

[32] Raman, S. K., Pentkovski, V. and Keshava, J. (2000), "Implementing streaming SIMD extensions on the Pentium III processor", *IEEE micro,* Vol. 20 No. 4, pp. 47-57.

[33] Satish, N., Kim, C., Chhugani, J., Nguyen, A. D., Lee, V. W., Kim, D. and Dubey, P. (2010), "Fast sort on CPUs and GPUs: a case for bandwidth oblivious SIMD sort", in *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, ACM, pp. 351-362.

[34] Shewchuk, J. R. 1994. An introduction to the conjugate gradient method without the agonizing pain. Carnegie-Mellon University. Department of Computer Science.

[35] Shewchuk, J. R. (1997), "A two-dimensional quality mesh generator and Delaunay triangulator", *Computer Science Division University of California at Berkeley, Berkeley, California, http://www.cs.cmu.edu/quake /triangle.html,* Vol. No., pp. 94720-1776.

[36] Shreiner, D. and Group, B. T. K. O. a. W. (2009), *OpenGL programming guide: the official guide to learning OpenGL, versions 3.0 and 3.1,* Pearson Education.

[37] Sul, I. H. (2009), "Parallel garment drape simulation of triangular mesh using GPU programming", *International Journal of Clothing Science and Technology,* Vol. 21 No. 4, pp. 252-269.

[38] Sul, I. H. (2010), "Fast cloth collision detection using collision matrix", *International Journal of Clothing Science and Technology,* Vol. 22 No. 2/3, pp. 145-160.

[39] Sul, I. H. and Kang, T. J. (2010), "Regeneration of 3D body scan data using semi-implicit particle-based method", *International Journal of Clothing Science and Technology,* Vol. 22 No. 4, pp. 248-271.

[40] Terzopoulos, D., Platt, J., Barr, A. and Fleischer, K. (1987), "Elastically deformable models", in *ACM Siggraph Computer Graphics*, ACM, pp. 205-214.

[41] Teschner, M., Kimmerle, S., Heidelberger, B., Zachmann, G., Raghupathi, L., Fuhrmann, A., Cani, M. P., Faure, F., Magnenat-Thalmann, N. and Strasser, W. (2005), "Collision detection for deformable objects", in *Computer graphics forum*, Wiley Online Library, pp. 61-81.

[42] Vassilev, T. I. (2010), "Comparison of several parallel API for cloth modelling on modern GPUs", in *Proceedings of the 11th International Conference on Computer Systems and Technologies and Workshop for PhD Students in Computing on International Conference on Computer Systems and Technologies*, ACM, pp. 131-136.

[43] Volino, P. and Magnenat-Thalmann, N. 2012. Virtual clothing: Theory and practice. 2000. Springer.

[44] Volino, P., Magnenat-Thalmann, N. and Faure, F. (2009), "A simple approach to nonlinear tensile stiffness for accurate cloth simulation", *ACM Transactions on Graphics,* Vol. 28 No. 4, pp. Article No. 105.

[45] Wang, H., O'brien, J. F. and Ramamoorthi, R. (2011), "Data-driven elastic models for cloth: modeling and measurement", in *ACM Transactions on Graphics (TOG)*, ACM, pp. 71.

[46] Witkin, A. (1997), "Physically based modeling: principles and practice constrained dynamics", *Computer graphics,* Vol. No., pp. 11-21.

[47] Yu, W., Kang, T. and Chung, K. (2000), "Drape simulation of woven fabrics by using explicit dynamic analysis", *Journal of the Textile Institute,* Vol. 91 No. 2, pp. 285-301.

[48] Zhang, D. and Yuen, M. M.-F. (2000), "Collision detection for clothed human animation", in *Computer Graphics and Applications, 2000. Proceedings. The Eighth Pacific Conference on*, IEEE, pp. 328-337.