EUROGRAPHICS '98 / N. Ferreira and M. Göbel
(Guest Editors)

Volume 17, (1998), Number 3

The Art of Knitted Fabrics, Realistic & Physically Based Modelling Of Knitted Patterns

M. Meißner and B. Eberhardt [†]

WSI/GRIS, University of Tübingen, Germany
iMAGIS / IMAG, B.P. 53, 38041 Grenoble Cedex 9, France

Abstract

In this paper we will present a system to use three dimensional computer graphics in garment design. This system is capable to visualize the "real", i.e. the physically correct, appearance of a knitted fabric. A fast visualization of a physically correct micro-structure garment is of crucial importance in textile industry, since it enables fast and less expensive product development. This system may be either used in the design of new products or teaching the art of knitted fabrics.
We use in our system directly the produced machine-code of the design system for knitting machines. A physical model, a particle system, is used to calculate the dynamics of the micro-structure of the knitted garment.
keywords: Physically based modeling, textile modeling, computer graphics, 3D-Models

1. Introduction

Having a look at the history of mankind, we have evidence that textiles exist since thousands of years. We know many different ways to produce garments from a single line of yarn, like braiding, knotting, weaving, knitting, etc. The two main methods however, compared to the amount of produced materials, are weaving and knitting. While weaving is simply the interlacing of weft and warp yarns, knitting is made by the interleaving of loops. Since weaving is technically much simpler we soon find in history mechanical devices to produce more woven textiles in less time.
Four centuries ago, the first knitting machine was invented by Rev. William Lee from village Calverton near Nottingham. Since then, knitted fabrics prospered in areas where elasticity, comfort and insulation are required. In fact knitted garments found their way in leisure-wear and even into haute couture. A large amount of knitted fabrics is produced on flat knitting machines which are very versatile in producing nearly arbitrary types of knitting patterns. Herein are at the same time the advantages and disadvantages in knitted fabrics. There are billions of ways to combine inter-meshing loops and each year four different collections of new fashion have to be invented by designers. This cost-intensive work has to be made faster and less expensive.
The approach using computer graphics may become part of producing fast and "real-looking" virtual textiles. Thus the expensive knitting machines are available for production instead of being needed for labor and cost-intensive pattern-design.
The visualization and simulation of textiles have already made its palce into the history in computer graphics. The draping of textiles has thoroughly been explored by several authors. One of the first was J. Weil [7]. He used a purely geometric approach of catenary surfaces. The research-group of N. Thalmann is known to have pushed cloth-modeling [2] to a high extend. Moreover, D. Breen et. al. proposed a particle system approach [1] together with an energy minimization process to simulate the draping of textiles. All these papers have a sophisticated simulation part in common in order to calculate the trajectories of the particles. In addition collision detection algorithms are elaborated suitable for cloth-modeling. A good survey on cloth-modeling techniques can be found in [5].

The visualization of the micro-structure of textiles however, especially the one of knitted fabrics, has so far not caught much attention within the computer graphics com-

[†] This work was done while this author was research assistant at WSI/GRIS, University of Tübingen, Germany

munity. But it is this micro-structure which is of vital importance to the textile industry.
In this paper we focus on the knitting process and not on the visualization of yarns which has been presented for instance in [4]. Here we want to present an application which allows an almost immediate visualization of a newly designed knitting pattern. Therefore, time consuming visualization techniques, like ray-casting in [4], are not applied. Although, one can always add it for a final rendering for advertising purposes. Nevertheless we want to calculate a natural impression of the pattern based on knitting data and physical properties of inter-meshing yarns.
In order to perform this, we take knitting-machine-data (WKT-data format of the Stoll company), generate a data structure representing it, and apply a particle system to simulate the physical behavior of the pattern. For visualization purposes we use simple drawing routines from the OpenGL library.

2. KnittSim

Here we present a new approach that is able to produce simulated samples of knitted fabrics from the original data produced by a designer. Usually, a knitting machine has to interpret this data and produce the final sample, which is fairly time consuming and expensive.
KnittSim is a system which fully simulates the entire knitting process. As a result, images of samples of knitted fabrics can be generated without even using a real knitting machine. This is not only a highly innovative step for industry but even more an extremely demanding process which has to include the physical properties of the material, i.e., yarn. The input data for our system is the same as the data which would be send to the real knitting machine. In our case it is a data format of the Stoll company, the largest knitting machine producer in Europe. Figure 1 shows a knitting machine from Stoll.
Our virtual knitting machine system, KnittSim, is implemented in an object oriented framework using C++. The system is equipped with an intuitive Graphical User Interface (GUI). It was developed using RapidApp including an Open Inventor viewer to enable three dimensional viewing of the knitted fabrics and interactive handling. The GUI is shown in Figure 2.

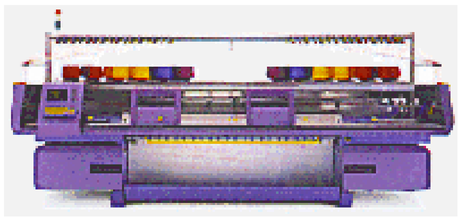
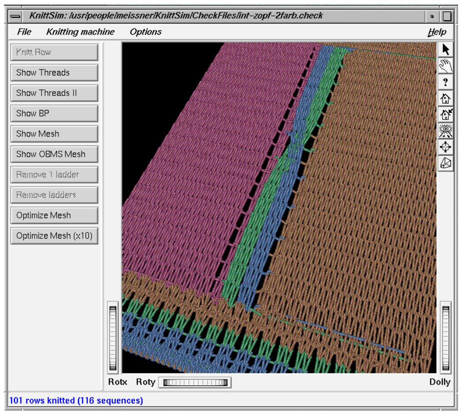


Figure 1: *Stoll knitting machine.*



101 rows knitted (116 sequences)

Figure 2: *Graphical User Interface of the KnittSim system.*

Global parameters of a knitting machine, i.e., the amount of needles per inch, needles per needle bed, length of needle bed, set of yarns, etc are taken into account for the simulation. An advantage of physically based simulation of knitted fabrics is the fact that we are able to *replace* any thread of the knitted fabric by different ones. On a real knitting machine this could only be achieved knitting the fabric again using different yarns. In contrast, our system simply allows the replacement of yarns and the simulation can be restarted.
Like a real knitting machine, KnittSim has two needle bed objects which can be equipped with any amount of needles. One rear and one front needle bed exist. Additionally, our virtual knitting machine is equipped with some more objects needed for the simulation process.
The knitting process is separated into three steps: *Topology Generation*, *Topology reduction*, and *Topology Refinement*. In the first step (topology generation), we generate a data structure containing all necessary information of the fabric by interpreting the input WKT-data. This step is done knitting along a grid and accepting non-physically based stretches, shears, etc. By storing the length of the initially distributed thread, we are capable of equalizing those distortions in the third step, the physically based topology refinement. Inbetween (topology reduction), the topology can be reduced by filtering out all instable interaction points, as for instance, ladders.

3. Data Structure

Representing a knitted pattern in an adequate data structure requires a highly flexible approach. Nowadays, there is almost no limit for the designers of patterns, using modern knitting machines, and every day new and very complex patterns are generated. This is extremely remarkable, knowing that a knitting machine can perform only a few operations.
A data structure being capable of storing all necessary information of a specific knitted pattern has to fulfill a vast num-

ber of requirements. We will itemize a few of them which are most important.

- Associated thread material information
- Thread course in 3D space
- Thread interaction points

It is necessary that we are able to "redraw" the knitted pattern in 3D space based on the generated data structure. Additionally all information needed for physical based modeling have to be available, i.e., the fabric is exposed to forces and has to be stretched adequately. Therefore, we developed a model that can represent all types of knitted patterns in the above described way. In order to understand this model and the underlying data structure, it is first necessary to have a closer look onto some examples of knitted patterns. Figure 3 (a) and (b) show a plain knitted pattern containing only knit stitches. Some more fancy patterns are shown in Figure 4.
How is the 3D curve of a thread influenced? Is it really necessary to store the entire curve in an explicit function or a parameterized representation? This would be quite expensive and fortunately there is an easier solution to this problem: Seen from an abstract point of view only regarding forces, a thread can be seen as a polygonal line. At points where the thread is exposed to deformation or forces, it gets bended. In the context of knitted patterns, we call such points of deformation bonding points (BPs), since threads bond together and interact, i.e., push against each other. From BP to BP, a thread is not exposed to any other force than gravity. Gravity does not cause a thread to hang loose within common knitted patterns due to the following reasons:

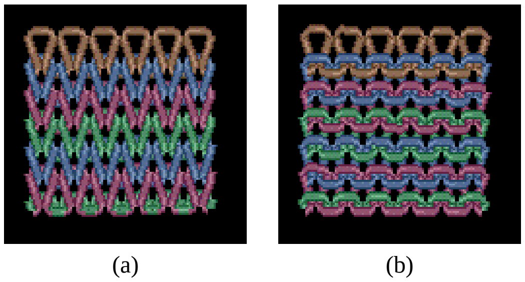

(a) (b)

Figure 3: *Example of a knitted pattern: (a) Front view (knit) (b) Rear view (purl).*

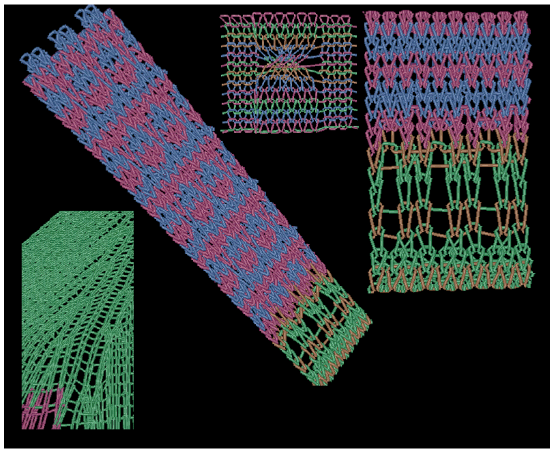


Figure 4: *Typical patterns of knitted fabrics.*

- Threads are light weighted.
- The distance inbetween BPs is small.

As we all know from daily experiences, threads can be pulled out of a knitted fabric and hang loose. Fortunately, this does not occur during the generation of the pattern itself and does not need to be included into the simulation.
Our model is simple but close to reality. In real knitted fabrics, slip of yarn is only observed within the range of two loops. Therefore, one can neglect the rubbing of the yarn against each other. Thus we may consider just the location of bonding points. However in a ladder we distribute the amount of free yarn around the neighboring yarn loops.

A thread can be seen as a chain of BPs, a bonding point course (BPCourse). From a highly abstract view, the basic structure of the knitted pattern is primarily independent of the real location of the individual BPs in space. There is a tight dependency of the distance of the BPs when it comes to the physically based modeling which has to pay attention to the forces inside the knitted pattern. But considering the basic structure, the curves of the threads depend on abstract BPs. BPs of different threads share the same spatial coordinate but this is stored only once and shared by different BPs. This circumvents duplicate information in the data structure. The overall knitted pattern is represented in a mesh structure composed by the BPs and the edges formed by the threads inbetween the BPs. Figure 5 (a) shows a knitted pattern, its BPs, and the resulting mesh structure of the knitted pattern.
It is obvious, that the thread course itself cannot be drawn by simply connecting BPs. More polygon segments are needed to get a better approximation of the curve at the bonding points which can also be seen in Figure 5(a). Additionally, the real thread course will not pass though the BPs but wind around them forming a loop. Nevertheless, the polygonal model used for the physically based simulation is valid to represent the interactions between threads.

As the pattern has to be stored in a flexible way using a linked list structure, we implemented our own template classes CList and CListContainer. An instance of CList provides the entity of a doubly linked list with numerous methods enabling fast access to the head and last element of the list, as well as further needed search methods. The template class CListContainer is an abstract container providing a pointer to the next and previous container, and a pointer to the object stored in this container. We will now elucidate the structure of a knitted pattern shown in Figure 6. It is a fairly small portion of a knitted pattern, simply two loops,

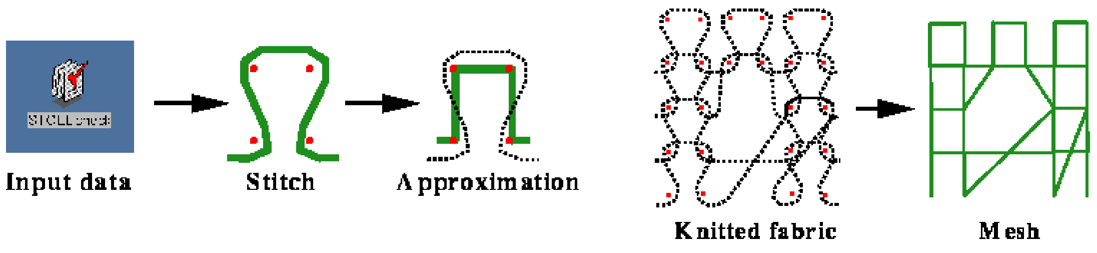


Figure 7: *Approximation of the thread course on a mesh.*

but it suffices to demonstrate the basic structure. The two stitches in this example are represented in a BPDataStructure, containing a list of two threads (BPCourse). Each of the BPCourses is itself a linked list of BPs. The first BPCourse consists of two and the second BPCourse of four BPs. As mentioned earlier, each BP is an abstract object providing information of the generation history of the BP, for instance: Was BP generated in rear or front needle bed, is it a top, bottom, right, or left BP. This is information necessary to "redraw" the thread based on the data structure. Each BP has a pointer to its spatial coordinates. Those coordinates are stored in a BPTable which essentially is a sorted linked list of BPTableEntries, each holding the spatial coordinates of the BPs. As mentioned earlier, there are BPs of different BPCourses sharing the same coordinates. Hence, they point to the same BPTableEntry in the BPTable circumventing the redundant storage of information mentioned earlier. In this case, the BPTable consists of four entries. To accelerate search operations, we keep the BPTable sorted.
The way we store a knitted pattern is an approximation. Instead of storing for each thread the curve in 3D, we store a mesh with edges from each BP to the neighboring ones representing the simplified structure of the knitted pattern. Figure 7 demonstrates the iterations from the input data to the final mesh structure for a simple pattern.

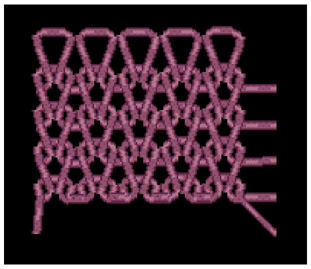
As mentioned earlier, we have to be capable of redrawing the knitted pattern based only on the data structure. This requires some additional information which we did not mention so far. For instance, each BP obtains information during its generation indicating whether it was generated in the rear or front needle bed, whether it is on the left or right side of a stitch, etc. This information allows us to redraw the thread course in a fast and easy way.
Since we are interested in interactivity, we cannot afford drawing any computation intense lines. Therefore, we simply draw line primitives (polygonal lines). To be more precise, we draw three lines each differently wide and bright. The widest is the darkest while the thinnest is the brightest. Furthermore, the thinnest line is dotted. Thus, the resulting line is conceived as being “three dimensional”.

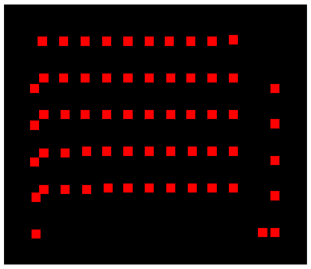
4. Topology Generation

Generating the previously described data structure out of the given input data format is a quite demanding process. A topology representing the relation of the threads to each other has to be generated. To have a better understanding of this process we will delineate the knitting machine and its possible operations.
Yarn carriers feed the yarn, coming off a yarn cone, into the knitting zone of a flat knitting machine where it is processed. Basically, there are three operations a knitting machine can perform: *Knit*, *transfer*, and *rack*.

Knit: To generate a stitch (knit, purl, or tuck), the needle has to be driven out to grab the yarn from the yarn carrier. The grabbed yarn can either be added onto the needle’s hook or be dismissed in case the needle is not fully driven out. Subsequently, the needle is driven back and, in case the thread was grabbed, a new stitch is generated. Figure 8 illustrates this process of generating a new stitch. There are more possibilities to generate stitches which we cannot delineate in this paper. For further information about the fairly complex knitting process we refer to [6].
Everytime a new row is knitted, a new BPCourse is inserted into the data structure. Correspondingly, every new stitch generates new BPs which are inserted into the CBPCourse. The related 3D coordinates are stored in a BPTableEntry which itself is added to the linked list of the BPTable. Note, first we have to check whether the BPTableEntry already exists to prevent redundancy. This operation is fast even for large patterns containing many BPs because the list is kept in sorted order. Adding a new row of stitches can overlap only with the BPTableEntries generated during the knitting process of the previous row. Hence, we only have to compare BPTableEntries with Y coordinate greater or equal of the previous row. In case an equivalent BPTableEntry exists, no new instance has to be inserted. The BP simply shares this BPTableEntry with one or more BPs of other BPCourses.

Transfer: The elements of a needle are transfered onto the needle of the opposite needle bed. Two situations are possible. The opposite needle is empty or the opposite needle already contains some elements. In the first case, we simply


(a)


(b)

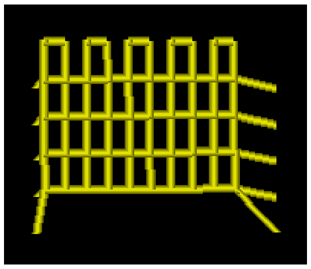

(c)

Figure 5: *Data structure of a knitted pattern: (a) Knitted pattern. (b) Corresponding BPs. (c) Corresponding mesh structure.*

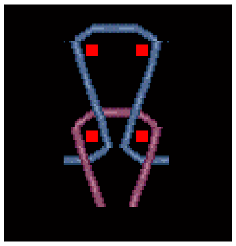


Figure 6: *Knitted pattern containing two knit stitches.*

modify the coordinates of the BPTableEntries relating to the BPs such that they are located within the other needle bed. This essentially results in adding or subtracting the needle bed distance to the BPTableEntry coordinates. In the latter case, we really have to transfer each BP from the old needle onto the other needle already holding BPs of the same or an-

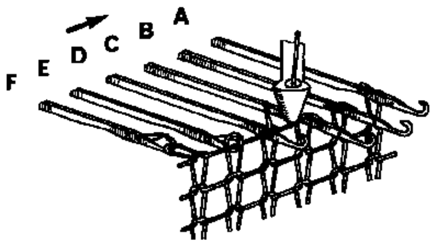


Figure 8: *Generation of a knit stitch.*

other thread. As a result, we obtain a reduced BPTable since four entries merge into two.

Rack: This operations does not affect the individual needle but the entire rear needle bed. It is either shifted to the left or to the right. In any case, we add or subtract the offset to all BPTableEntries which are affected by this operation.

The grid orientation of our approach introduces some *stretches* or *distortions* which can be seen in Figure 7. This is mainly due to the association of an Y-coordinate to each row. Once we generate a new row, the loops hanging on inactive needles are subject to stretching. Stretching a stitch or, to be more precise, the corresponding mesh element in the data structure does not affect the topology in any way. In the topology refinement step, those stretches are *equalized* corresponding to the physical forces. This is possible because we store the actual distributed amount of thread over each stitch. Hence, a stretch of the stitch (upper two BPs) results in a strong force pulling the BPs corresponding to the following formula:

$$F_i(x,y,z) = \sum_{k=0}^{n_i} V_k(x,y,z), \tag{1}$$

where i is the index of the mesh point, F_i is the force vector of i-th mesh point, n_i is the number of neighbors of i-th mesh point and V_k is the force-vector to the k-th neighbor from i-th mesh point, i.e. just a superpositioning of occurring forces. The force-vector V_k might be determined by empiric yarn data (used in our system) or be approximated to:

$$V_k(x,y,z) = C_k \cdot (\text{dist\_to\_neigh\_k - default\_dist})^{\alpha} \quad , \tag{2}$$

where again C_k is some material (yarn) dependent constant and α some integer $1,3,5$.

Truly, the topology can be influenced and changed by a combination of rack and transfer operations. A very simple example is a pattern containing two loops within the front needle bed. By transferring one loop to the other rear needlebed, racking it to the other side of the other loop, and transferring it back, a collision is produced due to crossing threads residing within the same plane. This has to be detected during rack and transfer operations. On such operations we test the involved thread segments for collision and either bend corresponding BPs until the collision is resolved or introduce new BPs.

5. Topology Refinement

Fitting the inter-meshing yarn loops directly into the produced mesh of BPs is hardly satisfying since at this point we have not yet considered the physical features of the knitted yarn loops in the newly designed knitting pattern. To simulate the correct physical behaviour, we adapted a particle system [3] to calculate the dynamics of the stretching, repelling and bonding of the yarn in the microstructure of the virtually knitted pattern. Since we have stored the actual length of the thread inbetween the BPs in the data structure of the bonding point mesh. We assume that spring forces pull or repel the neighboring particles toward each other. Thus we get a coupled particle system, as demonstrated in Figure 9.

Moreover, the bending of a yarn loop is simulated via another spring force "across" two bonding points, see Figure 10.

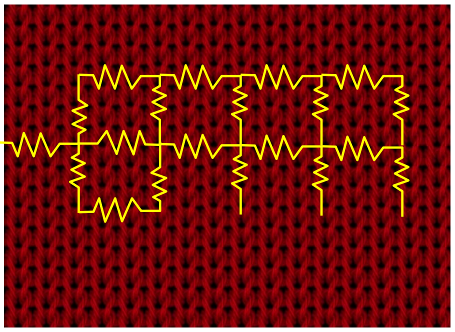


Figure 9: *Repell and stretch of bonding points.*

The slip of yarn at a bonding point is simulated via a rough heuristic. For instance, a ladder which resolves results in a slip of a yarn. Therefore, we distribute its length to the two neighbouring yarn loops. This is also a natural observation in "real" knitted fabrics.

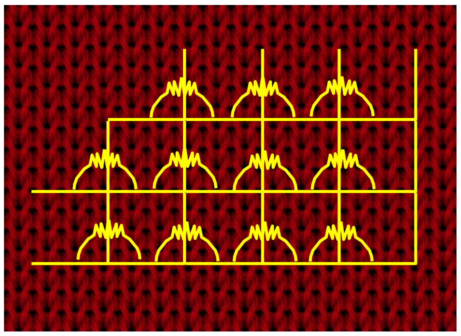


Figure 10: *Bending of yarn loops.*

Calculating the dynamics of the fabric is now a physical simulation of particles. Having the spring forces, we calculate the present energy distribution of the particle system via:

$$\begin{aligned} V &= \sum_{allsprings} \text{energy of spring}, & (3)\\ E_{kin} &= \text{total kinetic energy , and} & (4)\\ L &= E_{kin} - V\ . & (5)\end{aligned}$$

The used potentials for springs are: Let $\mathbf{x}_i$ be the location of the i-th BP and $\mathbf{x}_k$ the location of the k-th neighbor. In addition we denote with $\mathbf{x}_{m,i-1}$ and $\mathbf{x}_{m,i+1}$ the location of the successive BPs of BP i along the thread m passing through BP i. Let $\omega_{m,i}$ be the angle between the vectors $(\mathbf{x}_{m,i-1} - \mathbf{x}_i)$ and $(\mathbf{x}_{m,i+1} - \mathbf{x}_i)$ measured between 0 and 180 degrees. Then we have

$$\begin{aligned} V^{bend}_{i,m} &= C \cdot (\omega_i - 180^\circ)^2 & (6)\\ V^{repel}_{i,k} &= C \cdot (\|\mathbf{x}_k - \mathbf{x}_i\| - \text{defaultdist})^2 & (7)\\ V^{pull}_{i,k} &= C \cdot (\|\mathbf{x}_k - \mathbf{x}_i\| - \text{defaultdist})^4\ , & (8)\end{aligned}$$

here, C is always an appropriate constant.
The trajectories of the bonding points are obtained through a numerical solution of the well known Euler-Lagrange differential equation

$$\frac{\partial}{\partial t}\left(\frac{\partial L}{\partial v_i}\right) = \frac{\partial L}{\partial x_i}\ ,$$

which produces a second order ordinary differential equation of each coordinate of each bonding point.

We achieved good results using the fairly standard Cash-Carp Runge-Kutta solver with adaptive stepsize, however we have included other solvers in our system, ready to be used. Simulations may be seen in the following Section, where we present the results before and after the dynamic simulations (see in particular Figure 11).

Physical simulations depend on correct initial conditions. This triviality is in particular not always guaranteed in a straight forward approach of the realization of the knitting process. Mostly, collisions are detected using the mechanism described earlier. Unfortunately, in some fancy patterns, there might still be intersecting threads in the initial conditions present. Thus, in rare cases, we have to reconstruct a better starting condition of our initial net of bonding points. Fortunately, we found that a relocation (by hand) of just a few bonding points is sufficient to get good simulation results for those cases as well.

6. Results

We present some patterns which were fully generated using our virtual knitting machine KnittSim. A very interesting example is a cable pattern, commonly used by designers. In Figure 11, we show both, the knitted fabric and its mesh structure. Stretches and distortions generated in the grid oriented topology generation phase have been removed

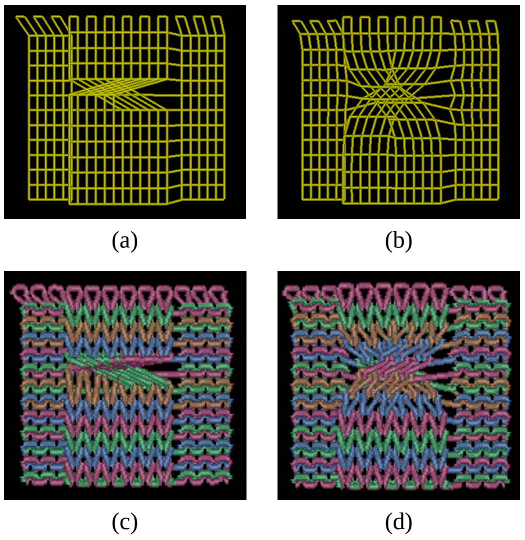


Figure 11: *Cable: (a) mesh of a cable before physically based refinement (b) refined mesh (c) real cable, as (a) (d) real cable, as (b).*

by the topology refinement phase. The image shows nicely the smooth curve of the threads forming the cable pattern.

Another example is shown in Figure 12(a). It is a knitted fabric containing a ladder. This is a stitch which is *instable* and will disappear once we stretch the fabric from outside. Such stitches can be detected and removed resulting in the pattern of Figure 12 (b). Note that there is still a stretch which is then removed during the topology refinement phase, see Figure 12 (c). The ladder removal can be done at any point in time, before or after the refinement.

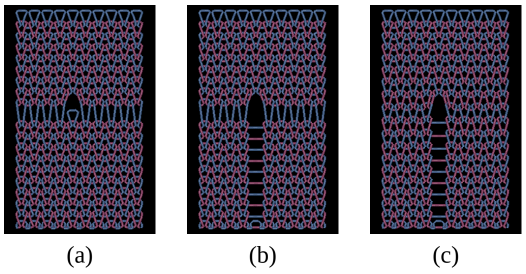


Figure 12: *Knitted fabric containing a ladder: (a) After topology is generated. (b) After reduction of topology. (c) Optimized.*

7. Conclusions

We introduced a system being capable of visualizing and demonstrating the whole knitting process of a given designed knitting pattern. This approach is a new step towards a less expensive production of knitted fabrics. It may be used in the design of patterns, teaching the art of knitting, and the verification of machine data. Our system incorporates the visualization and a physical simulation to achieve a most realistic (physically correct) appearance of the pattern.

8. Future Work

Currently, we are refining the topology generation step which lacks some properties. It is essentially necessary to detect **all** penetrations of threads. Therefore, we have to modify and extend the collision detection mechanism applied during the topology generation phase. Once this is included in the system, we will investigate how to incorporate extern forces which can be brought into the system, i.e., someone pulling the knitted pattern on two opposite sides. This will require a closer interaction of the topology optimization step and the data structure, i.e., the thread must be capable of sliding along a BP once a certain force threshold is reached exceeding the natural bonding force of threads.

9. Acknowledgements

We would like to thank the Stoll GmbH company and the state of Baden-Württemberg who funded the project. Further we would like to thank the people at IFA CAD/CAM, Heiko Denk and Tobias Hahn for their routines of the KnittSim system. Finally, we would also like to thank the people at WSI/GRIS, J. Hahn, R. Sonntag and J. Krämer. They provided us with routines of the RadioLab System and OBMS (Object based Modeling System) for the geometric representation of the knits.

References

1. David E. Breen, Donald H. House, and Michael J. Wozny. Predicting the drape of woven cloth using interacting particles. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 365–372. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

2. Martin Courshesnes, Pascal Volino, and Nadia Magnenat Thalmann. Versatile and efficient techniques for simulating cloth and other deformable objects. In Robert Cook, editor, *SIGGRAPH 95 Conference Proceedings*, Annual Conference Series, pages 137–144. ACM SIGGRAPH, Addison Wesley, August 1995. held in Los Angeles, California, 06-11 August 1995.

3. B. Eberhardt, A. Weber, and W. Strasser. A fast, flexible, particle-system model for cloth draping. *IEEE Computer Graphics and Applications*, 16(5):52–60, September 1996.

4. E. Gröller, R. Rau, and W. Straßer. Modeling and visualization of knitwear. *IEEE Transactions on Visualization and Computer Graphics*, 1(4):302–310, 1995.

5. Hing N. Ng and Richard L. Grisdale. Computer graphics techniques for modelling cloth. *IEEE Computer Graphics and Applications*, 16(5):52–60, September 1996.

6. Samuel Raz. *Flat Knitting - The New Generation*. Meisenbach Bamberg, Meisenbach, 1991.

7. J. Weil. The synthesis of cloth objects. *Proc. SIGGRAPH*, 20:49–54, 1986.