Pacific Graphics 2008
T. Igarashi, N. Max, and F. Sillion
(Guest Editors)

Volume 27 (2008), Number 7

Knitting a 3D Model

Yuki Igarashi[†1], Takeo Igarashi[1,2], and Hiromasa Suzuki[1]

[1]The University of Tokyo, Japan
[2]JST ERATO

Abstract

A knitted animal is made of a closed surface consisting of several knitted patches knitted out of yarn and stuffed with cotton (Fig. 1). We introduce a system to create a knitting pattern from a given 3D surface model (mainly designed for rotund animal models). A knitting pattern is an instructional diagram describing how to knit yarn to obtain a desired shape. Since the creation of knitting patterns requires special skill, this is difficult for nonprofessionals. Our system automates the process and allows anyone to obtain his or her original knitting patterns from a 3D model. The system first covers the surface of the model with parallel winding strips of constant width. The system then samples the strip at constant intervals to convert it into a knitting pattern. The result is presented in a standard visual format so that the user can easily refer it during actual knitting. We show several examples of knitted animals created using the system.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and TechniquesInteraction Techniques; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling Geometric Algorithms

1. Introduction

Computer graphics can be a powerful tool for designing physical objects. While traditional computer-aided design systems have been mainly used by professional users, inexpensive personal computers accompanied by recent progress in computer graphics and interaction techniques make it possible for nonprofessional users to benefit from computational systems in their personal handicraft activities. Previous attempts to build such tools include the design of paper craft toys [MS04, STL06] and stuffed animals [JKS05, MI07]. In this work, we apply a similar approach to another interesting application - that of designing knitted animals.

A knitted animal is made of a closed knitted yarn surface that is stuffed with cotton as shown in Fig. 2 (d). Typical knitted animals consist of several patches each of which corresponds to the body, head, arm, or leg as shown in Fig. 1. Each patch is topologically equivalent to a disk and knitted out a single strand of yarn forming a spiral pattern consisting of multiple rows. Each stitch in a row is usually connected with a stitch in the previous row and one in the next row, forming a square cell. However, a stitch is occasionally connected with two stitches in the next row (increasing stitch) or two stitches are connected with one stitch in the next row (decreasing stitch) as shown in Fig. 3 and Fig. 4.

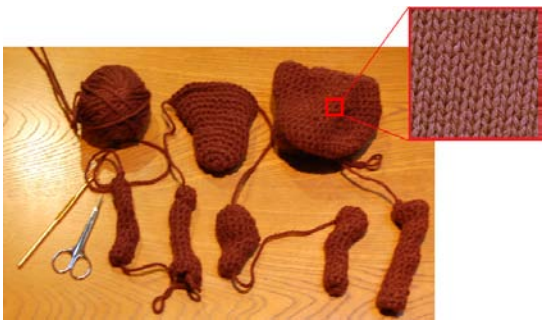
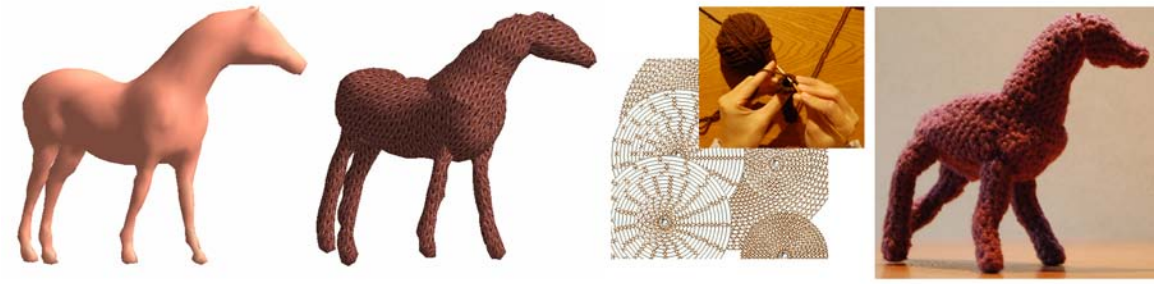


Figure 1: *Example of a knitted animal. A typical knitted animal consists of several circular and cylindrical patches.*

A knitting pattern is an instructional diagram describing how to knit yarn to obtain the desired shape. It shows a sequence of stitches in a circular or horizontal arrangement as

† e-mail:yukim@acm.org



(a) (b) (c) (d)

Figure 2: *The system automatically generates stripping of a 3D model for making a knitted animal from an input triangulated geometry. (a) Original 3D surface model; (b) a 3D virtual knitted animal model; (c) corresponding knitting pattern; (d) an actual knitted animal.*

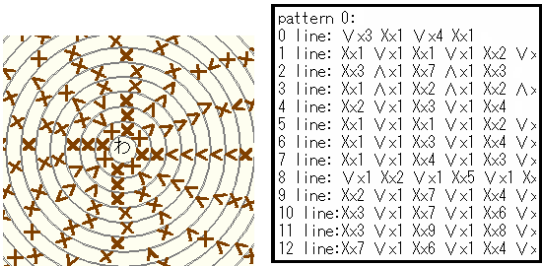


Figure 3: *Example of a knitting pattern. Left figure is a circular pattern and right one is a horizontal diagram. X is a regular stitch mark, V is an increase stitch mark, and Λ is a decrease stitch mark. We use a standard Japanese knitting pattern format in this work.*

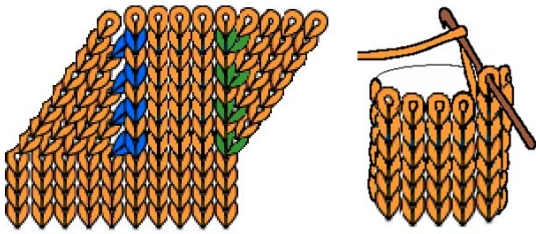


Figure 4: *Structure of knitting yarn. Increase stitches are shown in green, decrease stitches are shown in blue. Others are normal stitches. Our current implementation assumes purled knitting shown here, but basic algorithm should be applicable to other methods such as plain knitting.*

shown in Fig. 3. Since it is difficult for a nonprofessional to design an original knitting pattern, we developed a system to automate this task.

Our system takes a 3D surface model as input and produces a knitting pattern for knitting a real knitted animal. The system first segments the surface into several disjoint patches and then covers each patch by a winding constant-width strip. The system then resamples the strip at constant intervals to determine the individual stitches. Finally, the system generates a 2D knitting pattern by spatially arranging the stitches in the standard format. The user then can create a physical knitted animal by knitting the yarn in accordance with the pattern. Our system also presents a preview of the resulting knitted animal by applying a physical simulation to the model. Our current system is optimized for hand knitting and it is our future work to test our method with automatic knitting machines.

2. Related Work

Various methods have been proposed for making physical toys from virtual 3D models. Mitani and Suzuki [MS04] and Shatz et al. [STL06] presented methods for creating paper craft models from 3D models, while Julius et al. [JKS05] and Mori and Igarashi [MI06] demonstrated methods for creating stuffed animals from 3D models. The Plushie system [MI07] combines a 3D modeling process and physics simulation to design original stuffed animals. These systems transform a 3D surface into multiple 2D developable or quasi-developable patches. In this paper, we propose a method to convert a 3D surface into a single 1D chain of small cells for creating knitted animals. The Knitty system [IIS08] is an interactive design system for creating original knitted animals. It takes the user's sketching as input and generates a 3D knitted animal model and knitting pattern. However, it does not provide a mechanism to convert an existing 3D model to knitting pattern.

Various geometry processing methods have some aspects similar to our algorithm, but no existing method completely satisfies our particular needs. Several algorithms have been proposed to convert a 3D surface mesh into triangular strips [GE04, MS04] but they are not designed to generate strips with constant width. Our system generates a quad mesh with

a small number of triangles. However, existing methods for generating quad meshes [ACSD*03, MK04, DKG05] were not designed to generate quads of equal size. There is a method that covers a surface with quadrilaterals of a uniform size [LHM06], but we also require that cells can form a linear chain. Sweep-based models [HYKJ03, YK06] or generalized cylinder [SK97] are somewhat related to our approach, but again, they are not designed to produce strips of equal width.

Previous work on knit modeling and rendering has mostly focused on more microscopic structure of knitted materials. Examples of knit rendering can be found at [GRS95, SE98] and examples of knit modeling can be found at [GH02, ZS06]. Our work focuses on more global behavior of a knitted model and approximates each stitch with simple planar quads.

3. Algorithm

Our system takes a 3D manifold surface as input. We currently use relatively coarse mesh (1000-3000 vertices) to achieve interactive performance on a 1.1 GHz Pentium M PC. We assume the mesh to be reasonably well sampled; that is, the edge length is shorter than the size of a stitch.

Our system converts a 3D model into a knitted model in the following four steps illustrated in Fig. 5,

1) **Segmentation:** We first manually segment the surface into several disconnected patches. We currently do this manually (Fig. 5 (b)).
2) **Wrapping:** The system covers each patch with winding strips with equal width. This is achieved by computing a series of iso-contours with constant intervals (Fig. 5(c) and (d)).
3) **Re-sampling and meshing:** For each patch, the system places sampling points on each iso-contour at constant intervals and connects the nearby sampling points to form a quad mesh with occasional triangles. Quads represent a regular stitch and triangles represent increasing or decreasing stitches (Fig. 5 (e)).
4) **Pattern Generation and Simulation:** The system generates a knitting pattern (Fig. 5(f)) by arranging rows of stitches. The system also runs a simple physical simulation to predict the shape of the resulting knitted animal. (Fig. 5(g)).

3.1. Segmentation

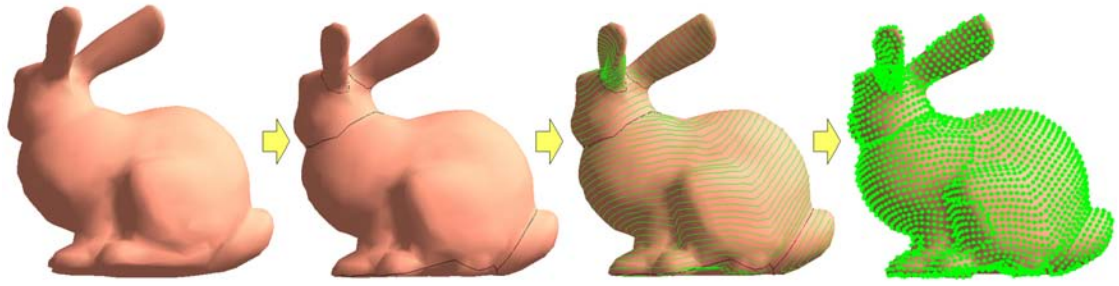
Various automated segmentation methods have been proposed [JKS05, LA06, LCWK07]. However, a knitted part forms a characteristic shape (rotund cylinder or a circle) and different segmentations yield very different final results. Therefore, it is difficult to obtain desired results for our purpose using general automatic segmentation methods. In addition, since the number of patches is usually small, the segmentation process is relatively easy to perform manually, and it can be considered to be part of the creative authoring process. Therefore, we currently use manual segmentation and provide a simple user interface to do this in which the user draws free-form lines on the 3D model surface to indicate the segmentation boundaries. Each patch must be topologically equivalent to a disk or a disk with holes. The system provides a warning if a patch has any other topology such as a sphere or torus. We plan to incorporate some automated method to assist in this manual segmentation process in the future.

3.2. Wrapping

The system first makes each patch topologically equivalent to a disk by filling the holes. That is, for a patch that has holes in it like the body of the bear, the user actually knits a "bowl" (Fig. 6). In our current implementation, the system identifies the longest boundary of a patch as the outer boundary and fills the inside of the remaining inner boundaries. Hole-filling is done by placing a new vertex at the center and connecting the center vertex with the boundary vertices.

The system then covers the patch by a series of iso-contours at constant intervals, starting with the outer boundary and proceeding inwards one by one. The computation works as follows (Fig. 7). The system first computes a distance field around the previous iso-contour and traces an iso-contour in which the value of the filed equals the desired interval. We use simple Euclidian distance in the 3D space here, not geodesic distance on the surface, because neighboring iso-contours are connected by stitches that can be well approximate by straight lines (not curves) in the final knitted model. We use a method similar to marching cubes (marching triangles) to obtain an explicit representation of a contour [LH87]. Specifically, the system computes the distance, d, from each mesh vertex to the previous iso-contour and checks whether d is smaller or greater than the fixed interval, d_0. Then, the system visits each mesh face and places a fragment of the new iso-contour on it if the face contains both vertices with $d > d_0$ and vertices with $d < d_0$. The new iso-contour is obtained by connecting all these fragments.

An iso-contour can be divided into multiple closed loops when the surface is branching (Fig. 8 a). In this case, the system chooses the largest loop as the next contour and continues the process. The other loops are closed at that point and no more iso-contours are generated inside of them (in this case, the artifact will be visible in the final result, prompting the user to specify an additional segmentation). At the end of this creating iso-contours process, the system creates a medial axis inside the last contour (Fig. 8 b). When the axis is sufficiently short, the system creates a vertex at the center of the last contour (Fig. 8 c).



(a) initial 3D surface model (b) segmentation (c) calculation of iso-contours (d) resampling of the iso-contours

(e) meshing (f) simulation and pattern generation

Figure 5: *System overview. (a) input of the 3D manifold surface mesh; (b) manual segmentation; (c) calculation of iso-contours; (d) wrapping strips along with the iso-counters; (e) resampling vertices given the iso-counters and meshing; (f) construction of knitted animal model and corresponding knitting pattern.*

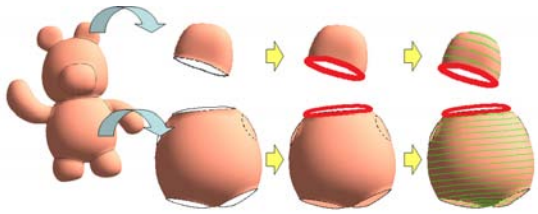
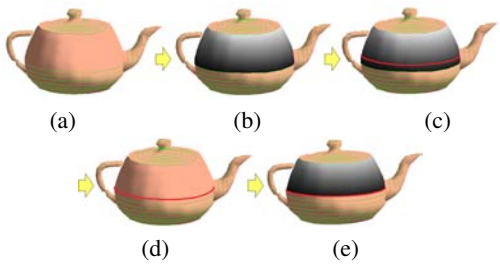


Figure 6: *Finding the seed boundary line. If the target patch has one boundary, the system chooses it as the seed boundary (upper). If the target patch has many boundaries, the system chooses the longest boundary as the seed boundary (lower).*



(a) (b) (c) (d) (e)

Figure 7: *Wrapping algorithm. (a) previous contour; (b) calculation of a distance field; (c) tracing the next contour; (d) the next contour; (e) calculation of the next distance field.*

3.3. Resampling and Meshing

In this step, the system first places sampling points along each iso-contour at constant intervals as shown in Fig. 9(a). The interval used is identical to the interval between iso-contours. The system then connects nearby sampling points on adjacent contours to create a mesh. The detail of making a mesh between two adjacent contours is as follows.

The system first visits each vertex of the first contour and connects the vertex to the closest vertex on the second contour as shown in Fig. 9(b). The system then visits each vertex of the second contour and connects it to the closest vertex on the first contour as shown in Fig. 9(c). We simply take a union of these edges to create the final mesh and this guarantees that all vertices will be connected to at least one vertex of the other contour as shown in Fig. 9(d). We observed that this simple method produces an ideal mesh for our purpose in which only a few vertices in a row are connected to a vertex in an adjacent contour. Exception is the end points. They are connected to around five to eight vertices.

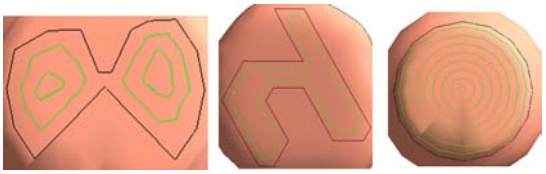


Figure 8: *An example of branching iso-contours.*

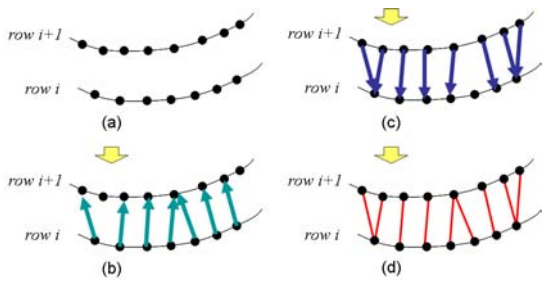


Figure 9: *Construction of a mesh between iso-contours.*

3.4. Rendering and Simulation

The resulting mesh consists of many quads and occasional triangles with nearly identical sizes. Quads represent regular stitches and triangles denote increasing or decreasing stitches. The system visualizes these stitches by assigning the appropriate stitch texture pattern to each cell as shown in Fig. 10.

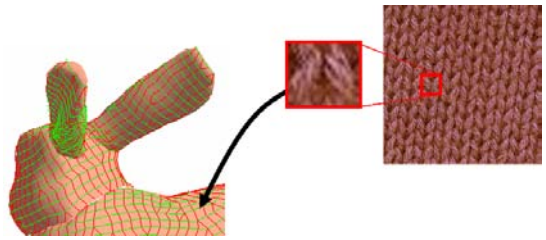


Figure 10: *The system visualizes the stitches by assigning the appropriate stitch texture pattern to each cell.*

Each row forms a closed loop of connected stitches, but it is necessary to specify a point to start knitting each row to arrange the stitches in a 2D knitting pattern. This point is also used to cut the closed loop open to generate a 2D knitting pattern. The system does this by selecting a random starting point on the first iso-contour and tracing the nearest vertex on the consecutive contours.

Several conventions (stitch marks) exist for describing knitting patterns such as circular and horizontal diagrams, but we choose to use a simple representation showing the number of consecutive stitches of each type (regular, increasing, and decreasing) as shown in Fig. 3. Knitting typically starts with a single center and proceeds outward, creating a circular pattern around it. Since our algorithm starts with an open boundary and proceeds inward the system places the rows in the reverse order when creating the knitting pattern.

The final shape of the knitted animal is different from the original 3D model because it is determined by the physical interaction between the knitted skin and cotton stuffing. Therefore, our system runs a physical simulation so that the user can check the shape of final result before actually starting to knit (Fig. 5(f)). We used the simple static method introduced in Plushie [MI07] for an interactive simulation on a standard PC. This simple method gives reasonable results sufficient for our purpose. However, the result still shows many visible artifacts (Fig. 12) and we plan to test more sophisticated methods in the future.

4. Results

Fig. 2(c) and Fig. 11 show actual knitted animals created using our system. The user can color the 3D model using a paint tool and the system automatically updates the color of the corresponding stitches on the knitting pattern. In the actual knitting process, the user changes the color of yarn referring the colored knitting pattern (Fig. 12). The system automatically marks the current knitting position on the knitting pattern column when the user pushes the up and down keys. Clicking the right mouse button on the knitting pattern also shows the user how to knit the particular stitch. The system also indicates the total knitting time and total length of yarn required to produce the model. Table 1 provides timing data for several different animals.

Figure 11: *Actual knitted animals using our system.*

5. Limitation

The models generated by our algorithm can represent subtle concavity and convexity of surface in the direction perpendicular to rows using decrease and increase stitches as shown

Table 1: *Timing data. (RAM1.5GB, 1.1 GHz Pentium M.)*

Model	#vertex	#patch	Segmentation	Wrapping	Meshing	Simulation
House	527	9	0.10 sec	5.22 sec	0.42 sec	15.43 sec
Bunny	1039	6	0.12 sec	13.91 sec	0.60 sec	22.95 sec
Teddy	1990	9	0.23 sec	7.76 sec	0.44 sec	6.71 sec
Teapot	527	4	0.09 sec	2.39 sec	0.37 sec	6.42 sec
Cow	787	12	0.06 sec	6.11 sec	0.45 sec	17.01 sec

Figure 12: *Coloring example. The colors painted on the model surface are also shown in the knitting pattern.*

in Fig. 13. However, it can not represent concavity and convexity in a direction parallel to rows because each row forms a circle. In the actual knitting process, professional users can represent desired concavity by pulling the surface using knit yarn inside of the body.

The quality of the current results is by no means perfect yet. Resulting models are often very different from hand-designed knitted animals. Knitting pattern should be more regular, having irregular stitches (increase and decrease) only near the patch borders. We need to incorporate a number of additional heuristics to solve these problems, which we plan to address in the future.

Figure 13: *Example of local feature. Left shows a dent of ear of bunny. Right shows a forefoot of cow without stuffing cotton.*

6. Conclusion

We presented an automatic method for producing knitting patterns from a 3D model. Our system first covers the surface of the model with winding strips of constant width. This is achieved by iteratively computing iso-contours at constant intervals starting from a boundary and moving inward. The system then samples the strip at constant intervals to convert it into a series of stitches. The final knitting pattern is presented in a standard format so that the user can easily refer to it during actual knitting.

This work demonstrates the effectiveness of designing physical objects by taking into account the material's physical property into account in the construction process. In the future, we would like to apply this approach to other physical objects such as leather crafts and rattan furniture.

References

[ACSD*03] ALLIEZ P., COHEN-STEINER D., DEVILLERS O., LEVY B., DESBRUN M.: Anisotropic polygonal remeshing. *ACM Transactions on Graphics (In Proceedings of SIGGRAPH 2003) 22*, 3 (2003), 485–493.

[DKG05] DONG S., KIRCHER S., GARLAND M.: Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometry Design, Special Issue on Geometry Processing 22*, 5 (2005), 392–423.

[GE04] GOPI M., EPPSTEIN D.: Single-strip triangulation of manifolds with arbitrary topology. *Computer Graphics Forum (EUROGRAPHICS) 23*, 3 (2004).

[GH02] GOKTEPE O., HARLOCK S.: Three-dimensional computer modeling of warp knitted structures. *Textile Research Journal 72*, 3 (2002), 266–272.

[GRS95] GROELLER E., RAU R., STRASSER W.: Modeling and visualization of knitwear. *IEEE Transactions on Visualization and Computer Graphics 1*, 4 (1995), 302–310.

[HYKJ03] HYUN D. E., YOON S. H., KIM M. S., JUTTLER B.: Modeling and deformation of arms and legs based on ellipsoidal sweeping. In *11th Pacific Conference on Computer Graphics and Applications (PG'03)* (2003).

[IIS08] IGARASHI Y., IGARASHI T., SUZUKI H.: Knitty: 3d modeling of knitted animals with a production assistant interface. In *Eurographics 2008 Annex to the Conference Proceedings* (2008), pp. 187–190.

[JKS05] JULIUS D., KRAEVOY V., SHEFFER A.: D-charts: quasi developable mesh segmentation. *Computer*

Graphics Forum (Proceedings of Eurographics 2005) 24, 3 (2005), 981–990.

[LA06] LIEN J. M., AMATO N. M.: Approximate convex decomposition of polyhedra. *Technical Report TR06-2002, Texas A M University* (2006).

[LCWK07] LU L., CHOI Y. K., WANG W., KIM M. S.: Variational 3d shape segmentation for bounding volume computation. *Computer Graphics Forum 26*, 3 (2007), 329–338.

[LH87] LORENSEN W., HARVEY C. E.: Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics (SIGGRAPH 87 Proceedings) 21*, 4 (1987), 163–170.

[LHM06] LAI Y. K., HU S. M., MARTIN R. R.: Surface mosaics. *The Visual Computer 22*, 9-11 (2006), 604–611.

[MI06] MORI Y., IGARASHI T.: Pillow: Interactive pattern design for stuffed animals. In *DVD publication at SIGGRAPH 2006 Sketches* (2006).

[MI07] MORI Y., IGARASHI T.: Plushie: An interactive pattern design for plush toys. *ACM Transactions on Graphics (ACM SIGGRAPH 2007) 23*, 3 (2007), Article No.45.

[MK04] MARINOV M., KOBBELT L.: Direct anisotropic quad-dominant remeshing. In *Proceedings of Computer Graphics and Applications (PG '04)* (2004), pp. 207–216.

[MS04] MITANI J., SUZUKI H.: Making papercraft toys from meshes using strip-based approximate unfolding. *ACM Transactions on Graphics (Proceedings of SIGGRAPH 2004) 23*, 3 (2004), 259–263.

[SE98] STRASSER W., EBERHARDT B.: Representation of knit fabrics. In *SIGGRAPH Course Notes: Cloth and Clothing in Computer Graphics* (1998), pp. F–1–18.

[SK97] SHAFF E. B., KUIJLAARS A. B. J.: Distributing many points on a sphere. In *Springer-verlag new york 19, 1* (1997).

[STL06] SHATZ I., TAL A., LEIFMAN G.: Paper craft models from meshes. *The Visual Computer: International Journal of Computer Graphics (Proceedings of Pacific Graphics 2006) 22*, 9 (2006), 825–834.

[YK06] YOON S. H., KIM M. S.: Sweep-based freeform deformations. *Computer Graphics Forum (Proceedings of Eurographics 06) 25*, 3 (2006), 487–496.

[ZS06] ZAHARIEVA-STOYANOVA E.: Algorithm for computer aided design curve shape form generation of knitting patterns. In *2006 IEEE International Conference on Automation, Quality and Testing, Robotics* (2006), pp. 327–331.