Automated Generation of Knit Patterns for Non-developable Surfaces

Mariana Popescu(✉), Matthias Rippmann, Tom Van Mele, and Philippe Block

Department of Architecture, Institute for Technology in Architecture, ETH Zürich, Stefano-Franscini-Platz 1, 8093 Zurich, Switzerland
mariana.popescu@arch.ethz.ch

Abstract. Knitting offers the possibility of creating 3D geometries, including non-developable surfaces, within a single piece of fabric without the necessity of tailoring or stitching. To create a CNC-knitted fabric, a knitting pattern is needed in the form of 2D line-by-line instructions. Currently, these knitting patterns are designed directly in 2D based on developed surfaces, primitives or rationalised schemes for non-developable geometries. Creating such patterns is time-consuming and very difficult for geometries not based on known primitives. This paper presents an approach for the automated generation of knitting patterns for a given 3D geometry. Starting from a 3D mesh, the user defines a knitting direction and the desired loop parameters corresponding to a given machine. The mesh geometry is contoured and subsequently sampled using the defined loop height. Based on the sampling of the contours the corresponding courses are generated and the so-called short-rows are included. The courses are then sampled with the defined loop width for creating the final topology. This is turned into a 2D knitting pattern in the form of squares representing loops course by course. The paper shows two examples of the approach applied to non-developable surfaces: a quarter sphere and a four-valent node.

Keywords: Knitting pattern · Non-developable surfaces · 3D knitting · Computational algorithm · Automated generation

Introduction

In architecture, textiles have been used as membranes forming tensile structures and have proven to be a feasible solution for the creation of resource-efficient formwork (Brennan et al. 2013; Veenendaal et al. 2011) or reinforcement for complex concrete geometries (Scholzen et al. 2015). Currently, most are produced as flat sheet material. Therefore, the creation of doubly curved architectural shapes with these fabrics requires extensive patterning. Knitting is a widely-used fabrication technology for textiles, which allows for the creation of fabrics with great variety in structure and the possibility of creating bespoke geometries similar to the final shape (near-net shape) without the need for patterning, avoiding waste through offcuts (Van Vuure et al. 2003; Hong et al. 1994; Abounaim et al. 2009). Within the current standard industrial machine width of 2.5 m (Aboumain 2010), architectural-scale elements with high curvature can

K. De Rycke et al., *Humanizing Digital Reality*,
https://doi.org/10.1007/978-981-10-6611-5_24

be fabricated to be used as skins (Thomsen et al. 2008), pavilions (Sabin 2017), or within hybrid systems of bending-active elements and knitted membranes (Ahlquist 2015a, b; Thomsen et al. 2015).

For the creation of a given piece of knit textile, a knitting pattern is necessary to define a set of instructions for the CNC-knitting machine to steer the knitting process. However, current knitting software offers only limited possibilities. Any custom, non-repetitive, non-developable knit pattern needs to be programmed by the user in a manner requiring detailed manipulation and understanding of knitting operations (ShimaSeiki 2017). Solutions allowing for easier manipulation of 2D knit patterns based on primitives, simulation of the resulting 3D shape and steering of the machine have been developed by McCann et al. (2016). For the creation of a given 3D geometry, the knit patterns contain increases (widening), decreases (narrowing), and short or incomplete rows (Fig. 1). The latter is equivalent to locally increasing the number of courses.

Strategies have been developed to generate curved and doubly curved shapes from knitted textiles for primitives such as cylinders, spheres and boxes through the use of mathematical descriptions of the shapes (Hong et al. 1994). If not automated, this process relies heavily on the proficiency of the user and it may be very difficult or impossible to achieve accurate, weft-knit fabrics for shapes beyond basic geometric primitives. Strategies for freeform surfaces, demonstrated by Thomsen et al. (2008; 2015), focus on translating a 3D shape into a 2D knitting pattern through the unrolling of developable surfaces, but do not solve non-developable surfaces.

In contrast to other industries where knitting is used for mass production (e.g. garment and shoe industry, automotive industry), the construction sector has a greater demand for non-repetitive modules using bespoke geometries to meet the requirements of contemporary architecture. The ability to create knitting patterns in a fast and flexible way for various 3D geometries is therefore especially important if knitting is to be used in construction. Possible applications demanding such flexibility are stay-in-place fabric formworks with integrated features for the guidance of reinforcement and other building elements.

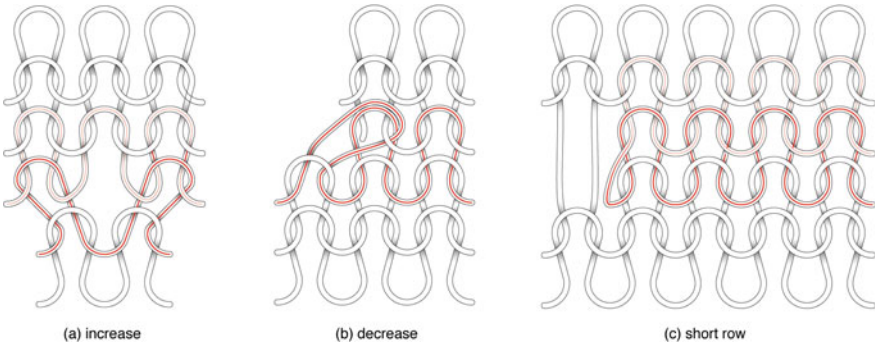

Fig. 1. Knit pattern manipulations: **a** increasing or **b** decreasing the number of loops from course to course; or **c** using short rows

This study presents an approach for directly creating knitting patterns on a given 3D geometry in an automated way without being constrained to developable surfaces. To achieve accurate results, constraints need to be taken into consideration. These are not only related to the variation in the knitting logic, but also to the accurate representation of the heterogeneous material behaviour. Similar to the work of Yuksel et al. (2012) or Igarashi et al. (2008), the presented approach relies on geometric descriptions, surface topology, loop geometry and course direction for the generation of accurate knitting patterns. In addition, the method accounts for the creation of short-rows, which follow the principles of machine knitting fabrication techniques.

In our case-studies we have applied the method among others to a quarter sphere and a four-valent node. The results for these two case studies are presented in Section "Results".

Methods

This section gives an overview of the methods used for the creation of patterns for a given 3D geometry.

As loops are of constant height and width, the knit topology is developed and represented using a quadrilateral network with only a few triangular exceptions that represent increases, decreases, or the starts/ends of short-rows. The quadrilateral network is generated with the following four steps:

1. **Patching**: Depending on the complexity of the 3D geometry, global singularities need to be defined manually to split the geometry in a number of patches.
2. **Course generation**: The surface is contoured and the courses are defined based on the given course height. The generated courses include short rows.
3. **Loop generation**: The courses found in the previous step are sampled with the loop width to define the loop topology for each course.
4. **2D knit pattern generation**: The resulting final topology is translated into a 2D knitting pattern containing each row to be knit with the corresponding number of loops.

For the case studies, we used an implementation of our methods using Rhinoceros 3D 5.0 (2017) as design environment. The implementation is developed using Python 2.7. (2001–2017) and is based on the compas framework (Van Mele 2017) using a network data-structure, which is a directed graph describing both the connectivity and the geometry of the knit (see Sections "Course Generation" and "Loop Generation").

Patching

An arbitrary quadrilateral mesh is the starting point for the generation of the knitting pattern. As a first step, depending on its complexity, the input geometry is segmented into several patches by defining singularities. These patches can coincide to the different pieces of material that need to be put together once the fabric is produced. However, as their primary role is to simplify the geometry and have better control over the pattern generation, they can be more numerous than required for the fabrication of

the final piece. Each patch can therefore be generated using multiple sub-patches. The pattern generation ensures that courses are aligned to both the start and end edges of the sub-patch, in the knitting direction. This results in matching courses between different sub-patches. It also offers the user control over the pattern alignment. The sub-patching can therefore be used as a way of controlling and aligning the pattern to specific constraints or desired features.

For the case studies presented in Section "Results", the input mesh was patched manually, and a knit pattern was generated for each patch. The following sections will describe this process schematically for a single patch.

Course Generation

A minimum of two guide curves, in course direction, are defined by the user to represent the start and end course of the piece. The user also defines the resolution for contouring and the loop parameters. Figure 2a shows the chosen knitting direction and the two guide curves defining the start and end of the piece. Figure 2b depicts the loop geometry and resulting parameters: course spacing/height c in warp direction and loop width w in weft direction. The loop parameters are considered fixed user inputs as they are directly related to the chosen knitting machine and yarn parameters (tension, knit point, gauge, yarn diameter etc.). For our experiments, the loop parameters were determined as presented in Munden (1959) and Ramgulam (2011) using samples knit on a computerised Brother KH970 and ShimaSeiki SWG 091N. These parameters form

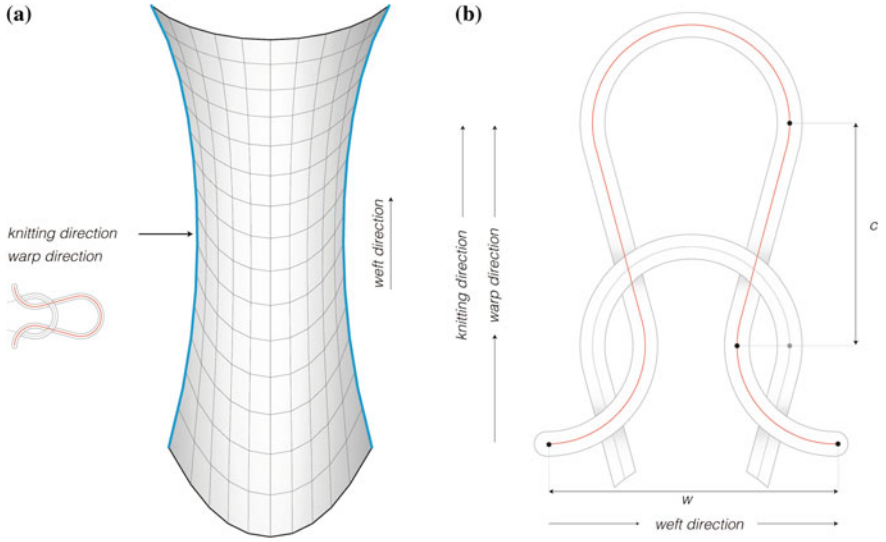

Fig. 2. **a** Mesh patch as input geometry for knitting pattern generation indicating chosen knitting direction and both start and end edges in course direction; and **b** loop geometry where c is the course spacing in warp direction (course height parameter in pattern generation) and w the loop width in weft direction (loop width parameter)

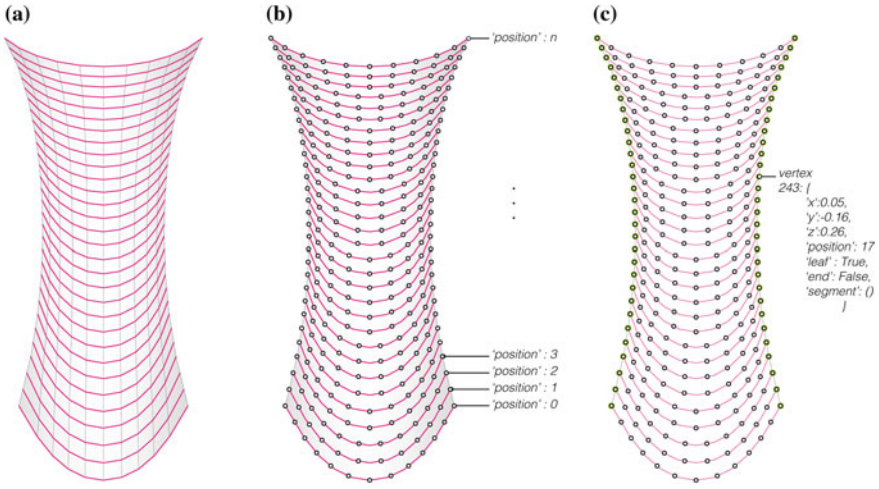

Fig. 3. **a** Contouring of the patch perpendicular to the course direction; **b** ordering and sampling of the contours with the defined course height; and **c** initial network instance with vertex attributes

the basis for generating the courses described in this section and the loop topologies described in Section "Loop Generation".

Given the user-defined parameters, the surface is covered with contours in the direction transverse to the course direction. The contours shown in Fig. 3a are created using level-sets computed on a distance field. These contours are ordered and subsequently sampled with the course height (Fig. 3b). A network of vertices and edges is initialised using the sample points.

Figure 3c depicts the initial network, where each vertex is given a '*position*' attribute corresponding to the contour line order. The '*leaf*' vertices (vertices with a single connecting edge) are also defined. At this point, the '*end*' and 'segment' attributes of the vertices have default values. Their use will be explained in Section "Loop Generation". Edges of the network store information about the directionality of the knitting. They represent the weft and warp direction of the knitting pattern and store this information in '*weft*' and '*warp*' attributes. Note that the edges in the initial network topology are neither '*warp*' nor '*weft*'. They define the contours.

First, the '*leaf*' vertices are connected to form the first course lines (Fig. 4a). Then, the vertices of the network are processed per contour, starting with the longest contour line (Fig. 4e). For each vertex of a contour, a set of potential connection vertices is defined as the four closest vertices on an adjacent contour (Fig. 4b, c). From these candidates, the vertex that creates the connection closest to perpendicular to the adjacent contour is selected (Fig. 4g).

However, when the angle formed by the two best candidates is similar, and the difference is less than 6 degrees, the candidate that forms the straightest line with the connection from the previous contour is preferred (Fig. 4h).

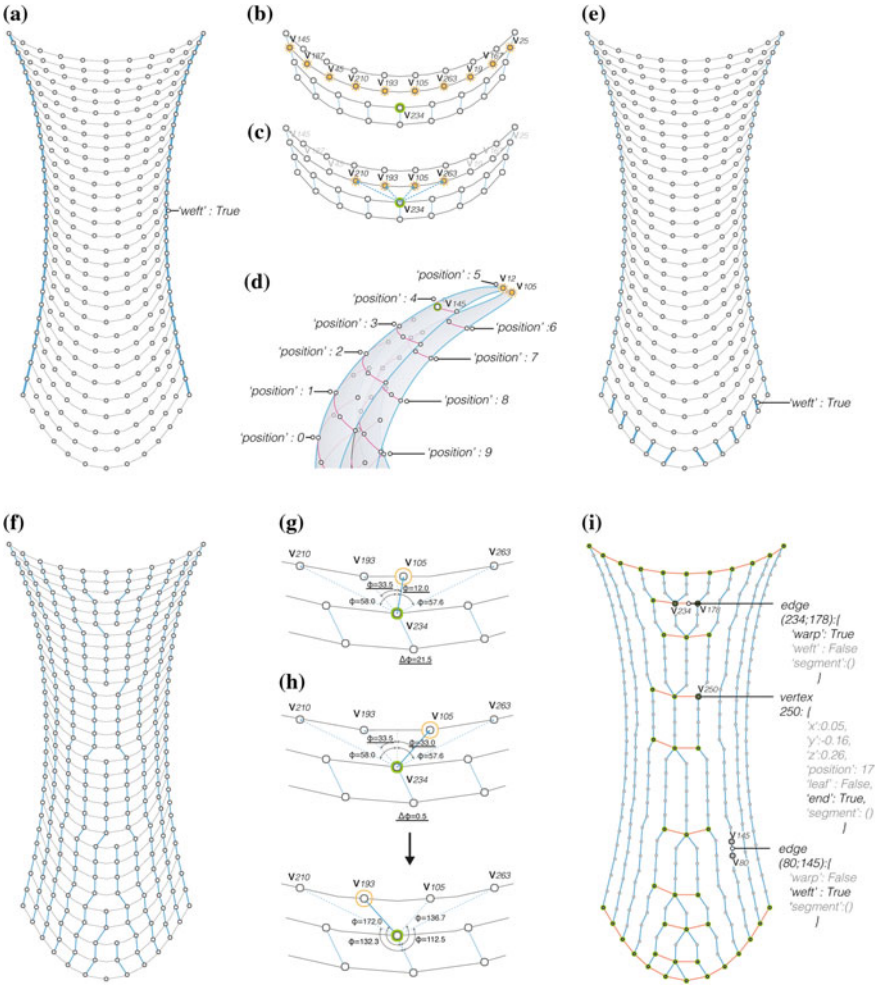

Fig. 4. **a** Connections added to all 'leaf' vertices; **b** set of connection candidates; **c** four closest connections in the set; **d** connection candidates for extreme curvature, based on 'position' attribute; **e** 'weft' connections for longest contour; **f** all 'weft' connections; **g** chosen connection based on minimum angle deviation from perpendicular to the current contour; **h** preferred connection out of similar candidates based on minimum angle change from previous 'weft' connection; and **i** resulting connections in 'weft' direction defining the courses, 'end' vertices, and first 'warp' connections

Note that the resulting '*weft*' edge is only added, if the connecting vertex has less than four already existing connections, and if it is not already connected to another vertex of the current contour.

By restricting the search for connections to vertices on the adjacent contours the approach can be applied to geometries with extreme curvature. The '*position*' attribute

of the vertices ensures that vertices that may be geometrically close but not part of the adjacent contour are ignored (Fig. 4d). The resulting network is shown in Fig. 4f.

In a second pass, all vertices with fewer than four connections are revisited and connected to the closest to perpendicular target in the direction in which no '*weft*' connection exists.

Finally, all vertices with more than four connections, their immediate neighbours over a non '*weft*' edge, and ones on the first and last contour are marked as '*end*'. The non '*weft*' edges connecting two '*end*' vertices are marked as '*warp*' (Fig. 4i).

We include a pseudocode (Listing 13) snippet for generating 'weft' connections, propagating from the starting position to the last position. The same logic is used to propagate the connections in opposite direction such that connections are created in both directions starting from the longest contour.

```
if start_position < last_position:
  current_position = start_position + 1
  while current_position < last_position:
    initial_vertex_list = [all_vertices_on_current_position]
    for vertex in initial_vertex_list:
      target_position = current_position + 1
      if number of vertex neighbours > 2:
        possible_connections = [closest_neighbours_on_target_position]
        most_perpendicular = [ordered_angles_possible_connections]
        if (most_perpendicular[0] - most_perpendicular[1]) < 6:
          connect (vertex, least_angle_change_connection)
        else:
          connect (vertex, most_perpendicular_connection)
    current_position = current_position + 1
    next_vertex_list = [all_vertices_on_current_position
    for vertex in next_vertex_list:
      if current_position != last_position:
        if number of vertex neighbours < 3:
          possible_connections = [closest_neighbours_on_target_position]
          connect (vertex, most_perpendicular_connection)
```

Listing 1. Pseudocode for generating 'weft' connections starting from the longest contour and propagating towards the last contour

Loop Generation

In this step, the courses, represented by all '*weft*' connections, are sampled with the loop width.

Before the final topology of the network can be created, a mapping network is initialised that keeps track of the connected '*end*' vertices (Fig. 5a). The network is created by finding all the '*weft*' edge chains between two '*end*' vertices. These '*weft*'

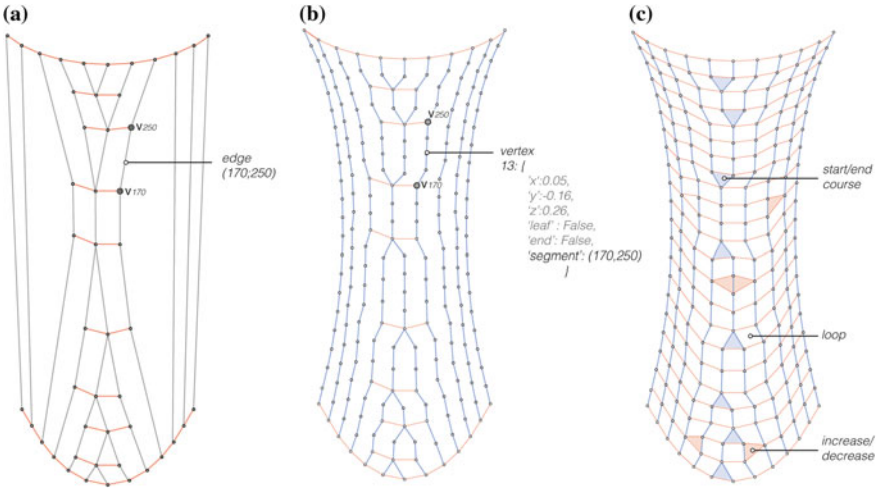

Fig. 5. **a** Mapping network—topological representation of the courses within the network; **b** final network topology based on sampling with the loop width before the generation of 'warp' connections; and **c** final network representing the topology to be knit including short rows, increases and decreases

edge chains are sampled with the loop width. The resulting points constitute the final vertices of the network while the previous vertices and edges that are neither '*weft*' nor '*warp*' are discarded (Fig. 5b). These vertices are also given a '*segment*' attribute, which identifies their position between two '*end*' vertices.

For the creation of all '*warp*' edges the same logic is applied as with the creation of the '*weft*' edges. The difference being that the restricted cloud to search for possible connections is now determined by the mapping network topology instead of the 'position' along a contour.

Figure 5c shows the final knit topology consisting of '*warp*' and '*weft*' edges. Each face of the network represents a loop. In '*weft*' direction, the triangular exceptions represent the start or end of a short-row, while the same exceptions in '*warp*' direction represent the increase or decrease in the number of loops within the next course.

2D Knitting Pattern Generation

For the translation of the topology to a 2D knitting pattern, a dual network is created where each vertex represents one loop and the edges retain the 'weft' and 'warp' directionality. Knowing the connectivity of the vertices through the edges, the topology can now be drawn as a pattern where every loop is represented by a square. Figure 6a shows the pattern representation resulting from the approach presented in this section. Figure 6b shows what the knitting pattern typically looks like in machine knitting software. Each square/pixel also represents a loop. The colour of the square gives additional information about machining operations such as transferring loops and inactivating needles (e.g. decreases), adding needles (e.g. increases), which bed the

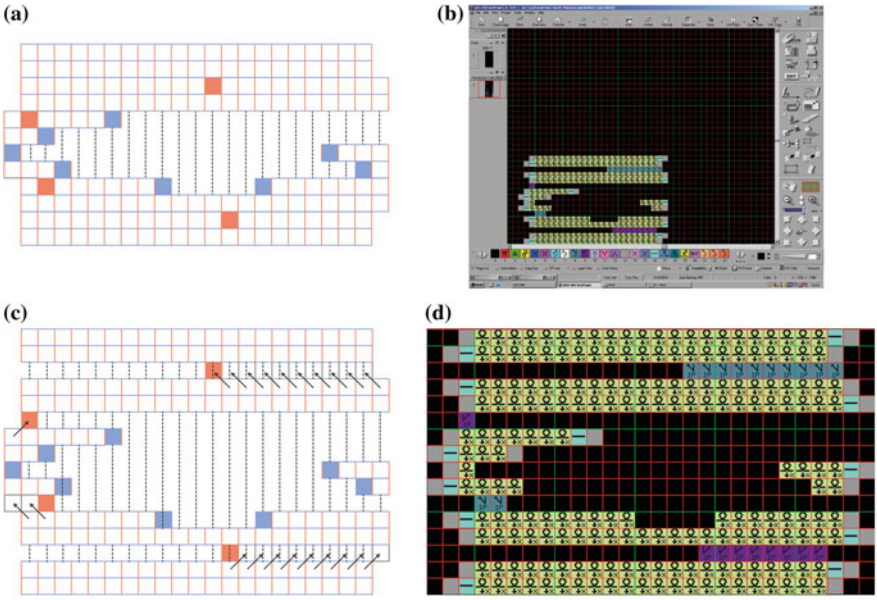

Fig. 6. **a** Resulting 2D pattern; **b** knitting pattern within SDS-one knitting machine software; **c** resulting knitting pattern showing the necessary transformations for knitting software; **d** resulting knitting pattern as represented in knitting software

needle is on etc. Figure 6c, d show the transformations applied to the pattern when translated to a knitting machine software specific pattern.

Results

The approach is tested on a series of knitted prototypes using various 3D input geometries, ranging from simple primitives to freeform surfaces. The results demonstrate that by using the presented approach, an even distribution of stitches and 3D fit of the textile can be achieved.

Reaching an accurate result is highly dependent on using accurate loop dimensions (width and height). The geometry of loops is dependent on a series of machining factors (tension settings, gauge etc.) and on the type of yarn used. With this in mind, calibration of the model can be done by determining the accurate loop dimensions through testing of plain knit samples of material.

Quarter Sphere

Primitives such as spheres and tubes are examples of non-developable surfaces for which patterns are available in most knitting software. Spherical shapes are usually described as patterns with a series of short-rows with repeating elliptical shapes. Figure 7 shows this approach as described by De Araujo et al. (2011).

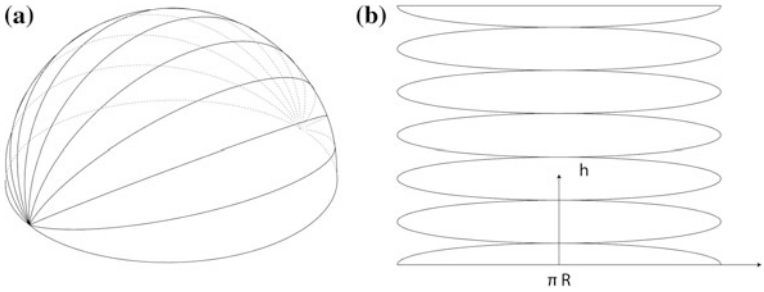

Fig. 7. Knitting approach for spherical form, after De Araujo et al. (2011, pp. 137–170): **a** theoretical 3D form; and **b** repeating knitting pattern

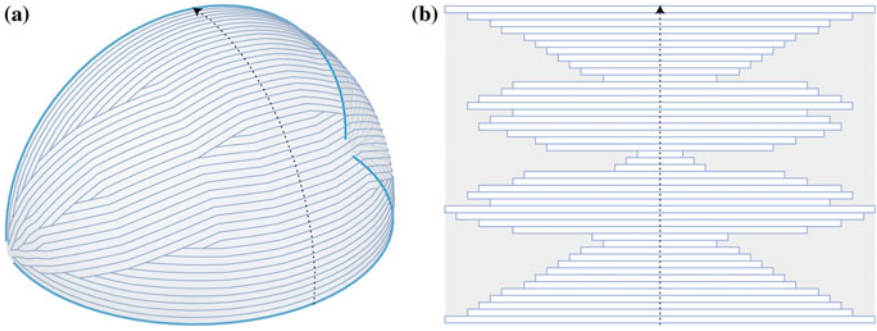

Fig. 8. Knitting pattern for a quarter sphere: **a** generated courses on the quarter sphere; and **b** 2D knitting pattern schematic

Figure 8 shows the results of the presented method applied to a quarter sphere. A large number of short-rows is necessary for creating this geometry. Noticeable is the symmetry and periodicity of the resulting pattern, which is comparable to rationalised patterns in known examples.

Four-Valent Node

This subsection presents the approach applied to a non-developable geometry of a four-valent node. This geometry is one for which knitting patterns as primitives do not readily exist. As the chosen geometry is more complex, all of the steps described in Section "Methods" are followed.

Figure 9a shows the initial patching of the geometry for the creation of the knitting pattern while Fig. 9b shows the patches that were knit as a single piece.

Figure 10a gives an overview of the resulting generated pattern, highlighting, in red, the short-rows used for the shaping of the geometry. Figure 10b shows the schematic course-by-course pattern to be knit for a piece of the node.

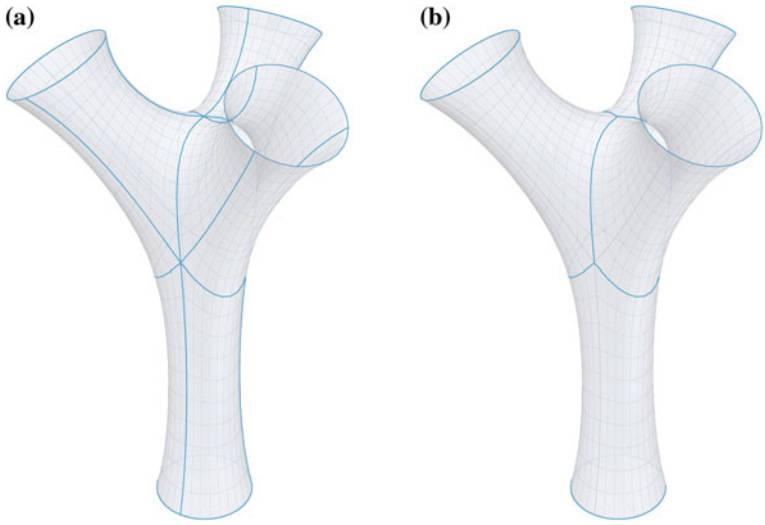

Fig. 9. **a** Patching of the surface for pattern generation; and **b** patches knit in one piece

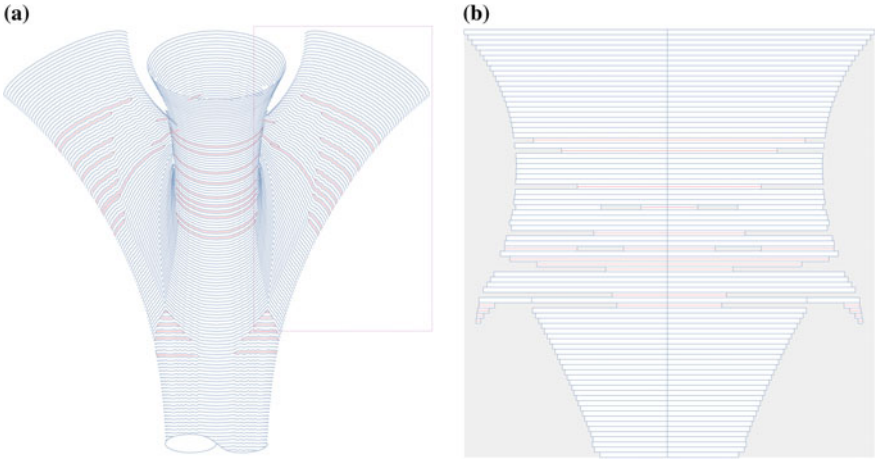

Fig. 10. **a** Courses for the four-directional minimal surface; and **b** knitting pattern for right branch (*box*) of the minimal surface

Figure 11 shows the resulting knit geometry using the generated pattern and highlights the short rows as they appear in the knit piece. Figure 12 shows the geometry as model, generated knit pattern and physical knit piece. Note that the knitted piece is tensioned into shape, which creates a discrepancy between the modelled shape and the knit shape as the modelled shape is patterned without taking into account a tensioned situation.

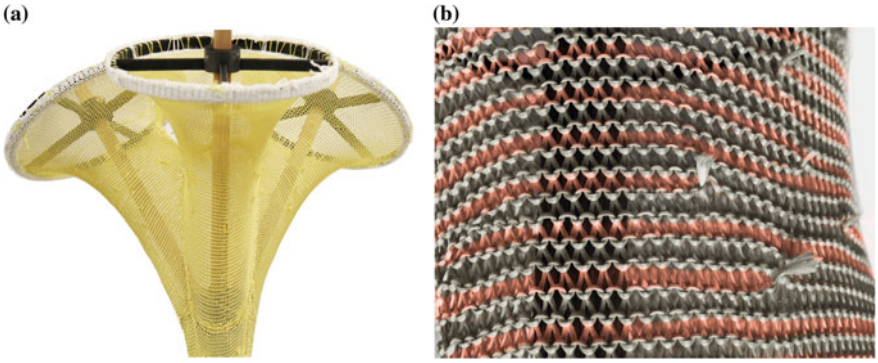

Fig. 11. **a** Physical prototype of minimal surface knit using the resulting knitting pattern; **b** short row features on knit prototype colourised for illustrative purposes

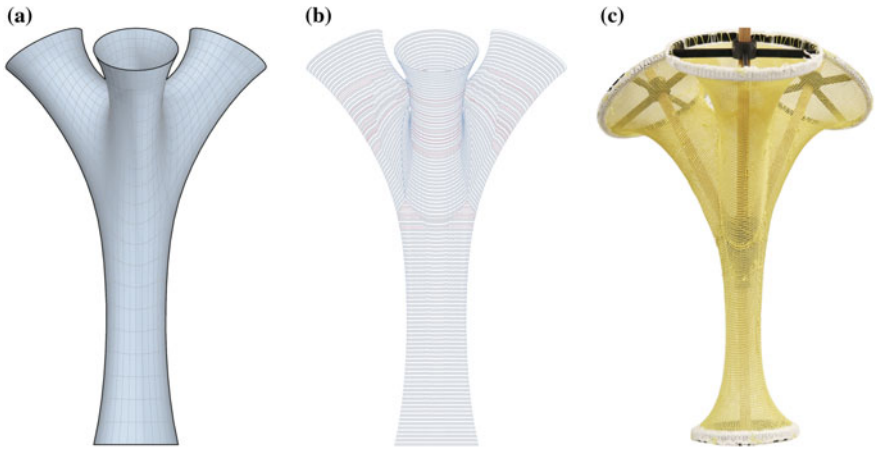

Fig. 12. **a** 3D geometry to be knit; **b** generated knit pattern on 3D geometry; **c** knitted and tensioned geometry

Discussion and Outlook

The presented approach produces knitting patterns from a given 3D geometry with accurate placement of short-rows and loops such that the input geometry can be created. The accuracy of the model in comparison to the real object is highly dependent on accurate measurements of loop geometry, which are directly correlated to the knitting machine parameters. Knowing these parameters, a good draping of the model with minimal loop distortion is achieved. Currently, all results are based on uniform loop parameters over the entire geometry. While it is possible to define varying loop dimensions and alignment for the separate (sub-)patches, loop variation is not possible within a single (sub-)patch.

Future developments will target the creation of more varied patterns with an eye on adjusting pattern densities and alignment to specified parameters and desired directions such as stress fields. The patching of the geometry plays an important role in being able to achieve the desired variations. Possible strategies for automated and optimised patching of a given geometry will be investigated in the future.

Furthermore, considering a tensioned fabric system (Fig. 11), the modelled/patterned geometry and tensioned result differ greatly. If the approach is to be applied to such systems, the inverse problem of computing an initial state given a known target tensioned state needs to be addressed.

On the fabrication side, the approach produces patterns that are in accordance to knitting machine functioning. However, transferring these patterns to the code needed for CNC knitting machines to operate is still a manual process. Specific machining instructions (carriage speed, yarn feeder position) are not part of the generated 2D patterns and need to be input in the knitting machine software. Presently, experiments have been done with ShimaSeiki's SDS-ONE (ShimaSeiki 2017) and will be further developed to create a more streamlined process.

When considering the machining process, further constraints need to be implemented on the pattern generation. For example, for the creation of short rows, needles on the machine are inactivated from one course to another. Depending on the mechanical possibilities of each machine, the recommended maximum number of needles that can be skipped, from a course to another, may be limited. Fabrication constraints such as these will be implemented in the pattern generation.

Acknowledgements. This research is supported by the NCCR Digital Fabrication, funded by the Swiss National Science Foundation. (NCCR Digital Fabrication Agreement#51NF40-141853). Machine knitting experiments and investigation into knitting machine and knitting software working has been done in collaboration with the Institute for Textile Machinery and High Performance Materials at the Technical University of Dresden.

References

Abounaim, M.D., Hoffmann, G., Diestel, O., Cherif, C.: Development of flat knitted spacer fabrics for composites using hybrid yarns and investigation of two-dimensional mechanical properties. Text. Res. J. **79**(7), 596–610 (2009)

Aboumain, Md.: Process development for the manufacturing of flat knitted innovative 3D spacer fabrics for high performance composite applications. PhD dissertation, Technical University of Dresden (2010)

Ahlquist, S.: Integrating differentiated knit logics and pre-stress in textile hybrid structures. In: Design Modelling Symposium: Modelling Behaviour, pp. 101–111 (2015)

Ahlquist, S.: Social sensory architectures: articulating textile hybrid structures for multi-sensory responsiveness and collaborative play. In: ACADIA 2015: computational ecologies, pp. 262–273 (2015)

Brennan, J., Walker, P., Pedreschi, R., Ansell, M.: The potential of advanced textiles for fabric formwork. Proc. ICE Constr. Mater. **166**(4), 229–237 (2013)

De Araujo, M., Fangueiro, R., Hu, H.: Weft-Knitted Structures for Industrial Applications, Advances in Knitting Technology, pp. 136–170. Woodhead Publishing Limited, Sawston (2011)

Hong, H., Fangueiro, R., Araujo, M.: The development of 3D shaped knitted fabrics for technical purposes on a flat knitting machine. Indian J. Fibre Text. Res. **19**, 189–194 (1994)

Igarashi, Y., Igarashi, T., Suzuki, H.: Knitting a 3D model. Pac. Graph. **27**(7), 1737–1743 (2008)

McCann, J., Albaugh, L., Narayanan, V., Grow, A., Matusik, W., Mankoff, J., Hodgins, J.: A compiler for 3D machine knitting. In: SIGGRAPH (2016)

Munden, D.: 26—the geometry and dimensional properties of plain knit fabrics. J. Text. Inst. Trans. **50**(7), T448–T471 (1959)

Python (Copyright 2001–2017) Python Programming Language. https://www.python.org

Ramgulam, R.B.: 3—Modeling of Knitting, Advances in Knitting Technology, pp. 48–85. Woodhead Publishing, Sawston (2011)

Thomsen, MR., Hicks, T.: To knit a wall, knit as matrix for composite materials for architecture. In: *Ambience08*, pp. 107–114 (2008)

Thomsen, MR, Tamke, M., Holden Deleuran, A., Friis Tinning, I.K., Evers, H.L., Gengnagel C., Schmeck, M.: Hybrid tower, designing soft structures. In: Design Modelling Symposium: Modelling Behaviour, pp. 87–99 (2015)

Rhinoceros (Copyright 2017) Rhinoceros Modeling Tools for Designers, Version 5. https://www.rhino3d.com

Van Mele, T.: Compas: framework for computational research in architecture and structures, softwareX (in preparation) (2017)

Van Vuure, A.W., Ko, F.K., Beevers, C.: Net-shape knitting for complex composite preforms. Text. Res. J. **73**(1), 1–10 (2003)

Veenendaal, D., West, M., Block, P.: History and overview of fabric formwork: using fabrics for concrete casting. Struct. Concr. **12**(3), 164–177 (2011)

Sabin, J. http://www.jennysabin.com/1mythread-pavilion (2017)

Scholzen, A., Chudoba, R., Hegger, J.: Thin-walled shell structures made of textile-reinforced concrete. Struct. Des. Constr. **1**, 106–114 (2015)

Schimaseiki: SDS-one APEX3. http://www.shimaseiki.com/product/design/sdsone_apex/ (2017)

Yuksel, C., Kalador, M., James, D.L., Marschner, S.: Stitch meshes for modeling knitted clothing with yarn-level detail. ACM Trans. Graph. **31**(4), 1–12 (2012)