COMMUNICATIONS IN
DEVELOPMENT AND ASSEMBLING OF TEXTILE PRODUCTS

Design tool for automated crocheting of fabrics

Jan Lukas Storck[1,*], Bjarte Alexander Feldmann[1], Yordan Kyosev[2]

[1]*Faculty of Engineering and Mathematics, Bielefeld University of Applied Sciences and Arts, Bielefeld, Germany*
[2]*Chair of Development and Assembly of Textile Products, Faculty of Mechanical Science and Engineering, Institute of Textile Machinery and High Performance Material Technology, Technische Universität Dresden, Dresden, Germany*
*Corresponding author *E-mail address:* jan_lukas.storck@hsbi.de

INFO

CDAPT, ISSN 2701-939X
Peer reviewed article
2023, Vol. 4, No. 2, pp. 254-272
DOI 10.25367/cdatp.2023.4.p254-272
Received: 23 March 2023
Accepted: 05 August 2023
Available online: 03 September 2023

ABSTRACT

In the context of developing a machine to automatically crochet fabrics, a suitable design tool tailored to the new technology and enabling its application is crucial. The paper offers first insights into the prototype of the crochet machine and presents the approach of such a design tool implemented in Python for creating, modeling and generating the machine instructions. With a graphical user interface (GUI), a flat crocheted fabric can be designed by arranging international crochet symbols for slip stitch (SL), single crochet (SC) and half double crochet (HDC). Built-in error checking mechanisms, following the rules of crochet and the machine's constraints, will aid inexperienced crocheters in this process. Based on the resulting computer representation as an array containing short strings for the respective stitches, a topology-based 3D model at the meso scale is automatically created as a preview of the designed crocheted fabric. Also, machine instructions to automatically crochet the fabric with the crochet machine prototype are generated by mapping the computer representation of the stitches to macros of G-code and appending them in a valid order. The straightforward design tool shows the capabilities of the crochet machine and is extensible for further enhancements. Through modeling, the structure of the machine-crocheted fabrics is presented for the first time. In comparison to manually crocheted fabrics, the machine-crocheted ones exhibit a technical front and back, since stitches are formed by the machine only from one side.

Keywords
crochet machine,
CAD,
modeling,
machine control,
crochet,
graphical user interface

1 Introduction

Crochet as a textile technology follows the basic principle of interlooping to mechanically work yarn into a textile fabric [1,2]. In comparison to knitting as the prominent interlooping technology, crochet is defined by the intermeshing of stitches “not only vertically with those in the previous row, as in knitting, but laterally as well — with others in the same row” [3]. Crocheted fabrics are generally used as clothing, home textiles or plush toys, but are so far not applied in the technical field [4,5]. However, research reports promising mechanical [6,7] and acoustic properties [8] and potential applications as textile sensors [9] or scaffolds for tissue engineering [10]. Automation of crochet with a properly accessible design tool for this new type of technology is therefore of high economic interest and could replace the poor working conditions under which commercially crocheted products are currently manually produced.

Contrary to their name, the established crochet (galloon) machines do not produce real crocheted textiles. This is because, as is usual with warp knitting machines, several yarns are processed, and the lateral connection of the stitches is realized via a weft inlay and not via interlooping [11]. Thus, a stitch is not formed according to crochet’s definition by drawing a loop through both the corresponding stitch in the previous row and the last formed stitch in the same row [2,3].

A first attempt of a machine to automate flat crocheting was reported in 2019 [12,13]. Currently, this approach of a real crochet machine is being extensively improved. Similar to single jersey flat knitting machines, the textile is suspended on horizontally arranged latch needles, but additionally, there is a single special needle opposite to this needle bed functioning as a crochet hook. The latter holds and manipulates the leading loop (LL) with which each stitch starts and ends [14]. In this context, it is also necessary to develop a method for designing machine-crocheted fabrics to operate the complex machine and present its possibilities.

Conventionally, manually crocheted fabrics are designed by manufacturing instructions in text form or by symbols representing the stitches on paper. Design tools for manually circular crocheted fabrics considering their construction rules were developed by researchers. Çapunaman et al. [15] propose a computational framework to generate crochet patterns corresponding to 3D objects as inputs, which can be designed with common computer-aided design (CAD) tools. By considering the individual style of the crocheter based on analyzed crochet swatches, instructions are output to crochet the 3D objects. With an alternative approach, Guo et al. [14] also compute text-based instructions for crochet patterns based on input 3D geometries. In this regard, stitches are represented by tiles, which are arranged automatically, to model and visualize the 3D textile to be crocheted. Besides the visualized mesh, instructions for manual crocheting are generated. Furthermore, Nakjan et al. [5] created a tool to specifically design 3D crocheted dolls (Amigurumi) by 2D sketches interpreted as 3D primitives (sphere, tear drop or cylinder), which are then compiled into crochet instructions.

Regarding the design of machine-produced textiles, commercial design systems are commonly used for versatile V-bed weft knitting machines [16]. M1plus from Stoll and KnitPaint from Shima Seiki are prominent examples of such and provide pixel-based programming interfaces [16-18]. In a tabular representation (columns for needles and rows for subsequent knitting cycles), stitches and operations represented by icons can be arranged graphically to design a textile fabric. Further visualizations, such as a simple running yarn notation or an elaborated 3D simulation of the yarn and the machine operations (in the case of KnitPaint), as well as functions for automatic error detection, simplify the development of knit programs [16,18,19]. Because these systems are predominantly machine-specific, research is conducted to enable high-level programming by manipulating 3D objects and focusing on the output structure independent of the specific machine [14,17,18,20].

3D previews of the designed textile and its patterns are state of the art and facilitate the development process [21,22]. Describing an idealized spline-interpolated yarn center path, taking into account the correct topology (relative orientation of the yarn segments to each other) at the meso scale, is a simple and flexible approach for generating such a preview [4,21,23,24]. This approach is, for example, implemented in the commercial Warp Knitting Pattern Editor 3D from TexMind [25]. With this program, warp knitted textiles can be designed by creating lapping diagrams for warp knitting machines with a graphical user interface (GUI) [26,27].

Following a similar approach, manually crocheted fabrics were modeled and subsequently used for finite element method (FEM) simulations in a recent publication [4]. These demonstrated anisotropic behavior and a uniform force distribution during displacement of the manually crocheted fabrics.

Here, a straightforward and extensible tool for designing machine-crocheted planar fabrics according to the current prototype of the first real crochet machine is presented. Such a tool is necessary because existing tools are not applicable to this new type of textile machine and, in addition to the hardware, software that enables the use of the machine must also be made available to potential users. In this context, international stitch symbols can be arranged with a GUI to create a fabric to be crocheted consisting of the basic stitch types, i.e. slip stitch (SL), single crochet (SC) and half double crochet (HDC). For convenience of use, there are functions for automatic error checking and ensuring that the designed fabric follows the rules of crochet and can be fabricated by the machine. The description of machine-crochetability is an important innovative aspect. From the computer representation, on the one hand, the G-code for controlling the machine is derived, and on the other hand, a 3D preview is generated by a topology-based model with parametric key points. The generated G-code can be loaded directly into the crochet machine to produce the designed crocheted textile. Describing the structure of machine-crocheted stitches is another original aspect.

2 Crochet machine design tool

The developed design tool for flat machine-crocheted fabrics is presented in this section. Firstly, the prototype of the novel crochet machine is introduced, then the GUI and error checking of the tool are described. Also, the approaches of generating the topology-based model as well as the G-code machine instructions are introduced.

2.1 Structure of the crochet machine

The developed tool is based on the specific capability of the prototype crochet machine currently in progress, which is a further development of the initial approach presented in references 12 and 13. The schematic setup of the crochet machine is depicted in Figure 1. Opposite a needle bed (1) with needles (2), on which the lastly formed crochet stitches are suspended, another special needle (3) is arranged. An individual needle selection is achieved with mechanic multiplexing by carriage 4. Needle 3 functions as the crochet hook, which forms the crochet stitches and always retains the current leading loop of the crochet process. This special needle in the carriage 5 can be driven to each stitch position according to the needle bed (currently 18 positions) to build there a new crochet stitch based on an old one. As another fundamental machine element, a yarn guide (6) moved by carriage 7 is involved in stitch formation by feeding the yarn into the special needle. Weights serve as a take down and stretch the fabric over a knock-over verge (8).

As can be seen in Fig. 1, the basic setup of the crochet machine prototype needs various axes for the complex crochet motion sequences. Thus, a mechatronic approach was chosen in contrast to the rather mechanical one of conventional textile machines [19]. In this regard, the machine's electric motors are controlled with G-code commands, similar to 3D printers or computerized numerical control (CNC) milling machines [13].

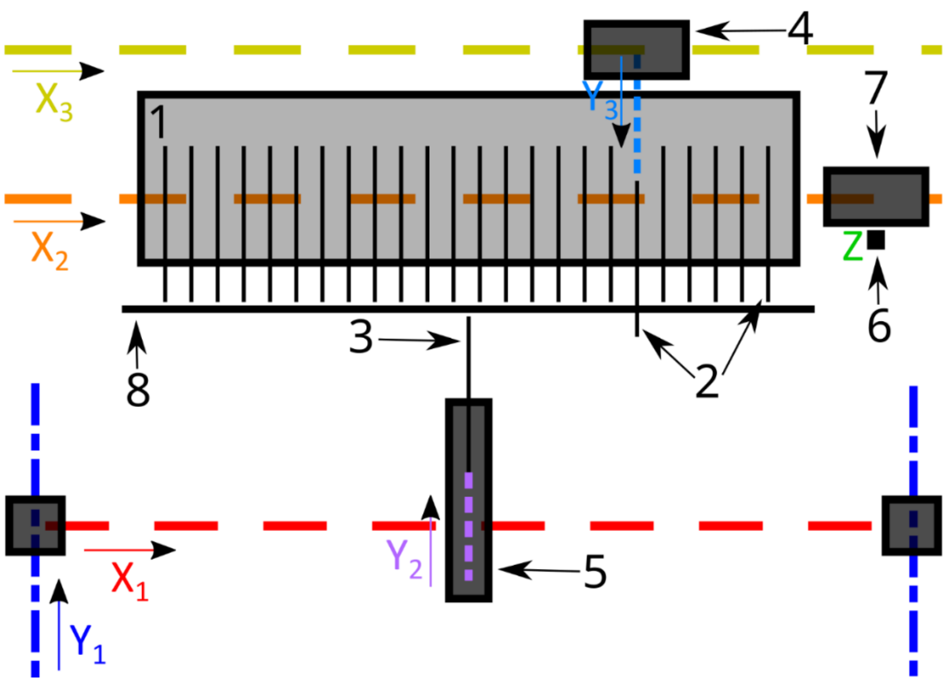


Fig. 1 Schematic structure of the crochet machine prototype viewed from top with indicated main axis. 1) needle bed, 2) needles, 3) special needle, 4) individual needle selection carriage, 5) special needle carriage, 6) yarn guide, 7) yarn guide carriage and 8) knock-over verge.

Due to the prototype status of the machine, no details can be shown here. Thus, the principle of the stitch formation is presented in a more general way in Fig. 2. In this example, the working stitch and its right neighbor were built in the previous row from right to left and are still suspended on the needles of the needle bed in the regions marked by the red dots. The LL of the last stitch of the current row (going to the right) is held by the special needle. To build a new SL, yarn is drawn through the working stitch from behind the fabric by the special needle, which establishes the vertical anchoring of the new stitch. This step is depicted in Fig. 2 b) as well as the yarn end going to the yarn stock. The newly created loop is then also pulled through the previous LL, which is the horizontal stitch anchoring and establishes a new LL, as shown in Fig. 2 c). The newly created SL is then suspended at the respective stitch position to finish the machine operation.

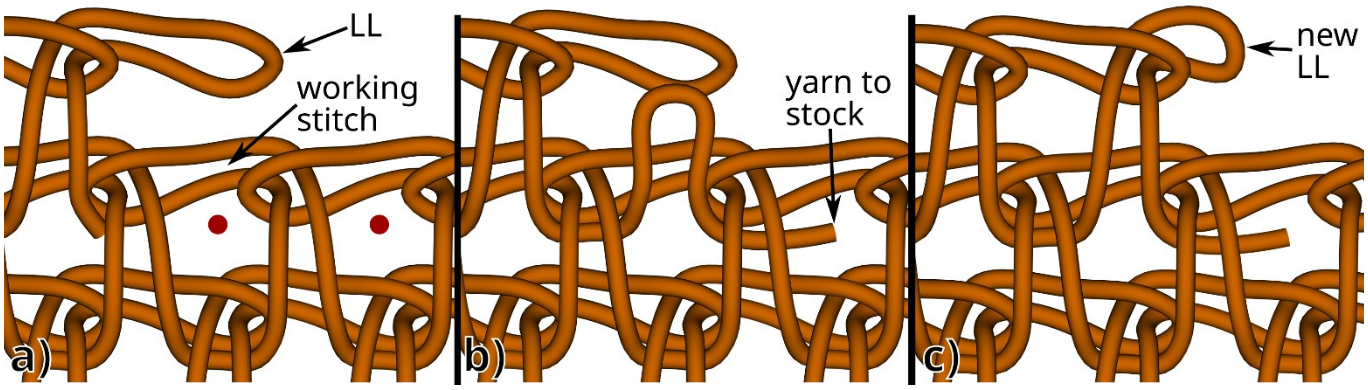


Fig. 2 Principle of creating a SL with the crochet machine. (a) Initial situation with the leading loop (LL) held by the special needle and the working stitch as well as the neighboring stitch suspended on the needles of the needle bed in the region marked by the red dots. (b) New yarn forming a loop and being drawn by the special needle through the working stitch. (c) Afterwards the loop is drawn as the new LL through the old one.

Creating a SC necessitates an additional step of drawing a second new loop through the first loop drawn through the working stitch (cf. Fig. 2 b)). This second loop is not drawn through the fabric but is drawn through the old LL after it has been drawn through the first loop. For HDC, as the most complex stitch the machine can currently create, an additional loop, not drawn through the fabric, is grabbed by the special needle before the steps of a SC are performed. The resulting stitch structures are depicted in Fig. 6 and 7.

In the recent version of the program, fabrics consisting of SL, SC and HDC can be created in arbitrary order. In general, the crochet machine is designed to automate flat crocheting and cannot crochet in the round. Therefore, only two-dimensional fabrics can be created and are in the focus of the design tool. Currently, stitch transfer is not possible but might be added in the future. Also, crochet through the back loop is not possible.

Due to the lateral interlooping of crochet stitches (cf. Fig. 2), the alternating crochet direction within each course must be considered. By default, the crochet direction of the first course with the foundation chain stitches (CHs), is set from right to left so that the stitches of the second course are assembled from left to right. CHs are also used for transitions, with which each course starts (except for the CH course). According to the height of the next stitch, one CH is used as a transition (T1) for SL and SC while two (T2) are necessary for HDC. The CHs are formed in the direction of the previous course and then placed at the corresponding needle positions of the course's start. In contrast to the other stitches, the loops of the CHs are pulled through only one instead of two previously formed stitches [2]. Thus, CHs do not count as crochet stitches, hence they can be found in other technologies such as a pillar stitch in warp knitting [28].

2.2 Graphical user interface and error checking

A pixel-based, machine-specific approach was chosen to correspond to the industrial standard programs [16-18] and therefore potential users are in principle familiar with the interface. Also, its tabular structure suits well the two-dimensional fabric structure. The existing approaches of design tools regarding manual crochet [5,14,15] cannot be adopted, because they deal with circular crocheting. Circular crocheting differs significantly from flat crocheting and the machine implementation also imposes some deviations to manual crocheting. The presented tool is implemented in Python 3 and built as cross-platform software.

The chosen approach of representing machine-crocheted fabrics with international crochet symbols, considering the crochet machine's operation, is presented in Fig. 3. In contrast to the alternative representation of crocheted fabrics with transitions outside the stitch columns, used for instance in reference 4 or 14, here, the construction of the fabric is more concisely represented, since it is intuitively comprehensible how the stitches are intermeshed with those above and beneath and on which needles they can be formed. The intermeshing is indicated in Fig. 3 by the red arrows, which also illustrate the stitch formation by drawing loops both from the previous stitch in the same course and from a stitch in the course below at the same wale. Consequently, wales correspond laterally to needle positions and courses chronologically to the stitches formed successively at the same needles. As usual, the top course corresponds to the last produced course. The first transition from course 1 to course 2 is a special case and does not have any stitches beneath it. This is due to the workflow of manually crocheting the first CH course onto the needles of the machine and putting the leading loop in the crochet machine's hook before the automated production starts.

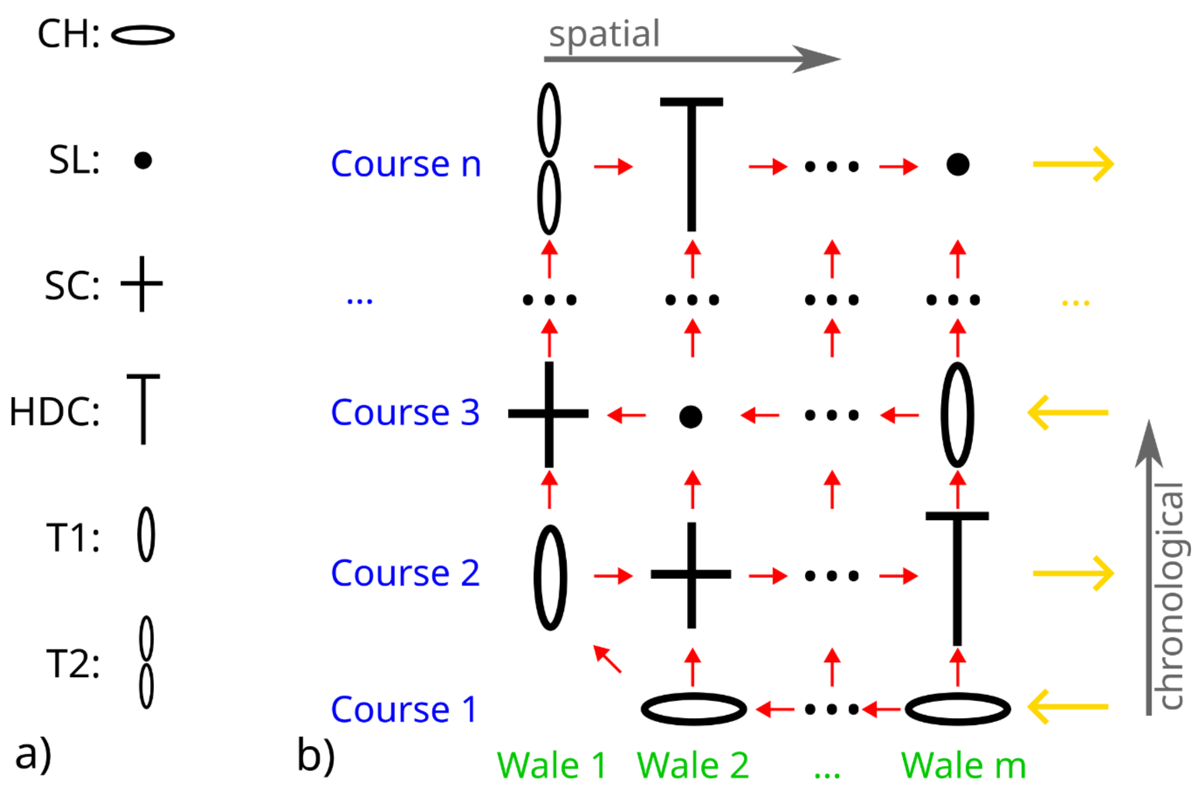


Fig. 3 (a) Description of the symbols used and (b) illustration of the representation of crochet fabrics in the GUI with exemplary stitches. Yellow arrows denote the crocheting direction of each course and red arrows show the connections of the stitches according to the crochet procedure. The ellipsis indicates further possible courses and wales. The needles of the machine are associated with the wales.

In addition to color differences, the international crochet symbols of the Craft Yarn Council, which provides guidelines and standardization for crocheted textiles [29], are used to label stitches in the developed GUI (cf. Fig. 3). This is, in principle, similar to conventional pixel-based programming interfaces [30]. A stitch type can be selected via the toolbar and a position in the fabric can be assigned by clicking on the respective tile. Stitches can be erased with the “blank tile” tool. The needles on which the stitches of the wales are created are defined by the needle indicator (NI) at the bottom of the GUI. According to the wale position, a stitch is automatically assigned to a specific needle position in the machine. This allocation based on the topology also remains in the array data structure. During production, stitches cannot be moved to different needles by the machine, so that no specific algorithms for needle scheduling or transfer planning are needed, which are for example discussed by McCann et al. [17] or Lin et al. [31] regarding knitting machines. However, the user can change the needle allocation in the GUI.

Besides common options, like loading and saving, the export to G-code and the model generation can be selected in the dropdown menu. The yarn tension can be set in the menu “Edit”. Fig. 4 depicts the developed GUI, which shows the technical front, with an exemplary fabric.

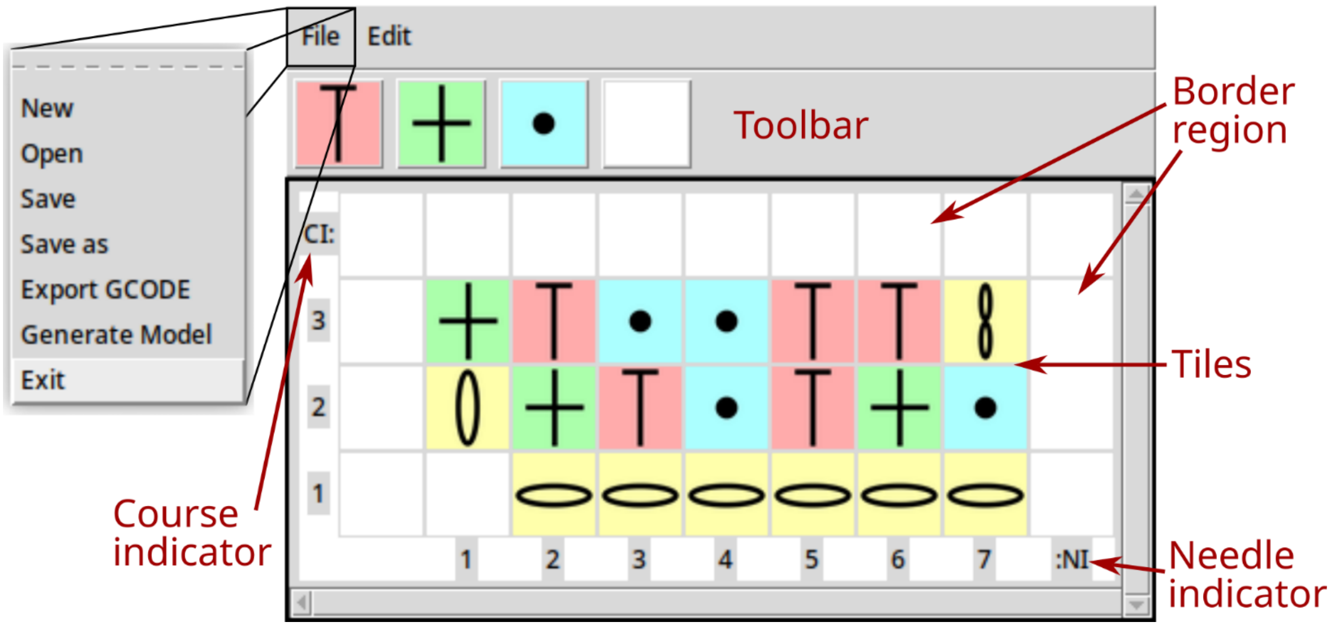


Fig. 4 Overview of the developed graphical user interface (GUI) with the computer representation of an exemplary crocheted fabric. The technical back of a topology-based model of the designed fabric is displayed in Fig. 8a.

The GUI is built using the Python version Tkinter [32] of the open-source GUI toolkit Tk. Stitch selection at the toolbar is implemented with buttons, while the tiles in the canvas region, which the stitches can be assigned to, are label widgets. By clicking on these modifiable labels, the internal representation in the data array is changed to the selection if that operation is determined to be valid by the program. This data array matches spatially the data displayed in the canvas region. However, to facilitate implementation, the indices of the data array (i and j, cf. Fig. 5 and the Python pseudocode in the supplementary materials) differ from the displayed NI and course indicator (CI) in that they start from zero and the courses are counted from top to bottom. NI and CI cannot be modified directly, instead, they are automatically designated or can be set manually in the "Edit" menu. By clicking the border regions surrounding the tiles, the canvas is automatically expanded in the direction the border region is facing and CI/NI are updated accordingly. For example, by clicking the left border region, a column of modifiable labels is added in between the left border column and the inner area containing the tiles, while the NI would be adjusted. If a row or column next to the border region contains no more stitches, it will be deleted automatically.

With the current state of the crochet machine prototype, there are the following constraints compared to manual crocheting:

- Only production of SLs, SCs, HDCs and transitions with one as well as two CHs are possible (constraint **a**)
- First course of CHs is to be manually crocheted from right to left with no CH beneath the first transition called first lay over (FLO) (**b**)
- Second course is always crocheted to the right by the machine (**c**)
- Fixed width, no increase or decrease (**d**)
- A stitch or transition must have a previously formed stitch or transition beneath (with FLO as exception) (**e**)
- Before each stitch in a course (according to the direction) has to be a previously created stitch or transition (**f**)
- Before a start position of the course, which has to be a transition, there must no other transition or stitch (**g**).

These machine specific constraints are checked by the program to aid users in designing crocheted fabrics with the GUI. A section of the program for error checking and ensuring that the designed crochet pattern is producible with the machine is shown as Python-based pseudocode in the supplementary material. Compliance checking with regard to the listed constraints **a** to **g** is indicated in the pseudocode accordingly.

In principle, error checking is based on completely traversing the array with the crochet pattern (as shown in Fig. 5) once, checking for any possible error that conflicts with the machine-specific constraints and general crochet rules. Thereby all possible errors can be found, which are considered. Due to the ongoing development of the machine, the error checking does not claim to be absolute. According to the machine-specific and general crochet rules reflected in the error checking (see also the supplementary materials), the machine-crochetability is defined.

Regarding the verification of compliance with important general crochet rules, it is, for example, checked that the crochet direction alternates with the courses or that HDCs follow on T2 while SLs or SCs follow on T1. The latter is implemented by inspecting the parent stitch beneath the transition and the target stitch (next stitch in crochet direction). Usually in manual crochet each course starts with a transition, nevertheless, this rule is additionally reflected in constraint **g** because it is a strict limit of the machine.

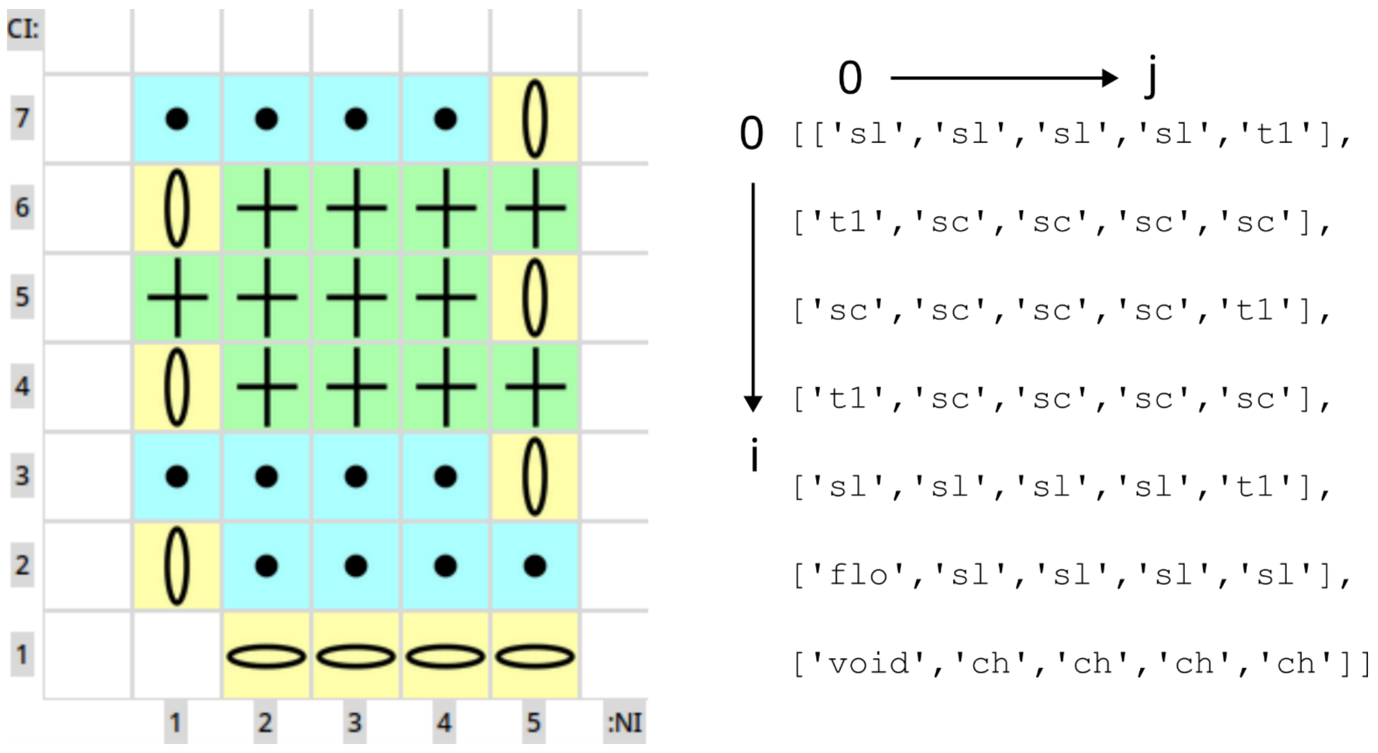
Constraint **g** is checked, as can be seen from the Python-based pseudocode shown in the supplementary material, by calculating the correct position of the transition of each course. There can only be one transition per course at this position. Also, everything that is not a void and is positioned before the transition (according to the crochet direction) is recognized as an error. The locations of the transitions correspond to the alternating crochet direction of each course. This pattern is based on the definition that the transition of the FLO is on the left and the corresponding course heads to the right. Thus, this complies with the constraint **c**. Furthermore, this also results in the direction of the first CH course to the left, which is a part of constraint **b**.

Regarding **b** and the special case of the first transition as the start of the automated production, it is controlled that there is no CH but a void in the crochet pattern under FLO. In general, as part of the error checking and for compliance with constraint **e**, it is checked for each stitch that there is a parent stitch in the course beneath it. Here the CH course is an exception, because it is the first course with no parent stitches.

The CH course is also taken as the basis for the fixed width of the fabric (restriction **d**) by checking that there are no more stitches in the other courses than the number of CHs + 1 (considering the special case of the missing CH under the FLO). Together with the calculation of the correct positions of the transitions, this ensures that the designed fabric has a constant width.

Moreover, constraint **f** is respected by searching for voids between stitches within a course. If there are voids between stitches, the rule is broken that before a stitch there must be previously formed stitch or transition, and an error is raised. Generally, the errors are marked by a text output and by a red border around the corresponding tile. Error resolving is currently left to the user to avoid correcting an error contrary to the user's intent when there are multiple causes. The user can also start the error checking at any time by clicking an option in the edit menu.

Compliance with constraint **a** is ensured by providing only the stitch types that can be created by machine for selection (cf. Fig. 4). Thus, algorithms for error checking are not needed in this case. As a further method for error prevention, the CHs of the first course are automatically set depending on the user assigned stitches in the second course. Also, a transition is automatically set when a stitch is assigned to a previously unoccupied tile above the existing stitches, whereby a new course is instantiated.



```
[['sl','sl','sl','sl','t1'],
 ['t1','sc','sc','sc','sc'],
 ['sc','sc','sc','sc','t1'],
 ['t1','sc','sc','sc','sc'],
 ['sl','sl','sl','sl','t1'],
 ['flo','sl','sl','sl','sl'],
 ['void','ch','ch','ch','ch']]
```

Fig. 5 Computer representation of a crocheted fabric graphically illustrated in the GUI at the left side and in an array with stitch label strings as the basic data structure at the right. Regarding the latter, the indices i and j of the internal data structure correspond to the iteration directions of the pseudocode in the supplementary material and are displayed in contrast to CI and NI. The special first transition is called first lay over (flo). The model generated from this information is shown in Fig. 9.

The structure of the designed machine-crocheted fabric is saved in a text file. Each stitch or transition is labeled with a short string and the topology is maintained by saving the courses in rows of an array, whereas the columns correspond to the stitch sequence of the GUI representation. An example of such an array representation of a crocheted fabric together with the respective GUI is depicted in Fig. 5. This simple representation contains all the necessary information about the fabric's structure. The used string labels correspond to the standard international crochet stitch representation in text form [29]. These strings can be mapped via dictionary data structures to G-code macros for automatically producing the corresponding elements or to the key point representation of the unit cell for modeling.

The array as the basic data structure for the design of automated crochet fabrics was intentionally created to be simple and clear. This allows experienced users, who are not dependent on the support of the GUI, to design crocheted fabrics with a simple text editor while benefiting from the preview in the form of modeling as well as from the automated generation of the G-code program for machine production.

2.3 Preview with topology-based crochet model

According to the topology-based modeling approach, the yarn path is simplified in that the exact geometry is not considered, but rather the relative orientation of the segments to each other [21,23]. The center yarn path at the meso scale for each stitch and transition is represented by a set of parameterized key points in a NumPy array, which can be seen as a unit cell. By virtually shifting and assembling the unit cells, a fabric can be created, which entails modifying the key point coordinates and appending the key points in one list in the correct order. A realistic representation can be achieved by spline-interpolation and volume sweeping along the yarn path. The freeware program TexMind Viewer [33] can be used for visualizing the modeled crocheted fabric. For storing the digital representation of the crocheted fabric, the open access Python library pytexlib [34] is used. Further details on the modeling approach can be found in reference 4 where manually crocheted fabrics were modeled. The present paper is based on this preliminary work. However, due to the different structures of manually and machine-crocheted fabrics, new unit cells had to be defined and the logic was extended.

The width and spacing of all stitches in the x-direction (L) is with the current prototype about 5 mm due to the pitch of the needle bed. For modeling purposes, a realistic yarn diameter is approximated to be 0.6 mm. Unit cells of different stitches can be shifted horizontally with multiples of the same translation vector to assemble a course. Unit cells shifted to form a course of three stitches of each type with a crochet direction to the left are shown in Fig. 6. The vertical spacing of the courses (y-direction) depends on the stitch height, which is different for distinct stitches and is influenced by the yarn tension. Hence, the stitch height can be adjusted with the yarn tension factor (YTF) according to equations 1 to 3, where H_{sl}, H_{sc} and H_{hdc} are the heights of the SLs, SCs and HDCs, respectively (cf. Fig. 6). L denotes the stitch's width, which the height is based on for realistic stitch size.

$$H_{sl} = YTF \cdot 1.25 \cdot L \tag{1}$$

$$H_{sc} = YTF \cdot 1.5 \cdot L \tag{2}$$

$$H_{hdc} = YTF \cdot 1.75 \cdot L \tag{3}$$

If different stitch types are in a course, the height of all stitches is adjusted to the highest type in that course. This is because, on the one hand, the machine's take-off elongates the stitches evenly and, on the other hand, a uniform stitch height is needed for a suitable connection to the next course in the modeling. The translation vector for vertical shifting depends on the height of the previous course.

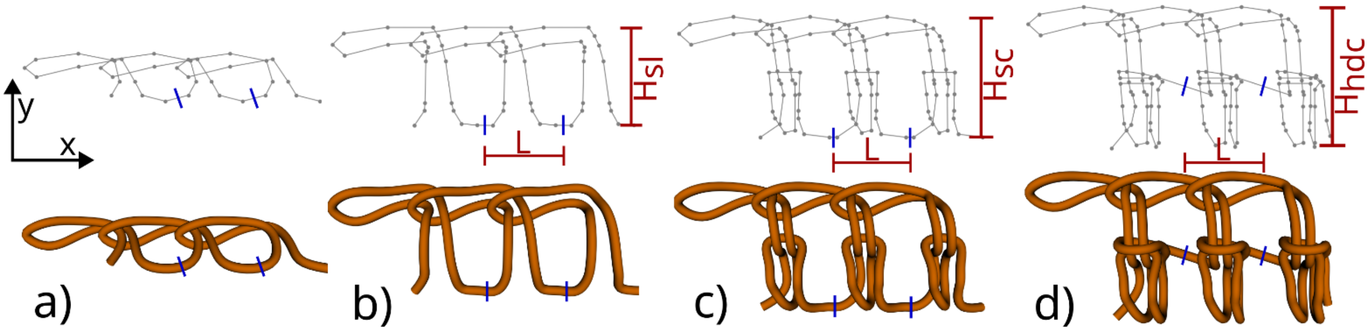


Fig. 6 Unit cells of the machine-crocheted stitches as key point representation with the coordinate system (top) as well as spline-interpolated and volume-swept models displayed with the TexMind Viewer (bottom). The blue lines separate the three unit cells shown as a course in each case. The heights and widths of the stitches are indicated. (a) Chain stitch (CH); (b) slip stitch (SL); (c) single crochet (SC); (d) half double crochet (HDC). Note that the CHs are not directly in the x-y-plane, but slightly tilted to get a more realistic interlooping to the following course.

To model the fabric, the GUI output array containing the topology and string labels for stitches and transitions (cf. Fig. 5) is iterated beginning with course 1 in wale m. Depending on the course's direction, the strings are mapped to left or right-pointing variants of the unit cells of the stitches. According to the topological position of the string labels in the array, corresponding translation vectors are added to the key point coordinates of the unit cells. Also, the key points are modified to match the required stitch length and height. The adjusted key point coordinates are appended in a monolithic list following the crochet sequence and iteration of the array. Regarding the assignment of the correct unit cell for a transition, the succeeding and previous stitches are considered to ensure appropriate intermeshing. The final key point list and yarn diameter are saved in comma-separated value (CSV) files, which can, for example, be opened with the TexMind Viewer for spline-interpolation and visualization.

2.4 Generating machine instructions

As stated in section 2.1, the prototype machine is controlled by G-code commands. In general, for forming a stitch, multiple G-code commands have to be executed. Relying only on a simple text or CSV data structure containing G-codes would require multiple lines to be changed for modifying a single stitch. In contrast to that, it is much simpler to work on an abstract data representation (the data array, as in Fig. 5 right side). In this case, to modify a single stitch only one cell in the data array has to be changed, additionally the data array preserves the topological information of the stitches. By doing so, however, it is crucial to have an efficient way to translate the abstract computer representation of the fabric to machine-executable commands (G-codes).

For this translation, the GUI output array containing the information about structure and topology (cf. Fig. 5) is used. Analogous to the generation of the topology-based model, the array is traversed according to the crochet sequence and, depending on the direction of each course, the exact stitch type (such as SL to the left or T2 to the right) is assigned as a string for each entry of the GUI output array. These specific strings are mapped utilizing dictionary data structures to text files containing the required G-codes for each stitch type as macros.

Starting with the second course (due to the manual building of the first CH course), the G-code instructions read for each stitch are appended consecutively and the result is written into a new text file. To save computing time, the G-code text file for each stitch type is read only once and stored in a dictionary, from which the data for subsequent stitches of the same type can be obtained. The G-code macros are programmed relatively and thus the absolute position of the machine elements does not have to be considered. This allows arbitrary sequence of stitches without needing to adjust the G-codes in the macros.

Advantageously, the abstract representation can also be more readily edited, more efficiently saved and more easily checked for errors. Especially the last part greatly simplifies the error checking algorithm, since to check one stitch a single entry of the array can be queried instead of multiple lines of G-code. All in all, this approach ensures machine-manufacturability but also improves the overall workflow.

Because the tool operates on the stitch level, a flexible interchangeability of the macros with the machine instructions is ensured. On the one hand, this is advantageous regarding the machine under development, and on the other hand, it enables the design tool to be used for alternative crochet machines in the future. The stitches could also be mapped to text files with instructions for manual crochet to generate crochet patterns in text form.

The G-code generated by the presented tool can be directly executed by the prototype crochet machine to produce a corresponding crocheted fabric. Currently, the freeware program cncjs [35] is used to send the G-code commands from a laptop to the crochet machine via an USB interface. Alternatively, the machine can receive the G-code via a memory card, similar to the typical use of a 3D printer.

3 Results and Discussion

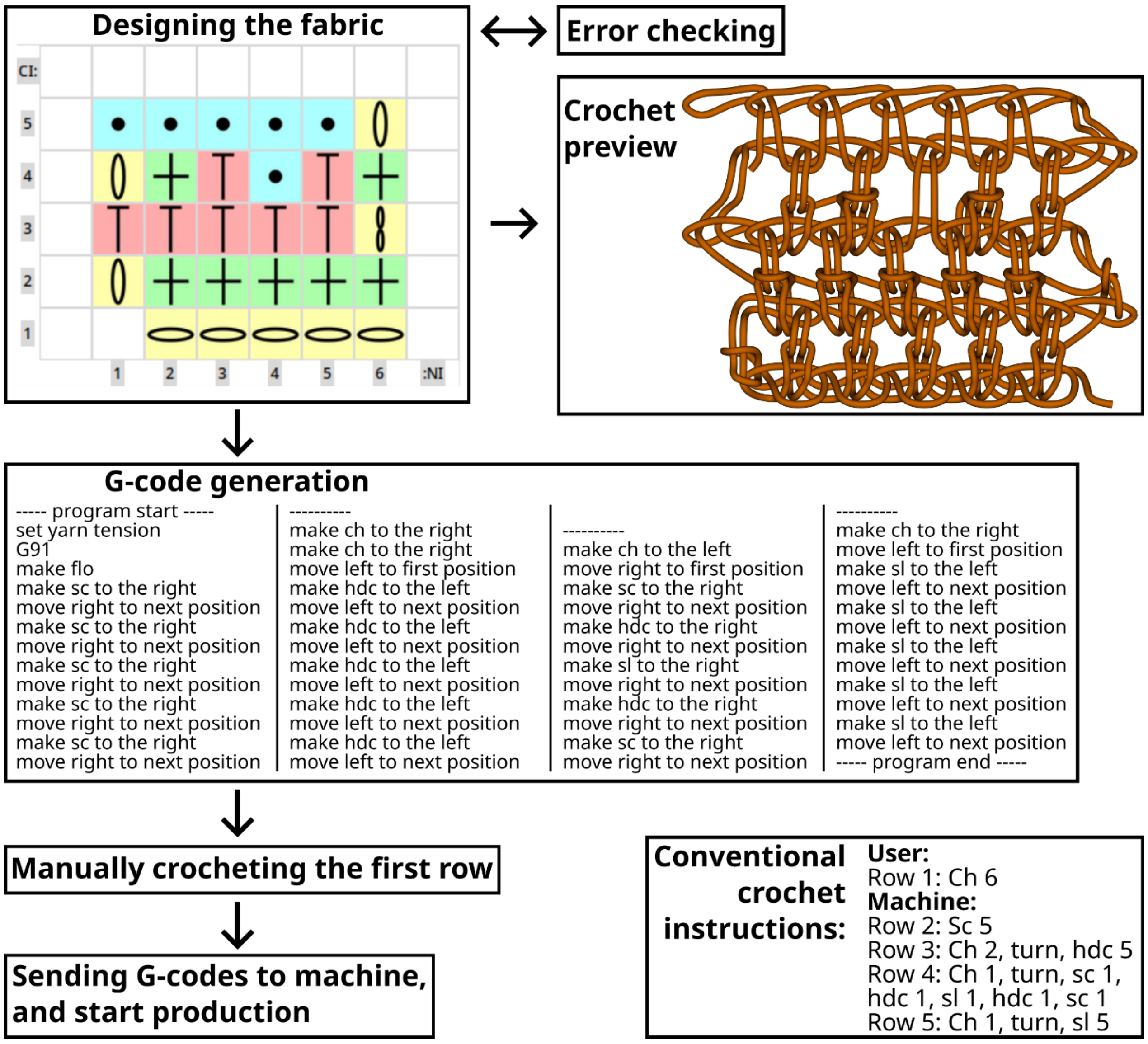
In the following, the workflow of the design tool is presented using an exemplary application and the approach is discussed. The structure of the crocheted fabrics automatically produced with the crochet machine is also compared with that of conventionally hand-crocheted fabrics, based on models.

3.1 Application of the tool

Since the program checks the validity of operations and patterns during the design, the designer can focus solely on creating the desired structure of the fabric. This facilitates the swift creation of fabric even for inexperienced crochet designers. During the process, an error check can be performed at any time highlighting errors for the user. Similarly, a topologically correct 3D preview can be created at one's convenience to be displayed with the freeware program TexMind Viewer. The CSV files of the models and the structure of the GUI can be saved for later use or rework. Once the design process is complete, the G-code can be generated automatically and used directly for production with the crochet machine. For this workflow, which is illustrated in Fig. 7, only the TexMind Viewer for the visualization of the 3D models and a program like cncjs for sending the G-codes are needed as external programs, both are available as freeware.

The pseudocode shown in Fig. 7 represents G-code macros for stitch formation. Due to the course direction, a distinction is made between stitches going to the left and stitches going to the right.

According to the structure of the textile, the macros of the stitches are repeated. For an easy understanding of the necessary production steps of the designed textile, conventional crochet instructions in text form are also given in Fig. 7. As shown in the workflow, the user must manually crochet the first row consisting of the CHs onto the machine needles before machine production can begin. Thereby, the sixth CH is the T1 of the second course.



Designing the fabric
CI:
5
4
3
2
1
1
2
3
4
5
6
:NI
Error checking
Crochet preview
G-code generation
Manually crocheting the first row
Sending G-codes to machine, and start production
Conventional crochet instructions:

```
----- program start -----
set yarn tension
G91
make flo
make sc to the right
move right to next position
make sc to the right
move right to next position
make sc to the right
move right to next position
make sc to the right
move right to next position
make sc to the right
move right to next position
----------
make ch to the right
make ch to the right
move left to first position
make hdc to the left
move left to next position
make hdc to the left
move left to next position
make hdc to the left
move left to next position
make hdc to the left
move left to next position
make hdc to the left
move left to next position
----------
make ch to the left
move right to first position
make sc to the right
move right to next position
make hdc to the right
move right to next position
make sl to the right
move right to next position
make hdc to the right
move right to next position
make sc to the right
move right to next position
----------
make ch to the right
move left to first position
make sl to the left
move left to next position
make sl to the left
move left to next position
make sl to the left
move left to next position
make sl to the left
move left to next position
make sl to the left
move left to next position
----- program end -----
```

User:
Row 1: Ch 6
Machine:
Row 2: Sc 5
Row 3: Ch 2, turn, hdc 5
Row 4: Ch 1, turn, sc 1, hdc 1, sl 1, hdc 1, sc 1
Row 5: Ch 1, turn, sl 5

Fig. 7 Workflow of using the developed design tool in an exemplary application. First, the fabric to be crocheted automatically is designed using the GUI and error checking, which ensures the manufacturability. Then a 3D model as preview of the crochet structure is generated and visualized with the TexMind Viewer. The G-code represented as pseudo code is generated automatically. Before the production starts, the first row consisting of CHs must be crocheted manually on the machine.

Modeling of the maximal yarn tension is restricted by the minimal stitch height, which ensures that all possible models are free of intersecting yarn segments. As default for the stitch height, a YTF of 1 is used. This default is compared to higher stitches representing lower yarn tensions in Fig. 8. As can be seen, the vertical distances between the interlooping regions, which are not altered to prevent intersections, are increased for higher stitches. Also, the CH course's height is not influenced because it is not produced by the machine. In the future, the exact relation between yarn tension during automated crocheting and resulting stitch height is to be investigated based on automatically crocheted samples with an improved crochet machine.

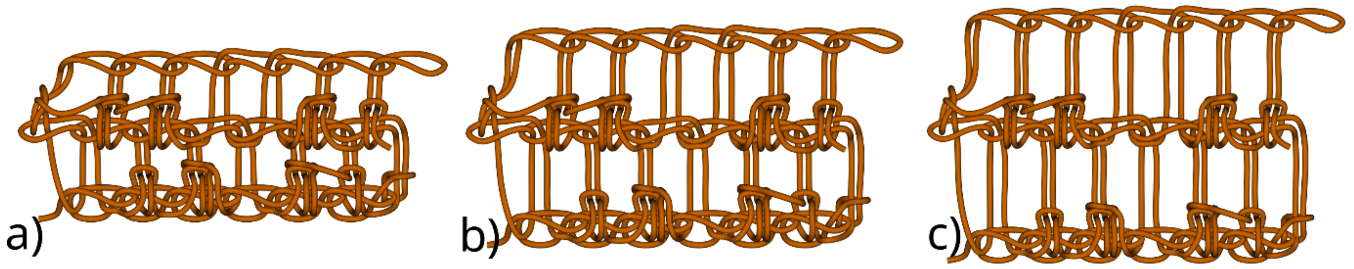


Fig. 8 Comparison of different stitch heights to enable modeling of different yarn tensions during manufacturing. The modeled fabric consists of all available stitch types and its structure is depicted in Fig. 4. Note that here the technical back is shown. (a) Lowest intersection free stitch height with yarn tension factor (YTF) of 1 used as default; (b) YTF of 1.25; (c) YTF of 1.5.

The concept of the presented approach of the first design tool for a true crochet machine is, in principle, similar to commercial design tools for knitting (such as M1plus or KnitPaint) with pixel-based programming, preview and error detection. However, the range of functions is much more limited, which is in particular due to the prototype status of the crochet machine. According to the limitation of the machine to only produce flat fabrics, the approach differs from tools presented in scientific literature to design 3D textiles for manual circular crocheting [5,14,15]. Thus, in contrast to the literature, 3D objects cannot be processed to represent them with crocheted stitches and generate instructions on how to crochet them. The processing of two-dimensional shapes in order to automatically calculate the arrangement of crocheted stitches is only reasonable when the functionality of the prototype has been extended by increase and decrease stitches and their specific geometries are known. It is planned to extend the tool with this and generally according to further developments of the machine. Moreover, the tools from the literature [5,14,15] refer to crochet in the round and not to planar crochet, which is automated by the crochet machine. However, similar to the work of Guo et al. [14], a 3D model of the crocheted textile is also created. Similarities to Çapunaman et al. [15] are that the intrinsic characteristics of the production process are taken into account (here needle spacing and yarn tension in contrast to individual crochet technique) and instructions for production are generated.

Compared to these crochet design tools from the literature [5,14,15], the goal of the tool presented here is not to design or efficiently construct 3D crocheted objects, but to enable the design of machine-made planar crocheted fabrics in an industrial context. The problem of the future application of a complex machine that automates a technology little known to the industry is solved by developing a stringent and extensible methodology for the design and control of the machine based on familiar tools. Without a dedicated design tool, the acceptance and application of the new machine is not given, because conventional tools and methods of driving textile machines cannot be applied here. The concise structure of the GUI with international crochet symbols and error checking allows users who are unfamiliar with crochet to easily develop a fabric to be crocheted by machine.

With the modeling it is possible to rapidly generate a preview, which is advantageous in terms of design processes [21,36]. Especially regarding crocheted fabrics, which are rather unknown to potential designers in the technical field, this is of great advantage. In addition to visualization, the automatically generated models can also be used for further simulative investigations, e.g. by means of FEM [4,24,37]. This enables the model-based development of crocheted fabrics.

Besides being specific to automated flat crocheting, the design tool is also specific to the presented prototype crochet machine because it is the only machine capable of producing planar crocheted fabrics, and due to general structures of machine-crocheted fabrics are so far unknown. The implemented error checking is partly related to general crochet rules but is also partly machine specific. However, the focus on individual stitches of the presented tool offers an approach for future expansion into a general, machine-independent design tool, which is the research trend regarding established knitting machines [17,18,20], since independent of the specific machine, the sequence in which the stitches are formed, given by the principles of crocheting, remains the same. Thus, when generating instructions for a specific machine, the stitches can be mapped simply to other text files with the appropriate machine commands. Therefore, instructions for other machines with a similar operation principle could be generated with the same GUI and tool. Assuming that alternative crochet machines also form the stitches only from one side, the topological modeling can also be seen as generally valid, because the topology of the stitches, alongside the production sequence, are given by the principles of crochet.

3.2 Model comparison of machine and manually crochet fabrics

In analogy to single jersey weft knitting machines and plain fabrics [38], machine-crocheted fabrics have a technical face and a technical back. This is because, in stitch formation, yarn is always drawn from the back to the front (cf. Fig. 2), creating face loops. The structural difference to manually crocheted fabrics, where face loops are created on both sides by turning the textile after each course, is illustrated in Fig. 9. By considering the SCs in the fabric's centers, the two sides of the machine-crocheted fabric (Fig. 9 a) and b)) can be clearly distinguished, whilst in the manually crocheted one (Fig. 9 d and e) they cannot be differentiated based on the SCs.

In the side view of the manually crocheted fabric (Fig. 9 f), the alternating side, from which the loops are drawn to form the stitches, can be easily observed in the SLs (courses 2, 3 and 7), which are aligned almost perpendicular to the previous course. In the machine-crocheted fabric, the SLs are more stretched (cf. Fig. 9 c) due to the fabric take-off. Furthermore, regarding machine-crochet, the loops of the SCs in the second course are drawn through the first course of CHs differently than in manual crochet, which can be seen in the different shapes of the lowest courses of a) and d). These differences have to be considered in the design of automatically crocheted fabrics. Thus, modeling is also important for skilled crocheters to mind the deviations from manual crocheting.

By the developed modeling, the topology of machine-crocheted fabrics is presented here for the first time. This illustration of the new textile structure accessible by the crochet machine is valuable not only for academic interest, but also necessary to demonstrate the potential of the novel crochet machine.

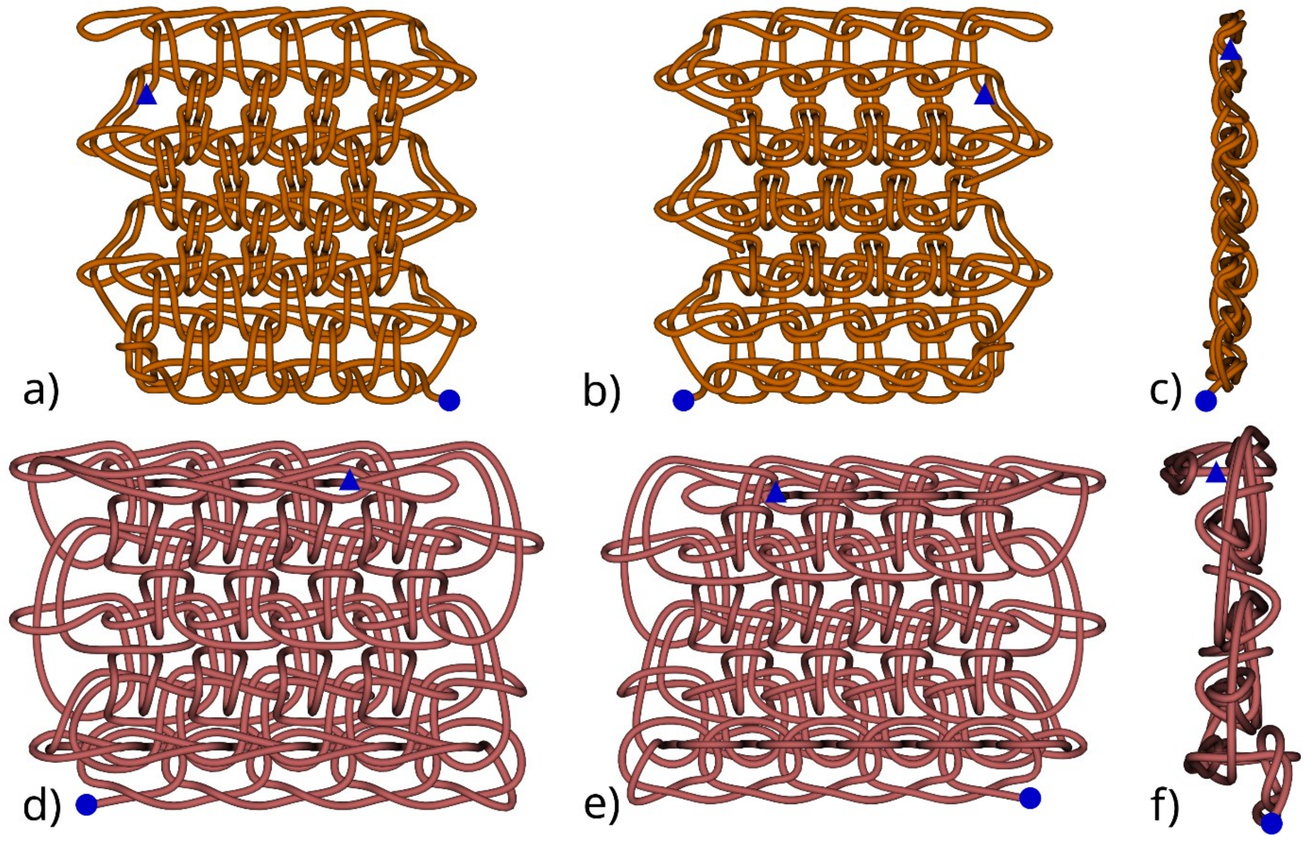


Fig. 9 Comparison of the modeled structure of automatically (top) and manually (bottom) crocheted fabrics. The stitch structure of the fabric is depicted in Fig. 5. The starting point of the yarn path is indicated by a blue circle and the end point by a blue triangle. Modeling of the manually crocheted fabric is conducted with the program presented in reference 4, and both models are visualized by the TexMind Viewer. (a) The technical face of the modeled machine-crocheted fabric; (b) the technical back of it; (c) the side view of the model; (d) one side of the modeled manually crocheted fabric; (e) the other side; (f) side view.

4 Conclusions

For a future automated production of crocheted textiles, not only an appropriate real crochet machine is needed but also an easy-to-use design tool tailored to crochet. In this regard, the developed tool is specific to the introduced flat crochet machine currently under development, because this machine is so far the only one and existing tools are not applicable in this case.

A planar fabric consisting of CHs, SLs, SCs and HDCs can be designed based on international crochet symbols, that can be arranged with a GUI. Automated error checking according to machine-specific and general crochet rules supports the user during the design process and ensures machine-producibility. In this regard, criteria for machine-producibility of crocheted textiles are defined.

Topology-based models with spline-interpolated key points on the meso scale are automatically generated to facilitate the rapid design process and to illustrate the unique structure of machine-crocheted fabrics. In contrast to manually crocheted fabrics, these exhibit a technical face as well as a technical back comparable to plain knitted fabrics, consequently they are slightly different in structure. In this context, the capabilities of a true crochet machine and the resulting stitch structures are presented for the first time.

Furthermore, the chosen array-based data structure as output of the GUI represents the crochet stitches by strings arranged topologically to the crochet structure and is used to generate the G-code for automatic production with the crochet machine prototype based on macros of machine instructions for building stitches. This allows for flexible adjustments to future advancements of the machine. The stitch level computing also provides the capability of using the design tool for future alternative crochet machines by simply mapping the stitches to other macros.

In the future, the crochet machine will be presented in more detail and automatically crocheted fabrics are planned to be compared with the models. Additionally, the relationship between yarn tension and stitch height is to be investigated so that the height of the stitches in the model can be calculated realistically. Also, it is planned to publish the code of the proposed design tool as open access after improving the documentation. Besides the benefits of using the design tool to construct crocheted samples in the context of ongoing machine development, the tool is intended in particular to demonstrate the possible structures of machine-crocheted fabrics and to aid potential users in accessing this new type of textile technology.

Author Contributions

J. L. Storck: conceptualization, software, investigation, writing – original draft preparation, visualization; B. A. Feldmann: software, investigation; Y. Kyosev: methodology, validation, supervision. All authors: writing – review and editing. All authors have read and agreed to the published version of the manuscript.

Conflicts of Interest

The authors declare no conflict of interest.

References

1. Spencer, D. J. An Introduction to Textile Technology. In *Knitting Technology: a comprehensive handbook and practical guide*, 3th ed.; Woodhead Publishing, 2001; pp. 1-6.
2. Karp, C. Defining Crochet. *Textile History* **2018**, *49*, 208–223, DOI: 10.1080/00404969.2018.1491689.
3. Emery, I. *The primary structures of fabrics: an illustrated classification*. Whitney Library of Design, 1995, ISBN: 0500016232.
4. Storck, J.L.; Gerber, D.; Steenbock, L.; Kyosev, Y. Topology based modelling of crochet structures. *Journal of Industrial Textiles* **2022**, *52*, 152808372211392, DOI: 10.1177/15280837221139250.
5. Nakjan, P.; Ratanotayanon, S.; Porwongsawang, N. Automatic Crochet Pattern Generation from 2D Sketching, 170–175, In *10th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2018; pp. 170-175, DOI: 10.1109/KST.2018.8426123.

6. Xu, Z.; Matsuoka, Y.; Deshpande, A.D. Crocheted artificial tendons and ligaments for the anatomically correct testbed (ACT) hand. In *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2015 IEEE International Conference on Robotics and Biomimetics (ROBIO), Zhuhai, 06.12.2015 - 09.12.2015; IEEE, 2015; pp. 2449–2453, DOI: 10.1109/ROBIO.2015.7419706.
7. Xu, Z.; Todorov, E.; Dellon, B.; Matsuoka, Y. Design and analysis of an artificial finger joint for anthropomorphic robotic hands. In *2011 IEEE International Conference on Robotics and Automation*, 2011 IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 09.05.2011 - 13.05.2011; IEEE, 2011; pp. 5096–5102, DOI: 10.1109/ICRA.2011.5979860.
8. Shi, L.; Wang, S.; Liang, N.; Zhang, X.; Niu, L.; Cheng, D.; Tang, X. Sound absorption of crochet fabrics with multi-plied yarns. *Applied Acoustics* **2022**, *199*, 109017, DOI: 10.1016/j.apacoust.2022.109017.
9. Bobin, M.; Amroun, H.; Coquillart, S.; Bimbard, F.; Ammi, M. SVM based approach for the assessment of elbow flexion with smart textile sensor. In 2*017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. 2017; pp. 2129–2134, DOI: 10.1109/SMC.2017.8122934.
10. Jiang, C.; Wang, K.; Liu, Y.; Zhang, C.; Wang, B. Textile-based sandwich scaffold using wet electrospun yarns for skin tissue engineering. *Journal of the Mechanical Behavior of Biomedical Materials* **2021**, *119*, 104499, DOI: 10.1016/j.jmbbm.2021.104499.
11. Spencer, D. J. Classes of warp knitting machines. In *Knitting Technology: a comprehensive handbook and practical guide*, 3th ed.; Woodhead Publishing Limited, 2001; pp. 289-312.
12. Grimmelsmann N; Fiedler J; Ehrmann A. Häkelmaschine. DE 10 2016 015 204 1 2018.06.21, 2018.
13. Grimmelsmann, N; Döpke, C; Wehlage, S; Ehrmann, A. The largest crocheting machine in the world. *Melliand Int.* **2019**, *25*, 99–100.
14. Guo, R.; Lin, J.; Narayanan, V.; McCann, J. Representing Crochet with Stitch Meshes. In *Symposium on Computational Fabrication, SCF '20: Symposium on Computational Fabrication*, Virtual Event USA, 05 11 2020 06 11 2020; Whiting, Emily, Hart, John, Sung, Cynthia, Peek, Nadya, Akbarzadeh, Masoud, Aukes, Dan, Schulz, Adriana, Taylor, Hayden, Kim, Jeeeun, Eds.; ACM: New York, NY, USA, 2020; pp. 1–8, DOI: 10.1145/3424630.3425409.
15. Çapunaman, Ö.B.; Bingöl, C.K.; Gürsoy, B. Computing Stitches and Crocheting Geometry. In *Computer-Aided Architectural Design. Future Trajectories*. Çağdaş, G., Özkar, M., Gül, L.F., Gürer, E., Eds.; Springer Singapore: Singapore, 2017; vol. 724; pp. 289–305.
16. Choi, W.; Powell, N. B. Three dimensional seamless garment knitting on V-bed flat knitting machines. *Journal of Textile and Apparel, Technology and Management* **2015**, *4* (3), 1-33.
17. McCann, J.; Albaugh, L.; Narayanan, V.; Grow, A.; Matusik, W.; Mankoff, J.; Hodgins, J. A compiler for 3D machine knitting. *ACM Trans. Graph*. **2016**, *35*, 1–11, DOI: 10.1145/2897824.2925940.
18. Yu, T.C.; McCann, J. Coupling Programs and Visualization for Machine Knitting. In *Proceedings of the 5th Annual ACM Symposium on Computational Fabrication (SCF '20)*. Association for Computing Machinery, New York, NY, USA, 2020; vol 7, pp. 1-10, DOI: 10.1145/3424630.3425410.
19. Spencer, D. J. Electronics in knitting. In *Knitting Technology: a comprehensive handbook and practical guide*, 3th ed.; Woodhead Publishing Limited, 2001; pp. 134-144.
20. Narayanan, V.; Wu, K.; Yuksel, C.; McCann, J. Visual knitting machine programming. *ACM Trans. Graph*. **2019**, *38*, 1–13, DOI: 10.1145/3306346.3322995.
21. Kyosev, Y.; Renkens, W. Modelling and visualization of knitted fabrics. In *Modelling and predicting textile behaviour*, Chen X (Ed.), Sawston: Woodhead Publishing, 2010, pp. 225–262.
22. Sha, S.; Geng, A.; Gao, Y.; Li, B.; Jiang, X.; Tao, H.; Luo, L.; Yuan, X.; Ke, H.; Hu, X.; Deng, Z.; Chen, Z. Review on the 3-D simulation for weft knitted fabric. *Journal of Engineered Fibers and Fabrics* **2021**, *16*, 155892502110125, DOI: 10.1177/15589250211012527.
23. Kyosev, Y. Introduction and problem definition. In: *Topology-based modeling of textile structures and their joint assemblies,* Kyosev, Y. Ed.; Cham: Springer International Publishing, 2019, pp. 3–9.
24. Renkens, W.; Kyosev, Y. 3D Simulation of warp knitted structures-new chance for researchers and educators. In *7th International Conference-TEXSCI*, 2010.
25. Warp knitting software on TexMind website. http://texmind.com/wp/products/warp-knitting-software/ (accessed 2023-02-19).
26. Kyosev, Y. Software for design of warp knitted structures. In *Warp knitted fabrics construction*. CRC Press, 2019; pp. 283-298.
27. Renkens, W.; Kyosev, Y. Geometry modelling of warp knitted fabrics with 3D form. *Textile Research Journal* **2011**, *81*, 437–443.
28. Spencer, D. J. Basic warp knitting principles. In *Knitting technology: a comprehensive handbook and practical guide*. 3th ed.; Woodhead Publishing Limited, 2001; pp. 286-297.
29. Craft Yarn Council Website. https://www.YarnStandards.com (accessed 2023-03-10).
30. Zaharieva-Stoyanova, E.; Bozov, S. CAD module for knitting patterns design. In *Proceedings of the 12th International Conference on Computer Systems and Technologies*, 2011; pp. 298-304.
31. Lin, J.; Narayanan, V.; McCann, J. Efficient transfer planning for flat knitting. In *Proceedings of the 2nd Annual ACM Symposium on Computational Fabrication*, 2018; pp. 1-7.
32. TkInter homepage. https://wiki.python.org/moin/TkInter (accessed 2023-03-16).
33. TexMind textile viewer on TexMind website. http://texmind.com/wp/products/neues-produkt/ (accessed 2023-02-19).
34. Python library pytexlib on GitHub. https://github.com/virtualtextiles/pytexlib/ (accessed 2023-08-04).
35. Cncjs homepage. https://cnc.js.org (accessed 2023-08-04).

36. Yan, X.; Gu, P. A review of rapid prototyping technologies and systems. *Computer-Aided Design* **1996**, *28*, 307–318, doi: 10.1016/0010-4485(95)00035-6.
37. Kyosev, Y. Applications of the topological generated models. In *Topology-Based Modeling of Textile Structures and Their Joint Assemblies: Principles, Algorithms and Limitations*, Springer International Publishing, 2019; pp. 221-232.
38. Spencer, D. J. The four primary base weft knitted structures. In *Knitting technology: a comprehensive handbook and practical guide*. 3th ed.; Woodhead Publishing Limited, 2001; pp. 60-81.

Supplementary materials

Python-based code for the error checking algorithm

```python
# GUI_Array is an array based on the proposed data structure,
# storing the stitches and their topological information, for example as seen in Fig. 3:
GUI_Array = [['sl', 'sl', 'sl', 'sl', 't1'],
             ['t1', 'sc', 'sc', 'sc', 'sc'],
             ['sc', 'sc', 'sc', 'sc', 't1'],
             ['t1', 'sc', 'sc', 'sc', 'sc'],
             ['sl', 'sl', 'sl', 'sl', 't1'],
             ['flo', 'sl', 'sl', 'sl', 'sl'],
             ['void', 'ch', 'ch', 'ch', 'ch']]
def errorTest(GUI_Array):
    # List of found errors
    ErrorList = []
    # The number of rows in the array
    NumOfRows = len(GUI_Array)
    # List of line lengths
    LengthList = []
    # List of found transitions
    TransitionList = []
    # Fabric start / end indices
    FabricStart = -1
    FabricEnd = -1
    OldFabricStart = -1
    OldFabricEnd = -1
    # Begin iterating over the GUI_Array in one-step increments, starting at the top row,
    # indices i and j are illustrated in Fig. 3 in relation to the crochet pattern
    for i in range(0, NumOfRows, 1):
        # Found transitions per line
        TransitionLineList = []
        # Old start / end for checking for stitches before the first found transition
        OldFabricStart = FabricStart
        OldFabricEnd = FabricEnd
        FabricStart = -1
        FabricEnd = -1
        # List of found voids in the fabric
        VoidList = []
        # List of potential voids in the fabric
        PotentialVoidList = []
        # Iterate over the elements in the line
        for j in range(0, len(GUI_Array[i]), 1):
            # Is the selected element a valid stitch?
            if GUI_Array[i][j] != 'void':
                if FabricStart < 0:
                    FabricStart = j
                # Satisfying constraint f) by checking for empty fields between fabric start
                # and fabric end within the line:
                # If a valid fabric end has been found
                # the potential void candidates become actual voids
                if FabricEnd < j:
                    FabricEnd = j
                    VoidList += PotentialVoidList
                    PotentialVoidList = []
                # Satisfying constraint e):
                # Check if a stitch has a parent stitch to attach to, ignore the last row
                if i < NumOfRows-1:
                    # Is the stitch not the first-lay-over (flo)?
                    if GUI_Array[i][j] != 'flo':
                        # Does the stitch have a parent stitch to connect to?
                        if GUI_Array[i+1][j] == 'void':
                            ErrorList.append((i+1, j, "Missing parent stitch"))
                    # flo found
                    else:
                        # Satisfying constraint b):
                        # Check if there is a parent stitch for the flo
                        if GUI_Array[i+1][j] != 'void':
                            ErrorList.append((i+1, j, "flo cannot have parent stitch"))
                if IsTransition(GUI_Array[i][j]):
                    TransitionLineList.append((i, j))
            # The element is unset / there is no stitch
            else:
                # Satisfying constraint f):
                # If a fabric start point has been found,
                # every void after it is highlighted as a potential void candidate
                if FabricStart > -1:
                    PotentialVoidList.append(j)
        # After finishing iteration of one line:
        # Calculate and save the line length, the elements between fabric start and end,
        # regardless of whether they are empty or not
        LineLength = FabricEnd-FabricStart+1
        LengthList.append(LineLength)
        # Save the found transitions
        TransitionList.append(TransitionLineList)
        # For every found void in the fabric throw an error
        for j in VoidList:
            ErrorList.append((i, j, "Missing stitch in fabric"))
        # Satisfying constraint c) by calculating the correct transition positions,
        # this ensures the correct crochet direction:
        # Scan the previous line, if there is one, to check for errors with transitions
        if i > 0:
            # Note: This function checks where the transition should be in this row,
            # not where a potentially misplaced transition is at
            TransitionLeftOrRight = checkTransitionSide(i-1)
            # Fabric start or end index used to determine the transition position
            FSE = None
            # The transition is on the left,
            # the fabric start index can be used to determine the transition position,
            # but might need to be adjusted
            if TransitionLeftOrRight == 'left':
                # If current row index is not in the last row
                if i < NumOfRows-1:
                    FSE = FabricStart
                else:
                    # Last row and a void below flo
                    if FabricStart > 0:
                        # flo has a void as parent stitch causing the fabric start to shift
                        FSE = FabricStart-1
                    # Last row and no void under the flo (error), the FSE has to be adjusted
                    else:
                        FSE = 0
            # The transition is on the right,
            # the fabric end index can be used to determine the transition position
            else:
                FSE = FabricEnd
            # The position, above the current row, where the transition should be at
            SelectedStitch = GUI_Array[i-1][FSE][0]
            # Check if there are obsolete stitches before the transition position
            ErrorStart = None
            ErrorEnd = None
            if TransitionLeftOrRight == 'left':
                ErrorStart = OldFabricStart
                ErrorEnd = FSE
            else:
                # Note: We have to adapt the index with '+1',
                # due to range(x) not counting the last element
                ErrorStart = FSE+1
                ErrorEnd = OldFabricEnd+1
            # Satisfying constraint g):
            # Iterate in one-step increments from the start of the obsolete stitches,
            # throw an error for each field if it is not a 'void'
            for n in range(ErrorStart, ErrorEnd, 1):
                # Is there a stitch before the transition?
                if GUI_Array[i-1][n][0] != 'void':
                    ErrorList.append((i-1, n, "Stitch before transition position"))
            # Is the stitch at the transition’s position an actual transition?
            if IsTransition(SelectedStitch):
                # Special case for the last row (first CH course),
                # since the transition that should be found is the flo
                if i == NumOfRows-1:
                    if SelectedStitch != 'flo':
                        ErrorList.append((i-1, FSE, "Missing flo"))
                # If not in the last row
                else:
                    # The parent stitch in the 'i'-row the transition connects to
                    Parent = GUI_Array[i][FSE][0]
                    # The target stitch in the 'i-1'-row the transition connects to
                    Target = None
                    if TransitionLeftOrRight == 'left':
                            # The transition is on the left,
                            # so the stitch following the transition is on the right ('+1')
                            Target = GUI_Array[i-1][FSE+1][0]
                        else:
                            # The transition is on the right,
                            # so the stitch following the transition is on the left ('-1')
                            Target = GUI_Array[i-1][FSE-1][0]
                        # Get the correct transition type given parent, target and direction
                        Transition = getTransition(Parent, Target, TransitionLeftOrRight)
                        # Is the selected stitch to the right transition type?
                        if SelectedStitch != Transition:
                            ErrorList.append((i-1, FSE, "Wrong transition at position"))
                # No transition was found, where there should be one
                else:
                    ErrorList.append((i-1, FSE, "Missing transition at position"))
                # Check if more than one transition was found in the row
                TransitionFound = False
                for t in TransitionList[i-1]:
                    # Is the selected transition the correct one with matching indices?
                    if (not TransitionFound) and t[0] == i-1 and t[1] == FSE:
                        TransitionFound = True
                    # Obsolete transition found
                    else:
                        ErrorList.append((t[0], t[1], "Obsolete transition at position"))
        # The length of the last row / CH course and one additional void field for the flo
        CH_LineLength = LengthList[-1]+1
        # Satisfying constraint d):
        # Check every line length, ignore the last row (':-1') as it is the fabric start
        for index, LineLength in enumerate(LengthList[:-1]):
            # Is the row length longer than the fabric start?
            if LineLength > CH_LineLength:
                ErrorList.append((index, 0, "Line longer than fabric start"))
            # Is the row length shorter than the fabric start?
            elif LineLength < CH_LineLength:
                ErrorList.append((index, 0, "Line shorter than fabric start"))
        # Notify the GUI for every error that has been found
        for Error in ErrorList:
            NotifyGUI(Error)
        return ErrorList
```