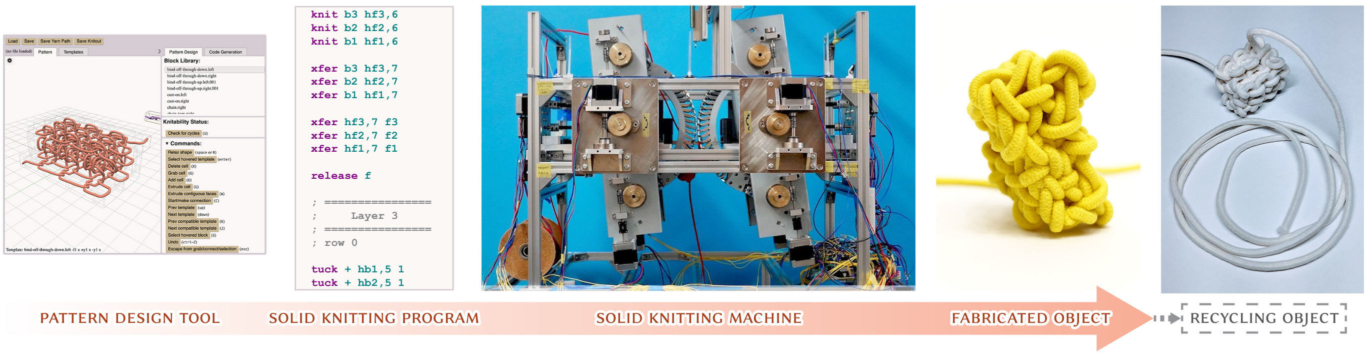
Solid Knitting

YUICHI HIROSE, MARK GILLESPIE, ANGELICA M. BONILLA FOMINAYA, and JAMES MCCANN,
Carnegie Mellon University, United States



```
knit b3 hf3,6
knit b2 hf2,6
knit b1 hf1,6

xfer b3 hf3,7
xfer b2 hf2,7
xfer b1 hf1,7

xfer hf3,7 f3
xfer hf2,7 f2
xfer hf1,7 f1

release f

; ==================
;      Layer 3
; ==================
; row 0

tuck + hb1,5 1
tuck + hb2,5 1
```

Fig. 1. Solid knitting is a new technique to fabricate dense, firm objects via knitting. The objects are built up layer by layer in the style of 3D printing, but are held together by intertwined strands of yarn rather than the use of adhesives. Here, we show our pipeline for creating objects using our solid knitting machine: our pattern design tool compiles shapes to low level instructions, which are executed by our solid knitting machine to fabricate objects. Once they have been fabricated, solid-knit objects can be recycled since the yarn can be recovered via unraveling.

We introduce solid knitting, a new fabrication technique that combines the layer-by-layer volumetric approach of 3D printing with the topologically-entwined stitch structure of knitting to produce solid 3D objects. We define the basic building blocks of solid knitting and demonstrate a working prototype of a solid knitting machine controlled by a low-level instruction language, along with a volumetric design tool for creating machine-knittable patterns. Solid knitting uses a course-wale-layer structure, where every loop in a solid-knit object passes through both a loop from the previous layer and a loop from the previous course. Our machine uses two beds of latch needles to create stitches like a conventional V-bed knitting machine, but augments these needles with a pair of rotating hook arrays to provide storage locations for all of the loops in one layer of the object. It can autonomously produce solid-knit prisms of arbitrary length, although it requires manual intervention to cast on the first layer and bind off the final row. Our design tool allows users to create solid knitting patterns by connecting elementary stitches; objects designed in our interface can—after basic topological checks and constraint propagation—be exported as a sequence of instructions for fabrication on the solid knitting machine. We validate our solid knitting hardware and software on prism examples, detail the mechanical errors which we have encountered, and discuss potential extensions to the capability of our solid knitting machine.

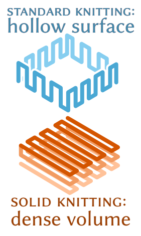
CCS Concepts: • **Applied computing → Computer-aided manufacturing**.

Additional Key Words and Phrases: knitting, solid knitting, knitting machine, fabrication, 3D printing

Authors' address: Yuichi Hirose, yuichih@cs.cmu.edu; Mark Gillespie, mgillesp@cs.cmu.edu; Angelica M. Bonilla Fominaya, abfominaya@gmail.com; James McCann, jmccann@cs.cmu.edu, Carnegie Mellon University, Pittsburgh, PA, United States.

ACM Reference Format:
Yuichi Hirose, Mark Gillespie, Angelica M. Bonilla Fominaya, and James McCann. 2024. Solid Knitting. *ACM Trans. Graph.* 43, 4 (July 2024), 15 pages. https://doi.org/10.1145/3658123

1 INTRODUCTION



In this paper, we introduce a new fabrication method called solid knitting. Unlike standard knitting, which makes hollow surfaces, solid knitting makes dense volumes by layering knit sheets—much as 3D printers layer plastic sheets. We envision a future where everyday objects like furniture or shoes—including soles—can be knit as one piece. Since the layers are topologically intertwined during fabrication, solid knitting requires no adhesives, allowing fabricated objects to be unraveled easily to recover the constituent yarn (Figure 1, *right*).

Contributions. This work makes four main contributions:

- We propose *solid knitting*, a novel fabrication method which can create dense, firm objects from a single strand of yarn.
- We present a *solid knitting machine* which automates the process of solid knitting.
- We introduce *augmented stitch volumes*, a volumetric mesh data structure for representing solid knitting patterns.
- We describe a scheduler to generate machine instructions from augmented stitch volumes.

After reviewing background material in Section 2, we contrast solid knitting with traditional knitting in Section 3, and describe our solid knitting machine in Section 4. We then present our stitch volume data structure, scheduler, and design system in Section 5, before exploring results in Section 6 and future work in Section 7.

2 BACKGROUND & RELATED WORK

Our solid knitting machine builds on existing work on knitting machines (Section 2.1), as well as design systems for 3D surface knitting (Section 2.2), and work on volumetric textile structures (Section 2.3). Before specializing to textiles, we also note that our solid knitting machine can also be viewed as a type of 3D printer, joining the ranks of nontraditional 3D printers which use, *e.g.*, layered paper sheets [Mcor Technologies 2014], needle felting [Hudson 2014], or laminated felt sheets [Peng et al. 2015] to create 3D objects.

2.1 Machine Knitting

While textile crafts date back to prehistory, knitting is thought to have been developed between 500 and 1200 CE [Rutt 1987, p.39] and knitting machines first appeared in 1589 [Felkin 1867, Ch. 3]. These early machines looked similar to those used today, although the latch needles used by modern knitting machines were only invented in the mid-19th century [Felkin 1867, Ch. 29; Spencer 2001, §3.14].

Knitting machines make fabric by pulling loops of yarn through other loops, often manipulating the yarn via latch needles (Figure 2). Latch needles feature a latch which rotates between the shank and the hook tip, creating open and closed states which hold and release loops as needed. To make a new loop, a knitting machine begins by extending a latch needle forwards. This motion slides any loop currently held by the needle toward the shank and opens the latch (Figure 2, *top*). Then the machine uses its *yarn carrier* to pull a new span of yarn across the open hook. Retracting the needle slides the loop over the latch and drops it off the needle tip, which simultaneously causes the latch to close. As a result, the new span of yarn passes through the old loop and becomes a new loop sitting on the needle (Figure 2, *bottom*). One fundamental limitation of knitting machines is that—unlike human knitters—they cannot pick up dropped loops. In order to knit through a loop at some point in the future, a knitting machine must keep hold of it on a latch needle. Consequently, industrial knitting machines feature a large number of latch needles. These needles are laid out in *needle beds* which can hold a row of loops. Standard V-bed knitting machines are composed of two large needle beds facing each other in an inverted V shape. For example, a Shima Seiki SWG091N2 15-gauge V-bed knitting machine has 540 needles on each bed [Shima Seiki 2013].

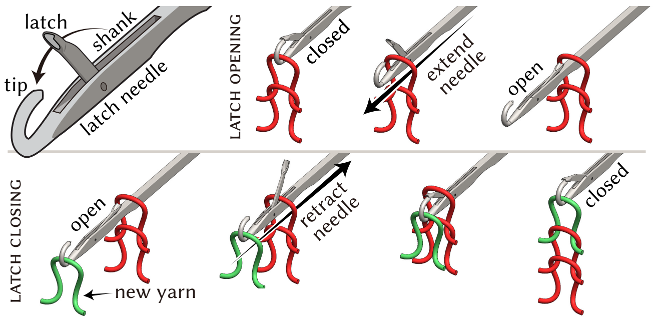


Fig. 2. Knitting machines manipulate loops of yarn using latch needles, whose latches rotate between the needle shank and tip. Extending a closed needle which holds a loop pushes the latch open (*top*). Conversely, retracting an open needle pushes the latch closed. Running new yarn through an open needle before retracting it creates a new stitch on the needle (*bottom*).

In the world of personal knitting, the Electro-knit and Knitic projects modified personal knitting machines to be controlled by computers [Guljajeva and Canet Sola 2013; Torres 2012]. Later open-source projects progressed to building entire machines, such as Circular Knitic for circular knitting machines and OpenKnit for V-bed knitting machines [Guljajeva and Canet Sola 2014; OpenKnit 2013]. OpenKnit eventually turned into the startup Kniterate [2015].

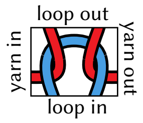
2.2 Knitting Pattern Design

Igarashi et al. [2008] explored the problem of handmade yarn surface fabrication, creating an interactive design system for making plush toys by hand crocheting. McCann et al. [2016] turned to machine fabrication and designed a compiler to convert high-level shape primitives into needle-level machine instructions represented as *knitout* code [McCann 2017]. Narayanan et al. [2018] built a system to generate knitout code from general 3D meshes (Popescu et al. [2018] developed a similar system for disc-shaped meshes). Narayanan et al. [2019] proposed the *augmented stitch mesh* as a higher-level representation of knitting patterns, generalizing the *stitch mesh* data structure that Yuksel et al. [2012] used for knit simulation. There have been a plethora of proposed design systems for knitting, including specialized systems focused on cloth patterns [Hofmann et al. 2019; Leaf et al. 2018], sketch-based design [Kaspar et al. 2021], high-level scripting [Hofmann et al. 2023], or complex shaping [Edelstein et al. 2022; Mitra et al. 2023], as well as more wholistic systems offering combined design of shape, texture, and color [Jones et al. 2021; Nader et al. 2021]. This work on knitting design has enabled cross-disciplinary research on knitting throughout graphics, fabrication, human-computer interaction, and human-robot interaction, *e.g.*, using knit fabric as soft sensors [Luo et al. 2021; Yu et al. 2023; Zlokapa et al. 2022], soft actuators [Albaugh et al. 2019; Luo et al. 2022], or materials with controllable elasticity [Liu et al. 2021]. However, almost all existing work has focused on designing patterns to fabricate *surfaces*, rather than fabricating *volumes*.

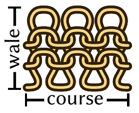
2.3 Volumetric Textiles

Traditional 2D textiles can be extended to 3D in a number of ways. Most existing work has focused on weaving, although there has been some investigation of knitting and crochet. *Knitting.* Albaugh et al. [2021] explored knit spacer fabrics, creating knit layers linked by monofilament filler yarn as an application of V-bed machines beyond traditional hand knitting techniques. *Crochet.* Closer to our work is *obsession series* [Lee 2009], a collection of dense volumetric objects made by intertwining layers of hand-crocheted fabric. However, given that even performing traditional crochet with a machine is an active area of research [Grimmelsmann et al. 2018; Perry et al. 2023], automating this volumetric form of crochet would be technically difficult. *Weaving.* Unlike knitting or crochet, which often use just a single yarn, weaving forms a fabric by interlacing a large number of vertical and horizontal threads. Wu et al. [2020b] proposed a design tool for 3D weaving and Wu et al. [2020a] presented a method to generate machine instructions from 3D solid models.

3 SOLID KNITTING

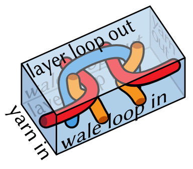


Traditional knit fabric is made by pulling loops through loops (Figure 3). Abstractly, an individual knit stitch can be seen as a rectangular tile, attached to four neighbors. It is connected to its left and right neighbors by the yarn and connected to its top and bottom neighbors by interlocking loops.

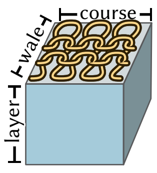


In a knitting pattern, these tiles are generally laid out in a grid. The horizontal yarn direction is the *course* direction, while the vertical loop direction is the *wale* direction (see inset).

Solid knitting extends this structure into a third dimension, stacking up layers of connected knit fabric to form a volume (Figure 4). Each solid knit stitch attaches to both a stitch of the previous row in the wale direction and a stitch of the corresponding row in the layer below. Consequently, each loop is also knit through twice: once by a stitch in the next row in the wale direction, and once by a stitch in the next layer. Knitting through a loop for the first time renders it *stable*—such loops will not immediately fall apart if pulled—but the stitch is not *complete* until it is knit through a second time (Figure 5). We include a hand-solid-knitting tutorial as supplemental material.



Just as a traditional knit stitch can be seen as a rectangular tile, a solid knit stitch can be seen as a box-shaped tile attached to six neighbors. It is connected to its left and right neighbors via the yarn and connected to its neighbors in the other two directions by interlocking loops.



In solid knitting patterns, these blocks are laid out in a 3D grid. We call the x and y directions the course and wale directions respectively, and call the new z direction the *layer* direction. Our design tool (Section 5.2) translates arrangements of such blocks into solid knitting instructions.

4 THE SOLID KNITTING MACHINE

Our solid knitting machine automates the process of solid knitting. As discussed in Section 3, solid knit objects are composed of stacked layers of stitches, each of which is knit through by another loop in the next layer to intertwine the layers together. In order to form stitches in this way, the machine needs two key capabilities: the ability to pull a new loop through a pair of existing loops, and the capacity to hold on to all of the loops in one layer until they are knit through again in the next layer. This holding function is unique to the solid knitting machine; traditional knitting machines only need to manipulate a single row of loops at a time, since loops are generally only knit through once by a stitch in the next row. Managing an entire layer of loops is the central problem of mechanized solid knitting.

4.1 Basic Configuration

To form and manipulate loops, our solid knitting machine has two beds of latch needles just as in a traditional V-bed knitting machine. The machine also has additional hooks, which we call *holders*, to hold on to stable but incomplete loops (Figure 7). The holding hooks are arranged in cylindrical arrays, allowing them to hold all of the loops in an entire 2D layer of solid knitting. By rotating these arrays, the machine can align any holding hook with the latch needle in its column, allowing the machine to transfer loops between the holding hook and needle in either the front or back bed. Between the needles, the solid knitting machine has *retaining needles*, which are used to tuck new spans of yarn into the holders. The machine has a single *yarn carrier* which runs yarn in between the two beds.

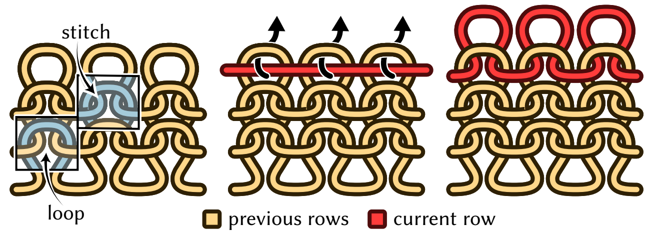


Fig. 3. Knit fabric is made up of loops which have been interlocked to form rows of stitches (*left*). To make a new row, one runs yarn across the top row of loops (*center*) and pulls the yarn through to create new loops (*right*). Each stitch is connected to its left and right neighbors directly by the yarn, and connected to the stitches below and above it by interlocking loops of yarn.

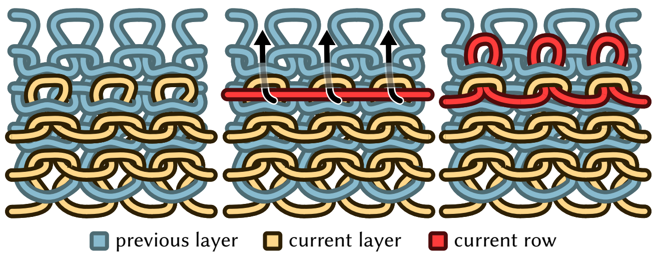


Fig. 4. Solid knit objects are comprised of stacked layers of knit fabric, each of which is made up of rows of stitches (*left*). To form a new row, one runs yarn across the latest row of loops on the top layer (*center*) and pulls the yarn through both these loops *and* loops from a row one layer down (*right*). Each stitch is thus connected to its left and right neighbors directly by the yarn, and connected to all other neighbors by interlocking loops of yarn.

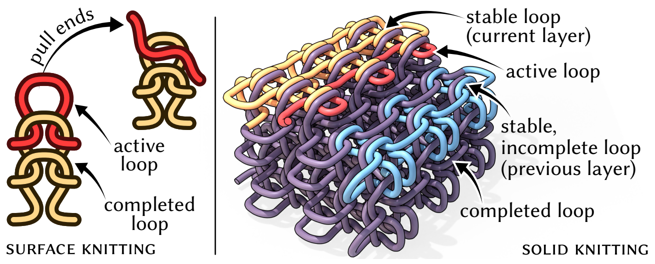


Fig. 5. *Left*: in traditional knitting loops in the top row are called *active*—or *unstable*—since pulling on the ends can unravel the loop, whereas the completed loops lower down remain stable when pulled. *Right*: in solid knitting, the most recent row of loops are also active and will unravel if pulled. The other loops on top of the object are stable, having been knit through once, but are not yet complete because they need to be knit through once more. All loops lower down in the object are completed.

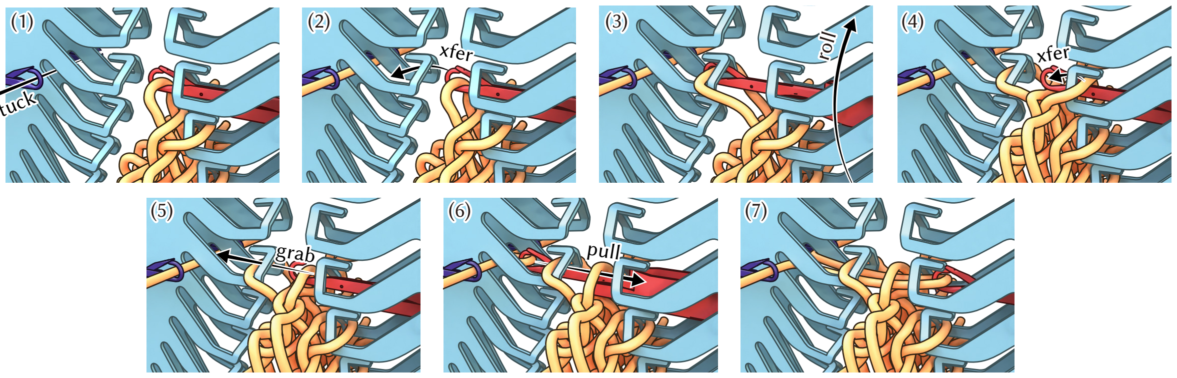


Fig. 6. Knitting the first row of a new layer on the solid knitting machine: (1) the machine first tucks a new span of yarn beneath the holder. Then, (2), it transfers a loop from a needle on the opposite bed onto the holder, where it can be knit through without being dropped. Next, (3), it rolls the holder upwards to access the next row of held loops, and then, (4), transfers a loop from this row onto the needle. Finally, the machine performs a knit operation, (5) pushing this needle forward, grabbing onto the new span of yarn tucked beneath the holder hook, and (6) retracting to pull the yarn through the loop on the holder and the loop on the needle. Note that the loop on the holder is retained to be knit through again in the subsequent layer, while the loop that was on the needle is released. (7) shows the completed solid knit stitch.

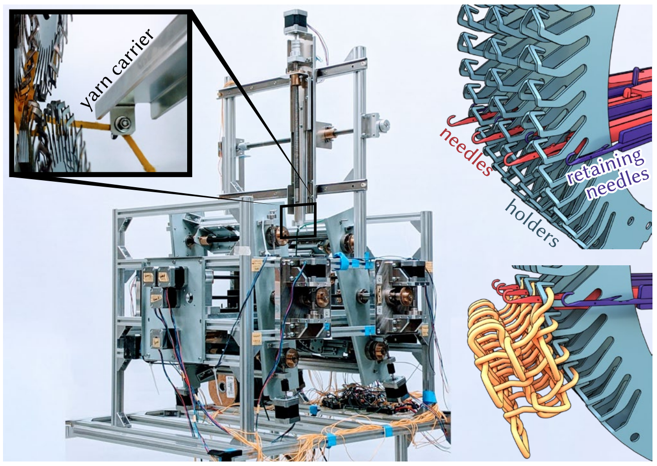
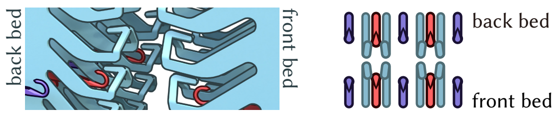


Fig. 7. The solid knitting machine has two needle beds, which are each made up of holders, needles, and retaining needles (*top right*). The holders have space to hold onto every loop in a layer of solid knitting (*bottom right*). The machine also has a single yarn carrier to feed in new yarn (*top left*).

Each holder hook is made up of two halves. Usually the holder is kept closed, with the two halves right next to each other. But for some operations the holder opens, separating the halves to provide access to the held loop.

holder closed
holder open

When illustrating the machine, we use perspective views from the side as well as more schematic top views. We draw the back bed at the left of side-view pictures and at the top of top-view pictures.



4.2 Forming Stitches

The process of solid knitting a stitch on our machine is illustrated in Figure 6. As this is the start of a new layer, one holder begins with all of the stable loops from the previous layer while the other holder begins empty. The machine starts by tucking yarn beneath a holder hook on the opposite bed and then pulls the new yarn through two loops: an active loop and a stable loop. It leaves the original active loop sitting on a holder—as the loop has become a stable loop to be knit through in the next layer—and simultaneously releases the original stable loop, which has now been knit through twice. Afterwards, the machine leaves the newly-formed stable loop on the opposite holder and holds a new active loop on its needle.

Broadly speaking, our machine moves loops between the two holder arrays while knitting a layer like hand knitters move loops between two needles while knitting a row (Figure 8, *top*). One holder holds the stable loops of the previous layer, while the other holds the stable and active loops of the current layer. Each new stitch pulls yarn through both an active loop of the current layer and a stable loop from the previous layer (Figure 8, *bottom*), turning the active loop into a stable loop held on the opposite bed and turning the yarn into the new active loop.

4.3 Basic Operations

The solid knitting process of Figure 6 can be summarized as:

(1) *tuck* yarn,
(2) *transfer* a loop from a needle to a holder hook,
(3,4) *transfer* a loop from a holder hook to a needle, and
(5,6,7) *knit* to form a new loop.

We call each set of these movements an *instruction*, and use them as minimum units of the machine motion. A list of the instructions serves as a program to control the solid knitting machine in a human-readable way, similar to G-code for 3D printers or knitout for traditional knitting machines [McCann 2017].

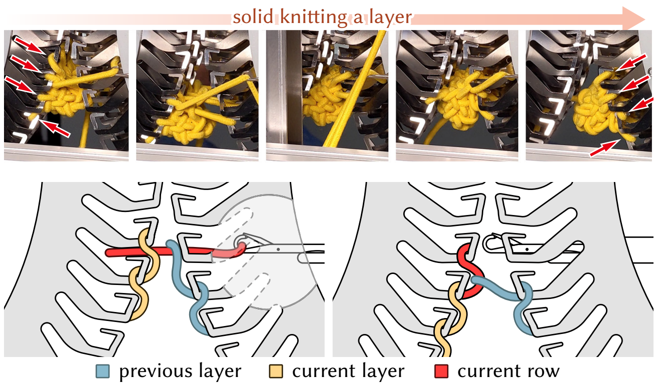


Fig. 8. *Top*: knitting a layer moves the object from one holder to the other. *Bottom*: while knitting, one holder holds the previous layer while the other holds the current layer. The newly formed loops (*i.e.*, the current row) are always transferred onto the current layer's holder.

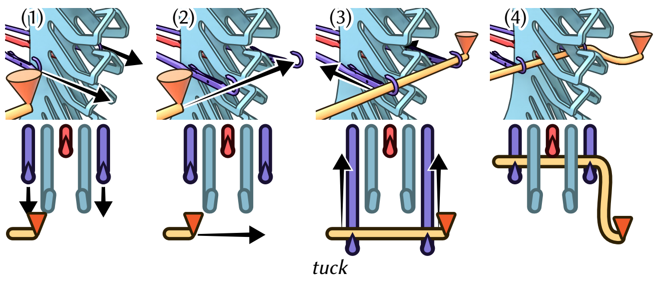
Our machine can perform the following *solid knitout* instructions. We write each instruction as a **name** followed by a list of `arguments`:

- **tuck** `D Hxy CS`: tuck a new span of yarn beneath holder hook `Hxy` using yarn carrier set `CS` in direction `D`.
- **xfer** `Nx Hxy`: move the loop on needle `Nx` to holder hook `Hxy` on the opposite bed.
- **xfer** `Hxy Nx`: move the loop on holder hook `Hxy` to needle `Nx` on the same bed.
- **knit** `Nx Hxy`: pull a loop through the loops on needle `Nx` and holder hook `Hxy` using yarn tucked beneath `Hxy`.
- **drop** `Nx`: drop any held loop from needle `Nx`. This is the same as knitting with no yarn tucked beneath the opposite holder hook.
- **release** `B`: let go of the yarn with the retaining needles on bed `B`.
- **roll** `H*y H*y`: rotate the holders so that the indicated rows on the front and back beds respectively align with the needles.
- **pause** `S`: temporarily stop knitting and display string `S` to the user, usually to request manual intervention.

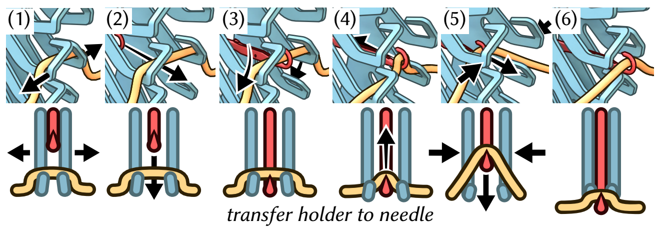
Formally, a direction `D` is either the symbol `+` or `-`. Needle locations `Nx` consist of a bed (`f` or `b`) followed by a needle index (*e.g.*, `f1` or `b2`). Holder locations `Hxy` consist of the character `h` followed by a bed location and a pair of holder indices (*e.g.*, `hf1,2` or `hb2,3`). We follow knitout in using carrier sets `CS`, although our current machine only has one yarn carrier.

Below, we describe these instructions in more detail. Unlike traditional V-bed machines, our current solid knitting machine does not offer a racking instruction to change the alignment of the beds; a discussion of the challenges involved with racking and possible solutions can be found in Section 7.4.

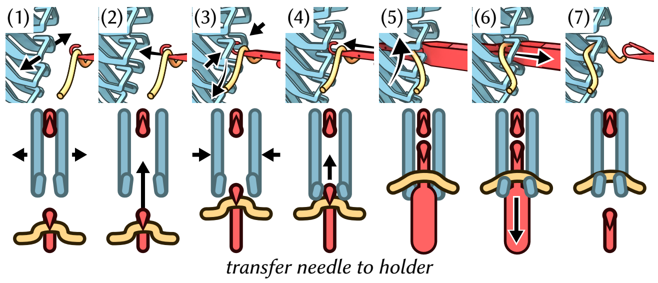
Tuck. The **tuck** instruction tucks a new span of yarn beneath a holder hook. First the machine extends the retaining needles (1) and runs the yarn carrier past them (2). Then it retracts the retaining needles (3) to catch the yarn and pull it under a holder hook.



Transfer. The **xfer** instruction transfers the loop on a needle to a holder hook or vice versa. To transfer a loop from a holder hook onto a needle, the machine first opens the holder hook (1), allowing the needle to extend and grab the yarn (2). The holder rotates to push the loops onto the needle (3), and then the needle retracts, pulling the loops off of the holder hook (4). Finally, the holder closes and the needle extends outwards again holding the loop (5).



To transfer a loop from a needle onto a holder hook on the opposite side, the machine first opens the holder hook (1). Then it moves the needle forward, bringing the tip of the needle between the two halves of the hook (2). It closes and rotates the hook downwards (3), and then moves the needle forward again (4). At this point the edge of the holder hook pushes against the held yarn loop, preventing the loop from passing too far inside of the hook. The holder then rotates upward to grab the loop (5). Retracting the needle closes the needle latch and completes the transfer (6, 7).



Knit. The **knit** instruction forms a new stitch by pulling yarn through two existing loops. To do so, the machine advances a needle through a loop held by the holder hook (1), grabs any yarn tucked beneath the holder hook, and then retracts it to pull the yarn through the old loop (2,3,4). This process forms a new loop on the needle. If there was also a loop on the needle to begin with, the new loop passes through both old loops.

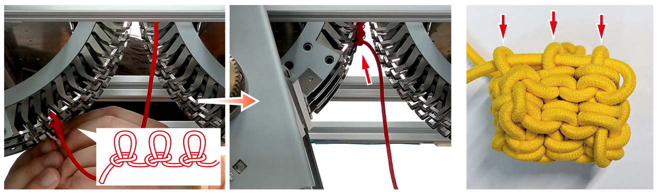
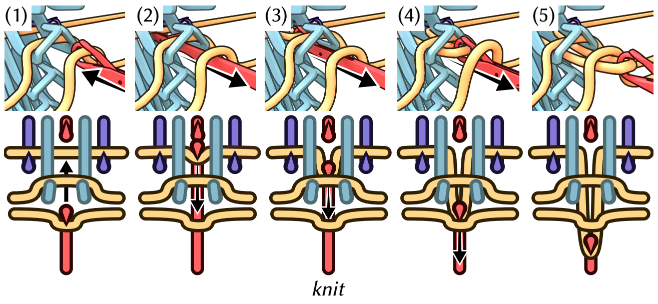


Fig. 9. *Left*: cast-on loops are manually made and hung onto the holder hooks for the first row of the first layer. The holders are rotated downward for the manual intervention and then back to the bed position. *Right*: the simplest method to bind off is passing the yarn through the last-row loops.



Drop. The **drop** instruction is the same as **knit** without yarn. No new loops are formed, and any loops on the needle drop off the tip. A loop on a holder hook can be dropped by transferring it to a needle and then dropping it from the needle.

Roll. The **roll** instruction rotates the holders to align the hooks of the given row with the needle bed. Such rotations happen implicitly during other instructions but **roll** can also be called explicitly, *e.g.*, to make space for manual intervention.

Pause. The **pause** instruction is used to pause the machine movement for manual intervention such as cast-on or bind-off.

4.4 Fabrication Capabilities

Our current prototype machine can knit cuboids and prisms. Cuboids are made by simply repeating the same number of rows per layer, while prisms are made by changing the number of rows per layer. For example, our machine can form a triangular prism by decreasing the number of rows as it progresses to the upper layers (Figure 14). A detailed explanation of the procedure can be found in Appendix A.

One could also make prisms by changing the number of stitches per row and keeping the number of rows per layer fixed. However, changing the number of stitches per row is more difficult mechanically, so we currently vary only the number of rows per layer. More detail on the challenges involved and the machine features necessary to create different classes of shapes are discussed in Section 7.3.

Cast On & Bind Off. In traditional knit fabric, the first and last row of stitches lack neighboring loops to keep them stable, so they are formed from special self-stabilizing stitches known as the *cast on* and *bind off* respectively. Similarly, the first and last row of stitches in a solid knit object must be cast on and bound off for stability. At the moment these operations are performed via manual

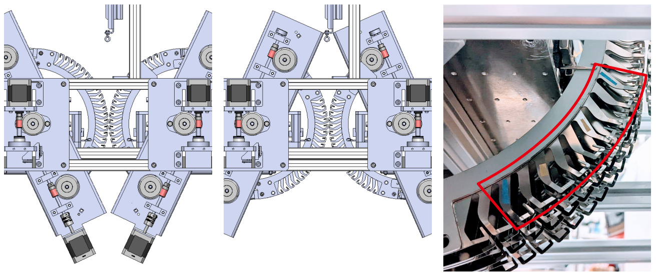


Fig. 10. The top and bottom part of the holder assemblies collide with each other if they are rotated too far (*left* and *center*), which makes nine hooks available, from the third to the eleventh from the bottom (*right*).

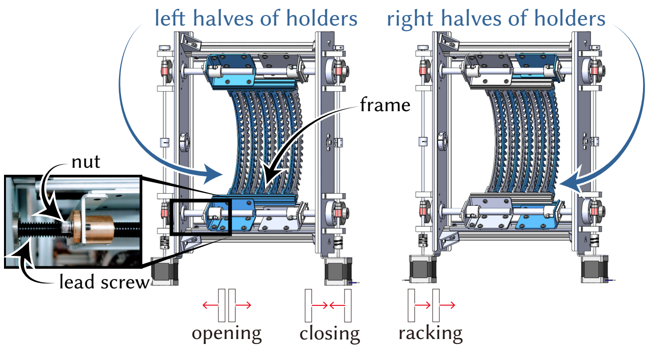


Fig. 11. Our machine's mechanism to open, close, and rack holders.

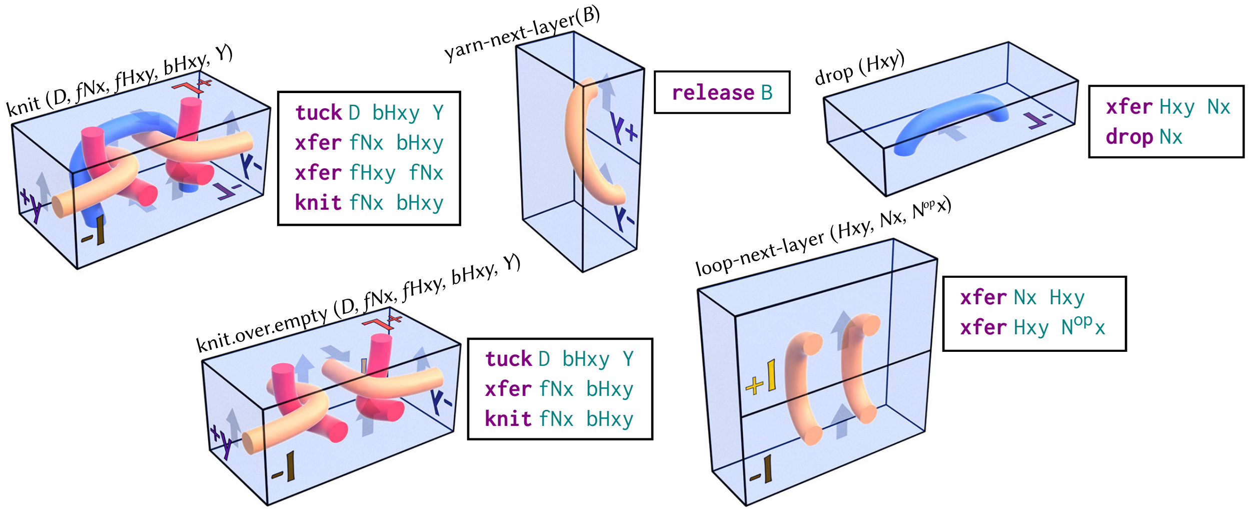
intervention (Figure 9). Note, though, that only the first row requires such stabilization: the rest of the stitches in the first layer will are automatically stable and require no further intervention.

Object Size. The current prototype has three latch needles on each bed, so it can work with a maximum of three stitches in the course direction. Each holder has eighteen hooks, but only nine hooks are available for use due mechanical constraints on the range of rotational motion (Figure 10). Because the last row of a layer is comprised of active loops which are to be used as the first row of the next layer, these nine available hooks can form eight rows of stable loops at maximum. Also, as discussed in Figure 26, the first layer and the row-increasing layer use one additional row. Hence, in case of cuboids, the current prototype can knit seven rows at most. There is no restriction on the number of layers in an object.

4.5 Mechanical Design

The machine is composed of steel and aluminum sheet metal parts, aluminum machined parts, and off-the-shelf mechanical and electronic components including aluminum rails, lead screws, gears, motors, and motor drivers.

Each needle is controlled by a bipolar stepper motor, 42SHD4404-24, via a lead screw. The current prototype has three needles per bed. Mechanically, all the right halves of the holders are connected together and controlled as one component, and so are the left halves. Each set of the right and left halves is controlled by a stepper motor, 42SHD4404-24, via worm gear, worm wheels, and lead screws on


knit (D, fNx, fHxy, bHxy, Y)
yarn-next-layer(B)
drop (Hxy)
loop-next-layer (Hxy, Nx, N^op x)
knit.over.empty (D, fNx, fHxy, bHxy, Y)
+y
-y
+l
-l
+L
-L

```
tuck D bHxy Y
xfer fNx bHxy
xfer fHxy fNx
knit fNx bHxy
```

```
release B
```

```
xfer Hxy Nx
drop Nx
```

```
tuck D bHxy Y
xfer fNx bHxy
knit fNx bHxy
```

```
xfer Nx Hxy
xfer Hxy Nopx
```

Fig. 12. Augmented stitch volume blocks and their associated program fragments. These blocks can be attached along their faces to represent solid knitting.

the top and the bottom (Figure 11). Rotation of these lead screws to the opposite direction widens or narrows the distance between the halves of the holders, which makes the holders open and closed. Rotation to the same direction makes the right halves and the left halves slide to the same direction, which is *racking* (detailed in Section 7.4).

The entire holder open-and-close unit described above is supported by rotational shafts which are controlled by stepper motors, 42SHD4404-24, via worm gears and worm wheels. This mechanism determines the rotational position of the holders. The motors are controlled by an Arduino Mega 2560 alternative (Whats Next? Green - WN00002) with DRV8825 stepper motor drivers.

5 AUGMENTED STITCH VOLUMES

We represent solid knitting patterns as *augmented stitch volumes*, volumetric meshes which encode individual stitches as blocks annotated with yarn geometry and machine instructions. The faces of these blocks are annotated with information about the yarns or loops which pass through the face, allowing one to infer dependencies between the blocks and verify the topological validity of the pattern. Augmented stitch volumes are a direct 3D generalization of the augmented stitch meshes introduced by Narayanan et al. [2019], which represent surface knitting patterns as polygonal surface meshes annotated with analogous yarn geometry, machine instructions, and dependency information.

Block types. Several basic block types are illustrated in Figure 12, along with their annotations. A full mesh built out of these blocks can be seen in Figure 13. The most fundamental block is the `knit` block (*top left*), which makes a standard solid knit stitch. We also provide a special `knit` block with no loops coming from beneath to model the first layer of a solid knit object (*bottom left*). The `loop-next-layer` block carries a loop of yarn from the ending edge of one layer to the starting edge of the next (*bottom right*), while the `yarn-next-layer` block carries the yarn from the final block of one layer to the starting block of the next (*top center*). The drop block finishes off a loop at the top of the object (*top right*). We also use special `yarn-in` and `yarn-out` blocks to denote the beginning and end of the yarn, and `cast-on` and `bind-off` blocks to denote the special stitches used to cast on the first row in a solid knit object and to stabilize the last row, although these operations must be done manually on our current solid knitting machine.

Block annotations. Augmented stitch volumes store program fragments associated with each block. The programs consist of a sequence of instructions which describe how to manipulate any loops and yarn which enter the block in order to form the loops and yarn which exit the block.

The program fragments associated with blocks are *unscheduled*, meaning that they do not use to specific locations on the solid knitting machine. Instead, they are written in terms of dummy variables `fNx`, `bNx`, `fHxy`, *etc.*, representing the front needle, back needle, front holder, *etc.*, which are used to make the stitch. During instruction generation, blocks are scheduled to concrete needle locations on the machine which are substituted in for these placeholder values.

Face annotations. Two blocks in a stitch volume may be attached by the yarn, by a wale-wise loop, or by a layer-wise loop. We annotate faces based on which type of connection they support. Faces labeled *yarn-in* (abbreviated as `-y`) represent locations where yarn can enter the block, and may only be attached to faces labeled *yarn-out* (`+y`). Similarly, faces labeled *wale-loop-in* (`-l`) may only attach to faces labeled *wale-loop-out* (`+l`), and faces labeled *layer-loop-in* (`-L`) may only attach to faces labeled *layer-loop-out* (`+L`). Finally, some faces may not admit any connections, in which case they are labeled with an `x`. We refer to such `x` faces as *unpairable* and all other faces as *pairable*. Connected faces induce a directed graph on the blocks of a stitch volume, which provides ordering information when generating instructions for the solid knitting machine.

Each face is also labeled with an *up* direction indicating how it should be attached to other blocks: when two blocks are attached along a face, their up directions must be aligned. In our implementation we provide several variants of the `knit` block with different up directions on their faces depending on whether the stitches are formed in the same direction or the opposite direction as the underlying stitches in the previous layer.

5.1 Instruction Generation

Our instruction generation algorithm is split into three passes: the first schedules each block onto a needle and holder and assigns it a direction, the second traverses the stitch volume and emits the instructions associated with each block, and the third reorders the emitted instructions to group instructions into distinct passes.

To schedule the blocks, our code greedily assigns needle and holder locations while enforcing the constraint that blocks connected by `y` faces must be scheduled onto adjacent needles (and thus onto horizontally-adjacent holder hooks), blocks connected by `l` faces must be scheduled onto vertically-adjacent holder hooks, and that the block direction must switch following each `yarn-next-row` or `yarn-next-layer` block. Then our code traverses the blocks of the pattern in topological order to generate *solid knitout* instructions (Algorithm 1). At each block, the code substitutes the block's assigned needle, holder, and direction into the block's associated program fragment and emits the resulting solid knitout code. Our topological ordering ensures that blocks are traversed layer by layer. Within each layer, blocks are traversed in yarn order. Once all such blocks have been processed, the code processes any `loop-next-layer` blocks and `drop` blocks, before processing the `yarn-next-layer` block and proceeding to the next layer. This traversal is managed using a priority queue.

Finally, our code groups generated instructions into passes for efficiency. For example, a row of knit stitches is formed by performing all **`tuck`**s first, then all **`xfer`**s from needles to holders, followed by any **`xfer`**s from holders to needles and finally performing all **`knit`**s. This reordering also allows our code to perform drops while no other loops are held on the needles, as discussed in Appendix A.

Preconditions. This instruction generation routine assumes that our stitch volumes satisfy three knittability constraints:

(1) Each pairable face must be connected to another face of the complementary type.
(2) The directed graph of blocks must be acyclic.
(3) The pattern must be composed of a single strand of yarn.

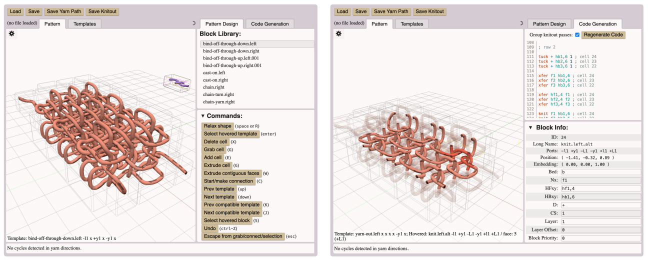
The first two conditions enforce topological validity of the stitch volume, ensuring that there will be no closed loops of yarn. They must be satisfied by any knittable stitch volume. By contrast, the third constraint is currently necessary since the solid knitting machine only has a single yarn carrier, but could be relaxed if designing patterns for future solid knitting machines or for hand knitters.

5.2 Pattern Design

We implemented a tool to support the design of solid-knit objects by connecting together different types of stitch volume blocks (Figure 13). The interface guides the user by snapping blocks to form connections between compatible faces. Individual knit layers can be isolated to facilitate edits within the object. Instructions are then generated for the machine via the method described in Section 5.1.

Algorithm 1: TraverseBlocks (pattern)

```
Input  : pattern                                   // collection of blocks
Output : $\mathcal{P}$                             // solid knitout program
$\mathcal{P}$ ← List()                             // initialize instruction list
toVisit ← PriorityQueue()                          // initialize priority queue
for block in pattern do                            // record degrees for topological sort
    block.dependencies = InDegree(block)
start ← FindBeginYarn(pattern)                     // locate begin-yarn block
toVisit.push(start, priority = 3)
while toVisit is not empty do                      // iterate in topological order
    currBlock ← toVisit.popTop()                   // get top priority block
    $\mathcal{P}$.push(currBlock.instructions)
    if currBlock.name = end-yarn then
        return $\mathcal{P}$                       // pattern finishes at yarn end
    foreach adj of currBlock.outConnections do
        adj.dependencies −= 1                      // record incoming edge
        if adj.dependencies == 0 then
            // Default priorities (0 is lowest priority)
            //    0: yarn-next-layer
            //    1: drop
            //    2: loop-next-layer
            //    3: all other blocks
            toVisit.push(adj, priority=adj.priority)
```



bind-off-through-down.left
bind-off-through-down.right
bind-off-through-up.left.001
bind-off-through-up.right.001
cast-on.left
cast-on.right
chain.right
chain-turn.right
chain-yarn.right

Template: bind-off-through-down.left -l1 x +y1 x -y1 x

No cycles detected in yarn directions.

```
; row 2

tuck + hb1,6 1 ; cell 24
tuck + hb2,6 1 ; cell 23
tuck + hb3,6 1 ; cell 22

xfer f1 hb1,6 ; cell 24
xfer f2 hb2,6 ; cell 23
xfer f3 hb3,6 ; cell 22

xfer hf1,4 f1 ; cell 24
xfer hf2,4 f2 ; cell 23
xfer hf3,4 f3 ; cell 22

knit f1 hb1,6 ; cell 24
```

ID:	24
Long Name:	knit.left.alt
Ports:	-l1 +y1 -L1 -y1 +l1 +L1
Position:	(-1.41, -0.32, 0.89)
Embedding:	(0.00, 0.00, 1.00)
Bed:	b
Nx:	f1
HFxy:	hf1,4
HBxy:	hb1,6
D:	+
CS:	1
Layer:	1
Layer Offset:	0
Block Priority:	0

Template: yarn-out.left x x x x -y1 x; Hovered: knit.left.alt -l1 +y1 -L1 -y1 +l1 +L1 / face: 5 (+L1)

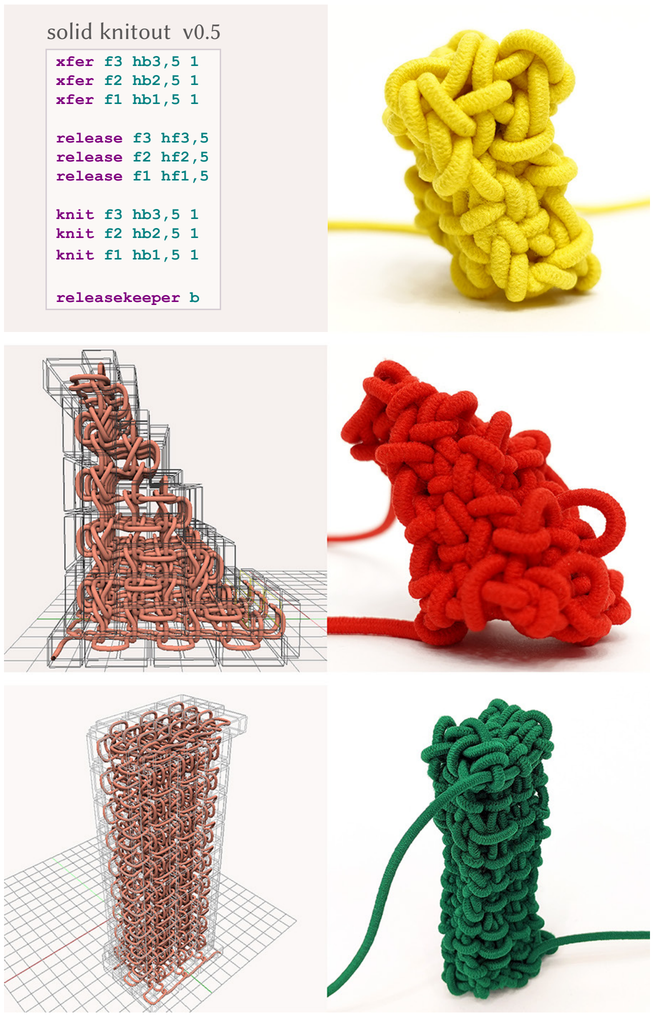
No cycles detected in yarn directions.

Fig. 13. *Left*: a stitch volume built from blocks in our design interface. *Right*: users can focus on a single layer of stitches, or generate a sequence of instructions for the whole pattern.

We chose to build a block-based pattern design tool rather than an automatic pattern generation tool, like slicer software for 3D printers, to make the best use of our current machine's limited layer size (three stitches × eight rows – Section 4.4).

Before generating instructions, our design tool must verify the three preconditions listed above. The first is enforced by construction, since the tool only allows compatible faces to be connected. To verify the second, we run Kahn's algorithm for cycle detection and display a warning if any cycles are found [Kahn 1962]. Finally, verifying the third condition amounts to checking that the pattern uses exactly one `begin-yarn` block and one `end-yarn` block.

The user can inspect the needle and holder locations that the scheduler assigns to each block, and can modify each block's priority or specify an offset requesting that the block's instructions be



solid knitout v0.5

```
xfer f3 hb3,5 1
xfer f2 hb2,5 1
xfer f1 hb1,5 1

release f3 hf3,5
release f2 hf2,5
release f1 hf1,5

knit f3 hb3,5 1
knit f2 hb2,5 1
knit f1 hb1,5 1

releasekeeper b
```

Fig. 14. Rectangular and triangular prisms constructed by the solid knitting machine. The yellow, small rectangular prism was designed by hand writing the previous version of solid knitout code (*top*).

emitted with a later layer. This is particularly helpful when scheduling **drop** instructions, which may need to be performed in later layers to avoid needle contention. More discussion of the scheduling concerns surrounding **drop**s can be found in Appendix A.

6 RESULTS

Figure 14 shows some machine-knit results: a short rectangular prism comprised of 3 stitches, 3 rows, and 8 layers (*top*); a triangular prism comprised of 3 stitches, 9 layers starting with the 6-row bottom layer and ending with the 1-row top layer (*middle*); and a long rectangular prism comprised of 3 stitches, 3 rows, and 20 layers (*bottom*). This triangular prism is made by dropping the stitches in the last row of odd layers, as detailed in Appendix A.2. Note that we hand-wrote instructions for the short rectangular prism in an older version of solid knitout, and we modified some parts of the generated instructions for the triangular prism, which was made before the instruction generation code was finalized. The instructions for the long rectangular prism were all automatically generated.

To execute a solid knitting program, our control software groups the instructions into passes—like carriage passes on a V-bed knitting machine—and sends the machine one pass at a time, pausing after each pass for the operator to inspect the result. This inspection was important as the machine often experienced mechanical errors requiring manual intervention to fix. We sent the next pass only after verifying that there was no error or fixing any errors that occurred. However, we occasionally missed errors—for instance, the loop in the inset was not properly knit through. Overall, mechanical errors occurred in about 53% of passes[1]

Stitch Size. In our design tool a stitch is treated as a cuboid with a ratio of 2.25 (course direction) : 1.125 (wale direction) : 1 (layer direction). This was based on the measurement of a hand-knit cuboid with 8 stitches × 8 rows × 8 layers was 54 mm × 27 mm × 24 mm.

The machine-knit cuboid of 3 stitches × 3 rows × 8 layers in Figure 14 (*top*) measured 34 mm × 20 mm × 41 mm, corresponding to a ratio of $34 : 20 : 15.4 \approx 2.2 : 1.3 : 1$. Its shape is close to our hand-knit exemplar, but knitting a gauge swatch would be required for more precise reproduction of a shape—just as in traditional knitting.

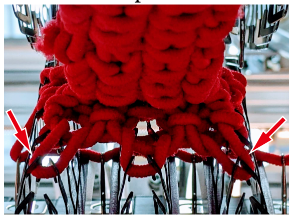
7 FUTURE WORK

In this section we begin by discussing strategies for reducing the occurrence of mechanical errors in the solid knitting machine (Sections 7.1 and 7.2), before proceeding to extensions that would allow the machine to fabricate a wider array of shapes (Sections 7.3-7.6) using a variety of materials (Section 7.7).

7.1 Loop Shape Consistency

The most frequent mechanical error is a failure to transfer a loop between the needle and the holder hook. The needle fails to grip the loops held by the holder or fails to transfer its loop onto the holder.

The root cause of these errors is that the shape of the loops—and of the hooks holding them—varies. Due to the elasticity of the yarn, the loops at the ends of rows deform significantly. One solution to this problem would be to use more rigid holders.



However, to do so, one would also need to develop needles that can change the size of the loop. The current transferring method relies on the fact that the holders are made of flexible flat springs, and would become impossible on a machine with more rigid holders. Instead, one might want to use needles with a mechanism for opening and closing, allowing the needles to be narrower than the holder pairs when releasing a loop from the holder and wider than the holders when transferring it to the holder as below.

[1]To be precise, we found errors in 180 out of 342 passes that we logged. Note that each pass typically consisted of several instructions, *e.g.*, `knit f1 hb1,7; knit f2 hb2,7; knit f3 hb3,7` is one pass.

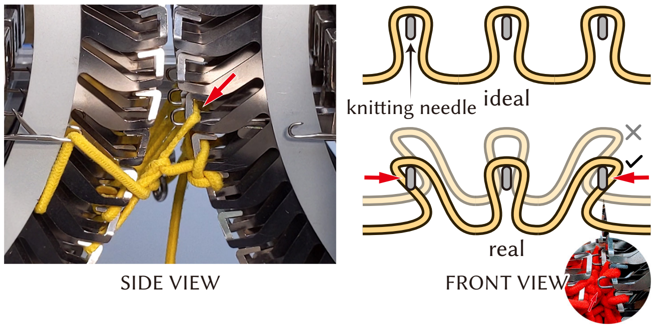
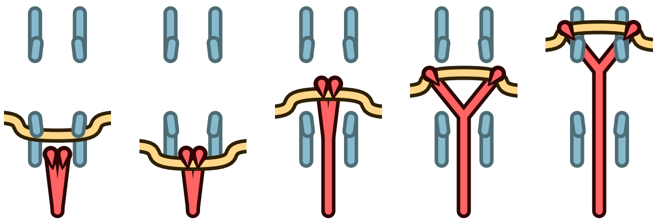
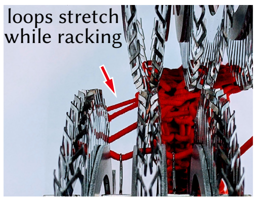


Fig. 15. The rotational position of the holder needs to be finely adjusted for successful transferring.



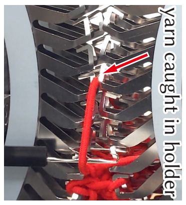
Currently, we manage this issue by adjusting firmware parameters. For example, in case the needle is releasing a loop from the holder hook, especially at the ends of rows, the needle may reach outside of the loop and will not be able to grip the loop. Therefore, the rotational position of the holder should be adjusted so that the needle advances just below the upper end of the loop (Figure 15). Of course, if the holder is rotated too far down, the needle will pass above the loop and transferring will fail in this case as well.



In reality, loop shapes differ depending on each holder hook shape due to holder bending and assembly accuracy and deflection, as well as differences in yarn tension when the loops are formed. This limits the solution using parameter adjustments.

Furthermore, if racking (Section 7.4) is added, the shape patterns of loops will diverge more than they do now, and the required parameter tuning will become infeasibly complicated. Therefore, we believe that the correct long-term solution to this problem is to make the loop shape consistent by mechanical improvement.

7.2 Tension Control



Currently, we manually rotate the spool of yarn to control tension. Careful tension control is essential because yarn tension affects loop shape consistency. Furthermore, if the yarn is too loose, it gets caught in unintended places during tuck operations. If it is too tight, however, it makes the needle's motor step out (*i.e.*, stop rotating). But tension control could likey be automated with a mechanism like those used on regular knitting machines (Figure 16).

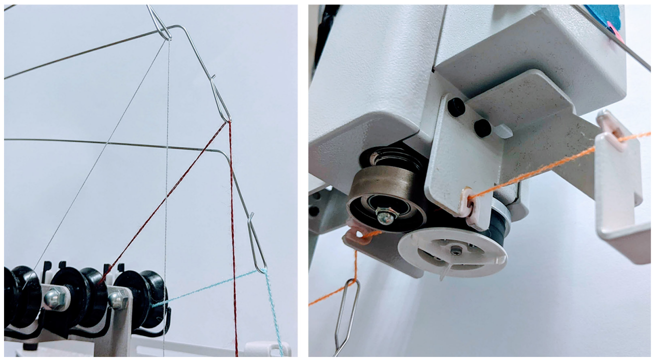


Fig. 16. Regular knitting machines have a mechanism to passively or actively control yarn tension using spring. For example, a Shima Seiki SWG091N2 knitting machine has both of them. The passive mechanism uses springs (*left*) while the active one uses motors in addition to springs (*right*). The solid knitting machine also needs a feature to control yarn tension.

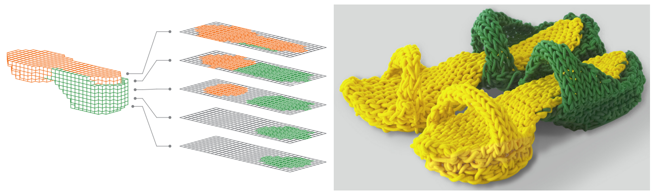


Fig. 17. *Left*: a model of sandals is converted into a solid knitting pattern by voxelizing with rectangular elements whose aspect ratio matches the stitches then slicing into layers. *Right*: solid-knit sandals constructed by hand from this pattern.

7.3 Shapes & Structures

In principle, one can solid knit any shape by slicing it into layers just as in standard 3D printing (Figure 17). However, as discussed in Section 4.4, our prototype only makes prisms at the moment. In this section, we discuss the steps involved in solid knitting different shapes and the additional capabilities required to fabricate them on a solid knitting machine.

To create a cuboid, our solid knitting machine stacks a sequence of identical rectangular layers of fabric. The number of stitches per row determines the width (length in the course direction), the number of rows determines the depth (length in the wale direction), and the number of layers determines the height (length in the layer direction). By varying these quantities over time, one can create more complex shapes. We classify the shapes into four types: cuboids, prisms, pyramids, and forked shapes (Figure 18). Our current machine can make the first two.

Cuboids. In order to knit a cuboid, one keeps the number of stitches and the number of rows a constant in all layers.

Prisms. In order to knit a prism, one can keep the number of stitches per row constant while varying the number of rows per layer. For example, as shown in Figure 18 (*center left*) a triangular prism can be created by decreasing the number of rows per layer. Of course, one could instead vary the number of stitches per row

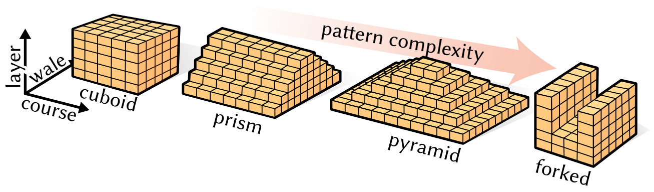


Fig. 18. Making the same number of stitches each layer creates a cuboid (*left*). Changing the number of rows per layer creates a prism (*center left*), and changing both the number of rows per layer and the number of stitches per row creates a pyramid (*center right*). More complicated techniques are required to make a forked shape (*right*).

while keeping the number of rows constant. However, it is easier to change the number of rows, which only requires changing the number of the holder hooks used per layer, rather than changing the number of stitches per row, which requires racking (Section 7.4). The difficulty is that decreasing the number of stitches in a row leaves behind an extra unstable loop, which must somehow be secured, whereas decreasing the number of rows in a layer leaves behind a collection of stable loops which can be dropped or unstable loops which can be secured without racking. Consequently, the current prototype can form prisms by varying the number of rows per layer, but cannot change the number of stitches per row.

Pyramids. In order to knit a pyramid, it is necessary to change both the number of stitches per row and the number of rows per layer. For example, a rectangular pyramid can be created by decreasing both the number of rows and the number of stitches in each layer to knit a rectangle that is smaller than the one in the previous layer (Figure 18, *center right*; Figure 20). Here the racking feature is absolutely necessary.

Forked shapes. In order to knit shapes forked in the wale (Y-) direction and/or the layer (Z-) direction, the machine has to have multiple yarn carriers to treat multiple yarns. When forked in the wale direction, the cross-section in the height direction, *i.e.*, one of the layers, is U-shaped. Given that a layer is filled by scanning in the course direction, it is impossible for the shape to be composed of only one yarn (Figure 19, *middle*). When forked in the layer direction, the cross section has disconnected parts, which also means it is impossible to knit with only one yarn (Figure 19, *right*). Since the current prototype has only one yarn carrier, it can only handle one yarn. Industrial knitting machines have multiple yarn carriers. This enables them to knit multicolored fabrics and complex shapes [McCann et al. 2016]. Similarly, solid knitting also requires multiple yarn carriers to knit shapes with branches in the wale and/or layer direction(s) as well as multicolored solid-knit objects.

Interfering shapes. In addition to the shapes described above, we need to consider the risk of collision when the object has a protruding cantilever from a large decrease of rows (Figure 21). It may be necessary to add a mechanism to hold the object down so that the cantilever remains bent away from the holders, similar to weights used on hand knitting machines.

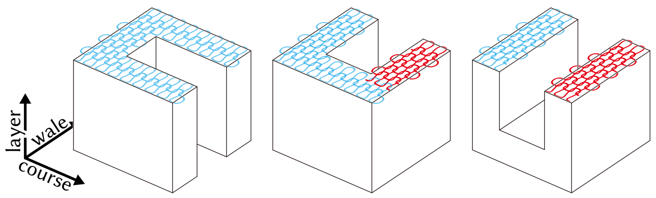


Fig. 19. Some forked shapes require multiple yarns.

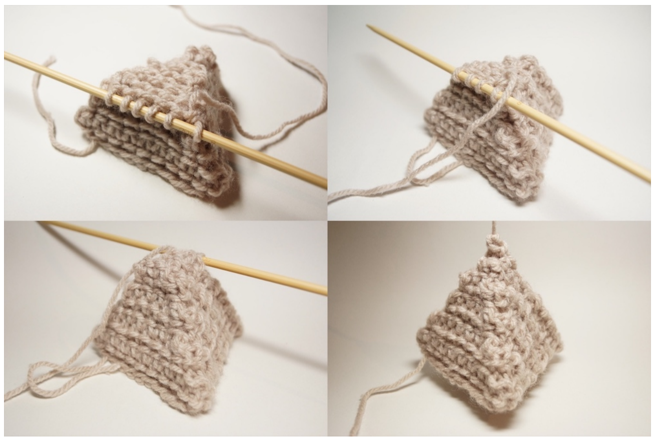


Fig. 20. Solid-knit pyramid constructed by hand.

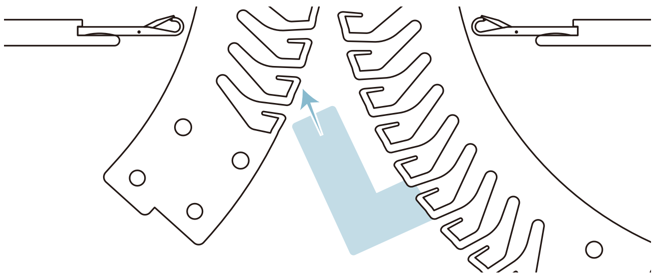


Fig. 21. A protruding cantilever could collide with the opposite side of holder.

7.4 Increases & Decreases

When knitting traditionally, one can shape the knit fabric by increasing or decreasing the number of stitches in each row. On standard V-bed knitting machines, these techniques are enabled by the *racking* mechanism, which translates one bed relative to the other. When combined with the ability to transfer loops between the beds, racking allows a user to stack loops on top of each other to decreases the number of stitches per row, or to spread loops apart on the bed to increase the number of stitches per row.

Our solid knitting machine is designed to accomplish increases and decreases by translating the holders horizontally, instead of the needle beds. As was briefly mentioned in Section 4.5, the holders are able to translate horizontally as an extension of their ability to open and close. Consequently, one could imagine trying to perform

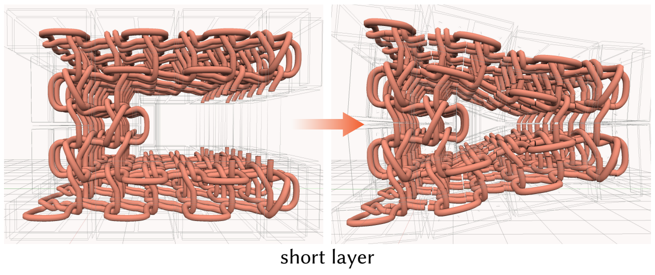


Fig. 22. A rough simulation of a short layer using our design tool.

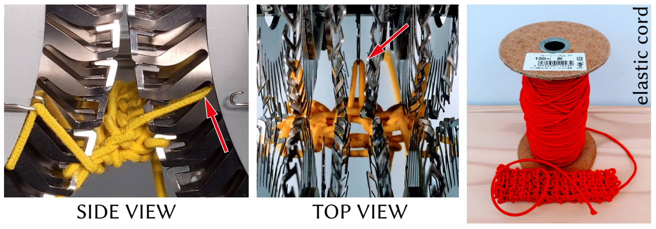


Fig. 23. While transferring a loop from a holder onto a needle, our solid knitting machine stretches the loop to several times its original length (*left, center*). To allow these deformations, we use elastic cord as our yarn (*right*).

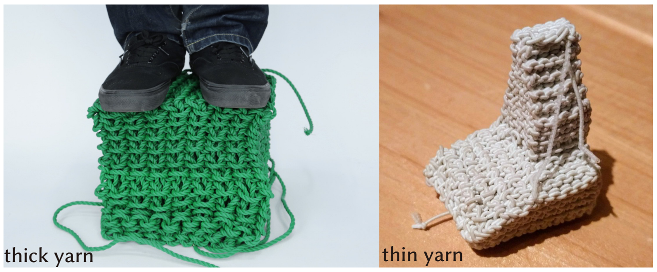


Fig. 24. Changing the yarn material changes the properties of solid-knit objects. Here we show two hand-knit examples. Using a thicker yarn can create sturdy objects quickly, but the results will be lower resolution (*left*). Conversely, using thinner yarn can create higher resolution objects but takes significantly longer (*right*).

increases or decreases by transferring loops onto the holders and using the holders to stack or spread out these loops. However, as mentioned in Section 7.1, such an approach to racking introduces deformations of the loop shapes and the holder hooks which cause mechanical issues with transferring between the needles and the holders. It would be interesting to explore other approaches to racking which incur less yarn distortion.

7.5 Short Rows & Short Layers

Short rows are another technique traditionally used by knitters to shape knit garments. The key idea is to leave loops in some rows un-knit—only knitting through them several rows later—causing the fabric to bend. Short rows are classically used in the heel of a sock and can be used to shape general 3D surfaces, as illustrated by McCann et al. [2016, Figures 3 & 15] and Liu et al. [2021, Figure 2].

Analogously, solid knitters can use a *short layer* technique—in which some rows of loops are only knit through in a later layer—to bend the shapes of solid-knit objects. In theory, short layers could be created using our current solid knitting machine, but they would induce even more loop shape inconsistencies which are not supported by our machine firmware (Section 7.1). Note, though, that short layers can be represented by our current augmented stitch volumes, since they are composed of the same basic stitches used in standard solid knitting. Using the design tool, it is possible to roughly simulate such deformed shapes (Figure 22).

7.6 Stitch Volume Singularities

Our stitch volumes, which build solid knitting patterns out of cubical stitches, can be viewed as regular *hexahedral meshes*. Extending the solid knitting machine to support shaping as discussed in Sections 7.4 and 7.5 would allow us to introduce *singularities* into these meshes, which is done in standard hexahedral meshing to produce high-quality meshes [Nieser et al. 2011, Section 2.2]. However, even in the general setting of hexahedral meshing, the exact behavior of these singularities and the constraints which they must satisfy to be meshable are an active field of study [Liu and Bommes 2023]. It would be interesting to investigate the validity of singularities in the special case of solid knitting. Solid knitting patterns are more restrictive than general hexahedral meshes, since one may only attach together faces of compatible types; this restriction eliminates a large class of symmetries which introduce complications into the mathematical theory of hexahedral mesh singularities [Palmer et al. 2020, Section 3.1], so the structure of singularities allowed in solid knitting patterns may be significantly simpler than the general case.

7.7 Materials

While solid knitting, our machine stretches the yarn loops dramatically. For instance, when transferring a loop from a holder hook onto a needle, the loop is pulled backwards and stretched to several times its original length (Figure 23, *left, center*). Consequently, we use elastic cord as our yarn. Specifically, our current prototype uses a 3 mm diameter *Kintenma Woolly Rubber* manufactured by Kawamura Seichu Co, Ltd. (Figure 23, *right*). The cord's elasticity also helps to keep loops on the holder hooks, preventing them from dropping off unintentionally. While solid knitting machines will probably need to keep using elastic cord in the near future, one can still experiment with different cords of different thicknesses (Figure 24).

8 CONCLUSION

This paper presented solid knitting, a new way of making dense, volumetric objects, and a solid knitting machine to automate the process. We also introduced a low-level language for representing machine instructions and a stitch volume data structure to encode solid knitting patterns as assemblies of blocks annotated with yarn information and machine instructions. The key feature distinguishing our solid knitting machine from traditional V-bed machines is its 2D array of holder hooks, used to hold all of the stitches in a layer of solid knitting.

There are many remaining open questions about solid knitting. Regarding the solid knitting machine, it is still not obvious how to effectively perform shaping via increases, decreases, short rows or short layers. Future developments could also include extensions to handle a variety of different yarn materials in order to fabricate objects at different scales. Introducing new shaping operations to the solid knitting machine would also lead to new questions in pattern design regarding the shapes of these stitches and their properties.

We believe that solid knitting should be a new and sustainable 3D printing technology. After the solid knitting machine becomes capable of knitting chairs, for example, this will enable a chair to be unraveled to its original yarn and to be re-knit into a table. Moreover, if there are the machines everywhere as a form of infrastructure, it would allow for the transmission of data for objects, rather than the objects themselves. For instance, a person who is moving to a new apartment can unwind the solid-knit furniture of the previous resident and reknit it into their own by using its data. Although our current achievements are still a long way from realizing this vision, this paper functions as an initial step toward the goal.

ACKNOWLEDGMENTS

Part of this work was supported by the INNO-vation Program of the Ministry of Internal Affairs and Communications, Japan, the 100BANCH GARAGE Program of Panasonic Holdings Corporation, and the MITOU Advanced Program of the Information-technology Promotion Agency, Japan. This material is partially based upon work supported by the National Science Foundation under Award No. 1955444, and upon the master's thesis and patent of our first author [Hirose 2014, 2022].

REFERENCES

Lea Albaugh, Scott Hudson, and Lining Yao. 2019. Digital Fabrication of Soft Actuated Objects by Machine Knitting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, New York, NY, USA, 1–13. https://doi.org/10.1145/3290605.3300414

Lea Albaugh, James McCann, Scott E. Hudson, and Lining Yao. 2021. Engineering Multifunctional Spacer Fabrics Through Machine Knitting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1–12. https://doi.org/10.1145/3411764.3445564

Michal Edelstein, Hila Peleg, Shachar Itzhaky, and Mirela Ben-Chen. 2022. AmiGo: Computational Design of Amigurumi Crochet Patterns. In *Proceedings of the 7th Annual ACM Symposium on Computational Fabrication (SCF '22)*. ACM, Article 5, 11 pages. https://doi.org/10.1145/3559400.3562005

William Felkin. 1867. *A History of the Machine-Wrought Hosiery and Lace Manufactures*. Longmans, Green, and Co., London. https://archive.org/details/historyofmachine00felkuoft/

Nils Grimmelsmann, Johannes Fiedler, and Andrea Ehrmann. 2018. Crochet machine. https://patentscope.wipo.int/search/en/detail.jsf?docId=WO2018114025 PCT Publication No. WO2018114025, Published June 28th., 2018.

Varvara Guljajeva and Mar Canet Sola. 2013. Knitic — The Revolution of Soft Digital Fabrication. http://dh2013.unl.edu/abstracts/ab-173.html

Varvara Guljajeva and Mar Canet Sola. 2014. *Circular Knitic*. http://var-mar.info/circular-knitic/

Yuichi Hirose. 2014. *Adding Undo-ability to Fabrication Process through the Development of the 3-D Knitting Machine*. Master's thesis. Keio University, Kanagawa, Japan.

Yuichi Hirose. 2022. Knitting machine and knitting method. https://patentscope.wipo.int/search/en/detail.jsf?docId=JP337722897 Japan Patent No. 7103557, Published July 2nd., 2020, Granted July 11th., 2022.

Megan Hofmann, Lea Albaugh, Ticha Sethapakadi, Jessica Hodgins, Scott E. Hudson, James McCann, and Jennifer Mankoff. 2019. KnitPicking Textures: Programming and Modifying Complex Knitted Textures for Machine and Hand Knitting. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology (UIST '19)*. ACM, 5–16. https://doi.org/10.1145/3332165.3347886

Megan Hofmann, Lea Albaugh, Tongyan Wang, Jennifer Mankoff, and Scott E Hudson. 2023. KnitScript: A Domain-Specific Scripting Language for Advanced Machine Knitting. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23)*. ACM, Article 21, 21 pages. https://doi.org/10.1145/3586183.3606789

Scott E. Hudson. 2014. Printing Teddy Bears: A Technique for 3D Printing of Soft Interactive Objects. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 459–468. https://doi.org/10.1145/2556288.2557338

Yuki Igarashi, Takeo Igarashi, and Hiromasa Suzuki. 2008. Knitting a 3D Model. *Computer Graphics Forum* 27, 7 (Oct. 2008). https://doi.org/10.1111/j.1467-8659.2008.01318.x

Benjamin Jones, Yuxuan Mei, Haisen Zhao, Taylor Gotfrid, Jennifer Mankoff, and Adriana Schulz. 2021. Computational Design of Knit Templates. *ACM Transactions on Graphics* 41, 2, Article 16 (Dec. 2021), 16 pages. https://doi.org/10.1145/3488006

Arthur B Kahn. 1962. Topological sorting of large networks. *Commun. ACM* 5, 11 (1962), 558–562. https://doi.org/10.1145/368996.369025

Alexandre Kaspar, Kui Wu, Yiyue Luo, Liane Makatura, and Wojciech Matusik. 2021. Knit Sketching: From Cut & Sew Patterns to Machine-Knit Garments. *ACM Transactions on Graphics* 40, 4, Article 63 (July 2021), 15 pages. https://doi.org/10.1145/3450626.3459752

Kniterate. 2015. *Kniterate*. https://www.kniterate.com/

Jonathan Leaf, Rundong Wu, Eston Schweickart, Doug L. James, and Steve Marschner. 2018. Interactive Design of Periodic Yarn-Level Cloth Patterns. *ACM Transactions on Graphics* 37, 6, Article 202 (Dec. 2018), 15 pages. https://doi.org/10.1145/3272127.3275105

Kwangho Lee. 2009. *obsession series*. http://www.kwangholee.com/

Heng Liu and David Bommes. 2023. Locally Meshable Frame Fields. *ACM Transactions on Graphics* 42, 4 (2023). https://doi.org/10.1145/3592457

Zishun Liu, Xingjian Han, Yuchen Zhang, Xiangjia Chen, Yu-Kun Lai, Eugeni L. Doubrovski, Emily Whiting, and Charlie C. L. Wang. 2021. Knitting 4D garments with elasticity controlled for body motion. *ACM Transactions on Graphics* 40 (Aug. 2021), 1–16. https://doi.org/10.1145/3450626.3459868

Yiyue Luo, Kui Wu, Tomás Palacios, and Wojciech Matusik. 2021. KnitUI: Fabricating Interactive and Sensing Textiles with Machine Knitting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1–12. https://doi.org/10.1145/3411764.3445780

Yiyue Luo, Kui Wu, Andrew Spielberg, Michael Foshey, Daniela Rus, Tomás Palacios, and Wojciech Matusik. 2022. Digital Fabrication of Pneumatic Actuators with Integrated Sensing by Machine Knitting. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. ACM, 1–13. https://doi.org/10.1145/3491102.3517577

James McCann. 2017. The "Knitout" (.k) File Format. [Online]. Available from: https://textiles-lab.github.io/knitout/knitout.html.

James McCann, Lea Albaugh, Vidya Narayanan, April Grow, Wojciech Matusik, Jennifer Mankoff, and Jessica Hodgins. 2016. A Compiler for 3D Machine Knitting. *ACM Transactions on Graphics* 35, 4 (2016), 11 pages. https://doi.org/10.1145/2897824.2925940

Mcor Technologies. 2014. Mcor IRIS HD Features & Specs.

Rahul Mitra, Liane Makatura, Emily Whiting, and Edward Chien. 2023. Helix-Free Stripes for Knit Graph Design. In *ACM SIGGRAPH 2023 Conference Proceedings (SIGGRAPH '23)*. ACM, Article 75, 9 pages. https://doi.org/10.1145/3588432.3591564

Georges Nader, Yu Han Quek, Pei Zhi Chia, Oliver Weeger, and Sai-Kit Yeung. 2021. KnitKit: A Flexible System for Machine Knitting of Customizable Textiles. *ACM Transactions on Graphics* 40, 4, Article 64 (July 2021), 16 pages. https://doi.org/10.1145/3450626.3459790

Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros, and James Mccann. 2018. Automatic Machine Knitting of 3D Meshes. *ACM Transactions on Graphics* 37, 3 (June 2018), 1–15. https://doi.org/10.1145/3186265

Vidya Narayanan, Kui Wu, Cem Yuksel, and James McCann. 2019. Visual knitting machine programming. *ACM Transactions on Graphics* 38, 4 (2019), 1–13. https://doi.org/10.1145/3306346.3322995

Matthias Nieser, Ulrich Reitebuch, and Konrad Polthier. 2011. Cubecover–parameterization of 3d volumes. *Computer graphics forum (SGP)* 30, 5 (2011), 1397–1406. https://doi.org/10.1111/j.1467-8659.2011.02014.x

OpenKnit. 2013. *OpenKnit*. https://openknit.org/

David Palmer, David Bommes, and Justin Solomon. 2020. Algebraic Representations for Volumetric Frame Fields. *ACM Transactions on Graphics* 39, 2, Article 16 (April 2020), 17 pages. https://doi.org/10.1145/3366786

Huaishu Peng, Jennifer Mankoff, Scott E. Hudson, and James McCann. 2015. A Layered Fabric 3D Printer for Soft Interactive Objects. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. 1789–1798. https://doi.org/10.1145/2702123.2702327

Gabriella Perry, Jose Luis García Del Castillo Y López, and Nathan Melenbrink. 2023. Croche-Matic: a robot for crocheting 3D cylindrical geometry. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, London, United Kingdom, 7440–7446. https://doi.org/10.1109/ICRA48891.2023.10160345

Mariana Popescu, Matthias Rippmann, Tom Van Mele, and Philippe Block. 2018. *Automated Generation of Knit Patterns for Non-developable Surfaces*. Springer Singapore,

Singapore, 271–284. https://doi.org/10.1007/978-981-10-6611-5_24

Richard Rutt. 1987. *The History of Hand Knitting*. Interweave. https://archive.org/details/historyofhandkni0000rutt

Shima Seiki. 2013. *SWG021N2/041N2/061N2/091N2 WHOLEGARMENT Knitting Machines*. https://www.shimaseiki.com/product/knit/swg_n2/

David J. Spencer. 2001. *Knitting technology : a comprehensive handbook and practical guide* (3rd ed.). Technomic, Lancaster, PA. https://doi.org/10.1533/9781855737556

Abigail Torres. 2012. *Electro-knit*. https://learn.adafruit.com/electroknit/overview

Rundong Wu, Claire Harvey, Joy Xiaoji Zhang, Sean Kroszner, Brooks Hagan, and Steve Marschner. 2020a. Automatic structure synthesis for 3D woven relief. *ACM Transactions on Graphics* 39, 4 (Aug. 2020). https://doi.org/10.1145/3386569.3392449

Rundong Wu, Joy Xiaoji Zhang, Jonathan Leaf, Xinru Hua, Ante Qu, Claire Harvey, Emily Holtzman, Joy Ko, Brooks Hagan, Doug James, François Guimbretière, and Steve Marschner. 2020b. Weavecraft: an interactive design and simulation tool for 3D weaving. *ACM Transactions on Graphics* 39, 6 (Dec. 2020), 1–16. https://doi.org/10.1145/3414685.3417865

Tianhong Catherine Yu, Riku Arakawa, James McCann, and Mayank Goel. 2023. uKnit: A Position-Aware Reconfigurable Machine-Knitted Wearable for Gestural Interaction and Passive Sensing using Electrical Impedance Tomography. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. 1–17. https://doi.org/10.1145/3544548.3580692

Cem Yuksel, Jonathan M. Kaldor, Doug L. James, and Steve Marschner. 2012. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Transactions on Graphics* 31, 4 (Aug. 2012), 1–12. https://doi.org/10.1145/2185520.2185533

Lara Zlokapa, Yiyue Luo, Jie Xu, Michael Foshey, Kui Wu, Pulkit Agrawal, and Wojciech Matusik. 2022. An integrated design pipeline for tactile sensing robotic manipulators. In *2022 International Conference on Robotics and Automation (ICRA)*. 3136–3142. https://doi.org/10.1109/ICRA46639.2022.9812335

A INCREASING & DECREASING ROWS

In this appendix, we describe how the solid knitting machine increases or decreases the number of rows knit in each layer (*e.g.*, when making prisms in Section 4.4). As a running example we consider an object which starts with three rows of stitches per layer, as was depicted in Figure 7, *right*. We begin with a schematic illustration of the process of knitting an ordinary layer (A.1), before detailing the modifications required to drop the last row (A.2), drop the first row (A.3), increase at the last row (A.4), or increase at the first row (A.5). The illustrations are shown in Figure 25. In principle, one can also increase or decrease in the middle of layers, but we restrict attention to these cases for simplicity.

A.1 Knitting a Standard Layer

The right holder starts holding three rows of stable loops plus a row of active loops which are also attached to the last row of stitches (1). We depict all active or stable loops, but omit completed loops from the diagram for visual clarity. The machine begins by transferring row d of active loops to the left holder, and rolls the right holder upwards to align the top row of stable loops with the transferred row of active loops (2). Next, the machine knits[2] through d and c to form e, and transfers the new row of loops onto the left holder (3). Then the machine knits through e and b to form f, and knits through f and a to form g.

A.2 Decrease at the Last Row

To decrease at the last row of a layer, we start in the scenario depicted in step (4) of A.1. Rows d and c, and e and b have been knit through, forming new rows e and f on the left holder, and leaving behind only row a on the right holder. Now, the machine can simply drop row a, which is stable (2). This leaves behind a layer consisting of the two stable rows d and e, and the one active row f (3), which the machine knits through, leaving behind two new stable rows and one new active row of loops on the right holder (4,5).

A.3 Decrease at the First Row

Decreasing at the beginning of a layer is slightly more complicated, since the machine cannot drop the active loops: instead, it must drop the first row of stable loops. We begin by depicting the machine holding three rows of stable loops and one row of active loops on the right bed (1). It transfers active row d to the left holder (2), and then drops stable row c from the right holder (3), leaving behind two stable rows on the right holder (4). Then, as usual, the machine knits through active row d and stable row b to form e (5), and similarly knits through active row e and stable row a to form f. This completes a layer consisting of two rows.

A.4 Increase at the Last Row

To increase at the last row of a layer, the machine begins by knitting through the three existing rows of the layer (1), as was depicted in A.1. Now the machine will create an additional row of stitches after the last row of this newly created layer. To do so, it knits through row g to form a new row h. Since there were no loops on the right holder during these knits, row h consists of traditional knit stitches rather than solid-knit stitches. However, if we try to proceed as usual, transferring row h to the right holder and solid knitting through rows g and h, our new solid knit stitches will not be properly stable (the topological problem is illustrated in Figure 26). Hence, we instead knit through row h to produce a new row i (3), transfer row i to the right holder, and then drop row h (4). Now the machine can safely knit through rows g and i (5) to produce a new row j of solid knit stitches. Proceeding through the other rows, we obtain a new layer with four rows of stable loops (6).

A.5 Increase at the First Row

To increase before the first row of a layer, one needs to cast on extra stitches onto the opposite holder before beginning the layer. However, as mentioned in the Section 4.4, cast on to our solid knitting machine currently requires manual intervention. A simple workaround is to instead perform an increase at the end of the previous layer via A.4, providing the required loops to knit the desired number of rows in the current layer.

[2]The steps required to knit this first row are illustrated in more detail in Figure 6

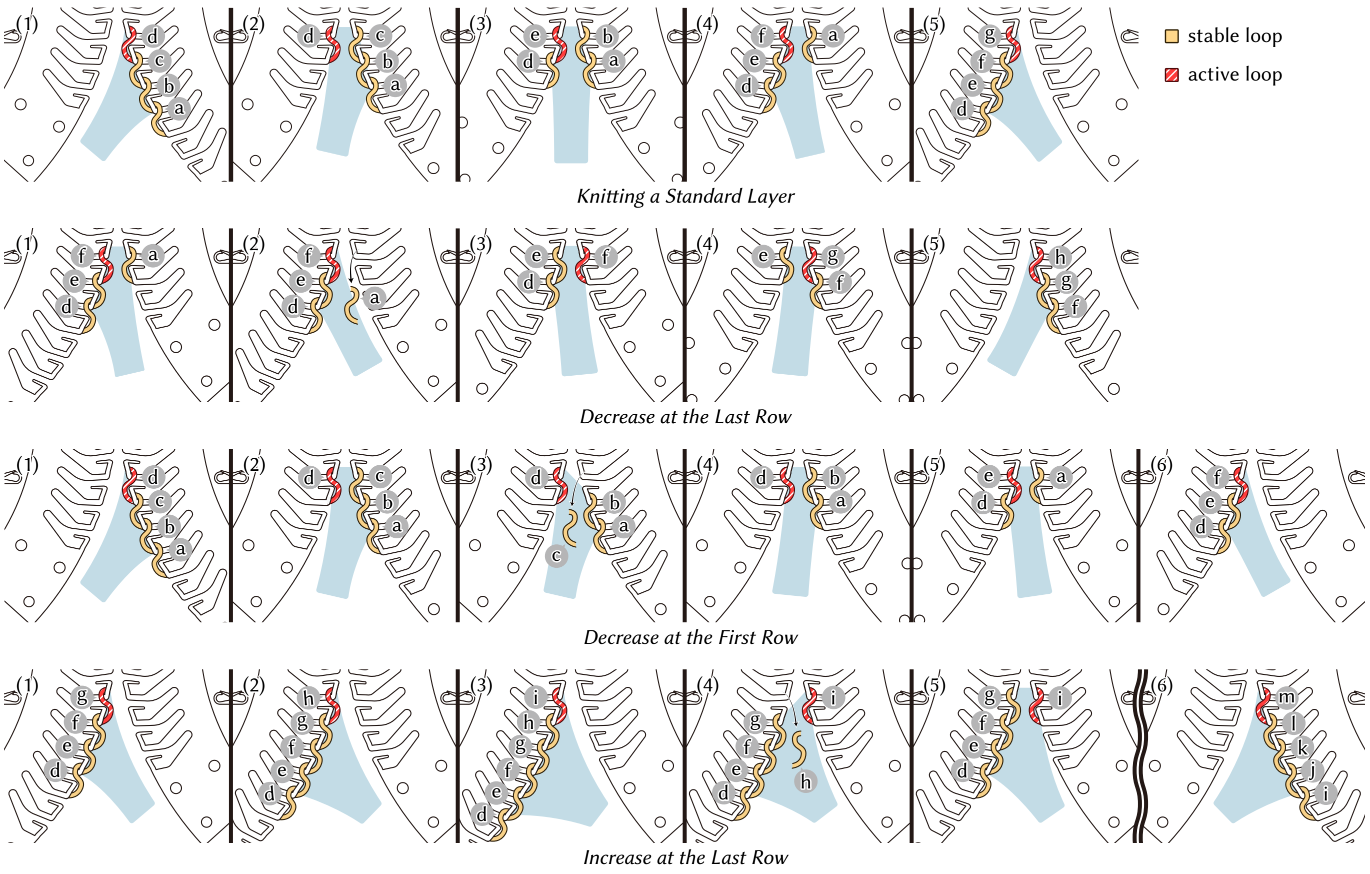


Fig. 25. Here we contrast the steps used to knit a standard layer (*top*) with the various increases and decreases described in Appendix A (*middle, bottom*).

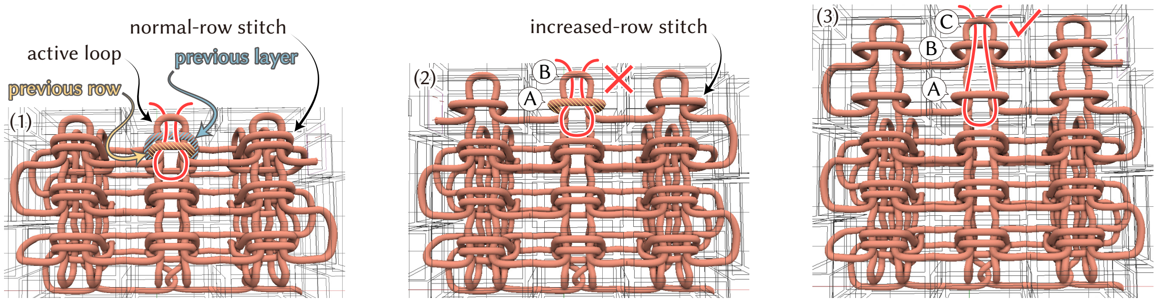


Fig. 26. In a normal row, the loop of the previous layer exists between the previous-row loop and the active loop, so there is no problem with solid knitting through these two (1). By contrast, increased rows have stitches as shown in the top row of (2) because there is no previous layer. This is the same as the first layer of solid knitting, *i.e.*, regular surface knitting. There is no other loop between the previous-row loop, A, and the active loop, B. If these two loops are solid-knit, B – which is formed through A – may return back to the other side of A and undo this stitch. Therefore, one should form another row loop C, and use B as the loop between A and C (3).