# Supplement to Wearable 3D Machine Knitting: Automatic Generation of Shaped Knit Sheets to Cover Real-World Objects

Kui Wu, Marco Tarini, Cem Yuksel, James McCann, Xifeng Gao

## 1 Cutting for Different Quad-(dominant) Meshes

The number of cuts generated by our method depends on the quality of input mesh, i.e. number of singularities and how well aligned they are. Fig. 1 demonstrates our method can robustly cut input quad-(dominant) meshes which are generated from previous papers [1], [2], [3], [4], [5], [6].

## 2 Knittability

The stitch mesh our system generates is guaranteed to be knittable:

As we discuss in Section 4, our flood fill algorithm is guaranteed to produce a topological disc (i.e., the cut-mesh) without any directional singularities. Since the cut-mesh is a topological disc, it cannot contain any holes. This means that the skeleton of the cut-mesh with the assigned directions is a *polytree* – and, thus, has an upward-planar embedding. This is a sufficient condition for machine-knittability (at least, from infinitely-thin, infinitely-stretchy yarn), as established by prior work [7]. More importantly, the procedure in the our knitting instruction generation (Section 7) will always succeed unless there is a directional singularity, since our algorithm will add extra cuts to the surface when necessary to guarantee the machine-knittability.

## 3 Local knitting machine oriented mesh modifications

Fig. 7 demonstrates four local mesh modifications to avoid mesh configurations that can lead to instabilities during machine knitting:

1) A short-row face below a decrease (Fig. 7a), will require the machine to knit one loop through three others, an operation it cannot always complete successfully. Our system, therefore, moves the short-row face either below or to another location along the same row.
2) If the wale edge of a short-row face is on the boundary (Fig. 7b), the stitch above the short-row face becomes unstable and unravels. Our system collapses the triangular stitch mesh face to avoid this situation.
3) Stitches at a boundary cannot be consistently moved by the knitting machine because there is no previous loop to hold the stitches down. Thus, our system turns decrease faces on the boundary (Fig. 7c) into regular stitches (collapsing the two course edges on the boundary).
4) At least two consecutive stitches must be created when adding loops to empty needles. Therefore, if there is only one casting-on stitch, our system collapses the corresponding course edge to form an increase stitch with one of its neighbors (Fig. 7d).

## 4 Knitting Order Determination

### 4.1 Random Knitting Order Generation

We generate a number of knitting orders for the patch-graph by using topological sort. Our sorting algorithm picks the patches of the garment one by one and places them to the next slot in the knitting order. A patch can only be picked if it is *ready*, meaning it does not have any patches below it in the wale direction that are not already picked. At the beginning, we mark all boundary patches that do not have any patches below them as *ready* and place them into a queue $\mathbb{Q}$. We begin our iterations by fetching a patch $P$ from $\mathbb{Q}$. At each iteration, we add the chosen patch $P$ to the knitting order $\mathbb{P}$. For all patches immediately above $P$ in the wale direction, we mark them as *ready*, if they do not have other patches below them that are not already picked. For the next iteration,

- If $P$ has only one ready patch that is directly above, we pick this patch as the next one.
- If $P$ has multiple ready patches directly above, we randomly pick one of them and add all others to $\mathbb{Q}$.
- If $P$ has no ready patch directly above, we randomly pick a patch from $\mathbb{Q}$.


- *K. Wu is with Massachusetts Institute of Technology.*
- *M. Tarini is with the University of Milan*
- *C. Yuksel is with University of Utah*
- *J. McCann is with Carnegie Mellon University*
- *X. Gao is with Florida State University*
- *Corresponding Author: Xifeng Gao, E-mail: gao@cs.fsu.edu*


*Manuscript received August 2, 2020; revised December 1, 2020.*

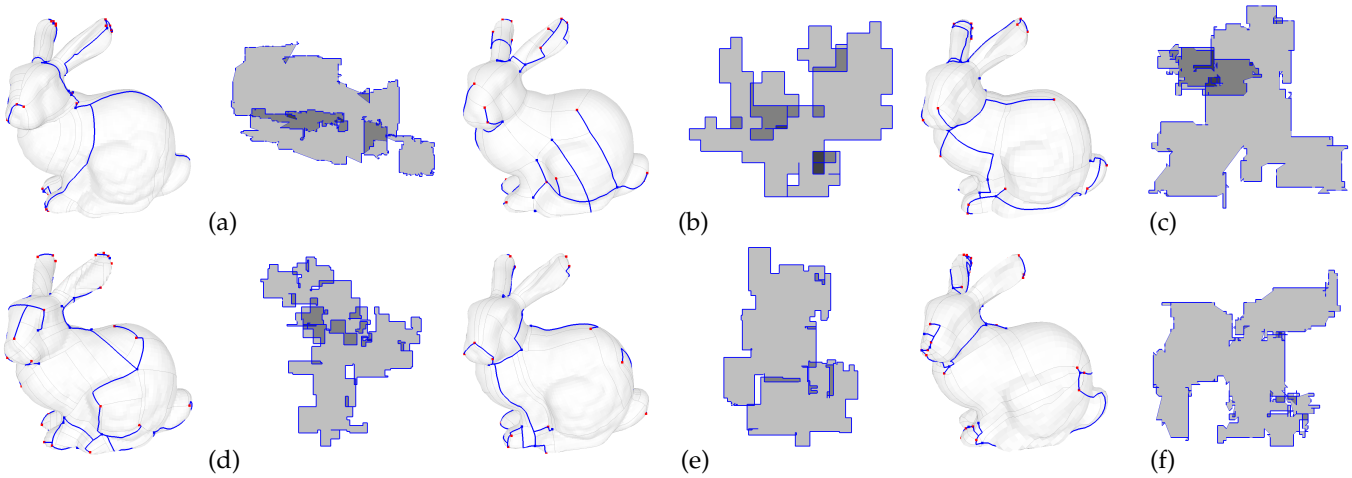


Fig. 1. The cut bunnies and the corresponding approximated paramteric spaces generated using the quad-(dominant) mesh from: (a) [1], (b) [2], (c) [3], (d) [4], (e) [5], and (f) [6].

This approach of picking the next patch favors continuity in the knitting order (by picking a patch directly above when possible), so that multiple patches can be knit together without cutting the yarn. The sorting is completed when $\mathbb{Q}$ is empty. Since the patch-graph is a DAG, this operation will always place all patches to the knitting order.

### 4.2 Knitting Simulation

For each knitting order, we simulate the knitting process on the machine bed following the given knitting order. The goal of the knitting simulation is to detect potential overlaps. We achieve this by keeping track of the occupancy of each needle on the machine bed. Starting with the first patch of the knitting order, we process each row of each patch in order to determine which needled are *occupied* during knitting. For each row, we mark all needles used by the bottom course edges of the row as *unoccupied*, if the bottom course edges are not on a boundary, since the loops held in those needles will be used for knitting new stitches through them. Then, we mark all needles used by the top course edges of the row as *occupied*, as the resulting stitches will be held by those needles. Note that a needle can only be marked as *occupied* only if it is *unoccupied* (and vice versa). During the simulation of a patch, we detect overlapping if we must mark an occupied needle as *occupied*. When we detect overlapping, we simply introduce a cut before this patch and start a new (unconnected) *piece* with this patch. Thus, the stitches of the first row of this patch are effectively converted to cast-on stitches.

At the end of the knitting simulation, it returns the number of additional cuts introduced (i.e. the number of overlapped patches detected). Notice that when an overlaps is detected, current traced patches are cut, even there is another patch connecting with traced patches without overlapping, but not traced yet. As an optional optimization, we check all separated pieces from the selected order and see if any of them can be connected with each other without overlapping, thereby reducing additional cuts.

## References

[1] W. Jakob, M. Tarini, D. Panozzo, and O. Sorkine-Hornung, "Instant field-aligned meshes." *ACM Trans. Graph.*, vol. 34, no. 6, pp. 189–1, 2015.
[2] N. Pietroni, E. Puppo, G. Marcias, R. Roberto, and P. Cignoni, "Tracing field-coherent quad layouts," *Comput. Graph. Forum*, vol. 35, no. 7, pp. 485–496, Oct. 2016.
[3] X. Gao, W. Jakob, M. Tarini, and D. Panozzo, "Robust hex-dominant mesh generation using field-guided polyhedral agglomeration," *ACM Trans. Graph.*, vol. 36, no. 4, pp. 114:1–114:13, Jul. 2017.
[4] X. Fang, H. Bao, Y. Tong, M. Desbrun, and J. Huang, "Quadrangulation through morse-parameterization hybridization," *ACM Trans. Graph.*, vol. 37, no. 4, Jul. 2018.
[5] J. Huang, Y. Zhou, M. Niessner, J. R. Shewchuk, and L. J. Guibas, "Quadriflow: A scalable and robust method for quadrangulation," *Computer Graphics Forum*, vol. 37, no. 5, pp. 147–160, 2018.
[6] K. Wu, X. Gao, Z. Ferguson, D. Panozzo, and C. Yuksel, "Stitch meshing," *ACM Trans. Graph. (Proceedings of SIGGRAPH 2018)*, vol. 37, no. 4, pp. 130:1–130:14, jul 2018.
[7] V. Narayanan, L. Albaugh, J. Hodgins, S. Coros, and J. McCann, "Automatic machine knitting of 3d meshes," *ACM Trans. Graph.*, vol. 37, no. 3, pp. 35:1–35:15, Aug. 2018.