HAND KNIT MESHING

by

CARMEN PARK

A THESIS

Presented to the Department of Computer Science
in partial fulfillment of the requirements for the degree of
Bachelor of Science.

JUNE 2026

An Abstract of the Thesis of

Carmen Park for the degree of Bachelor of Science
in the Department of Computer Science to be taken June 2026.

Title: Hand Knit Meshing

Approved: Tao Hou, Ph.D.
Primary Thesis Advisor

This project explores a computational pipeline for converting a simple closed 3D mesh into a quad dominant mesh that can be hand knit. To manage the complex parameter space of knitting, with a focus on left and right leaning increases and decreases.

The proposed pipeline operates in four core stages:

1. converting a sphere triangular mesh into a quad-dominant mesh into a knit mesh vertex, edge and facet abstraction;

2. using a simplified morse function and assigning timing values to each vertex given a starting and ending point;

3. iteratively selecting vertices and construcing quad faces to define a row structure;

4. and translating the row structures into a standard knitting instruction set as well as a newly constructed quad dominant `knitMesh`.

The resulting stitch map can be translated into plain-text row instructions. Physical validation and extension to non-convex shapes are left for future work.

Acknowledgements

I would like to thank my primary thesis advisor, Tao Hou, for his continued guidance and support throughout this past term. Despite this topic lying outside his primary area of expertise, he consistently made time to review my work, listen to my ideas, and help direct the project toward its final form.

I am also grateful to Hank Childs for his support as a secondary thesis advisor. His deep knowledge of the research process, encouragement during the early stages of topic selection, his enthusiasm for scientific visualization, and being able to take computer graphics with him in the fall, all have been consistently inspiring and helpful over the past year.

I would like to thank Daniele Panozzo for meeting with me last fall and first introducing me to this idea. Without seeing his contributions to the stitch meshing paper, I would not have embarked on this work. While that paper represents only a small part of his broader body of research in computer graphics, it had a profound impact on me.

Finally, I thank my family and friends for their unwavering support, for listening to my thoughts and frustrations, and for being present throughout this process.

I have greatly enjoyed this year of research. Although much of my time was spent simply getting up to speed with the relevant literature, I am excited by the prospect of continuing this work in the future.

Contents

List of Figures

Figure 1: Baby's Dress, Sarah Ann Cunliffe, 1851[2]

1 Introduction

Hand knitting has been a prominent part of human history for hundreds of years, with earliest "nålbinding" techniques[3] seen in a pair of socks from Egypt dating to the 3rd to 5th century AD. Earliest examples of double knitting in the Victoria and Albert Museum were made in North Africa around 1100 - 1300 AD during a period of Islamic rule. Knitting grew in popularity in the 1400s in Europe, and in the 1589 the stocking frame was the first knitting machine invented in Great Britain. Hand knitting has been an important part of historical textile industries. It was one of few acceptable ways for AFAB[4] people to make money in earlier societies, and while it is does not

[2]Hand knitted and beaded lace patterns that knitting machines could not replicate became very popular in the 19th century. They are made with extremely fine needles and tiny yarn. It is said this baby's dress had over a million stitches made with at least 6,300 yards of cotton. Cunliffe spent 5 months working 7 hours a day to complete it.

[3]Nålbinding was and is a precursor to knitting in the round where yarn is threaded through the eye of a needle and worked in round through a series of loops.

[4]Assigned female at birth

dominate the mainstream as it used to, knitting remains a relevant skill and means of creation.

Knitting patterns have been passed down through generations, emerging from a blend of trial and error, mathematical accuracy, and an understanding of shape and geometry. While there has been work towards automating or generating knitting patterns digitally, there are still no marketable programs that allow a user to design their own knitting pattern from a desired shape or form.

This difficulty stems partly from the discrete, loop-based nature of a knitted fabric. Unlike sheet materials like metal, paper or a weave, knitted stitches are interlocked loops that exhibit non-linear stretch or anisotropic behavior (with different properties along wales vs courses) and strong dependence on yarn type, tension, and needle size. Designing a knitting pattern for any given 3D shape requires solving the inverse problem of mapping the target geometry onto a grid of knit and purl stitches while strategically using increases, decreases, short rows, and stitch manipulations to achieve the correct curvature, drape, and edge finish. Trial and error remains dominant because even small changes in stitch placement or gauge can drastically alter the final object's size, fit, and stiffness. As a result, generating a reliable, printable knitting pattern from an arbitrary 3D model remains an open research problem that relates to the topology and geometry of the knitted loop network as well as the physical constraints of yarn under load.

While this paper does not claim to address the full scope of this problem, it does examine two important papers, Yuksel at al. [2] and McCann et al.[3] that both explore this problem in different ways.

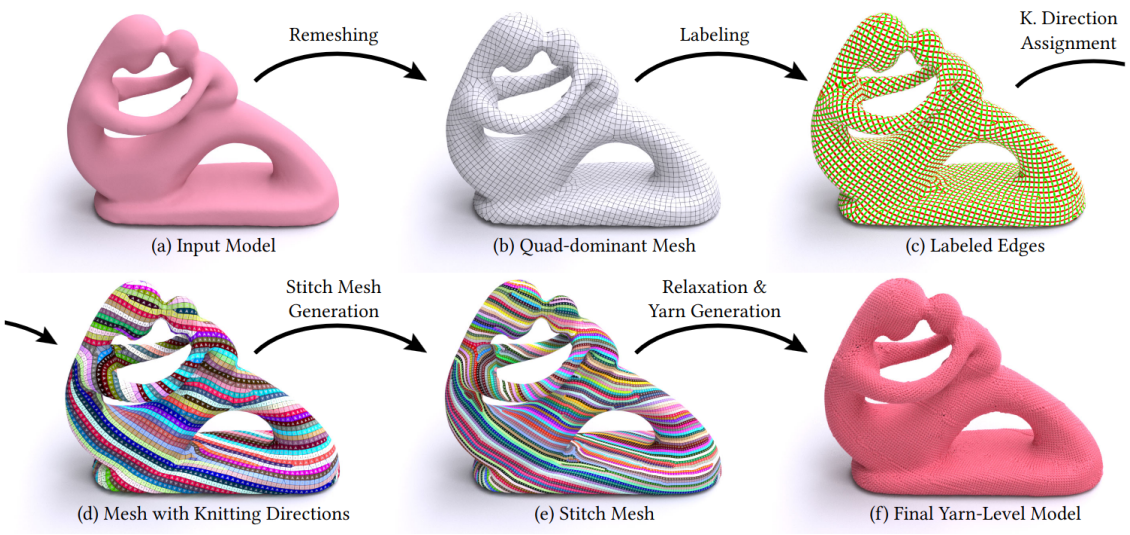


Figure 2: Yuksel et al. Pipeline

1.1 ”Stitch Meshing” Yuksel et al. 2018:

”Stitch Meshing”[2] introduces the first fully automatic pipeline to convert an arbitrary 3D shape into a digitally knitted model. It is based on a globally parameterized remeshing method that produces an isotropic quad dominant mesh resulting from the mesh’s 2-RoSy field. Knitting directions of each quad are assigned using a set of topological operations and a two-step global optimization over the mesh that minimizes the number of singularities. This mesh can then become a valid stitch mesh, where each quad represents a knit stitch and irregular faces represent knitted increases or decreases. The final yarn geometry is computed using a relaxation process. The resulting mesh is rendered using the physically based renderer Mitsuba, and it can be fabricated using 3D printing.

- (a) Pipeline takes an arbitrary input of a 3d model
- (b) Converts it to an isotropic quad dominant mesh with only quads and triangles
- (c) Each edge of the mesh are labeled as wale or course edge
- (d) Knitting directions over the mesh are assigned arrows to show orientation

- (e) Stitch mesh is generated

- (f) Final stitch mesh ”yarn-level model” is generated from stitch mesh via relaxation.

An important caveat of this paper is that the authors do not address the physical qualities of hand or machine knitting, nor can they guarantee that their pipeline's meshes are knittable. Because of this, my focus centers on their first step: the quad remeshing using 2-RoSy fields. The fact that the resulting mesh is not guaranteed to be knittable means that there may be invalid cycles of stitches, and short rows are not ensured to follow a consistent path. Similarly, the mesh may contain topological features that do not correspond to feasible knitting operations such as increases, decreases, or consistent loop connectivity.

Another feature of this paper that enables a cohesive digital knitting mesh is Yuksel et al.'s (2012) work on digitally modeling knitted clothing [4]. In their stitch mesh representation, each unit covers one quad, where the wale edge connects stitches and the course edge connects rows. Figure 3 illustrates a single stitch, a row of stitches, and a set of rows.

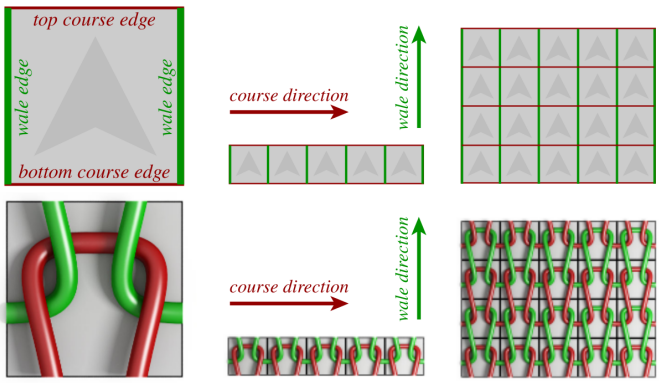


Figure 3: Yuksel et al. Stitch Mesh

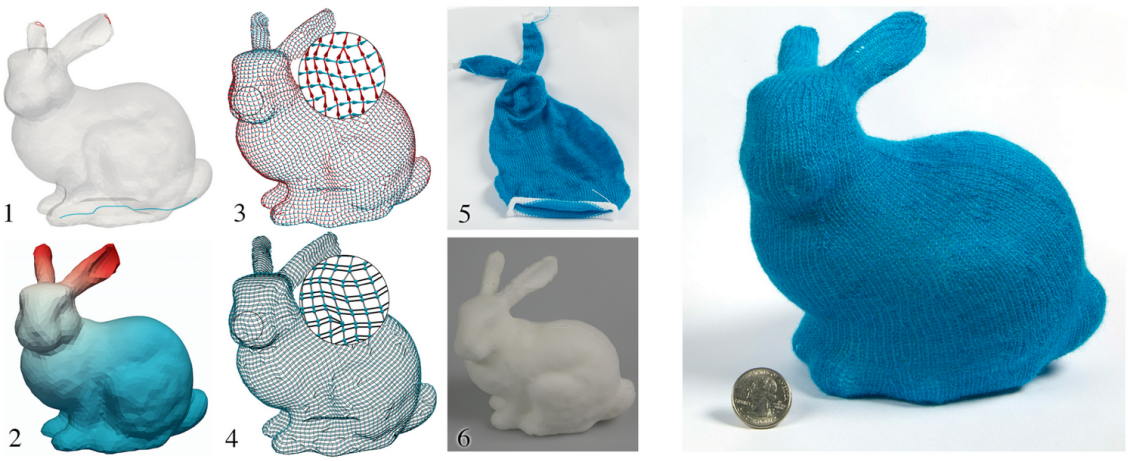


Figure 4: McCann et al. Pipeline

1.2 ”Automatic Machine Knitting of 3D Meshes” McCann et al. 2018

McCann el al.[3] presents the first computational approach to transform a 3D mesh directly into instructions for a computer-controlled knitting machine. Their pipeline takes user defined starting cycles on the mesh which are then:

1. interpolated to define a timing function from the beginning boundary to end;
2. the surface is remeshed to create a row and collumn graph that represents a knit structure;
3. processed by a tracing algorithm that incrementally builds a helix free, uniform and regular quad dominant mesh;
4. used to define a graph-based knit structure with rows, columns, and nodes, subject to strict rules that enforce machine knittability;
5. the pattern is machine knit and fitted over a similar 3D model of the target shape.

this paper raises important questions regarding the actual knittability of a 3D object and the technical requirements involved. The knitting graph enforces strict directionality, and the Reeb

timing function directs the flow of knitting across the mesh. Many aspects of this work informed my own approach.

The authors specifically contribute: (a) a succinct criterion for whether a surface can be machine-knit; (b) a field-guided procedure that produces a knit-mesh-compliant graph structure; (c) a tracing algorithm that transforms the graph into knitting-level operations; and (d) a scheduling algorithm that assigns machine-knitting needle locations and operations.

Several elements of this work proved particularly helpful. The knit graph provided a clear abstraction and rules for stitch connectivity. The Reeb graph and Morse function offered a topological foundation for directing the knit path. The tracing algorithm helped with the construction of a similar iterative method used in building my own quad mesh. The re-ordering algorithm was informative but less critical for my purposes, as hand knitting does not require the same level of machine-level instruction detail.

2 Research

Over fall and winter term, research was conducted on these two leading papers by Naranyan et al[3] and Wu et al.[2] and the subsequent papers they cited. I will begin this section by describing knitting with greater technical depth. I similarly will address quad remeshing, and more specifically N-RoSy field geneeration. Finally I will also look more closely at morse functions and how to construct a Reeb graph.

2.1 Knitting

(a) Stockinette (b) Garter Stitch (c) Reverse Stockinette

Figure 5: Three basic ways to make a knitted fabric

2.1.1 A Formal Definition

Knitting can be defined as the act of interlooping yarn to create a textile product. A stitch is defined as a single loop, with a knit or stocknette stitch being pulled forward through the previous loop and a purl or reverse stocknette stitch being pulled backward through the previous loop. Because of this structure, a stitch that is outward-facing on one side of the fabric appears as a purl on the opposite side when flipped over. A row is defined as a series of stitches that are connected to each other horizontally and a row increase is complete once you have looped through every stitch of the previous row.

Knitting is aided by knitting needles, which allow you to loop through the stitches. Knitting with two single-pointed needles creates a fabric with a clear beginning and end, where each row is disconnected from the next, and the resulting product is homeomorphic to a planar surface[5].

Another method is knitting with circular needles (or a set of three or more double-pointed needles); this allows the rows to be connected in a continuous spiral and makes a ”tube” shape, enabling you to create three-dimensional objects. Knitting in the round often means that you are

[5]Garder stitch is the result of stocknette knitting with double needles since the piece is alternating knitting right sides (RS) and wrong sides (WS). Garder stitch and reverse stocknette stitch (texturing) will not be addressed in this thesis but are worth mentioning as ways of knitting and also can influence the shape and form of a knitted object.

using stocknette sititch on the right side of the knit object[6].

Crocheting is not the same as knitting. It uses a single needle with a curved end and builds chains of stitches. since it operates at a single unit is much easier to build three-dimensional forms this way as there is no no dependency of stitches on the needle to account for. I will not be examining crocheting in this thesis, but future comparisons of both methods could lead to interesting findings.

Similarly, machine knitting and hand knitting, while often producing similar results, have fundamentally different constraints. Industrial machine knitting machines typically use a bed with two rows of hooks to hold stitches. A sliding carraige moves across the bed and pulls yarn through the hooks to form new stitches. There are constraints such as

- Decreases and increases are limited to two stitches at a time.
- Knitting must be continuous; for example you cannot pick up stitches from already completed work which is common with hand knitting.
- cast on and cast off sizes are constrained to maintain machine efficiency.

there are many other differences I have not addressed. While I had hoped to make strides away from a machine knitting implementation, my implementation mostly stayed within these constraints. That being said I would like to look at how hand knitting techniques can be modeled as they break away from the shared capabilities of machine knitting.

I have decided to refine my focus to to 4 particular increase and decrease techniques, a left and right leaning increase, and a left and right leaning decrease. I will mention short rows, but they were not incorperated in my findings.

[6]There is a caviat to this with short rows but this is only important if you are knitting

2.1.2 Casting On and Casting Off

Casting on is the term for making a foundation row of stitches on your needle and is necessary to start any knitting project. Casting off similarly is the final step of binding your stitches so that you can take it off the needles without it unraveling. Going into detail of the cast on and off process is not necessary for this paper, but understanding that casting on and off (both notated as CO) are how a knitting project begins and ends could be helpful to someone unfamiliar with the knitting process.

2.1.3 Increasing

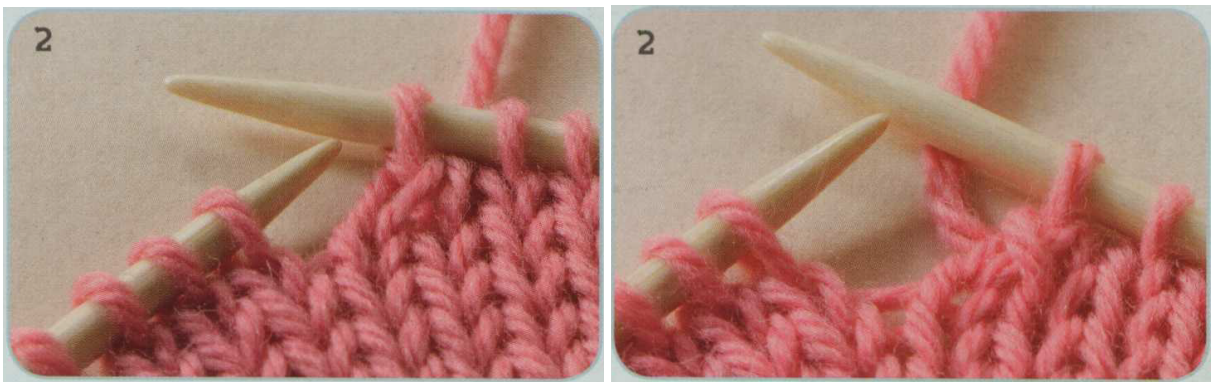
(a) Make One Right (M1R) (b) Make One Left (M1L) (c) M1R-M1L under needle

Figure 6: Make One Increases

The increasing stitch, make one stitch (M1), is the result of creating a loop between the yarn of two existing stitches and knitting it as a new stitch. Make one right (M1R) and make one left (M1L) are the right and left leaning versions of the make one increase respectively. They help make increase lines look more symmetrical, and result in a more polished final product.

2.1.4 Decreasing

Decreasing is the act of looping two stitches into one, leaving one less stitch in the current row than there was in the previous row. By default this folds one stitch underneath the other to

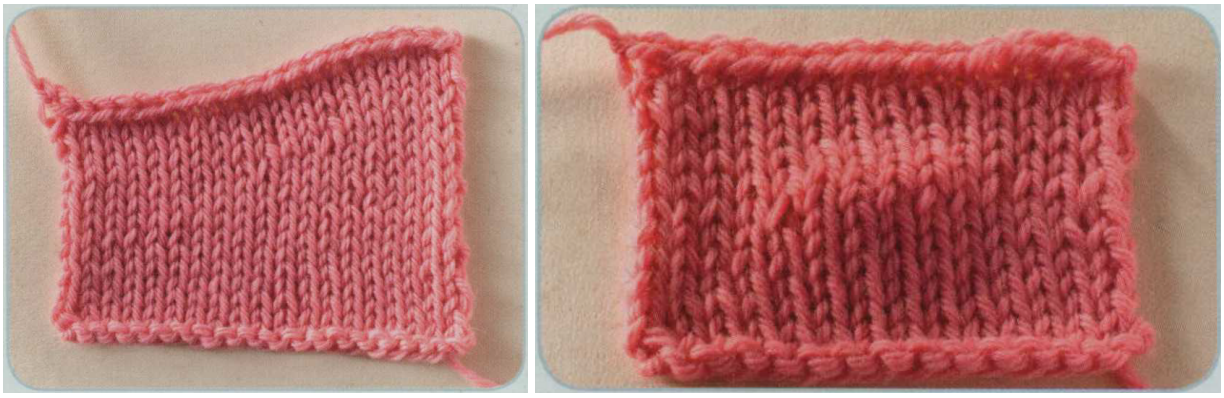


(a) Knit Two Together (K2tog) (b) Slip-slip-knit (SSK)

Figure 7: Right and left leaning decreases

create a sigle stitch. Knit two together (K2tog) is a common decrease technique that places the farthest of the two stitches on top, and it leans to the right. Slip-slip-knit (SSK) has the closest of the two stitches on top and means to the left.

2.1.5 Short Rows



(a) Short rows on one side (b) Short rows in the center

A short row is a knitting and crochet technique where you work partway across a row, turn around, and travel back. Insead of completeing one single row, short rows add two additional stitches[7] to a specific region of a knitted project. This process can be iterated over and over to

[7]One for traversal back, another for the traversal returning to the diverging point.

specifically change the curvature of a localized area of a knit object. Short rows are crucial for creating curves, 3D shapes, and adding length to targeted areas.

Some examples of conventional short row use is creating the heel of a sock, adding bust darts and raising the back in sweaters, making sculptural elements like ruffles, and so on. For hand knitting, wrap and turn or german short rows are necesary methods to avoid holes from making short rows.

2.1.6 Amigurumi

(a) Single arm construction (b) Connecting arms from above (c) Closing body from below

Figure 9: Knitting an amigurumi starfish

The word “amigurumi” literally translates in Japanese to “all encompassing knit”. It is a knit and crochet technique to create plush objects or any variety of shapes and forms. While amigurumi is very popular in the crocheting community, it is also slowly becoming popular in the knitting community. Singh constructs amigurumi patterns in the book “Amigrumi Knits”[5], with 20 patterns that all share simmilar strategies to the knitted starfish in figure 9.

In the instance of the starfish pattern, Singh first constructs the top of each arm, casting on at the base. She then decreases (alternating SSK, K2tog) so that the stitches come together along one center stitch or seam, and casts off at the point. The bottom half of the arm picks up stitches and purls perpindicular to the top half of the arm, casting off when both parts meet in the middle.

After five of these arms are constructed, she picks up stitches from the base of each arm to connect the starfish. This set of operations is not continuous or smooth, but rather she is patching many (in this case symmetrical) pieces into a closed surface.

while this is not how either papers I have looked at went about writing their knitting algorithms, it was most likely in the back of my mind when I wanted to aproach this problem in the first place. It adds many more potential opportunities of construction, and is something to look at in future implementations of this project.

2.1.7 Future Aspirations

Left and right leaning decreases offer a mechanism for shaping regions of changing curvature on a 2-manifold. With careful consideration, the grain of a knit can be intentionally manipulated across an object. This observation makes the project particularly promising, and I hope to eventually develop it into a tool that enables average knitters to design their own patterns.

Additionally, finding how to patching together separate knitted pieces could raise another quad remeshing question, as definitions of simplices and separatrices could shape knit patches so that not only continuous knit projects are represented here.

2.2 Quad Meshing

Bommes et al. [6] describes polygonal meshes as fundamental representations with applications in computer graphics, geometric modeling, simulation, architecture, scientigic visualization and other domains. A mesh is the representation of one large shape by many smaller polygonal cells, forming a cell complex.

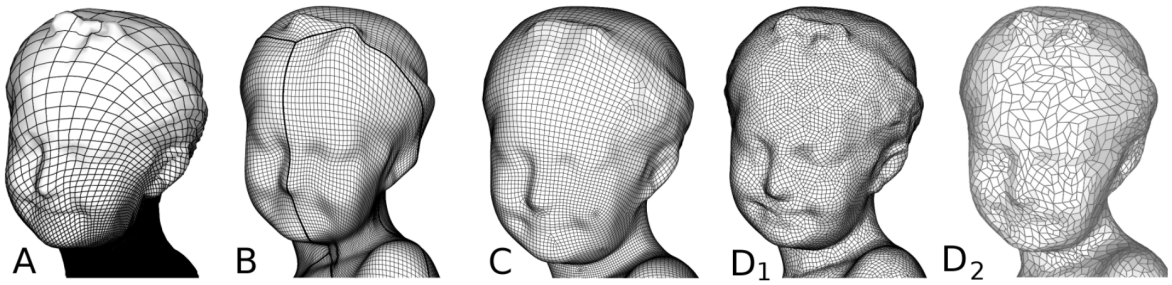


Figure 10: Quad Categories. A: regular; B: semi-regular; C:valence semi-regular; D_1, D_2: unstructured, D_2 more irregular than D_1.

2.2.1 Formal Definition

A two-dimensional polygonal mesh consists of vertices, edges , and facets [8]. The valence of a vertex is the number of its incident edges. The star of a vertex comprises its incident edges and facets; similarly, the star of an edge consists of its incident facets.

Meshes are typically conforming, meaning that any two faces share either a single vertex or a common edge. We assume meshes have a manifold configuration, i.e., there is no meeting at a single point [9]. An internal edge has a star of 2, while an external, or boundary edge has a star of 1. A boundary is composed of cycles of boundary edges. A mesh is watertight if all of its edges are internal.

Discrete functions (scalar, matrix, tensor) on a mesh can be defined by assigning values to vertices or facets. Continuous representations are obtained by interpolation, with bi-linear interpolation as a common method.

A *triangle mesh* is a mesh where all facets are triangles and a *quad mesh* is a mesh where all of ets facets are quadrilaterals. A *quad dominant mesh* has majority quadrilateral facets and a small number of triangular or pentagonal facets.

[8] points in space; lines connecting said points; polygons bounded by cycles of edges
[9] no ”bow tie” configuration

In a quad mesh, an internal vertex is called *regular* if it has a valence of 4, and *irregular* or *extraordinary* otherise. Since this project addresses closed meshes, every vertex is considered an internal vertex.

There is a natural relation between quad meshes and cross fields. A cross field is the assignment of a pair of orthogonal directions, or fields, to each surface point. It is like a vector field but without magnitudes. This is like an N Rotational symmetry field.

Index theory classifies singularities in vector fields, there is similar work in cross fields. For quad meshes, singularities correspond to extraordinary vertices. *Seperatrices* are integral lines that start at singularities. When all seperatricies start and end at singularities this creates natural partitions of a manifold.

2.2.2 Classification and Characteristics

There are several classes of quad meshes that can be seen in figure 10.

- (A) A regular quad mesh is globally mapped from a planar tiling to a mesh.
- (B) Semi regular quad meshes are when patches of quads are "glued" in accordance to its seperatricies. In finite element method (FEM) this is called a multi block grid where shere blocks correspond to patches. This is the most important class of meshes.
- (C) There is valence semi-regular where most of its vertices have a valence of 4, but not every valence semi regular mesh can be subdivided or partitioned into smaller patches. There are important algorithmic differences between valence and semi regular.
- (D) Quad meshes can also be unstructured which is categorized by a large portion of its vertices being irregular.

2.2.3 Quad Quality

Triangles possess several favorable properties: they are always planar, convex, and support linear interpolation of functions defined at vertices.

Quadrilateral quality is more complex. Planar quads are not always convex, and interpolating attributes on quads requires an extended definition of barycentric coordinates. Rendering is also more difficult, and a quad mesh must remain nearly planar, a property referred to as *planar quad* (PQ). Facet corners are ideally close to 90 degrees, opposite sides should be similar in length, and in a uniform quad mesh, every edge should be approximately equal in length.

Quad orientation and size. The number of irregular vertices is intrinsically related to geometric properties.

- **Regularity:** *Unstructured*, *valence semi-regular*, *semi-regular*, and *regular* meshes form a continuum of cases with increasing degrees of regularity.
- **Placement of irregular vertices:** As a rule of thumb, irregular vertices should appear in regions of strong positive or negative Gaussian curvature rather than where resolution changes are required.

Connectivity is an important factor in knitting. For most stitching patterns, a vertex valence of four represents a standard knit. Irregular vertices with valence five or three correspond to increases and decreases. A key aspect of knitting is that these irregularities affect the form and geometry of the knitted object. While most quad meshing works to minimize singularities, knit meshing looks to arrange symmetries around curvature lines. A knit mesh must as well have uniform quad lengths for each facet.

A naive way to convert any polygonal mesh into a quad mesh is to apply topological Catmull–Clark subdivision [7]. Yuksel et al.[2] focuses on N-RoSy field as means of quad generation because it more closely perserves uniform quad facets while considering the shape globally and focusing on primary curvatures.

2.3 N RoSy Fields

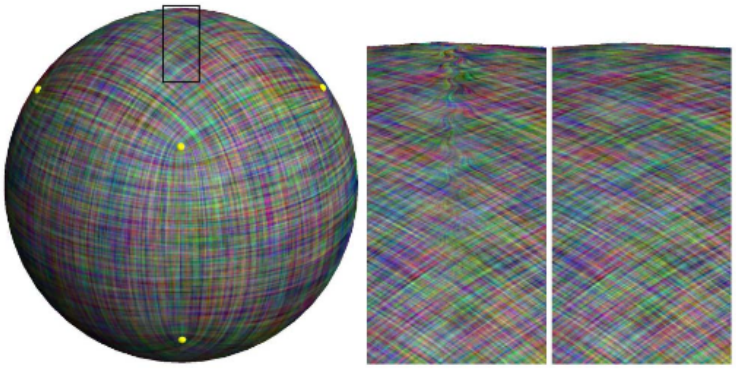


Figure 11: (a) A 4 RoSy field on a sphere forms a cube pattern. (b) A close up of the black box region from (a)[1]

Many objects in computer graphics and geometry processing can be described by *rotational symmetry fields*. N-way rotational symmetry (N-RoSy) fields are proposed as ways to model brush strokes, hatching, principal curvature directions, texture synthesis, and other related effects. N RoSy fields can be considered as a multivalued vector field, where at each point p there exist $n \in \mathbb{N}$ vectors that extend from p at angles $\frac{2\pi}{n}$. Common The singularities and seperatricies of a quad mesh can have a direct impact on the quality of the results[8].

Most modern N-RoSy field generation follows 5 steps that are simmilar to those laid out by Bommes et al's "Mixed-Integer Quadragulation" paper[9].

1. **Define constraints:** This process starts by specifying directions of the field at cartain points on a mesh. Constrsints help guide the fields to allign with specific geometric features or curvature.

2. **Represent field as an ”unrolled” vector field:** Each N-RoSy field it mapped to a ”direction” from the angles obtained by the number of rotational symmetries.

3. **Globally optimize for smoothness:** Within the constraints of this representation, solve a large optimization problem over the mesh. The end goal is to find the smoothest possible N-RoSy field.

4. **Identify and adjust singularities:** With a smooth field, majority vertices will be regular with a valence of 4. There will also be points where the field converges and diverges, or singularities. In finding a smooth field, one also can define optimal singularity location and number.

5. **Generate a clobal parameterization and extract the mesh:**‘Once these stages have been completed, a globally smooth parameterization is computed so that its iso-lines follow the directions of the quad field. A consistend quad mesh can then be extracted.

2.4 Morse function and Reeb Graphs

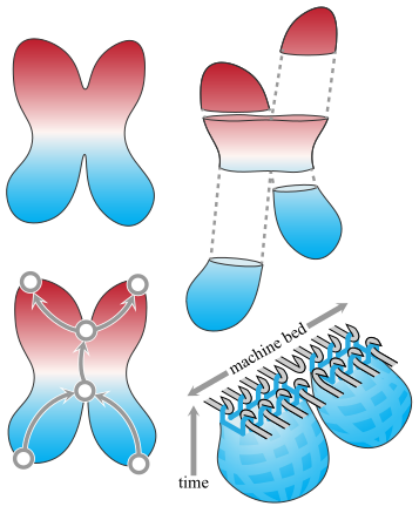


Figure 12: "An oriented 2D manifold-with-boundary $\mathcal{M}$ is knittable iff there exists a Morse function $f : \mathcal{M} \to [-1, 1]$ such that the reeb graph of f has upward planar embedding."[3]

Like McCann et al.[3] morse functions and Reeb graphs help answer the question of whether or not an arbirary 2D manifold $\mathcal{M}$ can be knit continuously. This is the same as asking whether or not the manifold can be described as a set of generalized cylinders from beginning to end. In order for $\mathcal{M}$ to be knittable, it needs at least 2 boundaries, and it must be oriented so that each cycle follows the previous one. They define a timing function $f : \mathcal{M} \to [-1.1]$ that assigns steps from the starting boundary to the end boundary. Each step layers a one dimensional closed curve (row) or curve (short row). They slightly perturb the value of f so that it guarantees lines, circles, or figure 8 shapes, which can all be held in the bed of a knitting machine.

A morse function is defined by Edelsbrunner and Harer[10, 160–174] as a smooth function $f : M \to \mathbb{R}$ on a compact smooth manifold if all critical points are non-degenerate, meaning that the Hessian matrix at each critical point is invertable[10].

[10]A Hessian matrix is a square matrix of all the second-order partial derivatives of a scalar-valued function. A critical point is a specific location on a geometric shape or a data field where the local topology, ie the number of connected pieces, holes or tunnels, changes.

A Reeb graph of f is obtained by contracting each connected component of the level sets $f^{-1}(t)$ to a single point. It is the quotient space $M/\sim$ where $x \sim y$ if $f(x) = f(y)$ and x and y belong to the same connected component of $f^{-1}(f(x))$.

On a 2D surface, a morse funtion has 3 types of critical points:

- **Local minimum:** Represented by a source node in the Reeb graph. This is where new contours appear as f increases. In the context of knitting, the local minimum marks casting on.

- **Saddle point:** This is where contours split or merge as you pass over a col[11]

  In the Reeb graph, a saddle is a node of degree 3. Importantly, this is where a contour can split into two (or two merge into one)

- **Local maximum:** Represented by a sink node. This is where contours disappear as f increases. For knitting, this corresponds to casting off or the end of a piece.

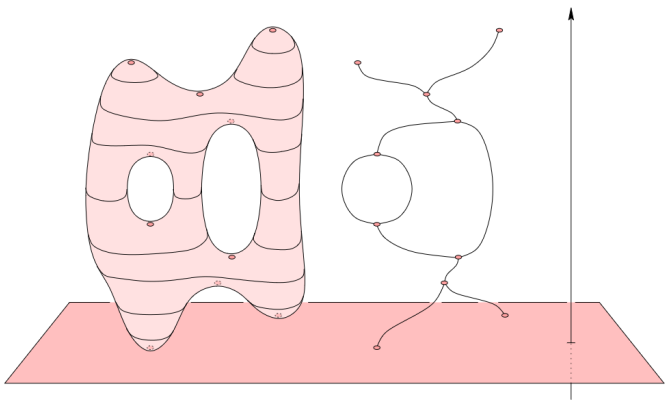


Figure 13: Level sets of the 2-manifold map to points on the real line and comontents of level sets map to points of the Reeb graph.

[11] A col is another name for a saddle point, and appears when the gradient is zero in a Morse function but is neither a minimum nor a maximum. A contour is another name for a level set, which is the set of all points on the manifold where the Morse function f takes the same value t

3 Method

Given that the majority of this work focused on understanding the existing body of research, implementing a full N-RoSy remeshing algorithm was beyond the scope of this thesis. Instead, a simplified algorithm was developed to demonstrate the core concepts.

The proposed algorithm takes as input a simple closed triangular mesh that is a sphere. It extracts the vertex, edge, and facet data, then assigns a scalar timing value to each vertex based on a user-defined starting point and ending point. The algorithm selects from the existing vertices those that lie within a specified distance threshold, then iteratively builds quads from the start boundary to the end boundary. Finally, it assembles these quads into a knittable stitch mesh and a knitting pattern to hand knit the mesh.

3.1 Setup and Build

The implementation was developed on a Linux system. Cross-platform compatibility has not yet been tested. The build process is managed with CMake.

3.1.1 Dependencies

Several external libraries were considered during development. The OpenGL Development (ogldev) library [11] is used for basic mesh import. GLM and GLAD were also tested, but both were ultimately replaced in favor of Qt. Qt was selected as the primary framework because it provides both a graphical user interface and robust support for the required backend functionality in C and C++.

3.1.2 Mesh Import and Data Structures

A basic mesh import is handled using a reader derived from ogldev [11], which supports loading arbitrary file formats. However, for consistency, the implementation primarily uses the Wavefront OBJ format, which was a familiar and straightforward choice. After importing, the mesh data is parsed and stored in a custom data structure called `knitMesh`. This object stores the vertices directly from the obj file and constructs edges from the facet definitions, providing a unified representation for subsequent processing.

3.1.3 Sourcing Meshes

Test meshes were obtained from several sources. Fertility meshes (OBJ format) were sourced from the Stitch Meshing repository by Yuksel et al.[12]. Additional triangular meshes were exported from Rhino the Autocad software. The Stanford Bunny and Dragon models were obtained from the Stanford 3D Scanning Repository[13]. The Utah Teapot was sourced from the University of Utah Computer Graphics archive[14].

Several additional test meshes were created using Blender, including a torus and a stretched sphere. Blender was also used for its quad remeshing tool, QuadriFlow, which operates as follows:

3.1.4 QuadriFlow

QuadriFlow [15] is a method for converting a triangle mesh into a clean, quad-dominant mesh. It improves upon the widely used Instant Field-Aligned Meshes (IFAM) algorithm by balancing local optimization speed with global solution quality. A common challenge with quad meshing is the correct placement of singularities (vertices where the mesh irregularly converges or diverges), and QuadriFlow addresses this through a three-stage process.

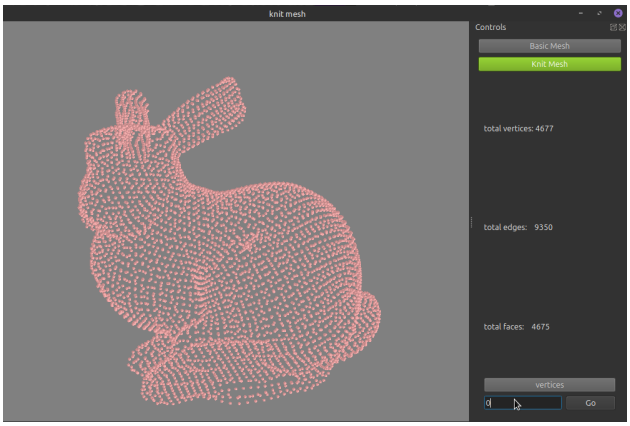


Figure 14: Early environment loading vertices from mesh.

1. **Field initialization:** The algorithm first computes an initial orientation and position field for the quads using a fast local smoothing method (similar to IFAM). This provides a reasonable starting point without expensive global computation.

2. **Singularity reduction via network flow:** The initial field typically contains too many singularities. QuadriFlow reformulates the problem of reducing singularities as a *minimum-cost network flow* problem. By mapping the mesh onto a network, it calculates the adjustments needed to remove singularities with the least "cost", yielding a near-optimal global solution in an efficient way.

3. **Final re-optimization:** With the integer adjustments from the second stage fixed, the continuous variables of the position field are re-optimized to produce the final, smooth, all-quad mesh.

Unlike N-RoSy fields which look for global optimization and mathematical accuracy, IFAM prioritizes computational efficiency and practical usability. It is well suited as a fast and efficient foundation for Blender's quad remeshing algorithm.

3.1.5 Class knitMesh

The `knitMesh` class was constructed to store both the original mesh data (vertices, edges, and facets) from an OBJ file and the derived data (new vertices, edges, and facets) of the resulting knit mesh. Internally, the class computes a bounding box for the graphical user interface and, in the same step, it defines its own starting and ending boundaries based on the maximum and minimum vertex coordinates. This is currently along the z-axis (corresponding to the y-direction in OpenGL).

The class provides a multitide of methods, like sending vertex data to the Qt widget, as well as auxiliary functions like retrieving adjacent vertices and generating unique identifiers. Some of these methods were not ultimately required but were retained for potential future use.

3.2 Proposed Algorithm

After importing the original vertices of a base mesh, the `knitMesh` is generated through four main steps:

1. computing a Morse function to assign a timing value to each vertex;
2. given a number of starting stiches creating the first boundary cycle;
3. iteratively constructing quadrilateral faces while assigning increases and decreases as needed;
4. outputting both a new mesh and a corresponding knitting pattern.

3.2.1 Reeb Graph Timing Assignment

For a simple shape like a sphere, the Reeb graph reduces to mapping each point on the surface to an interval based on a scalar function. I implemented a straightforward timing assignment that

projects vertices onto the line segment from a user-defined start point v_{start} to an end point v_{end}. The timing value $t(v)$ for a vertex v is computed as the normalized scalar projection of $(v - v_{\text{start}})$ onto the direction vector $d = v_{\text{end}} - v_{\text{start}}$:

$$t(v) = \frac{(v - v_{\text{start}}) \cdot d}{\|d\|^2}$$

This assigns $t(v_{\text{start}}) = 0$ and $t(v_{\text{end}}) = 1$, with intermediate vertices receiving values in $[0, 1]$ based on their linear position along the axis. The implementation assumes no saddles or critical points, which is sufficient for convex shapes. The pseudocode for this assignment is shown below.

Algorithm 1 Simple timing assignment (default Reeb)

Require: v_{start}, v_{end}, array of vertices $\{v_i\}$
Ensure: Timing values t_i for each vertex

```
d ← v_end − v_start
mag ← ‖d‖
t(v_start) ← 0, t(v_end) ← 1
for each vertex v_i do
    if v_i ≠ v_start then
        diff ← v_i − v_start
        t_i ← (diff · d)/(mag^2)
    end if
end for
```

3.2.2 Starting Boundary

After obtaining the timing values, I implemented a method to define the first CO cycle. This method takes a specified number of starting stitches and creates an edge for each stitch. It first locates the cycle of vertices whose timing values are closest to the starting boundary, identifies a center vertex among them, and then rotates from one start vertex to another to generate the desired number of edges in a cycle.

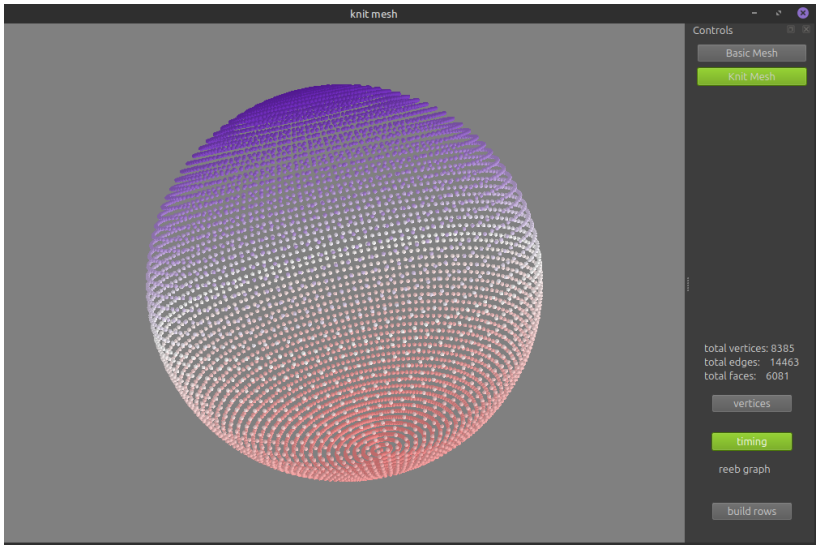


Figure 15: A simple reeb graph assignent to vertices on a sphere. Pink defines the starting boundary with indigo coloring at the end.

This approach is not universally applicable; it assumes that the vertices with the next timing values form a roughly circular arrangement. If they do not, the method will fail. For example, the starting boundary used by McCann et al. for the Stanford Bunny was a loop that had been stretched into a line in order to avoid knitting the base of the bunny and to better fit the constraints of an industrial knitting machine. For hand knitting however, the starting boundary should ideally be as small as possible, and this favors a circular cycle.

3.2.3 Building Rows and Collumns

Once a starting boundary cycle has been constructed, the row building process begins. The method, called `build_rows`, is divided into several steps. In a loop, it performs the following operations:

1. Retrieve the vertices from the previous row.

2. Obtain the next cycle of vertices based on their timing values.

3. From the next cycle, select the vertices that best maintain uniform edge lengths.

4. Build column connections, handling decreases and increases while generating the next row cycle.

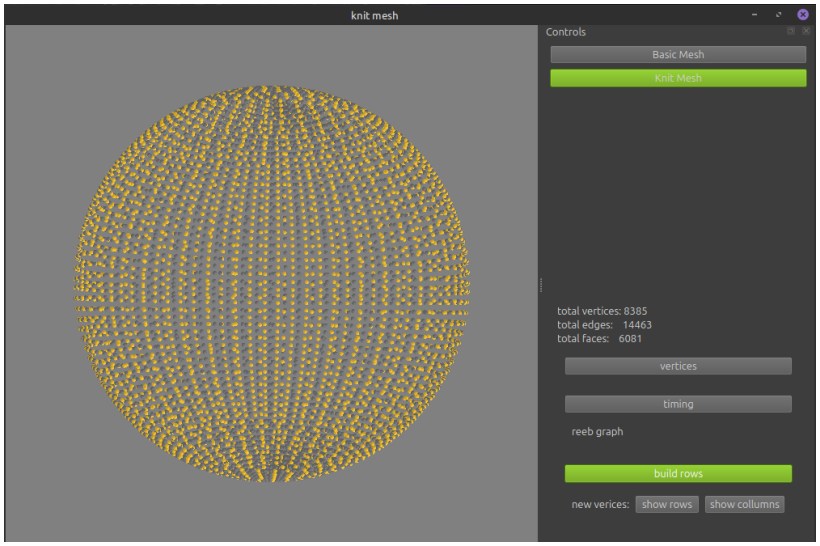


Figure 16: Vertices picked for knitmesh (I don't like how this looks and plan on replacing this or maybe all of these with a real diagrams if i have time).

Steps one and two were straightforward. The previous rows' knit vertices were already stored, and retrieving the next cycle of vertices based on timing required only a simple search.

Building the next row was more challenging. The second implementation assumes a starting side where each vertex has the smallest z-coordinate and a constant x-coordinate throughout. The next timing array was ordered by cycle order and by rotation. However, this approach does not generalize to all meshes, particularly those that are not well defined by a projection from a center point.

Vertices for the knit row are selected based on distance. If the distance falls within the stitch length, the vertex is accepted as a knit vertex, assigned as the new stitch, and the search proceeds to the next vertex. This method runs in $O(n)$ where n is the number of starting vertices in the timing cycle.

Once this process is complete, if the final stitch deviates from the desired length by less than $\pm\varepsilon$, all vertices are shifted backward by a small distance to even out the spacing.

An array is constructed that stores an integer value for each edge in a cycle, where:

- 0 represents a normal stitch,
- 1 represents M1R (make one right),
- 2 represents M1L (make one left),
- 3 represents K2tog (knit two together),
- 4 represents SSK (slip-slip-knit).
- 5 represents CO (cast on/off).

Each of these arrays is stored in the `knitMesh` object as an index map for later reference. To determine which facets become quads and which become triangles, the previous row is examined. The number of decreases can be found by comparing the stitch count (number of edges) of the current row with that of the current cycle.

If the previous row is a cast-on row:

- If there are as many increases as stitches in the current row, the first stitch is always normal (0), then the sequence alternates between increase types (e.g., 2, 0, 1, 0, . . .) to indicate that increases and knit stitches are interleaved.
- If there are fewer increases than stitches, (also alternating 2, 0, 1 but with more 0's inbetween since) the increases are distributed proportionally across the row.
- If there are more increases than stitches, do the first step but 0's get replaced with 1's and 2s.

Otherwise, if the previous row already contains stitches and increases are needed, left-leaning increases (M1L coded as 2) are placed one stitch to the left of their corresponding knit stitch, and right-leaning increases (M1R coded as 1) one stitch to the right. M1L and M1R are paired in the same regions where possible.

(I plan on making a figure for this and pseudocode here)

It is important to acknowledge that this approach does not prescribe a universal knitting method for all practitioners. Rather, it aims to achieve the desired effect of incorporating increases. This style of knitting has been observed and warrants further experimentation with respect to appearance, customizability, and other relevant factors.

3.2.4 Final Mesh and Pattern

After the row building process completes, a separate method uses the increase/decrease index map to construct the column edges. The knit vertices, column edges, and row edges are then exported to the graphical user interface, where the corresponding facets are also generated and delivered to the Qt widget for visualization.

The knitting pattern is extracted from the index map. The integer values defined earlier (e.g., 0 for normal stitch, 1 for M1R, 2 for M1L, 3 for K2tog, 4 for SSK) are translated into plain-text row operations, producing a human-readable pattern.

(cite appendix with pattern here).

Figure 17: Knit Cat from Yuksel et al.

4 Conclusion

This thesis explored a computational pipeline for converting a sphere triangular mesh into a quad-dominant representation related to hand knitting. A simplified Morse function (specifically, a linear projection of vertices onto an axis between user-defined start and end boundaries) was used to assign timing values. Based on this timing, a row-by-row construction algorithm was implemented. It selects vertices within a stitch-length distance, builds cycles of edges, and encodes increases (M1R, M1L) and decreases (K2tog, SSK) as integer labels (0–4) on edges.

The resulting `knitMesh` data structure stores vertices, edges, facets, and the stitch operation map. The pipeline was demonstrated successfully on convex shapes such as spheres. The stitch map can be translated into plain-text row instructions, providing a foundation for future hand-knitting pattern generation.

Several limitations must be acknowledged. The algorithm assumes that the next cycle of vertices forms a nearly circular arrangement and relies on a fixed projection axis; these assumptions do not hold for arbitrary manifolds. An attempt to use an explicit edge class introduced unnecessary complexity and was eventually simplified. Moreover, a full N-RoSy field was not implemented within the available time. QuadriFlow was examined as an alternative, but its vertex layout did not match the specific directionality of a hand-knitting graph. Future work should therefore focus on implementing a proper N-RoSy field to obtain a globally smooth cross field and optimal singularity placement, extending the row construction to arbitrary meshes, and finally validating the pipeline through hand-knitting of physical prototypes.

Despite these limitations, this work provides a concrete, working implementation of a timing-based row construction for knit-like meshes on simple geometries, and it clarifies the steps needed to move toward a fully general solution.

Knitting is a complex, multifaceted technique with many more aspects than have been introduced here. The continuation of this work could provide aid for knitters who want to design their own patterns without using traditional methods of constructing a knitting pattern.

References

[1] Jonathan Palacios and Eugene Zhang. Interactive visualization of rotational symmetry fields on surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 17(7):947–955, July 2011.

[2] Kui Wu, Xifeng Gao, Zachary Ferguson, Daniele Panozzo, and Cem Yuksel. Stitch meshing. *ACM Trans. Graph.*, 37(4), 2018.

[3] Vidya Narayanan, Lea Albaugh, Jessica Hodgins, Stelian Coros, and James McCann. Automatic machine knitting of 3d meshes. *ACM Trans. Graph.*, 37(3):35:1–35:15, August 2018.

[4] Cem Yuksel, Jonathan M. Kaldor, Doug L. James, and Steve Marschner. Stitch meshes for modeling knitted clothing with yarn-level detail. *ACM Trans. Graph.*, 31(4), July 2012.

[5] Hansi Singh. *Amigurumi Knits: Patterns for 20 Cute Mini Knits*. Creative Publishing International, Minneapolis, 2009.

[6] David Bommes, Bruno Lévy, Nico Pietroni, Enrico Puppo, Claudio Silva, Marco Tarini, and Denis Zorin. Quad-mesh generation and processing: A survey. *Computer Graphics Forum*, 32(6):51–79, 2013.

[7] Edwin Catmull and James Clark. Recursively generated b-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978.

[8] Jonathan Palacios and Eugene Zhang. Rotational symmetry field design on surfaces. *ACM Trans. Graph.*, 26(3):55–es, July 2007.

[9] David Bommes, Henrik Zimmer, and Leif Kobbelt. Mixed-integer quadrangulation. *ACM Transactions on Graphics*, 28(3):77:1–77:10, July 2009.

[10] Herbert Edelsbrunner and John Harer. *Computational Topology: An Introduction*. American Mathematical Society, Providence, RI, 2010. Pages 160–174.

[11] triplepointfive. ogldev: Opengl tutorials. https://github.com/triplepointfive/ogldev, 2018. Accessed: 2026-05-16.

[12] Cem Yuksel and Kui Wu. Stitch meshing, 2018.

[13] Stanford 3d scanning repository.

[14] Utah teapot.

[15] Jingwei Huang, Yichao Zhou, Matthias Niessner, Jonathan Richard Shewchuk, and Leonidas Guibas. Quadriflow: A scalable and robust method for quadrangulation. *ACM Transactions on Graphics*, 39(6), 2020.

[16] S. Dong, S. Kircher, and M. Garland. Harmonic functions for quadrilateral remeshing of arbitrary manifolds. *Computer Aided Geometric Design*, 22:392–423, 07 2005.

[17] Yu-Kun Lai, Miao Jin, Xuexiang Xie, Ying He, Jonathan Palacios, Eugene Zhang, Shi-Min Hu, and Xianfeng Gu. Metric-driven rosy field design and remeshing. *IEEE Transactions on Visualization and Computer Graphics*, 16(1):95–108, January 2010.

[18] Kui Wu, Hannah Swan, and Cem Yuksel. Knittable stitch meshes. *ACM Transactions on Graphics (TOG)*, 38(1):1–13, January 2019.

[19] The history of hand-knitting. Accessed: 2026-05-08.