COMPUTER-AIDED GEOMETRIC DESIGN FOR FORMING
WOVEN CLOTH COMPOSITES

By
Masaki Aono

A Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Major Subject: Computer Science

Approved by the
Examining Committee:

Michael J. Wozny, Thesis Adviser

Harry W. McLaughlin, Member

David L. Spooner, Member

Joshua U. Turner, Member

Peter R. Wilson, Member

Rensselaer Polytechnic Institute
Troy, New York

February 1994
(For Graduation May 1994)
COMPUTER-AIDED GEOMETRIC DESIGN FOR FORMING
WOVEN CLOTH COMPOSITES

By

Masaki Aono

An Abstract of a Thesis Submitted to the Graduate
Faculty of Rensselaer Polytechnic Institute
in Partial Fulfillment of the
Requirements for the Degree of
DOCTOR OF PHILOSOPHY
Major Subject: Computer Science

The original of the complete thesis is on file
in the Rensselaer Polytechnic Institute Library

Approved by the
Examining Committee:

Michael J. Wozny, Thesis Adviser

Harry W. McLaughlin, Member

David L. Spooner, Member

Joshua U. Turner, Member

Peter R. Wilson, Member

Rensselaer Polytechnic Institute
Troy, New York

February 1994
(For Graduation May 1994)
## CONTENTS

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>LIST OF TABLES</td>
<td>vi</td>
</tr>
<tr>
<td></td>
<td>LIST OF FIGURES</td>
<td>viii</td>
</tr>
<tr>
<td></td>
<td>ACKNOWLEDGMENTS</td>
<td>xvii</td>
</tr>
<tr>
<td></td>
<td>ABSTRACT</td>
<td>xix</td>
</tr>
<tr>
<td>1.</td>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>1.1</td>
<td>Contributions</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Organization of the Thesis</td>
<td>3</td>
</tr>
<tr>
<td>2.</td>
<td>PROBLEM STATEMENTS AND THE OUTLINE OF A CAD SYSTEM FOR FORMING WOVEN CLOTH COMPOSITES</td>
<td>5</td>
</tr>
<tr>
<td>2.1</td>
<td>Composite Materials in Aircraft and Other Vehicles</td>
<td>5</td>
</tr>
<tr>
<td>2.2</td>
<td>Current Composite Parts Manufacturing Process</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>Problems</td>
<td>9</td>
</tr>
<tr>
<td>2.4</td>
<td>Technology Needs and Goals</td>
<td>11</td>
</tr>
<tr>
<td>2.5</td>
<td>A Proposed CAD System for Forming Woven Cloth Composite Parts</td>
<td>12</td>
</tr>
<tr>
<td>3.</td>
<td>RELATED WORK</td>
<td>15</td>
</tr>
<tr>
<td>3.1</td>
<td>Tchebychev’s Cloth Model and its Derived Research Work</td>
<td>15</td>
</tr>
<tr>
<td>3.2</td>
<td>Mack &amp; Taylor’s Cloth Model for Fitting</td>
<td>17</td>
</tr>
<tr>
<td>3.3</td>
<td>Comparison of Research Work on Fitting Cloth</td>
<td>19</td>
</tr>
<tr>
<td>4.</td>
<td>MODEL DEFINITIONS</td>
<td>22</td>
</tr>
<tr>
<td>4.1</td>
<td>A Model for Woven Cloth Composite Plies</td>
<td>22</td>
</tr>
<tr>
<td>4.2</td>
<td>The Locking Angle and its Geometric Model</td>
<td>24</td>
</tr>
<tr>
<td>4.3</td>
<td>A Model for Slippage</td>
<td>29</td>
</tr>
<tr>
<td>4.4</td>
<td>A Model for 3D Surfaces</td>
<td>31</td>
</tr>
<tr>
<td>5.</td>
<td>FITTING A WOVEN CLOTH COMPOSITE PLY TO A SURFACE</td>
<td>35</td>
</tr>
<tr>
<td>5.1</td>
<td>Methods for Specifying Initial Conditions</td>
<td>35</td>
</tr>
<tr>
<td>5.1.1</td>
<td>Previous Method for Specifying Initial Conditions</td>
<td>36</td>
</tr>
<tr>
<td>5.1.2</td>
<td>A New Method for Specifying Initial Conditions</td>
<td>39</td>
</tr>
</tbody>
</table>
5.2 Mapping Calculations

5.2.1 Strategy for Mapping Calculations

5.2.2 Mapping Calculations for Auxiliary Mesh Points on the Base Path

5.2.3 Mapping Calculations for Regular Mesh Points NOT on the Base Path

5.3 Scanning Algorithms

5.3.1 The Case in Which Initial Paths are Defined by a Pair of Perpendicular Threads in 2D Space

5.3.2 The Case in Which Initial Paths are Defined by the Base Path and Sweeping Directions

5.3.3 An Approximation to the Mapping Calculation for an Indegree One Node

5.4 Boundary Adjustments for a Composite NURBS Surface

5.4.1 Requirements of Boundary Adjustments

5.4.2 A Data Structure for Quick Boundary Adjustments

5.4.3 Boundary Adjustment for the Mapping Calculation of Auxiliary Mesh Points

5.4.4 Boundary Adjustment for the Mapping Calculation of Regular Mesh Points

5.5 Plane Development

5.5.1 Previous Approaches to the Flattening

5.5.2 A Method for Obtaining Flattened Patterns

5.6 Formability

5.6.1 Shear Deformation Energy

5.6.2 Bending Deformation Energy

5.7 Examples of Fittings

5.7.1 An Octant of a Sphere

5.7.2 A Hemisphere

5.7.3 One Third of a Surface of Revolution

5.7.4 A Bump in a Panel

6. ANOMALOUS EVENTS AND THEIR PREVENTION BY INSERTING DARTS
6.2.1 Models for Darts ........................................... 110
6.2.2 Where Darts Should be Inserted ......................... 121
6.2.3 What Darts Should be Inserted ......................... 122
6.2.4 Overlapping Removal Algorithms ..................... 125
6.2.5 Examples of Inserting Darts ............................ 132

7. ALGORITHMS DERIVED FROM GEOMETRIC PROPERTIES OF A
   TCHEBYCHEV NET ........................................... 147

   7.1 Geometric Properties of a Tchebychev Net .............. 147
       7.1.1 Fundamental Equation of a Tchebychev Net ....... 147
       7.1.2 The Relationship between the Intrinsic Geometry
            of a Surface and the Thread Angle of a Tchebychev
            Net ............................................. 150
       7.1.3 Global Properties of a Tchebychev Net .......... 152

   7.2 Algorithms Derived from a Tchebychev Net Assumption .... 155
       7.2.1 An Algorithm for Fitting a Tchebychev Net to a
            Curved Surface ............................... 155
       7.2.2 An Algorithm Based on Global Properties of a
            Tchebychev Net .................................. 159
       7.2.3 An Algorithm for Finding a Good Initial Path .... 166

   7.3 Algorithms for Shape Optimization ...................... 188
       7.3.1 Historical Background ............................ 189
       7.3.2 Modifying a Surface by Editing Principal Curvatures
            .............................................. 191
       7.3.3 Modifying a Surface Shape by Editing Control Points
            .............................................. 191
       7.3.4 Modifying a Surface Shape by Finding Optimal Weights
            .............................................. 193
       7.3.5 Modifying a Surface Shape by Combining Control Polygon
            Adjustment and Optimal Weight Finding .............. 206
       7.3.6 Summary of the Methods for Shape Modification ... 221

8. CONCLUSION ............................................. 225

   8.1 Summary .................................................. 225
   8.2 Future Recommendations ................................. 229

LITERATURE CITED ............................................ 232

A. PARTIAL DERIVATIVES OF A NURBS SURFACE ................. 241

B. DATA STRUCTURES FOR A COMPOSITE NURBS SURFACE .... 244

C. DATA STRUCTURES AND ALGORITHMS FOR SCANNING MESH
   POINTS NOT ON THE BASE PATH ............................ 247
D. ALGORITHMS FOR OBTAINING AN INTERMEDIATE GAUSSIAN CURVATURE INTEGRAL ........................................... 251

E. THE RULE OF APPENDING FRACTIONAL FRAGMENTS TO BOUNDARY MESH POINTS IN BOUNDARY TRAVERSAL ....................... 253

F. COLOR ASSIGNMENT .......................................................... 259

G. IMPLEMENTATION OF A CAD SYSTEM FOR COMPOSITE PART LAYOUT: EXTERNAL FILE FORMATS ................................. 264
   G.1 Implementation of a CAD System ........................................ 264
   G.2 External Data File Format ............................................... 266
   G.3 Examples of External Data File Format ................................. 271
      G.3.1 External Data Files for a Surface of Revolution ............... 271
      G.3.2 External Data Files for an Object with a Convex Corner ... 273

H. GLOSSARY ............................................................................. 278
<table>
<thead>
<tr>
<th>Table</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table 3.1</td>
<td>Comparison of major research on fitting cloth to a surface.</td>
<td>20</td>
</tr>
<tr>
<td>Table 4.1</td>
<td>Locking angles of several fabric weaves. Based on Table 7 in [62].</td>
<td>28</td>
</tr>
<tr>
<td>Table 4.2</td>
<td>A set of parameters for our model of the locking angle that produces the observed locking angle in Table 4.1.</td>
<td>29</td>
</tr>
<tr>
<td>Table 5.1</td>
<td>Elastic constants for some fiber-reinforced composite materials with 60% unidirectional fibers by volume. Based on Table 3.4 in [28]. The unit for $E$ and $G$ is GPa (giga pascal), where $1\ Pa = 1\ Nm^{-2} = 10\ dyn\ cm^{-2}$.</td>
<td>88</td>
</tr>
<tr>
<td>Table 5.2</td>
<td>Initial conditions for the three fitting in Figure 5.25.</td>
<td>90</td>
</tr>
<tr>
<td>Table 5.3</td>
<td>Comparison of results with different initial conditions.</td>
<td>92</td>
</tr>
<tr>
<td>Table 5.4</td>
<td>Initial conditions for the fitting in Figure 5.31.</td>
<td>98</td>
</tr>
<tr>
<td>Table 5.5</td>
<td>Comparison of results with different initial conditions.</td>
<td>101</td>
</tr>
<tr>
<td>Table 5.6</td>
<td>Comparison of the two fitting results.</td>
<td>104</td>
</tr>
<tr>
<td>Table 6.1</td>
<td>Typical manufacturing faults in fiber composite laminates. Based on Table 6.2 in [44].</td>
<td>106</td>
</tr>
<tr>
<td>Table 7.1</td>
<td>The result of the optimal weight finding algorithm applied to the shoe-like object. For the weights $w_{3,j}$ ($0 \leq j \leq 6$), the optimal weight is sought such that $G(w_{3,j}) = 0$ with $\delta = 0.0$ in Equation (7.37). For the weights $w_{i,j}$ ($i \in {0, 1, 2}$, $0 \leq j \leq 6$), the corresponding weights $w_{6-i,j}$ are also changed into the same value to keep the symmetry of the resulting shape. For instance, if $w_{0,j}$ is changed into a new value, $w_{6,j}$ is also changed into the same new value, and the solution to $G(w_{0,j}, w_{6,j}) = 0.0$ with $\delta = 0.0$ is sought. See Figure 7.33 for identifying what control points constitute a pair that produces a symmetric result. Note that N/S in the entry represents No Solution, including the case where $w'_{i,j}$ becomes negative.</td>
<td>211</td>
</tr>
<tr>
<td>Table 7.2</td>
<td>The deviation value $D(w)$ between the original surface shape and the modified shape by changing the one or two weights that correspond to $w'_{i,j}$ in Table 7.1.</td>
<td>212</td>
</tr>
</tbody>
</table>
Table 7.3 Contribution of control points in determining the shape at the mesh point whose thread angle is 0.211 in the critical region $S_1$. Note that the control points with larger values have a greater impact on the shape than those with smaller values.

Table 7.4 Contribution of control points in determining the shape at the mesh point whose thread angle is 0.447 in the critical region $S_2$. 

vii
LIST OF FIGURES

Figure 2.1 Three major composite production forms. .................................. 6
Figure 2.2 A close-up of woven cloth fiberglass sheet. .............................. 7
Figure 2.3 A sheet of woven cloth graphite to be cut out on the table. .... 8
Figure 2.4 Kevlar/epoxy composites used for reinforcing the fuselage of a glider. ......................................................... 9
Figure 2.5 The basic flow in a CAD system for forming woven cloth composite. ................................. 13

Figure 4.1 (a) Plain weave ply and (b) the topology of initial mesh points. 23
Figure 4.2 Thread angle $\gamma$ at a mesh point $P_{i,j}$ .................................. 24
Figure 4.3 Geometry of in-plane shear deformation: (a) the original configuration and (b) a deformed configuration. 26
Figure 4.4 Geometric limit of shear deformation. .................................... 27
Figure 4.5 Several types of weaving patterns: (a) plain weave ($n = 2$), (b) twill weave ($n = 3$), and (c) 4-harness satin weave ($n = 4$). 28
Figure 4.6 Slippage effect. ............................................................... 30
Figure 4.7 Fittings to an octant of a sphere with slippage effect (dotted lines) and without slippage effect (solid lines). ......................... 32
Figure 5.1 Constrained yarn paths (previous method).............................. 36
Figure 5.2 Scanning order of mesh points (previous method).................. 37

Figure 5.3 Initial paths in 2D affect the flexibility in deformation; (a) the angle between a path and any thread intersecting the path is 90 degrees and the in-plane deformation is diagonally dominant (previous method); (b) the angle between a path and any thread intersecting the path is 45 degrees and the in-plane deformation is vertically dominant. 38

Figure 5.4 (a) Badly selected constrained yarn paths leave the uncalculated regions on the surface, and (b) a concave surface area may not be totally covered. 39
Figure 5.5 A new method for specifying initial conditions. 41

Figure 5.6 Intersection curves between a surface and a plane; (a) the intersection curve has a self-loop; (b) the intersection curve is an open curve. 47

Figure 5.7 Mapping calculation of an auxiliary mesh point on the base path. 49

Figure 5.8 Mapping calculation of a regular mesh point, with two known neighbors. 53

Figure 5.9 An example of a scanning based on the initial paths defined by a pair of perpendicular threads. 56

Figure 5.10 A dependency graph of the example in Figure 5.9. 57

Figure 5.11 A parallel scanning for the example in Figure 5.9. 58

Figure 5.12 An example of scanning with the initial paths defined by the base path and sweeping directions. 59

Figure 5.13 A dependency graph of the example in Figure 5.12. 60

Figure 5.14 An approximation of the mapping calculation for node (‘4’) with only one known neighbor (‘a’). 61

Figure 5.15 A complex scanning example that includes run-time recognition of the surface boundary. 65

Figure 5.16 An example of the base path on a composite surface. 68

Figure 5.17 An example of a composite NURBS surface made of seven NURBS patches. 68

Figure 5.18 Topological data for the example of Figure 5.17. 70

Figure 5.19 The code returned by PatchCode(SurfacePatch, u, v). 71

Figure 5.20 Boundary adjustment at the WEST boundary of the “old” patch, which corresponds to the EAST boundary of the “new” patch. 73

Figure 5.21 An example of a fitting to a composite surface made of several patches, where boundary adjustments are applied to the mapping calculations for some of the regular mesh points such as p(i, j) and p(m, n). Note that dotted lines represent patch boundaries. 76
Figure 5.22 The result of the boundary traversal applied to the example of Figure 5.15. 83

Figure 5.23 An example of plane development. Black circles denote boundary mesh points, white circles denote “mapped” mesh points, and bold edges denote the fractional fragments of boundary mesh points. The boundary is obtained as the dotted line. 84

Figure 5.24 Fractional fragments appended to the boundary mesh points: (a) no fragments added to both the fitting and the plane development, and (b) fragments added to both the fitting and the plane development. 85

Figure 5.25 Fittings to an octant of a sphere and their plane developments. 91

Figure 5.26 Fitting results with an actual sheet of woven cloth fiberglass, corresponding to the theoretical fittings shown in Figure 5.25. 93

Figure 5.27 3D fitting results with the starting point at the north pole; (a) the wefts take 90 degrees to the base path, and (b) both wefts and warps take 45 degrees to the base path. 94

Figure 5.28 3D fitting results with colors assigned, based on the thread angle values ($T$); (a) the wefts take 90 degrees to the base path, and (b) both wefts and warps take 45 degrees to the base path. 95

Figure 5.29 2D plane development patterns corresponding to the fitting results of Figure 5.28; (a) the wefts take 90 degrees to the base path, and (b) both wefts and warps take 45 degrees to the base path. 96

Figure 5.30 Three views of a quadrant of a surface of revolution. 97

Figure 5.31 Fittings to a third part of a surface of revolution with six different initial conditions. 99

Figure 5.32 Plane development patterns used for the fitting to a third of a surface of revolution. 100

Figure 5.33 Distribution of thread angles between wefts and warps at mesh points projected above the associated plane developments. 102

Figure 5.34 A surface with a bump. 103
Figure 5.35 Two fittings to a surface with a bump; (a), (b) are the fitting results, and (c), (d) are the 2D plane development patterns corresponding to (a) and (b), respectively.  

Figure 6.1 Anomalous events in terms of the thread angle at a mesh point between a weft and a warp: (a) the thread angle goes below the allowable minimum ($\gamma_{\text{lock}}^{\text{min}}$) and (b) the thread angle exceeds the allowable maximum ($\gamma_{\text{lock}}^{\text{max}}$).  

Figure 6.2 An illustration of a wrinkle.  

Figure 6.3 Cases for which there are no mapping solutions. The surface profiles show (a) breaking and (b) bridging.  

Figure 6.4 Stitched darts (before and after insertions).  

Figure 6.5 Trimmed darts (before and after insertions).  

Figure 6.6 An example of the insertion of a stitched dart ((a) before and (b) after the insertion).  

Figure 6.7 The data structure for the example in Figure 6.6.  

Figure 6.8 An example of a corner stitched dart.  

Figure 6.9 The data structure for the example in Figure 6.8.  

Figure 6.10 An example of the insertion of a polygonal cut.  

Figure 6.11 An example of the insertion of a linear cut.  

Figure 6.12 An example of the insertion of a polygonal cut: (a) A fitting example where a gap and an overlap occur, and (b) the plane development pattern.  

Figure 6.13 Algorithm for determining a shape of a dart to be inserted into a lower critical region.  

Figure 6.14 Triangular mesh approximation for finding overlaps.  

Figure 6.15 A point-triangle inclusion test, where the projection of mesh point P is located inside triangle ABC.  

Figure 6.16 An example of a triangle tree: (a) original triangle mesh, (b) a hierarchically represented triangle mesh, and (c) the triangle tree corresponding to (b).
Figure 6.17 (a) A typical overlapping situation, (b) non-safe removal, and (c) one correct solution to overlap removal. 130

Figure 6.18 An object with a convex corner made up of seven NURBS patches: three planes, three cylindrical parts, and one spherical part. 133

Figure 6.19 Fitting to an object with a convex corner. 133

Figure 6.20 (a) Plane development pattern generated from the fitting for the surface with convex corner, and (b) distribution of thread angles at mesh points projected above the plane development. 134

Figure 6.21 Fitting results with colors based on thread angle values; (a) 3D fitting and (b) 2D plane development pattern. 135

Figure 6.22 Stitched dart insertion: (a) 3D fitting and (b) 2D plane development pattern with a stitched dart. 136

Figure 6.23 Linear cut insertion: (a) 3D fitting and (b) 2D plane development pattern with a linear cut. 137

Figure 6.24 An example of the overlapping removal algorithm applied to a woven cloth ply, starting with a linear cut (loop = 1). A black circle denotes an edge-touching mesh point, and a white circle denotes an overlapping mesh point. The iteration stops when the OPL becomes empty as depicted in Figure 6.13. In other words, it stops when no more overlapping mesh points remain. 138

Figure 6.25 A polygonal cut obtained by applying the overlap removal algorithm: (a) 3D fitting and (b) 2D plane development pattern with a polygonal cut. 139

Figure 6.26 A woven cloth sheet (Kevlar), where a polygonal cut is specified with black lines. 140

Figure 6.27 The fitting result with a sheet of woven cloth Kevlar having a polygonal cut. 141

Figure 6.28 Fittings to an object with a concave corner: (a) 3D fitting result with no darts, (b) 3D fitting result with a linear cut, (c) 2D plane development pattern corresponding to (a), and (d) 2D plane development pattern corresponding to (b). 143

xii
Figure 6.29 (a) A fitting to a shoe-like object with colors based on the thread angle values at each mesh point and (b) the 2D plane development. .......................... 144

Figure 6.30 2D plane development patterns with trimmed darts: (a) two linear cuts are inserted, (b) polygonal cuts obtained by sequential method, (c) polygonal cuts obtained by cut distance priority method, and (d) polygonal cuts obtained by thread angle priority method. .......................... 145

Figure 6.31 3D fitting results with trimmed darts: (a) two linear cuts are inserted, (b) polygonal cuts obtained by sequential method, (c) polygonal cuts obtained by cut distance priority method, and (d) polygonal cuts obtained by thread angle priority method. 146

Figure 7.1 Mapping of a Tchebychev net onto a 3D surface. ................. 148

Figure 7.2 A surface region enclosed by four threads: two wefts \( x = x_1 \) and \( x = x_2 \), and two warps \( y = y_1 \) and \( y = y_2 \). ...................... 153

Figure 7.3 A facet enclosed by \( u = u_i, \ u = u_{i+1}, \ v = v_j, \) and \( v = v_{j+1} \). The total integral Gaussian curvature of a surface is approximated by summing up the integral Gaussian curvature of the entire facets. ...................... 161

Figure 7.4 Prediction of the thread angle \( \alpha_3 \) in a fitting to a quadrant of a sphere with radius \( R = 10 \) and the thread distance \( d = 0.7 \). The starting point is assumed to be at the north pole of the sphere. ................................. 163

Figure 7.5 Plots of the intermediate Gaussian curvature integrals projected above the plane developments for each different fitting of Figure 5.25. ................................. 166

Figure 7.6 Plots of the intermediate Gaussian curvature integrals projected above the plane developments for each different fitting of Figure 5.31. ................................. 167

Figure 7.7 Correspondence between a sequence of points on a geodesic in UV space and XYZ space. ............................... 173

Figure 7.8 Two fittings to a half cone; the fitting result of (a) produces shear deformations because the base path is not a geodesic, and (b) produces no shear deformation because the path is a geodesic. ............................ 176
Figure 7.9  Plots of thread angles at mesh points projected above the associated plane developments, corresponding to Figure 7.8 (a) and (b), respectively. ........................................ 176

Figure 7.10  (a) A $(u, v)$ parametric representation of a cone, where $R$ is the radius of a circle on $Z = 0$ plane, and (b) the 2D plane development pattern of a cone, where $0 \leq \theta \leq \pi \sin \alpha$. .............. 177

Figure 7.11  (a) Fitting to a cornered object with better initial conditions, and (b) the distribution of thread angles at mesh points projected above the plane development of (a). ....................... 179

Figure 7.12  Fitting results with colors based on thread angle values: (a) 3D fitting with better initial conditions, and (b) the plane development pattern. ........................................ 180

Figure 7.13  Results of fitting actual woven cloth fabrics (fiberglass) onto an object with a convex corner: the result on the left has its initial conditions corresponding to Figure 6.19 and the result on the right has its initial conditions corresponding to Figure 7.11 (a). ........................................ 181

Figure 7.14  The graph of minimum thread angles ($\gamma_{\text{min}}$) for the fittings to a cornered object with the better initial conditions by varying the distance between adjacent crossing. .................. 182

Figure 7.15  A free surface with several geodesics passing through the point where the Gaussian curvature is maximum in the surface. One of the geodesics, denoted by $g^*$, is generated by aligning the initial direction with a principal direction at the point. .... 184

Figure 7.16  Gaussian curvatures with colors for the free surface shown in Figure 7.15. ........................................ 185

Figure 7.17  3D fitting results with four different initial conditions: (a) the starting point is at $K = |K_{\text{max}}|$, the base path is aligned with a geodesic having the initial direction as the principal direction at the point; (b) same as (a) except that the starting point is not at $K = |K_{\text{max}}|$, but at the center of the surface; (c) same as (a) except that the base path has the initial direction not in the principal direction; (d) same as (a) except that the base path is not a geodesic, but a planar curve. .............. 186
Figure 7.18 2D plane development patterns with four different initial conditions: (a) the starting point is at $K = |K_{max}|$, the base path is aligned with a geodesic having the initial direction as the principal direction at the point; (b) same as (a) except that the starting point is not at $K = |K_{max}|$, but at the center of the surface; (c) same as (a) except that the base path has the initial direction not in the principal direction; (d) same as (a) except that the base path is not a geodesic, but a planar curve.

Figure 7.19 Gaussian curvatures with colors: (a) the original shape, and (b) the (globally) modified shape.

Figure 7.20 Fitting results for the new shape with colors assigned, based on the thread angle values: (a) 3D fitting and (b) 2D plane development pattern.

Figure 7.21 Plot of thread angles between wefts and warps at mesh points projected above the plane development, corresponding to the fitting of Figure 7.20.

Figure 7.22 Plots of the intermediate Gaussian curvature integrals ($|I_{K_{ij}}|$), corresponding to the fittings of Figure 6.21 and Figure 7.20.

Figure 7.23 The original shape of an octant of a sphere defined by a rational biquadratic B-spline surface.

Figure 7.24 The plot of $|I_K|$ as a function of $w_{1.1}$ and $w_{2.1}$.

Figure 7.25 An example of shape modification; (a) is the original shape of the octant of a sphere, (b), (c), and (d) are modified shapes whose $I_K$ values are equal to $\pi/4$.

Figure 7.26 An example of shape modification with colors assigned to the control points and control polygon edges of Figure 7.25, where $W$, attached to the color bar, denotes the value of the weights of the control points.

Figure 7.27 Deviations with colors for the shapes shown in Figure 7.25, where $D$, attached to the color bar, denotes the deviation defined by Equation (7.39). Clearly, (d) has minimum deviation among the three ((b),(c), and (d)).

Figure 7.28 Gaussian curvatures with colors for the shapes shown in Figure 7.25.

Figure 7.29 3D fitting results with colors for the shapes shown in Figure 7.25.
Figure 7.30 The plot of thread-angle for the shoe-like example with the same initial conditions as Figure 6.29. ........................................ 208

Figure 7.31 Gaussian curvatures with colors for the shoe-like object. ........... 208

Figure 7.32 The plot of intermediate Gaussian curvature integrals, $I_{K_{i,j}}$, for the shoe-like object. ............................................... 209

Figure 7.33 Control points for the shoe-like object. The control points with an asterisk represent the on-surface control points. ........... 210

Figure 7.34 An example of degree elevation applied to an octant of a sphere; (a) shows the original control polygon, and (b) shows the control polygon after degree elevation. ........... 215

Figure 7.35 3D fitting results to the shoe-like object where shape modifications are applied: (a) the original shape, (b) the modified shape obtained by the optimal weight finding algorithm alone, (c) the modified shape obtained by adjusting the on-surface control points $c_{0,2}$ and $c_{6,2}$, assuming the deviation threshold $h_D = 100.0$, and by applying the optimal weight finding algorithm for $w_{1,5}$ and $w_{5,5}$, (d) the same as (c) but with the deviation threshold $h_D = 400.0$. ................................. 221

Figure 7.36 Deviations with colors for the shapes shown in Figure 7.35. ........ 222

Figure 7.37 Flowchart for modifying a surface shape. ............................ 224

Figure 8.1 Overall process flow of the CAD system for simulating the layout of woven cloth composites. The comment put in parentheses represent the sections and the algorithms that describe the process. ......................................................... 230

Figure F.1 The hue hexagon. A hue value $\theta$ is measured by the angle from “Red.” ................................................................. 260

Figure F.2 Two different color assignments to the thread angles for a fitting to an octant of a sphere under the set of initial conditions given by Table 5.2 (c) in Section 5.7.1; (a) colors are assigned with the non-linear function for the fiberglass sheet, and (b) colors are assigned with the non-linear function for the Kevlar sheet. ............................... 263
ACKNOWLEDGMENTS

I am very grateful to my research advisor and the chairman of my doctoral committee, Dr. Michael J. Wozny, for his guidance during the last four years. This work has been improved as a result of his stimulating comments and advice, as well as valuable opportunities for public presentations that I have received from him.

I would like to thank Dr. Harry W. McLaughlin, Dr. David L. Spooner, Dr. Joshua U. Turner, and Dr. Peter R. Wilson for serving on my doctoral committee. I greatly appreciate the consideration each of the committee members has taken in my progress during the last few years.

I would also like to thank Dr. David E. Breen for his continuous encouragement and technical advice in many aspects throughout my thesis work. It was he who motivated me to conduct this thesis project. I must thank Mr. Paolo Denti for his contribution to the darting problems and for stimulating discussions during his visit at DMI (Design and Manufacturing Institute). Thanks are also due to DMI staffs, Ms. Mary N. Johnson, Ms. Ilene Blanchet, and Ms. Paula Popson for their continuous assistance from the beginning of my Ph.D. research here at RPI.

I am also thankful to Dr. Volker Paedelt, Technical Manager of RPI’s Center for Composite Materials and Structures, for his guidance on how woven cloth composites are placed on the fuselage of a glider, and for giving us sample composite fragments (Kevlar, fiberglass, and graphite).

I am gratefully indebted to many people who helped me with providing references related to my thesis project. Among them, my sincere thanks are due to Dr. Barry P. Van West of Boeing Defense & Space Group for the references on “locking angle” and “slippage” of composite materials, to Dr. Sandra L. Samelson of State University of New York, New Paltz for the references on the Tchebychev net including her master and Ph.D. theses, and to Dr. Haruki Imaoka of Nara Women’s University in Japan for the references on the mechanical aspects of garment design.
The financial support from Overseas Scholarship Program of IBM Japan Ltd. is greatly acknowledged.

Finally, I would like to dedicate this thesis to my family, especially to my wife Tomoko and my son Haruki. Without their help, patience, and love, I could not have finished a Ph.D.
ABSTRACT

Woven cloth composite ply materials have been extensively used to reinforce the structure of automobiles and aircraft, due to their outstanding weight-to-strength ratio and their remarkable flexibility in “fitting” to complex geometries. Here, “fitting” means that a ply is deformed and applied to a 3D surface such that it is in contact with the surface everywhere. However, it is not always possible to fit a woven cloth composite ply to doubly curved surfaces. In these instances it is necessary to cut pieces, or darts, out of the cloth in order to accomplish the fitting.

The goal of this dissertation is to investigate the geometric aspects of forming 3D woven cloth composite parts, and to provide a better understanding for the fitting mechanism. Specifically, this dissertation focuses on the following algorithms: (1) the algorithm for fitting a 2D ply of woven cloth composites onto a 3D curved surface, (2) the algorithm for obtaining the 2D “flattened” shape that corresponds to the area used for producing the fitting, (3) the algorithm for inserting darts where necessary, and (4) the theories and the algorithms derived from our model for woven cloth composite ply, in order to predict and prevent anomalous events such as wrinkling and tearing.

We propose a computer aided design (CAD) system that incorporates these algorithms and assists in the lamination process for forming 3D woven cloth composites. In order to implement such a CAD system, the following requirements must be met: (1) an accurate model of woven cloth composite plies, (2) a flexible model for 3D surfaces, (3) a mapping (fitting) method between 2D and 3D spaces, (4) a method for obtaining 2D plane developments, (5) a method for inserting darts where necessary, and (6) a method for predicting and preventing anomalous events.

We then describe our solutions to each of these requirements. Specifically, for requirement (1), a Tchebychev net cloth model, or a cloth model in which threads are treated as inextensible, is presented. For requirement (2), a NURBS (Non-Uniform...
Rational B-Spline) surface model is adopted. For requirement (3), a numerical method for mapping a mesh point in 2D space onto a 3D curved surface is proposed, emphasizing the vital effect of the initial conditions. The solution to requirement (4) is shown to be straightforward due to our model for woven cloth.

For requirement (5), several dart models are proposed, followed by the description of algorithms for inserting darts. For requirement (6), anomalous events are first defined, and then two major algorithms for preventing them are described. The first algorithm automatically finds good initial conditions that lead to the fitting with the least possibility of anomalous events, assuming that the 3D surface shape is fixed. The second algorithm modifies the surface shape either by finding optimal weights of the control points that are used to define the given NURBS surface, or by adjusting the control polygon, assuming that the initial conditions for the fitting are fixed. Both of these algorithms are based on the differential geometry properties derived from the Tchebychev net assumption of the cloth.

Computer simulation examples with graphical renderings are included wherever applicable. These graphical tools are shown to be quite useful for the CAD system for simulating the lamination procedure in order to intuitively analyze the kinematic conformability prior to actual manufacturing.
CHAPTER 1
INTRODUCTION

With the advent of lighter and stronger composite materials over the last two decades, more and more metals have been replaced by composite materials in aircraft and other vehicles. It is not surprising that the airframes of certain aircraft (e.g., Lear Fan 2100) are totally made of composite materials, mainly graphite/epoxy [44]. With this trend of replacing metals with composite materials, new lamination technologies have been developed to manufacture composite materials in order to form an airframe. During the lamination process, plies of composite sheets must be cut to the required size, with the required fiber orientation, and laid up onto the surface of a die or molding tool prior to curing. It is the process of deforming a woven cloth composite sheet into a 3D shape by pressing it onto a mold without gaps and wrinkles that we will refer to as a “fitting.” Unfortunately, the cutting and orientation process has not yet been fully automated; it has primarily been implemented with manual procedures. It is therefore a time-consuming and inaccurate process which is in need of design automation tools.

In this dissertation, we focus on the geometric aspects in the lamination process for forming woven cloth composites and propose a computer-aided design (CAD) system that incorporates the technologies presented here. With this system, designers can simulate and compare a variety of fittings under a given set of initial conditions by means of a computer graphics display, predict the resulting 2D flattened patterns, and prevent anomalous events such as wrinkling and tearing.

1.1 Contributions

From a global viewpoint, the contributions of this thesis are twofold: (1) the development of a methodology for constructing a CAD system for simulating the lamination of woven cloth composites, and (2) the improvement of the basic
understanding of the mechanism as to how a woven cloth composite “ply” deforms its shape to fit into complex geometries.

Componentwise, we have achieved the following:

- development of a new mapping (fitting) method which simulates a wide variety of fittings to an arbitrary composite NURBS (Non-Uniform Rational B-Spline) surface composed of one or more NURBS surface patches

- development of an algorithm for producing a 2D “flattened” pattern corresponding to the area in the woven cloth composite ply used for a fitting

- development of algorithms for inserting several types of “darts” (stitched darts, polygonal cuts, and linear cuts), in order to prevent anomalous events

- development of algorithms derived from a Tchebychev net assumption (or the assumption of inextensibility of a thread in cloth) including a method for predicting the thread angle based on the quantity called the intermediate Gaussian curvature integral, an algorithm for automatically finding good initial conditions for a fitting, assuming a fixed 3D surface shape, and an algorithm for modifying a NURBS surface shape either by finding optimal values for the weights of the control points of the surface, or by adjusting the control polygon, assuming fixed initial conditions for a fitting

- development of graphical tools to visually identify what a 3D fitting result looks like, what a 2D “flattened” pattern corresponding to a 3D fitting result looks like, where excessive shear deformation may occur, where “darts” should be inserted, and how the Gaussian curvature for a given surface as well as the intermediate Gaussian curvature integral for a 3D fitting result is distributed

- partial verification of the validity of our models and the method for fitting and dart insertion by comparing the simulation results with those obtained by actual fitting experiments
1.2 Organization of the Thesis

The remainder of the thesis is divided into seven more chapters. Chapter 2 first describes the characteristics of woven cloth composite materials used in aircraft and other vehicles, presents current problems during the design and manufacturing of woven cloth composite parts, and lists several requirements for solving the problems. The requirements include: (1) an accurate model of woven cloth composite sheets, (2) a flexible model for 3D surfaces, (3) a mapping method between 2D cloth space and 3D surface space, (4) a method for obtaining 2D plane developments, (5) a method for handling “darts”, and (6) a method for predicting and preventing anomalies that may occur during the lamination process.

Chapter 3 presents a background survey of research on “fitting” cloth. Chapter 4 presents a model of woven cloth composites, a model for a “locking angle”, a model for “slippage”, and a model for 3D surfaces. Requirements (1) and (2) are covered in this chapter.

The mapping method (or fitting method) based on numerical computations by solving a system of non-linear equations and the method for obtaining plane development patterns are described in Chapter 5, which correspond to requirements (3) and (4). Chapter 5 also focuses on the following three sub-topics. The first describes the method for specifying initial conditions which lead to a very flexible fitting, where the initial conditions include the reference point both in 2D cloth space and 3D surface space, the initial orientation of the woven cloth, and the initial path with which the cloth is aligned. The second sub-topic describes the method for modifying the mapping calculations when they involve two or more adjacent surface patches. The third sub-topic describes the formability, or the measure of how much energy is necessary to deform a given 2D woven cloth composite ply into a given 3D surface. Several fitting simulations together with a physical experiment are also presented in Chapter 5. The main emphasis throughout these simulations is how significantly the initial conditions affect the resulting fitting configuration.

Chapter 6 discusses the problems of anomalous events such as wrinkling and
tearing which may occur during the lamination process, and presents the method for preventing them by inserting “darts.” Thus, requirements (5) and (6) are covered in this Chapter. Alternative methods for preventing anomalous events, based on the differential geometry properties of the deformed woven cloth composite ply, are mentioned in Chapter 7. They include the method for finding good initial conditions for a fitting when a 3D surface is fixed, and the method for modifying a surface shape into a shape free of anomalous events when the initial conditions are fixed.

Graphical tools, including 3D wireframe display as well as shaded image display for visualizing fitting results and their statistics, are demonstrated through simulation examples in Chapters 5 to 7. Finally, Chapter 8 summarizes our current work and points out possible future recommendations.
CHAPTER 2
PROBLEM STATEMENTS AND THE OUTLINE OF A CAD
SYSTEM FOR FORMING WOVEN CLOTH COMPOSITES

This chapter first describes the characteristics of composite materials used for aircraft and other vehicles, then discusses problems and requirements in the current composite parts manufacturing process, and finally presents the outline of a proposed CAD system for forming woven cloth composites.

2.1 Composite Materials in Aircraft and Other Vehicles

Composites consist of a reinforcing material suspended in a “matrix” material (e.g. epoxy) that stabilizes the reinforcing material and bonds it to adjacent reinforcing materials [44]. As illustrated in Figure 2.1, the three major composite production forms are: (a) chopped fiber, (b) unidirectional tape, and (c) bidirectional woven cloth fabric (sometimes called broadcloth or broadgoods).

Metals and chopped fiber reinforced composites are isotropic, having the same material properties in all directions.

Unidirectional fiber-reinforced composites, like wood, are strongest in the direction along the axis of the fibers. Common arrangements for tailoring fiber orientation include 0 degree, 90 degree, and ±45 degree orientations of fibers with respect to the principal axis direction. Generally, a thin sheet (≈ 1 mm) of fiber-reinforced composites having θ degree orientation with respect to the principal axis is referred to as θ ply, or simply ply. However, since a unidirectional fiber-reinforced composite ply is not strong in the direction perpendicular to the fiber axis, it is usually the case that alternate layers of plies at 0, 45, 90 degree orientation are laid-up to form a stronger composite. A unidirectional tape is usually preimpregnated with the matrix material, and thus it is called “prepreg.”

Woven cloth fabrics are bidirectional, with fibers running at 0 and 90 degrees.
They may also be prepreg, but preforms (partially cured form) such as commingled fabrics [106] are more popular due to their drapability and ease of handling. Structurally, woven cloth is composed of vertical threads (called warps) and horizontal threads (called wefts). Wefts and warps are interwoven to form a sheet of cloth as seen in Figure 2.2.

Today’s aircraft employ advanced composites of the fiber-reinforced type (production forms (b) and (c)), partly because they have outstanding strength-to-weight ratio, and partly because their structural properties may be tailored to the expected loads in different directions [22]. In the following, we refer to a particular fiber-reinforced composite by writing the name in the format “fiber/matrix.” The main fibers presently used in aircraft applications are graphite, boron, aramid (e.g. Kevlar49\(^1\)), and glass and the main matrix material is epoxy resin [78]. For example, the Boeing 757 and 767 aircraft use graphite/epoxy for many of the control surfaces, and a hybrid aramid-graphite/epoxy composite for the wing-to-fuselage fairing, undercarriage doors, engine cowlings, and fixed trailing-edge panels. A graphite/epoxy composite is popularly referred to as “carbon fiber composite.”

\(^1\)Kevlar is a trademark of Du Pont.
Figure 2.2: A close-up of woven cloth fiberglass sheet.

Among fiber-reinforced composites, bidirectional woven cloth (fabric) composites has a salient feature that they can “fit” to fairly complex geometries by changing the angle between wefts and warps. Yet not much is known as to the mechanism of how they deform to fit into complex geometries. In order to shed some light on this problem, our attention in this thesis is paid to this type of composites.

2.2 Current Composite Parts Manufacturing Process

Here we consider the overall current manufacturing process for 3D composites. By 3D composites, we mean the final 3D shapes of composite plies fitted and deformed onto a specific portion of a curved surface. We could also think of 3D composites as macro-composite structures, consisting of multiple layers of plies, and each ply as a micro-composite structure. Not surprisingly, the process of making these 3D composites is slow, tedious and devoid of most design and manufacturing automation.

The design stage of making 3D composites begins with specifying the 3D shape of the object. Next, the position and shape of each ply that comprises the 3D part must be specified. The designer, at this point, must determine how each flat sheet of reinforcing material will be oriented on the complex 3D shape of the part. Once the
Figure 2.3: A sheet of woven cloth graphite to be cut out on the table.

Outline of each sheet is specified on the 3D shape, a 2D pattern must be described that can be cut out of a flat sheet, and subsequently deformed to fit completely on the predefined 3D shape and outline. Figure 2.3 shows an example of a rolled sheet of woven cloth graphite to be cut out on the table.

During the manufacturing stage, a mold is produced from the 3D shape description, and the 2D patterns are cut out of sheets of the reinforcing material. A laminator then takes the sheets and places them on the mold, following the instructions of the designer. For non-structural parts, it is necessary to deform many of the plies during the manufacturing process, in order to force the plies to lie completely on the mold. In many cases, the reinforcing materials reach their limit of deformation, and must be cut and “darted” in order to force them to lie completely on the predefined surface. This kind of cutting and darting is unacceptable for structural parts. For these parts, the designers must take extra care in designing parts with
simpler shapes that do not require darting. If complex shapes are required, designers must spend significant time determining how to specify the ply outlines in order to remove the need for darting. Once each individual sheet has been applied manually, the part is placed in an autoclave and cured. The final step involves trimming the excess edge material to produce the finished part. Figure 2.4 shows a fuselage of a glider reinforced with Kevlar/epoxy composites.²

2.3 Problems

Currently, there are many problems and inefficiencies associated with the design and manufacture of 3D composite parts. For many segments of the process

²This picture was taken by the author with permission of Dr. V. Paedelt at the composite manufacturing laboratory in RPI.
there exists little or no design automation tools. Though some computer-based
tools do exist for the design and structural analysis of the original 3D shapes, the
rest of the design and manufacturing process is done “by hand.” Technologies and
tools for specifying the ply outlines on the 3D shapes are unavailable. Given the
3D ply outlines, the 3D pattern must be “flattened” in order to produce the 2D
pattern for cutting the plies. It is here, in the 2D pattern making process, where
there is the greatest need for design technology. Also, as stated earlier, the 2D plies
are deformed and darted in order to force them to fit on the 3D surface. There
is a need to create design technologies and tools that can automatically create the
2D pattern, including the darts, that take into consideration the true deformation
properties of the reinforcing material and the steps taken to press the ply onto the
mold. Without design tools of this kind, the decisions as to where the darts should
be placed and how the sheets are applied to the molds are left to the individual
laminator on the factory floor.

Granted, at this point in time, the laminator is the most qualified person to
make the darting and adjusting decisions, but leaving that decision up to the in-
dividual laminator has some ramifications. For example, two laminators will not
make the same part in exactly the same way. Each laminator will apply the sheet
in a slightly different fashion, inserting darts in slightly different positions. This
not only produces inconsistent parts with different structural properties, but it is
also time-consuming and inefficient, because each laminator must spend time deter-
mining how to lay down the sheet and where to place the dart. Additionally, the
laminator must manually cut out the dart with a pair of shears and place a patch
over the dart seam. Since the most commonly used graphite/epoxy composite is
substantially more expensive than aluminum at the present time, the abundance
of wasted fragments may be costly if a laminator cannot skillfully and judiciously
insert darts.

In the worst case, wrinkles and folds may appear on the resulting shape of
the 3D composite. Wrinkles and folds are considered serious defects because non-uniform reinforcement of the structure might lead to an early fracture. Therefore, it is imperative that no wrinkles or folds should remain on the 3D composites.

2.4 Technology Needs and Goals

Given the growing number of matrix composite parts in today’s advanced aircraft and vehicles, it is extremely important to produce technologies and tools to support the automated design and manufacturing of these advanced structures. Specifically, the following technologies and tools are required:

- an accurate model for woven cloth composites, that captures most of the observed properties of advanced composite materials,

- a flexible model for 3D surfaces, that makes it possible to represent shapes of aircraft and other vehicles,

- a mapping method between 2D woven cloth sheet and 3D surface, considering deformations of the cloth,

- a method for predicting “flattened” patterns that are used for fitting a sheet to a specific portion of the surface,

- a method for inserting one or more “darts” wherever necessary,

- a method for predicting and preventing anomalies such as wrinkling, folding, and tearing.

When the above listed technologies and tools are implemented on a computer, a designer can simulate the lamination process and automatically generate the 2D patterns necessary for a 3D composite part. The important decisions as to what is the best ply outlines and orientations for a particular part can also be made with the assistance of these tools. The designer may iteratively test various ply configurations and application strokes to see which one produces the best fit to the 3D part with a
minimum number of darts and adjustments, or with minimum deformation energy. Once the designer has completed the design and generated a 2D pattern with darts, all laminators are given the same pattern for the same part, and are instructed how to place the ply on the mold in a consistent fashion. This modeling and simulation capability will also lay the groundwork for automatically determining the 2D pattern and 3D pressing strokes. Thus we can potentially save time and money, and increase quality, by introducing computerized tools that meet the above requirements.

2.5 A Proposed CAD System for Forming Woven Cloth Composite Parts

In order to overcome the problems of current 3D composite part design and manufacturing, we propose a computer-assisted environment in which designers can simulate the lamination process in an iterative fashion. As we have seen in the previous section, the bottleneck in current composite layout design lies in the process of fitting a 3D configuration. It is imperative that a computer-aided system for woven cloth composite layout design incorporate the ability to accurately and quickly evaluate a 3D fitting configuration. Here we present an overall CAD system as a solution that satisfies the technology requirements mentioned in the previous section.

Figure 2.5 illustrates the basic flow of a computer-aided woven cloth composite layout design process. We assume that 2D woven cloth composites and 3D surface geometry data are provided as databases.

A typical session in our integrated environment proceeds as follows: First, a designer chooses a certain 2D woven cloth composite from the material database and a certain surface shape from the geometry database. Second, he/she specifies the initial conditions. This corresponds to the process “specify the initial conditions” in the figure and includes the specification of the starting point for the fitting (mapping), a direction to which the initial path must be aligned, and one or more pressing strokes that correspond to sweeping directions from the initial path. These initial conditions must be sufficient to uniquely determine the mapping calculation.
Figure 2.5: The basic flow in a CAD system for forming woven cloth composite.
which follows.

As the name implies, the “mapping” process computes the 3D coordinate of each 2D mesh point of a woven cloth composite ply when it is deformed and fitted onto the curved surface. It is usually a complex function of the 3D surface shape, the deformation properties of the material and the steps taken to press it onto the 3D surface. Therefore, in our proposed system, the entire mapping process is approximated by repeatedly calculating the mapping of a discrete mesh point from the 2D cloth space into the 3D space of the curved surface. During the mapping computation, situations may occur in which there is no solution to the mapping, or, if there is a solution, it is unacceptable because it exceeds the deformation limit of the given material. In such cases, “exception handling” must be performed. The exception handling may include the insertion of “darts”, the modification of the surface shape locally, the subdivision of the surface region, and/or the replacement of 2D material.

If everything goes well, the result of the fitting is displayed on a screen and verified. If the designer is satisfied with the result, then he/she will proceed to the “plane development” process; otherwise he/she must go back to the specification of initial paths. The plane development process (or “flattening” process) is responsible for identifying what portions of the 2D woven cloth composite ply are actually used for the mapping. This data is necessary not only for understanding how laminators should cut the material with a pair of shears, but also for evaluating the feasibility of the cutting. If he/she is not satisfied with the predicted result of the flattened patterns, he/she can re-initiate the whole process.

In Chapters 5 through 7, we will elaborate on the individual functions in our proposed CAD system.
CHAPTER 3
RELATED WORK

This chapter discusses previous work related to our research. Since there are many technical subjects involved in our research as stated in the CAD system outline in the previous chapter, it may be verbose to present previous work in all subjects. Instead, our attention is focused on the subject of “fitting” a woven cloth sheet to a curved surface. For the remaining subjects, we discuss their historical backgrounds where they are described.

Historically, there have been two independent research directions related to the “fitting” of a woven cloth sheet to a curved surface. One direction has its roots in the study of a mathematical cloth model by Tchebychev [103]. Another direction of research has its roots in the study of fitting a real cloth material onto a specific surface, initiated by Mack and Taylor [61]. A common characteristic between the two directions of research is the assumption that the threads in a woven cloth sheet are inextensible. In spite of this similarity, however, no explicit interplay has ever been observed so far.

3.1 Tchebychev’s Cloth Model and its Derived Research Work

Here, the emphasis has been on how to model cloth and how to analyze the mathematical and mechanical properties of the model, rather than on how to fit a given piece of cloth to a surface. As will be discussed later, the fundamental model for woven cloth composites in this thesis relies on Tchebychev’s assumption where cloth is modeled as a continuum in which the threads are treated as continuously distributed and inextensible. This assumption is commonly referred to as a “Tchebychev net” assumption. Tchebychev himself only gave a brief description of this concept in his lectures, and it was Darboux [25] who publicized Tchebychev’s idea, and who supposedly defined the term “Tchebychev net.”
The same continuum model was used by Rivlin [81, 82, 83] in his mechanical theory of networks of inextensible cords and in the generalization of this theory by Adkins [1, 33]. Specifically, Rivlin [81] formulated a continuum theory of plane deformations of networks formed by inextensible cords, and obtained general solutions of both displacement and traction boundary-value problems. This theory is most directly applicable to large deformations of wide-mesh nets, since the network is idealized as having no elastic resistance to distortion. Although Rivlin’s theory gives a reasonable approximation to the large-deformation behavior of closely-woven fabrics, such networks do exhibit elastic resistance at the crossings. Adkins [1] generalized Rivlin’s theory by allowing nonlinear elastic shearing response. Then the theory can be used to model closely-woven fabrics which do exhibit elastic resistance to shear deformation.

Pipkin developed these theories in a series of papers. He discussed various kinds of singularities including singular fibers, collapsed regions, and folds [69]. Inextensible networks with slack [70] were also posed, in which a continuum model of networks of inextensible fibers that can support tension, but not compression, was assumed. When wrinkling is observed in deformation, a modified theory that allows the possibility of a continuous distribution of infinitesimal wrinkles was shown [73]. He described the equilibrium of a Tchebychev net [72] that has been spread over a given curved surface by formulating it with a system of non-linear hyperbolic partial differential equations, first derived by Servant [96]. As an example, Pipkin formulated the equilibrium of a Tchebychev net spread over a paraboloid. Wang and Pipkin [110, 111] extended the above theories by incorporating the effect of bending stiffness on plane deformation of a Tchebychev net.

The equilibrium of a Tchebychev net was also investigated by Kuznetsov [54, 55] in the study of cable-band membranes, assuming only tensile forces as the external load and a surface of revolution as the surface to which a net was fitted. He then showed a closed formula with two arbitrary constants for the condition of static equilibrium, and revealed the statical-geometric interrelations instead of the
ordinary constitutive relations usually employed in structural analysis.

The important features of a Tchebychev net are more or less related to the theory of global differential geometry, particularly to the Gauss-Bonnet’s theorem [27, 41, 100]. Samelson gave a condition that whether or not a piece of cloth modeled by a Tchebychev net can be globally fitted to a given curved surface which is modeled by a two-dimensional Riemannian manifold [91, 92, 94]. As an example, a closed hemisphere was proved to admit a global Tchebychev net [93].

All these efforts on a Tchebychev net have been made analytically; no efforts have been ever made from numerical standpoints.

3.2 Mack & Taylor’s Cloth Model for Fitting

The study of fitting a real cloth material onto a specific surface was initiated by Mack and Taylor [61]. They defined several basic assumptions for a model of woven cloth and investigated the mathematical conditions necessary for fitting a piece of cloth to a surface of revolution. As an example, they carried out experiments by fitting their model to a sphere. Their assumptions include the inextensibility of threads, which is equivalent to the Tchebychev net assumption. In addition to that, they assumed that the distances between adjacent horizontal weft threads and between adjacent vertical warp threads are much smaller than the largest radius of curvature on the surface.

Partly because Mack and Taylor gave an intricate formula in terms of a differential equation that held only for a surface of revolution, and partly because they investigated an analytic solution based on the integration of the equation, since the appearance of their paper there has not been much progress on this kind of approach. The first computational method for fitting woven cloth to a 3D surface was proposed by Robertson et al. [84, 85]. They reduced the fitting problem to that of solving an intersection problem between two spheres and a quadric. Both Mack and Taylor’s, and Robertson’s methods do not provide a flexible way to define the initial conditions for a fitting. Specifically, Mack and Taylor assumed that initially the
threads should be positioned at 45 degrees to the meridian of a surface of revolution. Robertson et al. implicitly assumed in their experiment of fitting woven cloth to a sphere that two perpendicular longitudes were selected to be aligned with a weft and a warp, respectively. Both approaches provide little flexibility in changing the orientation of a given piece of woven cloth on the surface.

Heisey and Haller [38] first attempted to numerically solve a differential equation similar to Mack and Taylor’s. The initial conditions in their technique are slightly more flexible than the previous methods, allowing the threads to have an arbitrary orientation at the start of the fitting process. What was missing was a technique for fitting woven cloth to an arbitrary surface other than a surface of revolution.

Bergsma and Huisman [14] described a deep drawing of fabric reinforced thermoplastics in order to produce thin walled composites. They assumed pivoted joints and trellising type deformation in the fabric, and performed a computer simulation and compared the result with practical experiment. The simulation program was able to indicate whether a product can be deep drawn or not. Unfortunately, the applicable surface was limited to an open, rectangular box.

Van West [106, 107] was the first to consider the fitting of woven cloth commingled fabrics to a fairly arbitrary surface by modeling it as a bicubic Hermite patch. Like Robertson’s method, he reduced the fitting calculation to an intersection problem between two spheres and a bicubic Hermite patch. As for the initial conditions, he assumed that two perpendicular threads were selected from 2D cloth space, and that each was aligned with a predefined sequence of equidistant points on the surface. This was a further enhancement as well as great departure from Heisey’s method in that any curves could be chosen on the surface as initial paths, instead of choosing just one angle parameter. Van West also defined two exceptional cases: wrinkling and bridging. However, his way of coping with such exceptions was restricted.
There are two drawbacks to the approach that Van West identified in his thesis. (1) It is impossible to exactly express quadrics with bicubic Hermite patches. For example, a sphere cannot be represented exactly with these patches. (2) The initial paths represented by sequences of equidistant points on the surface are calculated by hand. The first drawback may not cause a serious problem since a good approximation to a sphere is possible even with bicubic Hermite patches. The second drawback imposes a large burden on users because calculating sequences of equidistant points on the surface in itself is not a straightforward task, unless the surface has a very simple shape.

Prakash et al. [75] address the inefficiencies involved in the manual construction of composite parts and describe an interactive system called AUTOLAY to improve it. They model a ply as a set of polygons and their composite forms are assumed to be unidirectional tapes. Therefore, their method for approximating flat pattern development can be applied to a limited set of 3D surface shapes such as strips, panel and shell type components which have only shallow curvatures.

3.3 Comparison of Research Work on Fitting Cloth

Table 3.1 compares major research efforts including this thesis on “fitting” woven cloth to a surface in terms of surface models, methods for specifying initial paths, methods for the mapping calculations of mesh points, and miscellaneous other features.

Our unique points over previous researches are as follows:

• flexibility in representing and controlling a 3D surface with NURBS surfaces (See Section 4.4)

• ability to simulate a great variety of fittings, based on a flexible way of specifying initial conditions (See Sections 5.1, 5.2, and 5.4)

• ability to predict the 2D flattened pattern necessary for producing a composite ply fitted to a given 3D surface region (See Section 5.5)
Table 3.1: Comparison of major research on fitting cloth to a surface.

<table>
<thead>
<tr>
<th>Surface Model</th>
<th>Initial Path</th>
<th>Mapping Calculation</th>
<th>Other Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mack &amp; Taylor (1956)</td>
<td>surface of revolution</td>
<td>differential equation (analytic)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>45 degree to the meridian (fixed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Robertson et al. (1981)</td>
<td>sphere, cone</td>
<td>intersection (two spheres and a quadric)</td>
<td>2D numerical plane development</td>
</tr>
<tr>
<td></td>
<td>two longitudes from the pole to the equator (fixed)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Dow et al. (1986)</td>
<td>sphere cylinder</td>
<td>unknown (numeric)</td>
<td>1. limited slippage effect</td>
</tr>
<tr>
<td></td>
<td>two perpendicularly intersecting curves</td>
<td></td>
<td>2. locking angle measure</td>
</tr>
<tr>
<td>Heisey &amp; Haller (1988)</td>
<td>surface of cycl. coord. system</td>
<td>differential equation (numeric)</td>
<td></td>
</tr>
<tr>
<td></td>
<td>any angle to the meridian (one degree of freedom)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Van West (1990)</td>
<td>bicubic Hermite patch</td>
<td>intersection (two spheres and a Hermite patch)</td>
<td>limited wrinkling and bridging treatment</td>
</tr>
<tr>
<td></td>
<td>a sequence of points (user supplied)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Aono (this thesis)</td>
<td>NURBS surface</td>
<td>1. planar curve, 2. geodesic path</td>
<td>1. locking angle model</td>
</tr>
<tr>
<td></td>
<td></td>
<td>intersection (two spheres and a NURBS surface, etc.)</td>
<td>2. slippage effect</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. 2D plane development</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>4. anomalous events prevention</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(1) dart insertion</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(2) finding good initial conditions automatically</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>(3) shape optimization</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>5. various display tools</td>
</tr>
</tbody>
</table>
• ability to insert darts where excessive shear deformation is predicted (See Section 6.2)

• ability to automatically provide good initial conditions that lead to the least possibility of anomalous events such as wrinkling and tearing, given a fixed 3D surface shape (See Section 7.2.3)

• ability to modify the surface shape which is likely to produce anomalous events, given fixed initial conditions (See Section 7.3)

• capability of a variety of graphic displays in order to visually evaluate a fitting result (See examples in Chapters 5 through 7)

These unique features will be discussed in subsequent chapters.
CHAPTER 4
MODEL DEFINITIONS

In this chapter, an accurate model of woven cloth composites, a flexible model for representing 3D surfaces, a geometric model for a “locking angle”, and a model for “slippage” will be presented.

The first two models correspond to the requirements stated in Section 2.4. For the model of a locking angle, a comparison is made to verify the validity of the model with measured locking angles. For the model of slippage, a fitting simulation is provided for visually assessing the validity of the model. The verification with respect to the model of woven cloth composites will be demonstrated through examples in Chapters 5 through 7. For the model of 3D surfaces, a NURBS surface representation is adopted because it is widely accepted as a standard representation for a curved surface in various graphical standards [47, 66, 67].

4.1 A Model for Woven Cloth Composite Plies

Here we present our model for woven cloth composite plies, that accounts for most of the properties of advanced composite materials such as glass, aramid, graphite, and boron. A woven cloth composite ply can be regarded as a woven fabric, consisting of horizontal and vertical threads interwoven in a specific fashion. The ply generally exhibits strong tensile strain resistance in the thread directions and weak shear strain resistance [28]. Consequently, a woven cloth composite deforms chiefly by changing angles between horizontal and vertical threads without elongation or shrinkage of its threads. Considering this, we have made the following assumptions for our woven cloth composite plies:

- The cloth consists of vertical and horizontal inextensible threads.
- There is no slippage at a crossing of a vertical thread (warp) and a horizontal thread (weft) when the cloth is deformed.
A thread segment between adjacent crossings is straight.

Although these assumptions are simple, they provide a good approximation to most of the advanced woven cloth composites such as fiberglass/epoxy, Kevlar/epoxy, and graphite/epoxy. Specifically, with the first assumption, the deformation is limited to “shear” deformation, and this assumption has been validated experimentally [10, 84]. Likewise with the second and third assumptions, cloth can be treated as a discrete mesh of “pin-fixed” points which are able to rotate around each mesh axis. These assumptions can be made because the actual woven materials used for composites are usually embedded in a sticky resin matrix and the distance between adjacent thread crossings is very small. In Section 4.3, we extend the second assumption to allowing a slippage effect.

Without loss of generality, we implicitly make the following auxiliary assumptions. Firstly, the initial shape of the woven cloth is assumed to be rectangular. Secondly, there is no physical difference between wefts and warps. Finally, regardless of the weaving pattern of the given woven cloth composite ply, we assume that a ply is modeled as a linked network of mesh points, where each mesh point is defined at the crossing between a weft and a warp, as illustrated in Figure 4.1 (b).
Figure 4.2: Thread angle \( \gamma \) at a mesh point \( P_{i,j} \)

Figure 4.1 (a) shows typical weaving pattern called a plain weave in which wefts and warps are alternately interwoven both vertically and horizontally. The mesh point defined as above is also called a regular mesh point. Later, we introduce another mesh point called an auxiliary mesh point (See Section 5.1.2). Topologically each mesh point is linked with four adjacent mesh points with equal distance at the outset.

For the discussion to follow, we define a thread angle, \( \gamma \), as the angle between the line segment \( P_{i,j}P_{i-1,j} \) and \( P_{i,j}P_{i,j-1} \) as illustrated in Figure 4.2, where \( P_{i,j} \) is a mesh point of interest and \( P_{i-1,j} \) and \( P_{i,j-1} \) are the two neighboring mesh points along a weft and a warp, and whose fittings are assumed to be done prior to the fitting of \( P_{i,j} \).

4.2 The Locking Angle and its Geometric Model

Here we present a geometric model of a “locking angle”, which serves as an important indicator of the beginning of an anomalous event caused by excessive shear deformation. Although a woven cloth composite ply is very flexible, it is well-known that there is a limit of the allowable shear deformation in terms of the thread
angle. This thread angle is called locking angle or jamming angle. Usually, a locking angle refers to the minimum thread angle (denoted by $\gamma_{\text{lock}}^{\min}$), and no further shear deformation may occur at an angle less than that. For the sake of later discussion, we shall define the maximum thread angle $\gamma_{\text{lock}}^{\max}$ as $\pi - \gamma_{\text{lock}}^{\min}$.

The locking (jamming) angle of a given woven cloth composite ply is determined by many factors. Dow et al. [62] describes that weave parameters such as the number of threads per inch, thread width, and spacing between threads determine the jamming angle. Another important parameter is the type of weave. For instance, it is observed that a plain weave, where threads are interwoven through every thread they cross, will jam at an angle much higher than an 8-harness satin weave, where the threads are interwoven only at every eighth thread they cross. Similarly, Ishikawa states in his paper [51] that the satin weave, particularly 8th harness satin weave, is widely used because it is more pliable and easier to fit onto a mold than the plain weave cloth.

Lindberg et al. [59] experimented with a number of commercial fabrics and investigated the formability in terms of shearing and buckling. They provide the table that includes the values related to $\gamma_{\text{lock}}^{\min}$ for 66 commercial fabrics, ranging from 0.1 (cotton with 5-harness satin weave) to 50 degrees (silk with plain weave). For woven cloth composite materials, Dow et al. [62] shows that most commercial fabrics exhibit jamming at angles less than 35 degrees.

Unfortunately, none of the previous papers describe the model of the locking angle. Here we present a model for the locking angle, that accounts for the phenomena described above, and discuss the validity of the model. In the following, we first describe the geometrically determined shear limit and then discuss the effects of the types of weave on our model of the locking angle.

As shown in Figure 4.3, let the distance between adjacent crossings be $d$, and the width of a thread be $h$. Assume $d \geq 2h$. As the ply undergoes shear deformation from Figure 4.3 (a) to (b), the thread angle $\gamma$ decreases from $\pi/2$. If we assume in-plane shear deformation, the distance $L$ shown in Figure 4.3 (b) decreases with
Figure 4.3: Geometry of in-plain shear deformation: (a) the original configuration and (b) a deformed configuration.

the increase of the angle $\theta$. There is a relation between the thread angle $\gamma$ and $\theta$ as

$$\gamma = \pi/2 - 2\theta.$$ 

The distance $L$ is expressed as the following formula:

$$L = d - 2\frac{h}{\cos 2\theta} = d - 2\frac{h}{\sin \gamma}.$$ 

Obviously, the distance $L$ must be non-negative. Geometric locking occurs when the distance $L$ becomes zero as illustrated in Figure 4.4. By setting $L = 0$ in the above equation, the thread angle at the geometric locking ($\gamma_{\text{geom}}^{\text{lock}}$) is calculated as

$$\gamma_{\text{geom}}^{\text{lock}} = \arcsin\left(\frac{h}{d}\right).$$ \hspace{1cm} (4.1)

For instance, if $d = 0.1$ inch and $h = 0.025$ inch, then $\gamma_{\text{geom}}^{\text{lock}} = 30$ degrees. Let us denote the distance between adjacent wefts as $d_f$, and the distance between adjacent warps as $d_w$. In Equation (4.1) we have assumed that $d = d_f = d_w$. Suppose $d_f \neq d_w$. Then we have a more general formula for $\gamma_{\text{geom}}^{\text{lock}}$ as follows:

$$\gamma_{\text{geom}}^{\text{lock}} = \max(\arcsin(\frac{h}{d_f}), \arcsin(\frac{h}{d_w})).$$
Figure 4.4: Geometric limit of shear deformation.

where \( \text{max} \) is a maximum function. Note that actual locking usually occurs at an angle \( (\gamma_{\text{lock}}) \) equal to or greater than \( \gamma_{\text{geom}}^{\text{lock}} \).

Up to this point we have assumed that the given woven cloth composite ply is a plain weave. Let us consider the effect of the type of weave upon the locking angle. There exists many types of weaving patterns [51] which can be essentially classified by the \( \text{repeat number}, n \), as illustrated in Figure 4.5. The value of \( n_w \) (\( n \) in the weft thread direction) is defined such that a weft thread is interlaced with every \( n_w \)-th warp thread. In other words, the same weaving pattern appears after every \( n_w \) warp thread in the weft thread direction. In general weaving structures, \( n_f \) (\( n \) in the warp thread direction) may differ from \( n_w \). However, for convenience, we assume that \( n_f = n_w = n \). Among the patterns, the weave of \( n = 2 \) is called \textit{plain weave} and \( n = 3 \) is one type of \textit{twill weave}. The minimum number of \( n \) for the \textit{satin weave} is 4, and this satin is usually called ‘claw foot satin’ or ‘Turkish satin.’

To capture the difference of the shear limit based on the type of weave, we provide a simple model as follows:

\[
\gamma_{\text{lock}}^{\text{min}} = (1 + \alpha \log_{10}(\frac{n}{n-1}))\gamma_{\text{geom}}^{\text{lock}} + \beta,
\]

(4.2)

where \( n \) is the repeat number (\( n > 1 \)), \( \alpha \) is a nonnegative constant (\( \alpha \leq 1 \)), and \( \beta \) is a constant such that \( |\beta| \ll \gamma_{\text{geom}}^{\text{lock}} \). With this model, it is obvious that the woven cloth ply of the plain weave has a locking angle greater than the woven cloth ply.
Figure 4.5: Several types of weaving patterns: (a) plain weave \( n = 2 \), (b) twill weave \( n = 3 \), and (c) 4-harness satin weave \( n = 4 \).

Table 4.1: Locking angles of several fabric weaves. Based on Table 7 in [62].

<table>
<thead>
<tr>
<th>Type of Weave</th>
<th>Thread Count/Inch</th>
<th>Locking Angle ( \gamma_{\text{lock}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Plain</td>
<td>50 ( \times ) 50</td>
<td>37.7</td>
</tr>
<tr>
<td>Plain</td>
<td>15 ( \times ) 10</td>
<td>36.5</td>
</tr>
<tr>
<td>4 Harness Satin</td>
<td>36 ( \times ) 55</td>
<td>28.5</td>
</tr>
</tbody>
</table>

of the satin weave. For instance, if \( \gamma_{\text{geom}}^{\text{lock}} = 30 \) degrees, \( \alpha = 0.5 \), and \( \beta = 0 \), then \( \gamma_{\text{min}}^{\text{lock}} = 34.5 \) degrees for the plain weave and \( \gamma_{\text{min}}^{\text{lock}} = 31.9 \) degrees for the 4-harness satin.

In order to verify our model, we quote the experimental data obtained by Dow et al. [62] as listed in Table 4.1. Unfortunately, Table 4.1 provides no direct parameters such as \( h \) and \( d \). We therefore need to infer these parameters with some additional assumptions. Specifically, we assume that the distances \( d_f \) and \( d_w \) are obtained by \( d_f = 1/c_f \) and \( d_w = 1/c_w \), respectively, where \( c_f \) and \( c_w \) represent the thread count per inch in the weft and warp thread direction. For example, the woven cloth in the second row in Table 4.1 has 15 \( \times \) 10 threads per inch, thus
Table 4.2: A set of parameters for our model of the locking angle that produces the observed locking angle in Table 4.1.

<table>
<thead>
<tr>
<th>$d_f$</th>
<th>$d_w$</th>
<th>$h/d_f$</th>
<th>$h/d_w$</th>
<th>Locking Angle ($\gamma_{\text{lock}}$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.020</td>
<td>0.020</td>
<td>0.250</td>
<td>0.250</td>
<td>37.7</td>
</tr>
<tr>
<td>0.067</td>
<td>0.056</td>
<td>0.203</td>
<td>0.245</td>
<td>36.5</td>
</tr>
<tr>
<td>0.027</td>
<td>0.018</td>
<td>0.136</td>
<td>0.202</td>
<td>28.5</td>
</tr>
</tbody>
</table>

$d_f = 1/15 \approx 0.067$ and $d_w = 1/10 = 0.100$. For each example, we also have to specify the thread width $h$ or the ratio $h/d$. We assume the ratio $h/d$ ranges from 0.10 and 0.25. In summary, we define parameters as listed in Table 4.2. Note that the constant parameters $\alpha$ and $\beta$ are assumed to be 0.808 and 0.000, respectively, for all the cases. Then locking angles obtained by assigning the parameters to our model become equal to those in Table 4.1. The important thing that we want to emphasize here is not finding the exact answer to a set of parameters that match the observed locking angles, but with reasonable assumptions, being able to produce similar results as those observed.

4.3 A Model for Slippage

The slippage of weft threads relative to warp threads in a woven cloth ply has been observed, resulting from transverse displacements of threads [59, 14], as illustrated in Figure 4.6. Slippage makes the fabric locally tighter or looser. It is an unwanted effect that is difficult to control. It was shown experimentally that slippage occurs just before the fabric locks [106]. Potter [74] examined that although slippage does occur at the crossing between a weft and a warp, the pin-fixed assumption remains valid in most cases.

Dow et al. [62] proposed a model for the slippage effect expressed as:

\[
\text{Thread spacing/Original Spacing} = 1 + S(1 - \frac{\theta}{90}),
\]

where $S$ is a slippage parameter, and $\theta$ is a minor angle between intersecting threads.
Their model has a linear relationship between the amount of slippage and the thread angle. Assuming that the slippage parameter $S$ is positive, as the thread angle becomes small, the thread spacing becomes large due to slippage. Since this slippage effect is based on the observed behavior when the fabric is subjected to uniaxial extension with no lateral constraints, the model may not be valid when biaxial extension or compression is applied.

Considering the previous approaches and experiments regarding the slippage effect, we have made the following assumptions in order to capture the slippage effect during the lamination process:

- No thread extends or shrinks due to slippage, but the distance between adjacent crossing between wefts and warps may be altered.
- Slippage occurs when a thread angle exceeds either the minimum slippage angle ($\gamma_{\text{slip}}^{\text{min}}$) or the maximum slippage angle ($\gamma_{\text{slip}}^{\text{max}}$), where $\gamma_{\text{lock}}^{\text{min}} \leq \gamma_{\text{slip}}^{\text{min}}$ and $\gamma_{\text{lock}}^{\text{max}} \geq \gamma_{\text{slip}}^{\text{max}}$.

Figure 4.6: Slippage effect.
The minimum and maximum slippage angles introduced here are determined by many factors including those affecting the locking angle as well as frictional forces applied at the crossings. With the above assumption, when slippage is predicted to occur, the distance between adjacent wefts \( (d_{\text{weft}}) \) and the distance between adjacent warps \( (d_{\text{warp}}) \) become \( d_{\text{weft}}' \) and \( d_{\text{warp}}' \) as illustrated in Figure 4.6. \( d_{\text{weft}}' \) and \( d_{\text{warp}}' \) are given by the following formula:

\[
\begin{align*}
    d_{\text{weft}}' &= d_{\text{weft}}(1 + S_{\text{weft}}) \\
    d_{\text{warp}}' &= d_{\text{warp}}(1 + S_{\text{warp}})
\end{align*}
\]

where \( S_{\text{weft}} \) and \( S_{\text{warp}} \) represent the percentages of the slippage effect along weft threads and warp threads, respectively. \( S_{\text{weft}} \) and \( S_{\text{warp}} \) are usually a function of the thread angle and the material used for each thread. As long as the same material is used for both wefts and warps, we assume \( S_{\text{weft}} = S_{\text{warp}} \). It should be remembered that no thread extends or shrinks even when slippage occurs.

Our model for slippage is different from Dow’s model [62] in that the slippage effect is assumed to occur only when the thread angle exceeds certain values, instead of linearly varying with the change of the thread angle.

Figure 4.7 compares the result of fitting to an octant of a sphere with slippage effect (dotted lines) and the result without slippage effect (solid lines). This simulation result conforms to the result obtained by Dow et al. [62] with a 10% slippage effect.

As Potter noted [74], the slippage effect is usually very small. This is especially true when we use a woven cloth composite ply impregnated with sticky resin. Therefore, without loss of generality, we will neglect the slippage effect, in the fitting simulation examples in Chapters 5 through 7.

### 4.4 A Model for 3D Surfaces

It is important to provide a flexible model for 3D surfaces, allowing us to represent shapes of aircrafts and other vehicles. A common feature of the geometry
Figure 4.7: Fittings to an octant of a sphere with slippage effect (dotted lines) and without slippage effect (solid lines).
of aircrafts and other vehicles is that they are almost completely composed of curved surfaces. Polygonal models are not appropriate because extremely large numbers of polygons may be required to provide sufficient accuracy for representing arbitrary curved surfaces. Primitive instancing and its operational extensions like constructive solid geometry (CSG) modeling [80] may be applicable as long as the surface shape is simple enough to be easily assembled from a given set of primitives. In fact, most of the previous physical fitting experiments were carried out by using simple shapes such as a sphere and a cone [84, 85]. It is therefore preferable to be able to represent such simple shapes exactly, in order to compare the simulation results with experimental data. The biggest drawback of CSG-like models in our context is that they cannot assure sufficient smoothness along the boundary of connected primitives. This is a serious problem when we consider smooth surfaces typically seen in aircrafts and automobiles.

Considering the above, we have adopted a NURBS (Non-Uniform Rational B-Spline) surface model [29, 87], a parametric representation with rational polynomials, as our 3D surface model. It allows the exact representation of quadrics (e.g. sphere) and surfaces of revolution, it allows local shape control, and it provides us with sufficient smoothness between adjacent patches. In addition, NURBS has been adopted in various graphic standards such as IGES [47], PDES [66], and PHIGS+ [67] for the representation of curved surfaces. The NURBS representation does have some drawbacks. These are:

- It may be expensive to evaluate the value of a given rational polynomial, if the degree of the polynomial is high.

- It is not closed under simple operations such as rotation. Therefore, numerical errors may be accumulated if we apply such operations repeatedly [98].

The salient features of a NURBS surface model, especially the ease of synthesizing and modifying a shape, outweigh its drawbacks. The formula of a NURBS surface is given as follows [29]:
\begin{equation}
r(u, v) = \frac{\sum_{i=0}^{Cu-1} \sum_{j=0}^{Cv-1} w_{i,j} c_{i,j} N_i^m(u) N_j^n(v)}{\sum_{i=0}^{Cu-1} \sum_{j=0}^{Cv-1} w_{i,j} N_i^m(u) N_j^n(v)},
\end{equation}

where variables associated with NURBS surface have the following meanings:

- $c_{i,j}$: control points of $(i, j)$ position,
- $w_{i,j}$: weights of $(i, j)$ position,
- $Cu$: number of control points along $u$–axis,
- $Cv$: number of control points along $v$–axis,
- $\{u_0, ..., u_{Cu+m-2}\}$: knot vector along $u$–axis,
- $\{v_0, ..., v_{Cv+n-2}\}$: knot vector along $v$–axis.

And $N_k^n(t)$, called \textit{B-spline basis function} is recursively defined as:

\begin{align*}
N_k^n(t) &= \frac{t - t_{k-1}}{t_{n+k} - t_{k-1}} N_k^{n-1}(t) + \frac{t_{n+k} - t}{t_{n+k} - t_k} N_{k+1}^{n-1}(t), \\
N_k^0(t) &= \begin{cases} 
1 & \text{(if } t \in [t_{k-1}, t_k) \text{)} \\
0 & \text{(otherwise)}
\end{cases}
\end{align*}
CHAPTER 5
FITTING A WOVEN CLOTH COMPOSITE PLY TO A SURFACE

The essential function in our proposed computer-aided design system for composite part layout is a flexible method for fitting a 2D woven cloth composite ply to a given 3D curved surface. In this chapter we focus on this issue, by describing how to specify initial conditions for fitting (mapping), how to perform the mapping calculation of mesh points between 2D cloth space and 3D surface space, how to scan mesh points during the mapping calculation, and how to adjust the mapping calculation when the thread on which the mesh point of current interest extends across the boundary of one NURBS patch to its adjacent one. Finally, we will present several examples of fittings which are obtained by the algorithms developed here.

5.1 Methods for Specifying Initial Conditions

Initial conditions must be sufficient, but not excessive, in order to uniquely determine the mapping between a point on a piece of cloth in 2D space and the corresponding point on a surface in 3D space. If, for instance, they are over-specified, it is likely that no solution will satisfy the given initial conditions. If, on the other hand, they are under-specified, a unique solution may not be found. Initial path specifications are very important because the flexibility of the mapping problem is determined by the flexibility of the initial path specification.

In the following, we will first describe previous methods for specifying initial conditions for the fitting problem and then present our new approach. It should be noted that initial conditions are classified into two components: (1) initial conditions for 2D space and (2) initial conditions for 3D space.
5.1.1 Previous Method for Specifying Initial Conditions

Although there are minor differences in previous approaches [84, 85, 14, 107], a common assumption among them is the specification of initial conditions by fixing two yarn paths in both 2D and 3D spaces. Such yarn paths are sometimes called “constrained yarn paths.” In 2D space, one weft yarn and one warp yarn are selected as a pair of constrained yarns. They are mutually perpendicular to each other. In 3D space, each constrained yarn path is mapped into a curve, defined by a sequence of equidistant points on the given surface, as shown in Figure 5.1. The distance between adjacent points on the 3D surface is assumed to be exactly equal to the distance between adjacent points in 2D space.

By specifying a pair of constrained yarn paths, both the 2D cloth region and the 3D surface region are generally divided into four regions. The mapping calculation of points between 2D and 3D spaces is carried out for each quadrant independently.

This method has the following advantages: (1) it is easy to calculate the mapping between 2D and 3D points on the initial paths because the cloth threads are aligned with the paths, and (2) it is straightforward to scan all the mesh points during the mapping calculation because scanning along warp yarns and weft yarns will produce the same results due to symmetry. This second advantage is illustrated in Figure 5.2 as an example of scanning $m \times n$ mesh points.
The advantages of the previous method stem from its simplicity in specifying 2D initial conditions. However, there are several problems associated with the technique for specifying initial conditions.

The first problem is that the method does not allow sufficient flexibility in specifying the constrained yarn paths in 2D space. It only allows the paths to lie on perpendicular warp and weft threads as shown in Figure 5.1. It does not allow the specification of arbitrary constrained yarn paths where the angle between a path and any thread intersecting the path is some value other than 90 degrees. Figure 5.3 illustrates the in-plane shear deformation with two different initial paths. Specifically, Figure 5.3 (a) corresponds to the previous method, and the in-plane shear deformation tends to be diagonally dominant. On the other hand, Figure 5.3 (b) shows a different method for specifying initial paths, in which the angle between a path and any thread intersecting the path is 45 degrees. The in-plane shear deformation with this method tends to be either horizontally or vertically dominant.

The second problem is that the method imposes on users the cumbersome task of calculating a sequences of equidistant points on the 3D surface to be fitted.

Figure 5.2: Scanning order of mesh points (previous method).
Figure 5.3: Initial paths in 2D affect the flexibility in deformation; (a) the angle between a path and any thread intersecting the path is 90 degrees and the in-plane deformation is diagonally dominant (previous method); (b) the angle between a path and any thread intersecting the path is 45 degrees and the in-plane deformation is vertically dominant.
As long as the shape of the surface is simple (such as a sphere), this may not be a serious task. If, however, we deal with an arbitrary curved surface (such as a sculptured surface), it is as difficult as determining the whole mapping between 2D and 3D points.

The third problem is that there is a possibility of leaving “uncalculated regions” (See Figure 5.4) within the original surface area. There are two cases in which this problem arises. The first case may occur when the constrained yarn paths on a 3D surface are badly selected so that the mapping calculation cannot reach into the shaded region as in Figure 5.4 (a). The second case may occur when the surface area has a concave shape, as in Figure 5.4 (b), preventing the mapping calculation from reaching into the shaded region, regardless of the yarn paths.

5.1.2 A New Method for Specifying Initial Conditions

Here we describe a new method for specifying initial conditions. With this method, the first two problems in the previous method can be overcome. The solution to the third problem (“uncalculated surface region”) will be addressed in later sections.
5.1.2.1 Initial conditions in 2D space

Our method for specifying initial conditions in 2D space is based on the actual manufacturing practices utilized by laminators during the lamination process. Laminators usually fix one arbitrary path not necessarily aligned with a particular thread. Then, they sweep the ply along arbitrary directions. This process is analogous to the process of ironing out wrinkled clothing after washing and drying. A crucial concept here is that the resulting deformation on a 3D surface generally depends on the history of how the ply has been swept so far.

In order to simulate this practice, we define one initial path instead of defining two perpendicular initial paths. We refer to this as the base path or the guide line. As shown in Figure 5.5, the base path is not necessarily aligned with any weft or warp thread, but must be a straight line in 2D space. A starting point must be chosen somewhere on the base path, but it is not expected to coincide with a thread crossing. When a base path is specified as above, auxiliary mesh points are automatically defined as the intersections between the base path and threads, as shown in open circles in Figure 5.5. Once auxiliary mesh points are defined, the topology of the initial (regular) mesh points (shown in Figure 4.1 (b)) has to be modified.

The base path generally subdivides the 2D space into two half-spaces, and for each half-space, we define a sweeping direction with a 2D vector. It is the sweeping direction that determines the order of mapping calculations among regular mesh points belonging to the half-space.

As a special case, if we align the base path with a specific thread and if we allow the two sweeping directions to be parallel with each other and perpendicular to the base path, the result is equivalent to the previous method. By choosing the sweeping directions judiciously and taking advantage of the local surface geometry, it is possible to decrease the “uncalculated surface regions.”
5.1.2.2 Initial conditions in 3D space

The result of fitting a ply depends not only on the initial conditions in 2D space, but also on those in 3D space. As we mentioned earlier, the initial paths in 3D space with the previous method have to be calculated by hand as a sequence of equidistant points on the surface. This task is as hard as performing a mapping calculation of a mesh point between 2D and 3D spaces. In order to solve this problem, we will propose several different methods for specifying initial paths in 3D space which are able to automatically calculate the mapping of every mesh point on the base path in 3D space. The thrust of our solutions will be to define a curve on the surface that corresponds to the base path, instead of defining a sequence of points on the surface. The differences between the methods therefore stem from the differences in defining a curve on the surface. There are many ways to define such a curve. When defining the 3D curve as the base path it is important that it be easy to specify and that it should lead to flexible fitting results. In the following subsections, we list different methods for specifying the curve on the surface as the base path in 3D space and discuss their advantages and disadvantages.
• [Method 1] Defining the Base Path as an Isoparametric Curve

If the surface is parametrically defined, the simplest way of defining the base path in 3D is an isoparametric curve (i.e. \( u = \text{const.} \) or \( v = \text{const.} \)), lying on the given parametric surface. This curve is sometimes referred to as a coordinate curve [101]. The advantage of this method is its simplicity and the fact that it completely frees us from calculating explicit points on the surface. There are, however, serious disadvantages. The most serious problem occurs when the surface is made of composite patches where no global parametrization is possible. In this case, \( u = \text{const.} \) and \( v = \text{const.} \) curves are discontinuous at the boundary between adjacent patches. In other words, we cannot continue the mapping calculation beyond the patch where the mapping calculation is initiated. The second disadvantage is inflexibility, we cannot initially choose a path other than \( u = \text{const.} \) and \( v = \text{const.} \) curves.

• [Method 2] Defining the Base Path as a General Parametric Curve

A natural extension to the above method is to define a general parametric curve on the surface represented by \( u = u(t) \) and \( v = v(t) \), where \( t \) is an arbitrary parameter. At the boundary of two patches, the original curve has to be replaced by another parametric curve, and the continuity condition must be set appropriately. In general, given a composite surface made of several patches, we need to define a set of parametric curves. The advantage of this method is that it allows us to continue the mapping calculation beyond a patch even if the global parametrization of the two patches may not be possible. The disadvantage of this method is that characteristic points and the associated data must be specified in order to define the parametric curves. For example, if each curve is represented as a NURBS curve, its control points as well as its knot vectors and weights have to be specified. This may involve a large number of specifications and may be equivalent in the complexity to specifying a sequence of equidistant points.
• **[Method 3] Defining the Base Path as the Intersection between a Plane and the Given Surface**

Defining a parametric curve on the surface suffers the same problems as the previous method for specifying initial conditions. A simpler method for specifying a curve on the surface is to define the curve as the intersection between two surfaces; one is given by the surface to which a ply is fitted, and the other is a plane passing through the starting point for the fitting. We call this plane the **base plane**, for the base path must lie on the plane. All that is needed is to specify a base plane in 3D space is a reference point and a normal vector to the base plane. It is therefore very simple to specify, yet it is significantly more flexible than the approach based on isoparametric curves. The disadvantage of this method is that the curve is limited to a planar curve in 3D space. In spite of this limitation, this method is attractive because it provides a practically reasonable initial path from the standpoint of deformation energy accumulated. As will be described later, if the angle between a path and any thread intersecting the path is not constant, the fitting result will produce deformation energy along the path. Since a non-planar curve (except a geodesic) produces more distortion (in terms of geodesic curvature) along the path, using a planar path as the base path in 3D is a reasonable choice. In the following section, we will discuss a method for automatically obtaining a sequence of points on the intersection curve between a plane and the surface. Note that examples in a later section in this chapter are generated by the base path in 3D space with this method.

• **[Method 4] Defining the Base Path as the Intersection between a Ruled Surface and the Given Surface**

An extension to the above method is to define the initial path as the intersection curve between a ruled surface and the given surface. A ruled surface is a generalization of linear interpolation and can be expressed as either
\[ \mathbf{r}(u, v) = (1 - v)\mathbf{p}_1(u) + v\mathbf{p}_2(u) \] or \[ \mathbf{r}(u, v) = \mathbf{p}_0(u) + v\mathbf{p}_1(u), \]

where \( \mathbf{p}_0(u), \mathbf{p}_1(u), \) and \( \mathbf{p}_2(u) \) are space curves called directrices, and \( u, v \) are arbitrary parameters. Clearly, a plane is a special case of ruled surfaces. Similarly, we could replace a ruled surface by an arbitrary algebraic surface or by another NURBS surface. Usually these types of extensions increase input parameters, and eventually they will become almost equivalent, in terms of complexity, to defining a sequence of equidistant points on the surface as the previous method for specifying initial conditions. Thus, there is a tradeoff between the ease of specification and the flexibility in defining the base path on the surface.

- **[Method 5] Defining the Base Path as a Special Curve Passing Through the Starting Point**

Another method for specifying the base path is to define it as a special curve by considering local differential geometry around the starting point. We present two such special curves: (1) lines of curvature passing through the starting point and (2) geodesics passing through the starting point.

It is well-known that lines of curvature passing through a point are determined intrinsically and uniquely except at an umbilic point [27, 39, 101]. Moreover, for a non-umbilic point, two lines of curvature are always perpendicular to each other. We may reduce the distortion along the base path by taking advantage of this feature.

On the other hand, if we define the base path as a geodesic passing through the starting point for a fitting, we may obtain the fitting with the smallest amount of woven cloth materials because a geodesic path has the shortest length between two points on the surface [27, 101]. Furthermore, since no distortion results along the base path in terms of geodesic curvature, the deformation energy accumulated along the path can be minimized (See Chapter 7).

These are the advantages of utilizing a special curve (either a line of curvature or a geodesic). The disadvantage of this method is that we have to solve
non-linear differential equations to obtain a sequence of points on the paths [30, 43] for both curves. It is therefore relatively expensive to calculate the points on the base path and it is error prone if they are poorly implemented. In addition, both curves may become closed curves, especially if the surface is doubly curved such as an ellipsoid [39]. Nevertheless the method provides us with the basis for discussing an optimal configuration of fittings, and thus we incorporate the above idea into our algorithm for finding a set of good initial conditions, which will be described further in Chapter 7.

5.2 Mapping Calculations

In this section, we will describe the algorithm that maps 2D points onto a 3D surface. We will first describe our strategy for the mapping algorithm, then focus on the problem of mapping a sequence of auxiliary mesh points on the base path, and will finally focus on the problem of mapping regular mesh points not on the base path.

5.2.1 Strategy for Mapping Calculations

As will be described in the following two subsections, both the problem of mapping a sequence of auxiliary mesh points on the base path and the problem of mapping regular mesh points not on the base path are reduced to the surface-surface intersection problem (or the SSI problem) [5, 40, 43, 76]. The SSI problem is an active area of current research and will not be addressed in detail here. Instead, we briefly describe the four major methods for solving the SSI problem: (1) the algebraic method, (2) the lattice evaluation method, (3) the recursive subdivision method, and (4) the marching method. We will discuss which one is best suited to the mapping calculations for mesh points. The criteria for applicability of the methods to the mapping calculations include the following:

- The method should be able to generate the mapped points on the intersection curve easily.
The method should produce the mapped points accurately in the usual numerical sense.

- The method should produce the mapped points fast enough for allowing us to make an interactive CAD system for composite layout.

The *algebraic method* solves the SSI problem, by using the implicit representations of the two intersecting surfaces. For certain simple cases, such as plane/plane, the intersection curve is obtained explicitly. However, this method is not suitable for generating points on the curve. In addition, if the degree of the implicit representation of a surface is high, it can be very time-consuming to obtain the intersection curve [40].

The *lattice evaluation* method uses a discrete lattice. The surface-surface intersection at the lattice points are evaluated and an approximation of the intersection curve is found. This method is suitable for obtaining the cross sectional contours of an object, such as a human body, at different heights. However, the lattice evaluation method has the same problem as the algebraic method, and it is not suitable for generating points on the intersection curve. In addition, the accuracy of the result depends on the size of the lattice. If there are local “bumps” or “spikes” within the discrete lattice unit, for instance, it may not be easy to isolate the true solution.

The *recursive subdivision method* solves the SSI problem by subdividing the original problem into smaller problems, based on the “divide-and-conquer” paradigm [2]. The recursive subdivision method is effective, provided that the subdivided subproblems are simple enough. The simplest subproblem in the SSI problems is probably the plane/plane intersection. Therefore, the original SSI problem is eventually reduced to a collection of plane/plane intersection problems with this method. However, it may be very costly in terms of both space and time, if the given surfaces are heavily curved.

The *marching method* generates a sequence of points on the required intersection curve by stepping from point to point in a direction specified by the local
differential geometry. This method is most appropriate for our applications because it satisfies all of the above criteria. The marching method is, however, not a panacea. For example, suppose that the true intersection curve between the two surfaces has a loop as shown in Figure 5.6 (a). Then we may not be able to find the mapped points on the curve over the entire loop. The similar problem may occur when the true intersection curves consist of multiple disjointed parts. These things have to be taken into account especially when we consider the mapping calculations for a sequence of auxiliary mesh points on the base path defined as the intersection between the given (NURBS) surface and the base plane. Therefore, the base plane must be supplied such that it always transversely intersects with the NURBS surface as shown in Figure 5.6 (b).

In the following subsections, we will describe the mapping calculations in terms of the SSI problem and present the algorithms based on the marching method.

5.2.2 Mapping Calculations for Auxiliary Mesh Points on the Base Path

Here we will focus on the mapping calculations for auxiliary mesh points on the base path, assuming that the base path is defined as the intersection between a
given (NURBS) surface and the base plane. Another definition of the base path as a geodesic curve in 3D and the generation of points on the geodesic will be described in Section 7.2.3.

Although it might be desirable to derive a closed form solution for the intersection curve between a given NURBS surface and the base plane, it is not necessary. What is really needed is the sequence of points on the base path. The marching method is appropriate for this case. In order to apply the marching method to the mapping calculation of mesh points on the base path, we assume that the distance between adjacent (auxiliary) mesh points on the base path is kept fixed both in 2D and 3D spaces. With this assumption, the mapping calculation of mesh points on the base path is reduced to the intersection problem between the base plane, a sphere whose radius is the distance between two adjacent mesh points, and a given NURBS surface as shown in Figure 5.7. Thus, we have reduced the problem of automatically calculating the mappings of sampled points on the base path in 3D space to repeated applications of an intersection problem.

Let \((x_0, y_0, z_0)\) represent a mesh point already known on the intersection curve in 3D space and \(d_{xyz}\) represent the step distance in 3D space between the mesh point and the next auxiliary mesh point to be calculated. The problem of obtaining the next point, lying on the base plane whose normal is given by \((x_e, y_e, z_e)\), is expressed by the following system of non-linear simultaneous equations.

Plane:

\[
x_e(x - x_0) + y_e(y - y_0) + z_e(z - z_0) = 0, \tag{5.1}
\]

Sphere:

\[
(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = d_{xyz}^2, \tag{5.2}
\]

NURBS Surface (degree \(m \times n\)):

\[
\mathbf{r}(u, v) = \frac{\sum_{i=0}^{C_u-1} \sum_{j=0}^{C_v-1} w_{i,j} c_{i,j} N_i^m(u) N_j^n(v)}{\sum_{i=0}^{C_u-1} \sum_{j=0}^{C_v-1} w_{i,j} N_i^m(u) N_j^n(v)}. \tag{5.3}
\]

Substituting Equation (5.3) into Equations (5.1) and (5.2), we obtain:
Figure 5.7: Mapping calculation of an auxiliary mesh point on the base path
\[ F_1(u, v) = \begin{aligned} &x_e(x(u, v) - x_0) + y_e(y(u, v) - y_0) \\ &+ z_e(z(u, v) - z_0) = 0 \end{aligned} \]
\[ F_2(u, v) = \begin{aligned} &(x(u, v) - x_0)^2 + (y(u, v) - y_0)^2 \\ &+ (z(u, v) - z_0)^2 - d_{xyz}^2 = 0 \end{aligned} \] (5.4)

where \( \mathbf{r}(u, v) = (x(u, v), y(u, v), z(u, v)) \).

At this point, we have two simultaneous non-linear equations with two variables. This system of non-linear equations can be solved by the Newton-Raphson method [20], assuming the existence of first partial derivatives of the NURBS surface with respect to variables \( u \) and \( v \). Let \((u^{(k)}, v^{(k)})\) denote the pair of variables at the \( k \)-th iteration. The relationship between \((k+1)\)-th pair and the \( k \)-th pair is expressed as follows:

\[
\begin{bmatrix} u^{(k+1)} \\ v^{(k+1)} \end{bmatrix} = \begin{bmatrix} u^{(k)} \\ v^{(k)} \end{bmatrix} - \begin{bmatrix} \Delta u^{(k)} \\ \Delta v^{(k)} \end{bmatrix},
\] (5.5)

where

\[
\begin{bmatrix} \frac{\partial F_1}{\partial u} & \frac{\partial F_2}{\partial u} \\ \frac{\partial F_1}{\partial v} & \frac{\partial F_2}{\partial v} \end{bmatrix} \begin{bmatrix} \Delta u^{(k)} \\ \Delta v^{(k)} \end{bmatrix} = \begin{bmatrix} F_1(u^{(k)}, v^{(k)}) \\ F_2(u^{(k)}, v^{(k)}) \end{bmatrix}.
\] (5.6)

The elements of the Jacobian matrix in Equation (5.6) with respect to \( u \)-axis is given as:

\[
\frac{\partial F_1}{\partial u} = \begin{bmatrix} x_e \\ y_e \\ z_e \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial x_e}{\partial u}(u^{(k)}, v^{(k)}) \\ \frac{\partial y_e}{\partial u}(u^{(k)}, v^{(k)}) \\ \frac{\partial z_e}{\partial u}(u^{(k)}, v^{(k)}) \end{bmatrix},
\]
and

\[
\frac{\partial F_2}{\partial u} = 2 \begin{bmatrix} x(u^{(k)}, v^{(k)}) - x_0 \\ y(u^{(k)}, v^{(k)}) - y_0 \\ z(u^{(k)}, v^{(k)}) - z_0 \end{bmatrix} \cdot \begin{bmatrix} \frac{\partial x}{\partial u}(u^{(k)}, v^{(k)}) \\ \frac{\partial y}{\partial u}(u^{(k)}, v^{(k)}) \\ \frac{\partial z}{\partial u}(u^{(k)}, v^{(k)}) \end{bmatrix}.
\]

The Jacobian element with respect to \(v\)-axis is given in the same way. More details on partial derivatives of a NURBS surface are described in Appendix A.

The initial condition defines the first approximated point at a position distance \(d_{uv}\) in UV space from the starting point in the direction \(r_a = (u_a, v_a)\) as follows:

\[
\begin{bmatrix} u(0) \\ v(0) \end{bmatrix} = \begin{bmatrix} u_0 \\ v_0 \end{bmatrix} + d_{uv} \begin{bmatrix} u_a \\ v_a \end{bmatrix},
\]

(5.7)

where

\[
r_a = u_a \frac{\partial r}{\partial u} + v_a \frac{\partial r}{\partial v}.
\]

The distance \(d_{uv}\) is approximated by the following equation:

\[
d_{uv} = d_{xyz} R,
\]

(5.8)

where \(R\) is the ratio of the length of a unit vector \(\tilde{V}_{xyz}\) in XYZ space to the length of the corresponding vector \(V_{uv}\) in UV space. At the starting point on the base path, \(\tilde{V}_{xyz}\) is given as a parameter. Once a next mesh point is calculated, it is updated every time by the following formula:

\[
\tilde{V}_{xyz} = \frac{(P_{-1} - P_0)}{|P_{-1} - P_0|},
\]

where \(P_{-1} = (x_{-1}, y_{-1}, z_{-1})\) (previous point) and \(P_0 = (x_0, y_0, z_0)\) (current point). \(V_{uv}\) is obtained by the orthonormal projection of \(\tilde{V}_{xyz}\) onto the tangent plane at \(P_0\).

The terminating condition is given as follows:

\[
|F_1(u^{(k+1)}, v^{(k+1)})| + |F_2(u^{(k+1)}, v^{(k+1)})| < \varepsilon,
\]

(5.9)
where $\varepsilon$ is chosen as a suitably small scalar value.

### 5.2.3 Mapping Calculations for Regular Mesh Points NOT on the Base Path

Not surprisingly, the mapping calculation for (regular) mesh points not on the base path is similar to the intersection calculation given in the previous section. We have to deal with two cases: (1) when only one neighbor of the current mesh point is known, and (2) when two neighbors of the current mesh point are known. Note that a “neighbor” here refers to a mesh point adjacent to the mesh point of interest. Case (1) will be treated in the next section. Here we will focus on case (2).

Figure 5.8 illustrates how to perform the mapping calculation for case (2). Given that the positions of the mesh points $p(i-1, j)$ and $p(i, j-1)$ on the NURBS surface are known, it is straightforward to calculate the position of $p(i, j)$. Since it is assumed that the warp and weft threads are inextensible, the point $p(i, j)$ can be defined by the intersection of a sphere of radius $d^{wef}$ centered at $p(i-1, j)$, a sphere of radius $d^{warp}$ centered at $p(i, j-1)$, and the NURBS surface, which is represented by the following system of non-linear simultaneous equations:

**Sphere 1:**

$$(x - x_1)^2 + (y - y_1)^2 + (z - z_1)^2 = (d^{wef})^2,$$  \hspace{1cm} (5.10)

**Sphere 2:**

$$(x - x_2)^2 + (y - y_2)^2 + (z - z_2)^2 = (d^{warp})^2,$$  \hspace{1cm} (5.11)

**NURBS Surface (degree $m \times n$):**

$$r(u, v) = \frac{\sum_{i=0}^{Cu-1} \sum_{j=0}^{Cv-1} w_{i,j} c_{i,j} N^m_i(u) N^n_j(v)}{\sum_{i=0}^{Cu-1} \sum_{j=0}^{Cv-1} w_{i,j} N^m_i(u) N^n_j(v)},$$  \hspace{1cm} (5.12)
Since the subsequent calculations are basically similar to the mapping calculations for points on the base path, only the details pertaining to the initial and terminating conditions are described here.

In order to apply the Newton-Raphson method expressed by Equation (5.5), variables for the initial condition must be represented in UV space. Let \( p(i, j)_{uv} \) denote a mesh point represented in UV space at a crossing of \( i \)-th weft thread and \( j \)-th warp thread. We assume that \( p(i - 1, j)_{uv} \) is on the same weft thread with \( p(i, j)_{uv} \), and \( p(i, j - 1)_{uv} \) is on the same warp thread with \( p(i, j)_{uv} \). Let us define the \textit{weft gradient vector} \( \mathbf{v}^{\text{weft}}_{uv} \) at \( p(i - 1, j)_{uv} \) as

\[
\mathbf{v}^{\text{weft}}_{uv} = p(i - 1, j)_{uv} - p(i - 2, j)_{uv}.
\]

Likewise we define the \textit{warp gradient vector} \( \mathbf{v}^{\text{warp}}_{uv} \) at \( p(i, j - 1)_{uv} \) as

\[
\mathbf{v}^{\text{warp}}_{uv} = p(i, j - 1)_{uv} - p(i, j - 2)_{uv}.
\]
Let us then define two points \( p(i, j)_{uvw}^{weft} \) and \( p(i, j)_{uvw}^{warp} \) as follows:

\[
\begin{align*}
  p(i, j)_{uvw}^{weft} & \equiv \begin{bmatrix} u_{i,j}^{weft} \\ v_{i,j}^{weft} \end{bmatrix} = \begin{bmatrix} u_{i-1,j}^{weft} \\ v_{i-1,j}^{weft} \end{bmatrix} + (d_{uvw}^{weft})\tilde{v}_{uvw}^{weft} \\
  p(i, j)_{uvw}^{warp} & \equiv \begin{bmatrix} u_{i,j}^{warp} \\ v_{i,j}^{warp} \end{bmatrix} = \begin{bmatrix} u_{i,j-1}^{warp} \\ v_{i,j-1}^{warp} \end{bmatrix} + (d_{uvw}^{warp})\tilde{v}_{uvw}^{warp}
\end{align*}
\]  

(5.13)

where \( \tilde{v}_{uvw}^{weft} \) and \( \tilde{v}_{uvw}^{warp} \) are normalized weft and warp gradient vectors in UV space and distances \( d_{uvw}^{warp} \) and \( d_{uvw}^{weft} \) are obtained in a similar fashion as \( d_{uvw} \) in Equation (5.8). Then, we approximate the initial point for the Newton’s method with the following formula:

\[
\begin{bmatrix} u^{(0)} \\ v^{(0)} \end{bmatrix} \equiv \begin{bmatrix} u^{(0)}_{i,j} \\ v^{(0)}_{i,j} \end{bmatrix} = \frac{1}{2} \left( \begin{bmatrix} u_{i,j}^{weft} \\ v_{i,j}^{weft} \end{bmatrix} + \begin{bmatrix} u_{i,j}^{warp} \\ v_{i,j}^{warp} \end{bmatrix} \right).
\]  

(5.14)

This assumption works well in most cases where the UV parameterization has a nearly linear relationship with the XYZ parameterization in 3D Cartesian space.

The terminating conditions are expressed with the normal convergence condition given in Equation (5.9), together with the additional conditions given as follows:

\[
\begin{align*}
  \gamma^{(k+1)} & < \gamma_{lock}^{\min} \\
  \gamma^{(k+1)} & > \gamma_{lock}^{\max}
\end{align*}
\]  

(5.15)

where

\[
\begin{align*}
  \gamma^{(k+1)} & = \arccos \frac{(x_1-x^{(k+1)})(x_2-x^{(k+1)})}{|x_1-x^{(k+1)}||x_2-x^{(k+1)}|}, \\
  x_1 & = \mathbf{r}(u_{i-1,j}, v_{i-1,j}), \\
  x_2 & = \mathbf{r}(u_{i,j-1}, v_{i,j-1}), \\
  x^{(k+1)} & = \mathbf{r}(u^{(k+1)}, v^{(k+1)}).
\end{align*}
\]

These conditions (Equation (5.15)) correspond to the situation where a given 2D ply cannot be fit to a 3D surface because the thread angle between weft and
warp exceeds $\gamma_{\text{lock}}^{\min}$ or $\gamma_{\text{lock}}^{\max}$. Note that $\gamma_{\text{lock}}^{\min}$ and $\gamma_{\text{lock}}^{\max}$ are called minimum locking angle and maximum locking angle respectively, and are described in more details in Section 4.2.

5.3 Scanning Algorithms

The mapping algorithms mentioned in the previous section are applied to each mesh point in a given 2D ply. Scanning algorithms, on the other hand, are applied to a set of mesh points. The purpose of a scanning algorithm is to determine the order of the mapping calculations through the mesh points. This order has to conform to the given initial conditions including the base path and the sweeping directions. In general, the resulting mapping calculations depend on the scanning order of the mesh points. Before describing our scanning algorithms, we first mention the scanning based on the previous method for specifying initial conditions, where a pair of perpendicular threads called “constrained yarn paths” are given. We then define the scanning problem as a traversal of a given DAG (Directed Acyclic Graph), which allows us to evaluate the computational complexity for the (parallel) scanning of the mesh points expressed as nodes in a DAG. We finally focus on our scanning algorithm, together with a method for approximating the mapping calculation for an indegree one node in a given DAG.

5.3.1 The Case in Which Initial Paths are Defined by a Pair of Perpendicular Threads in 2D Space

Here we describe the scanning based on the previous method for specifying initial conditions, where a pair of perpendicular threads called “constrained yarn paths” are given. Suppose, for example, a pair of perpendicular initial paths labeled from ‘a’ to ‘g’ are given as shown in Figure 5.9. Consider the mapping calculations of the mesh points not on the initial paths labeled from ‘1’ to ‘9’ in Figure 5.9.

In order to calculate the mapping of a mesh point not on the initial paths, two neighboring mesh points must be already known, as we mentioned in the previous
Figure 5.9: An example of a scanning based on the initial paths defined by a pair of perpendicular threads.

section (See Figure 5.8). The only mesh point that satisfies this condition at the outset in this example is mesh point ‘1’. Once the mapping calculation of mesh point ‘1’ is done, we can proceed to either mesh point ‘2’ or ‘4’. In other words, in order to calculate the mapping of mesh points ‘2’ and ‘4’, mesh point ‘1’ must be known in advance. A similar argument is applied to the remaining mesh points.

Figure 5.10 shows this dependency relation in a DAG (Directed Acyclic Graph), which we call the dependency graph of mesh points in a given 2D ply. Nodes in the graph represent mesh points, and arcs represent the dependency relations among the mesh points. Two nodes linked by an arc are neighbors to each other. Every node has at most two incoming arcs and at most two outgoing arcs. The mapping calculation of a mesh point can be performed only after the mapping calculations of the two neighboring nodes, corresponding to the nodes with incoming arcs, are done.

For instance, node ‘5’ has two incoming arcs from nodes ‘4’ and ‘2’, and two outgoing arcs to nodes ‘8’ and ‘6’. The mapping calculations of nodes ‘4’ and ‘2’ must precede the mapping calculation of node ‘5’, which in turn must precede those of nodes ‘8’ and ‘6’. Since there is no loop in a DAG, the correct scanning order
can be obtained by a topological sorting in time $O(N + E)$ [23, 52], where $N$ is the number of nodes, and $E$ is the number of arcs in the DAG.

It should be noted that the solution to a valid scanning order is generally not unique, but the result of the mapping calculations is the same in this particular example. For instance, the following are two valid scanning orders, but produce the same fitting result:

$1 \rightarrow 4 \rightarrow 2 \rightarrow 7 \rightarrow 5 \rightarrow 3 \rightarrow 8 \rightarrow 6 \rightarrow 9$

and

$1 \rightarrow 4 \rightarrow 7 \rightarrow 2 \rightarrow 5 \rightarrow 8 \rightarrow 3 \rightarrow 6 \rightarrow 9$.

This phenomenon occurs when every node in the DAG has exactly two incoming arcs. In such a case, it is possible to parallelize the scanning algorithm. As shown in Figure 5.11, all the mesh points can be scanned in five steps with three processors in this example. Assume that $a(i, j)$ denotes a two dimensional array that represents given mesh points, where $1 \leq i \leq m$ and $1 \leq j \leq n$. Assume also that $a(1, j)$ and $a(i, 1)$ are given as a pair of perpendicular initial paths. Then with a CREW (Concurrent Read Exclusive Write) PRAM model [23], we can scan all the mesh points in $O(m + n)$ steps with $O(max(m, n))$ processors. This can be easily proved.
by considering that the depth of a DAG for the two dimensional array \(a(i, j)\) with
the above assumptions becomes \(O(m + n)\).

5.3.2 The Case in Which Initial Paths are Defined by the Base Path and Sweeping Directions

The above argument is not always applicable when the initial paths are given by the base path and sweeping directions as discussed in Section 5.1. This is because there may exist a node in a given DAG that has only one incoming arc.

Consider Figure 5.12, where regular mesh points are given by a \(4 \times 4\) network of nodes labeled from ‘1’ to ‘16’. Auxiliary mesh points on the base path are given by a sequences of nodes labeled from ‘a’ to ‘f’, and the sweeping directions are assumed to be perpendicular to the base path as shown with arrows in the figure.

In terms of topology, once auxiliary mesh points ‘a’ to ‘f’ are defined, neighboring relations are altered. For example, nodes ‘7’ and ‘8’ are originally neighbors to each other. Once the base path is defined, however, they are no longer neighbors. Instead nodes ‘7’ and ‘b’ are new neighbors, and so are nodes ‘8’ and ‘b’.

Regular mesh points are divided into two groups by the base path. Each group is scanned in the corresponding sweeping direction. Since the scanning of
mesh points in each group can be done independently, we will describe the group that includes node ‘1’.

Figure 5.13 shows the dependency graph corresponding to the example in Figure 5.12. Assuming that nodes labeled from ‘a’ to ‘f’ are already known, we first calculate the mapping of a group of nodes ‘4’, ‘7’, ‘10’, and ‘13’. The exact order among them is determined by the distance between the nodes and the base path measured along the sweeping direction. For instance, mapping calculation of node ‘3’ can be done once nodes ‘4’ and ‘7’ are available. In general, the mapping calculation of a particular node in the dependency graph can be done once all the nodes that have outgoing arcs to the node of interest are available.

It should be noted that there are nodes with only one incoming arc, such as nodes ‘4’ and ‘13’ in Figure 5.13. We hereafter call these nodes *indegree one nodes*, and other nodes *indegree two nodes*. It is this indegree one node that corresponds to case (1) in Section 5.2.3 on mapping calculations for regular mesh points. The
indegree one nodes require special consideration because the ordinary mapping algorithm that requires two known neighbors (case (2)) cannot be used to calculate them. This is the source of the “uncalculated surface regions” problem as shown in Figure 5.4. In the following section, we first describe a mapping calculation for indegree one nodes, and then describe how to scan both auxiliary and regular mesh points.

5.3.3 An Approximation to the Mapping Calculation for an Indegree One Node

Indegree one nodes require a special mapping algorithm. It is impossible to uniquely determine the mapping of an indegree one node with only one known adjacent mesh point. Additional information must be supplied in order to perform the mapping from 2D to 3D.

We have developed a heuristic which assumes that the local geometry is preserved around the indegree one node to be calculated. Specifically, we assume that the angular relationship between a thread and the base path at a mesh point in 2D space is preserved on the tangent plane at the mapped point in 3D space, as illustrated in Figure 5.14.

For example, suppose we are looking at the mapping of node ‘4’, where the
4

a
tangent plane

θ
3D

thread

base path

2D

d

(a)

4

θ
tangent plane

at node ‘a’

3D

(b)

Figure 5.14: An approximation of the mapping calculation for node ‘4’ with only one known neighbor (‘a’).

The angle in the 2D space between the base path and the thread passing through nodes ‘4’ and ‘a’ is $\theta$ as in Figure 5.14 (a). Then, as shown in Figure 5.14 (b), this angular relationship is preserved on the tangent plane at the mapped point of ‘a’ in 3D.

Thus, the problem of mapping node ‘4’ is reduced to the intersection problem between a plane (shown with dotted lines passing through nodes ‘4’ and ‘a’), a sphere whose radius is $d$, and a NURBS surface. All the mapping calculations beyond node ‘4’ from the base path are dependent upon this approximation, as shown in the dependency graph of Figure 5.13. This is why the entire mapping calculation is generally affected by the scanning order.

In order to obtain the plane equation which passes through nodes ‘4’ and ‘a’, we first calculate the direction vector from ‘a’ to ‘4’ at the mapped point of ‘a’ in 3D space, by rotating the direction vector of the base plane by $\theta$ around the surface normal at ‘a’. The plane equation is then determined by this direction vector and the surface normal as expressed in the following formula:
\[
\begin{vmatrix}
x & y & z & 1 \\
x_0 & y_0 & z_0 & 1 \\
n_x & n_y & n_z & 0 \\
d_x & d_y & d_z & 0 \\
\end{vmatrix}
= 0,
\]

(5.16)

where \((n_x, n_y, n_z)\) denotes the surface normal at ‘a’, \((d_x, d_y, d_z)\) denotes the direction vector from ‘a’ to ‘4’, and \((x_0, y_0, z_0)\) denotes the mapped point of ‘a’. Equation (5.16) represents a general plane equation used to approximate the mapping of a mesh point with only one known neighbor.

### 5.3.3.1 Scanning Auxiliary Mesh Points on the Base Path

Scanning auxiliary mesh points on the base path, at first view, seems to be done by simply repeating the mapping calculations of mesh points on the initial paths. However, it is not that straightforward partly because the distance between adjacent mesh points on the base path is not necessarily constant, and partly because the base path normally includes crossings both on weft and on warp threads. Clearly an algorithm that first scans intersections between the base path and wefts, and then scans intersections between the base path and warps does not work. The output of a scanning algorithm must be a sequence of mesh points in the order they appear along the base path.

Algorithms used to draw lines on a raster graphics display may be adapted to solve the problem of scanning mesh points on the base path. In other words, both problems are analogous since their output must be a sequence of points in a particular direction in 2D space. The following algorithm scans the mesh points on the base path in the order they appear, based on the DDA (Digital Differential Analyzer) algorithm [86] used in generating line drawings on a 2D raster graphic display.
Algorithm 5.1 (SCANNING AUXILIARY MESH POINTS ON THE BASE PATH)

Procedure SCAN-AUXILIARY-MESH-POINTS($d_x$, $d_y$, $x_0$, $y_0$);

\[ \Delta_x = \frac{1}{|d_x|}; \quad \Delta_y = \frac{1}{|d_y|}; \quad i = i_1 = \lfloor x_0 \rfloor; \quad j = j_1 = \lfloor y_0 \rfloor; \]

if $|i_1 - x_0| < \varepsilon$ then $i_1 = i_1 - 1$; endif
if $|j_1 - y_0| < \varepsilon$ then $j_1 = j_1 - 1$; endif

$i_2 = i_1 + 1; \quad j_2 = j_1 + 1$;

if $d_x > 0$ then $\Delta_i = 1; \quad \text{Length}_x = \frac{i_2 - x_0}{d_x}$;
else $\Delta_i = -1; \quad \text{Length}_x = \frac{i_1 - x_0}{d_x}$;
endif

if $d_y > 0$ then $\Delta_j = 1; \quad \text{Length}_y = \frac{j_2 - y_0}{d_y}$;
else $\Delta_j = -1; \quad \text{Length}_y = \frac{j_1 - y_0}{d_y}$;
endif

while $i \geq i_{\text{min}}$ and $i \leq i_{\text{max}}$ and $j \geq j_{\text{min}}$ and $j \leq j_{\text{max}}$

\[ \text{if} \ |\text{Length}_x - \text{Length}_y| < \varepsilon \text{ then} \]

/* coincident on a lattice */
\[ i = i + \Delta_i; \quad j = j + \Delta_j; \]
\[ \text{Length}_x = \text{Length}_x + \Delta_x; \]
\[ \text{Length}_y = \text{Length}_y + \Delta_y; \]
\[ x = i; \quad y = j; \]

else if $\text{Length}_x < \text{Length}_y$ then

/* intersecting with a warp */
\[ i = i + \Delta_i; \quad \text{Length}_x = \text{Length}_x + \Delta_x; \]
\[ x = i; \quad y = y_0 + \frac{d_y}{d_x} \times (x - x_0); \]

else

/* intersecting with a weft */
\[ j = j + \Delta_j; \quad \text{Length}_y = \text{Length}_y + \Delta_y; \]
\[ x = x_0 + \frac{d_x}{d_y} \times (y - y_0); \quad y = j; \]

endif

\[ \text{MakeGrid}(x, y); \]

endwhile

Note that $(d_x, d_y)$ denotes the gradient vector of the base path with respect to the horizontal line in 2D space, assuming that each element is not equal to zero, and $(x_0, y_0)$ denotes a starting point. MakeGrid$(x, y)$ in the above algorithm defines a
new mesh point located at \((x, y)\), modifies the topology of the adjacent two mesh points (either WEST and EAST neighbors or SOUTH and NORTH neighbors), and performs a mapping calculation. In the case of the base path given in Figure 5.12, this algorithm defines mesh points ‘d’, ‘c’, ‘b’, and ‘a’ in this order and performs the mapping calculations, assuming mesh point ‘d’ is a starting point. Remaining mesh points ‘e’ and ‘f’ are defined after switching the direction \((dx, dy)\) to \((-dx, -dy)\). Note that original mesh points labeled from ‘1’ to ‘16’ are defined in advance. The efficiency of the above algorithm can be significantly improved by utilizing integer arithmetic [18].

5.3.3.2 Scanning Regular Grid Points NOT on the Base Path

Once auxiliary mesh points on the base path have been mapped to 3D, the scanning order of the remaining regular mesh points must be determined. During the scanning process, indegree one nodes in the dependency graph and the surface boundary may be encountered.

Consider a detailed example shown in Figure 5.15. Suppose regular mesh points are labeled from ‘1’ to ‘49’, and auxiliary mesh points on the base path are labeled from ‘a’ to ‘j’. Node ‘f’ represents the starting point which is not on any thread. Suppose also that the dotted lines indicate the surface boundary that we will encounter during the mapping calculation. Note that the location of the surface boundary is determined when the mapping calculations reach the boundary at run time. The scanning algorithm must identify the surface boundary at run time, keeping the scanning order consistent with the given initial conditions.

Once nodes ‘a’ to ‘j’ (except ‘f’) are calculated, the scanning algorithm proceeds as follows. Let us first define the frontier of the scanning as a group of mesh points that include candidates for the next mapping calculation. Initially, it consists of mesh points adjacent to those on the base path. For example, in Figure 5.15 we have two independent frontiers: \{'7’, ‘13’, ‘20’, ‘26’, ‘33’, ‘39’, ‘46’\} and \{'14’, ‘21’, ‘27’, ‘34’, ‘40’, ‘47’\}. For the sake of simplicity, we focus only on the first frontier.
The frontier is sorted at creation time in the order specified by the sweeping direction. For example, if we assume the sweeping direction is perpendicular to the base path, the first element in the frontier is the nearest mesh point to the base path. If there are two or more mesh points that have the same distance to the base path, the mesh point nearest to a given starting (reference) point is chosen. The frontier is therefore regarded as an ordered list whose elements are mesh points. In addition, we maintain the number of known neighbors for each list element. In this particular example, the result of the sorting is \{<‘33’,2>, <‘20’,2>, <‘46’,1>, <‘7’,1>, <‘39’,1>, <‘13’,1>, <‘26’,1>\}, where the second number appearing in each list element represents the number of known neighbors at sorting time. For instance, node ‘33’ has two known neighboring nodes ‘g’ and ‘h’.

We then proceed to the mapping calculation for the first element in the frontier list. The first element is extracted from the list, and its number of known neighbors is examined. If it is equal to two, the ordinary mapping calculation based on the intersection between a NURBS surface and two spheres is performed. If it is equal to one, the mapping based on the intersection between a NURBS surface, a plane
and a sphere is performed. In either case, if the mapping calculation is successfully completed, the extracted element is replaced by its unknown neighbors. Specifically, if the unknown neighbor is already in the frontier list, its number of known neighbors is incremented; otherwise the new elements are sorted and inserted into the frontier list and their number of known neighbors is set to one. If the mapping calculation terminates because the mesh point is outside the NURBS surface, the element is simply removed from the list. This process is repeated until the frontier list is empty. In order to find the surface boundary at run time, the in-out determination of the mapped point is done during the mapping calculation by checking if \((u^{(k+1)}, v^{(k+1)})\) defined in Equation (5.5) is inside or outside the NURBS patch of interest. Appendix C lists the algorithm for scanning regular mesh points, and the associated data structure for a mesh point and the frontier list.

The complexity of the algorithm is evaluated as follows. Suppose we have \(N = n^2\) mesh points. Since the base path has \(O(n)\) mesh points, the initial frontier also has \(O(n)\) elements. The sorting of this list takes \(O(n \log n)\) time. Because the number of remaining unprocessed elements are \(O(N) = O(n^2)\), we need \(O(n \times N) = O(n^3)\) to do the insertion sorting for these elements in the worst case. We can reduce the time considerably either by using a bucket sort [23] instead of an insertion sort, by maintaining a segment tree and using a tree insertion [77], or by pre-sorting the mesh points.

### 5.4 Boundary Adjustments for a Composite NURBS Surface

Provided that a solution to the mapping calculation exists, and if it is within the original surface region which includes the starting point, we can proceed to the next mapping calculation repeatedly in a specified scanning direction, until either the surface region is totally fitted or the given 2D ply is exhausted. Care must be taken, however, when the mapping calculation reaches the boundary of the current surface region, especially in the case where the surface is defined by a composite NURBS surface.
In this section, we focus on this problem. We first describe the requirements of boundary adjustments by using an example. We then present the data structure necessary for quick adjustments at the boundary. Finally, we describe the algorithms for boundary adjustments for the mapping calculations of mesh points. For auxiliary mesh points, the boundary adjustment algorithm is applied to a thread segment that straddles two adjacent patches. For regular mesh points, on the other hand, the boundary adjustment algorithm is applied to two thread segments that straddle two or more adjacent patches.

5.4.1 Requirements of Boundary Adjustments

Figure 5.16 illustrates an example of the base path that passes though multiple patches. Since the mapping calculations of both auxiliary and regular mesh points are done in UV parametric space as we formulated in the Newton’s method in Section 5.2, we have to calculate UV variables at each mesh point. The UV variables include the local UV coordinate value \((u, v)\) at the mesh point and the direction vectors \((d_u, d_v)\) of the threads (wefts and/or warps) passing though the mesh point.

As long as the mapping calculation is done within a particular surface patch, the calculation of a set of new UV variables is straightforward, given the known neighboring UV variables. When the mapping calculation reaches the boundary of a surface patch, however, calculating a set of new UV variables becomes a problem. At the boundary, we have to re-specify a set of UV variables in the UV space of the new surface patch. As long as the global parametrization of the UV variables is possible, this re-specification may not be a significant issue. However, If we are faced with a situation where the global UV parametrization is impossible, as in the example of an object with a convex rounded corner illustrated in Figure 5.17, we must replace the UV variables represented in UV space of the old NURBS patch by the UV variables represented in UV space of the new NURBS patch adjacent to the old one.

Specifically, for the initial guess in the mapping calculations for the auxiliary
Figure 5.16: An example of the base path on a composite surface.

Figure 5.17: An example of a composite NURBS surface made of seven NURBS patches.
mesh points given by Equation (5.7), we have to re-specify \( \mathbf{r}_a = (u_a, v_a) \) vector at the boundary. For the initial guess in the mapping calculations of regular mesh points given by Equation (5.14), we need to re-specify \( p(i, j)_{uv|wef} \) and \( p(i, j)_{uv|warp} \) which are defined in Equation (5.13).

5.4.2 A Data Structure for Quick Boundary Adjustments

Here we consider a data structure suitable for boundary adjustments in the mapping calculations. Suppose every NURBS patch is defined independently without any topological links between the NURBS patches. Then, at the boundary, we have to search in an exhaustive way for the adjacent patch that may include the next mesh point. For \( O(m) \) numbers of mesh points with \( O(n) \) numbers of patches, this may take \( O(mn) \) time, which is too time-consuming for practical purposes if \( m \) and \( n \) are large numbers.

A simple method for avoiding such an exhaustive search is to maintain the topological data among patches, and to exchange the UV variables by using the topological data. There have been substantial previous efforts for keeping the topological data among adjacent geometrical objects. The winged-edge data structure [11], the DCEL (Doubly Connected Edge List) [77], and the Radial Edge Structure [112, 113, 114] are three of such topological data structures, but they have been used primarily to describe the topological data for polyhedral objects.

In our context, we mainly deal with the case where every boundary curve is expressed by one of the following: (1) \( u = u_{min} \), (2) \( u = u_{max} \), (3) \( v = v_{min} \), and (4) \( v = v_{max} \). In addition, we assume that each surface patch is simply connected and there are no holes inside. Considering these, we have developed a topological data structure for a composite NURBS surface as described below. We assume that a composite NURBS surface is made of a set of NURBS patches, and each NURBS patch as its data structure includes four adjacent pointers that correspond to WEST \( (u = u_{min}) \), EAST \( (u = u_{max}) \), SOUTH \( (v = v_{min}) \), and NORTH \( (v = v_{max}) \) boundaries, respectively. Figure 5.18 illustrates the pointers that correspond to the
example shown in Figure 5.17. Although the data structure generally expects four adjacent patches, it can also handle a degenerate case where two of the boundaries coincide. For example, patch 7 in Figure 5.17 has a degenerate point that corresponds to $u = u_{\text{min}}$ boundary. In Figure 5.18, it is expressed by the NULL pointer for the WEST pointer of patch 7.

We can now classify the type of the boundary according to the combination of the four directions. For each boundary of a surface patch, we have up to four different types of boundaries of the adjacent patch, and in total, sixteen combinations are possible. For instance, in Figure 5.18, patch 7 has an EAST-WEST boundary with respect to patch 5, a SOUTH-EAST boundary with respect to patch 4, and a NORTH-WEST boundary with respect to patch 6. In the following, we describe the problems of boundary adjustments during the mapping calculations of mesh points,
5.4.3 Boundary Adjustment for the Mapping Calculation of Auxiliary Mesh Points

When we deal with a composite NURBS surface, UV space variables must be adjusted and replaced at the boundary between two patches. These variables are defined only locally on each patch, and they are used for the mapping calculations of both auxiliary and regular mesh points. Here, we focus on the boundary adjustment for the mapping calculations of auxiliary mesh points.

Once we set up the four pointers for each patch, the task of replacing the UV variables at the boundary between two patches becomes straightforward. Let $PatchCode(SurfacePatch, u, v)$ denote a procedure that returns a code which represents the location of the point $(u, v)$ in UV space relative to the current NURBS patch, pointed to by $SurfacePatch$. Specifically, code value 0 means that the point $(u, v)$ is inside the patch pointed to by $SurfacePatch$, and non-zero code values imply that the point $(u, v)$ is outside the patch (See Figure 5.19).

**Figure 5.19:** The code returned by $PatchCode(SurfacePatch, u, v)$.
This function is similar to the so-called “area code” in the 2D clipping algorithm [63, 86] to clip line segments with a clipping window for computer graphics display. The list of PatchCode(SurfacePatch, u, v) and the associated data structures are shown in Appendix B.

If we assume that the radius of the curvature of each boundary curve surrounding a patch is sufficiently large compared to the distance between adjacent wefts or warps, we can linearly interpolate the line segment which extends from a mesh point \((u_0, v_0)\) to the adjacent mesh point \((u, v)\), and which intersects with a boundary of the patch between the two end points. Note that we assume that \((u_0, v_0)\) is located in the original patch, and \((u, v)\) is located out of the original patch. The boundary adjustment algorithm for the line segment between \((u_0, v_0)\) and \((u, v)\) is given as follows:

**Algorithm 5.2 (BOUNDARY ADJUSTMENT OF A LINE SEGMENT BETWEEN TWO ADJACENT MESH POINTS)**

1. If \(\text{PatchCode}(\text{SurfacePatch}, u, v) \neq 0\) then do the following; otherwise return.

2. Calculate the intersection between the line segment and the boundary by linear interpolation. (Let \((u_{0\text{old}}, v_{0\text{old}})\) denote this point expressed in the “old” UV space parametrization.)

3. Represent the intersection in terms of the adjacent UV space parametrization. (Let \((u_{0\text{new}}, v_{0\text{new}})\) denote this point expressed in the “new” UV space parametrization.)

4. Represent the direction vector \((u_a, v_a)\) in terms of the adjacent UV space parametrization. (Let \((u'_a, v'_a)\) denote the new vector.)

5. Modify the step distance \(d\) by linear interpolation. (Let \(d'\) denote the adjusted step distance in UV space.)
Figure 5.20: Boundary adjustment at the WEST boundary of the “old” patch, which corresponds to the EAST boundary of the “new” patch.

6. Represent the end point \((u, v)\) in terms of the adjacent UV space parametrization. (Let \((u', v')\) denote this point.)

Figure 5.20 illustrates the boundary adjustment when a line segment, extending from \((u_0, v_0)\) to \((u, v)\), straddles the WEST boundary of the “old” patch. Suppose \(u_{\text{min}}, u_{\text{max}}, v_{\text{min}},\) and \(v_{\text{max}}\) denote WEST, EAST, SOUTH, and NORTH boundaries of the “old” patch. Suppose also \(u'_{\text{min}}, u'_{\text{max}}, v'_{\text{min}},\) and \(v'_{\text{max}}\) denote WEST, EAST, SOUTH, and NORTH boundaries of the “new” patch. Then, we can refine the above algorithm by taking the boundary adjustment of the WEST boundary, for instance, as follows:

In the following algorithm, we assume that \(\mathbf{x}_0 = (x_0, y_0, z_0)\), \(\mathbf{x}_{\text{old}}^0 = (x_{\text{old}}^0, y_{\text{old}}^0, z_{\text{old}}^0)\), and \(d_{xyz}\) is a step distance in \(\mathbb{E}^3\) space.
/* PatchCode(SurfacePatch, u, v) = WEST */

newpatch = oldpatch → W;

\( u_{old}^0 = u_{\text{min}}; \)

\( v_{old}^0 = v_0 + \frac{u_{\text{min}}-u_0}{u-u_0}(v-v_0); \)

if oldpatch = newpatch → E then

\( u_{new}^0 = u'_{\text{max}}; \)

\( v_{new}^0 = v'_{\text{min}} + \frac{v_{old}^0-v_{\text{min}}}{v_{\text{max}}-v_{\text{min}}}(v'_{\text{max}} - v'_{\text{min}}); \)

\( u'_a = u_a; v'_a = v_a \)

else if oldpatch = newpatch → W then

\( u_{new}^0 = u'_{\text{min}}; \)

\( v_{new}^0 = v'_{\text{min}} + \frac{v_{max}^0-v_{old}^0}{v_{\text{max}}-v_{\text{min}}}(v'_{\text{max}} - v'_{\text{min}}); \)

\( u'_a = -u_a; v'_a = -v_a \)

else if oldpatch = newpatch → S then

\( u_{new}^0 = u'_{\text{min}} + \frac{v_{old}^0-v_{\text{min}}}{v_{\text{max}}-v_{\text{min}}}(u'_{\text{max}} - u'_{\text{min}}); \)

\( v_{new}^0 = v'_{\text{min}}; \)

\( u'_a = v_a; v'_a = -u_a \)

else if oldpatch = newpatch → N then

\( u_{new}^0 = u'_{\text{min}} + \frac{v_{max}^0-v_{old}^0}{v_{\text{max}}-v_{\text{min}}}(u'_{\text{max}} - u'_{\text{min}}); \)

\( v_{new}^0 = v'_{\text{max}}; \)

\( u'_a = -v_a; v'_a = u_a \)
endif

d' = d \times \frac{\|x_0-x_{old}\|}{a_{xyz}};

\begin{pmatrix}
  u' \\
  v'
\end{pmatrix}
= \begin{pmatrix}
  u_{new}^0 \\
  v_{new}^0
\end{pmatrix} + d' \times \begin{pmatrix}
  u'_a \\
  v'_a
\end{pmatrix};

It should be noted that the boundary adjustment of the UV space variables
from one patch to another is done simply by following the WEST pointer in constant
time, and no exhaustive search is needed.

5.4.4 Boundary Adjustment for the Mapping Calculation of Regular
Mesh Points

Here we will describe boundary adjustments for the mapping calculation of
regular mesh points. Recall that the initial guess for the mapping calculation must
be represented in the UV space parametrization expressed by Equation (5.14). This
initial guess works well, provided that the two previous points \( p(i-1, j) \) and \( p(i, j-1) \)
are in the same patch. If, however, one or two of them are in a different patch,
we cannot simply employ the equation. This is because the UV space variables are
local to each patch. In the following, we assume without loss of generality that
\( p(i-1, j) \) and \( p(i, j-1) \) are in different patches, and that neither of them are on the
base path. We also assume that \( p(i-1, j) \) is on the same weft thread with \( p(i, j) \),
and \( p(i, j-1) \) is on the same warp thread with \( p(i, j) \). The mapped point \( p(i, j) \) is
either in the same patch with \( p(i-1, j) \), in the same patch with \( p(i, j-1) \), or in
a patch different from both. We hereafter refer to the first two cases as Case (A),
and the last case as Case (B). Figure 5.21 demonstrates the two cases; Specifically,
the mapped point \( p(i, j) \) is in the same patch 2 with \( p(i-1, j) \), but \( p(i, j-1) \) is
in patch 3. This corresponds to Case (A). On the other hand, the mapped point
\( p(m, n) \) is in patch 4, but \( p(m-1, n) \) is in patch 3, and \( p(m, n-1) \) is in patch 5.
This corresponds to Case (B). Generally speaking, the mapping calculation for Case
(B) is more difficult than the mapping calculation for Case (A), because it involves
more patches.

We have developed three different methods for obtaining the point \( p(i, j) \) when
the above assumptions hold:
Figure 5.21: An example of a fitting to a composite surface made of several patches, where boundary adjustments are applied to the mapping calculations for some of the regular mesh points such as \( p(i, j) \) and \( p(m, n) \). Note that dotted lines represent patch boundaries.
• **Method 1**

1. Approximate the intersections between the boundary and the line segments \( p(i-1,j)p(i,j) \) and/or \( p(i,j-1)p(i,j) \) when they straddle between the two adjacent patches. Let them denote \( p(i-1,j)' \) and \( p(i,j-1)' \).

2. Apply the mapping calculations for regular mesh points mentioned in Section 5.2.3, by regarding \( p(i-1,j)' \) and \( p(i,j-1)' \) as the known neighbors. Note that the distances between adjacent crossings along weft threads \( (d^{weft}) \) and along warp threads \( (d^{warp}) \) in Equation (5.13) must be modified similarly.

• **Method 2**

1. Identify all the NURBS patches that may be involved in the mapping calculations. Elevate the degrees of the NURBS patches, such that all the patches identified above have the same degree.

2. Merge them into a global NURBS patch temporarily and perform the mapping calculation to obtain \( p(i,j)' \), where \( p(i,j)' \) represents the mapped point in the UV space of the temporarily generated patch.

3. Re-map the point \( p(i,j)' \) to the point \( p(i,j) \), represented by the corresponding UV space of an originally defined patch.

• **Method 3**

1. Calculate the surface normal vectors \( n(i-1,j) \) and \( n(i,j-1) \) at the points \( p(i-1,j) \) and \( p(i,j-1) \), respectively. Let \( (x_0,y_0,z_0) = (p(i-1,j)_{xyz} + p(i,j-1)_{xyz})/2 \) and \( (n_x,n_y,n_z) = (n(i-1,j) + n(i,j-1))/2 \).

2. Calculate the intersection between the two spheres whose centers are at \( p(i-1,j) \) and \( p(i,j-1) \), and the plane \( n_x(x-x_0)+n_y(y-y_0)+n_z(z-z_0) = 0 \). Let \( (x_i,y_i,z_i) \) denote this intersection.
3. Calculate the intersection between the same two spheres as above and the ray \((x - x_i)/n_x = (y - y_i)/n_y = (z - z_i)/n_z\). The intersection finally gives \(p(i, j)\).

The first method is the simplest of the three, and it is the most efficient. Unfortunately it produces only a rough approximation to the \(p(i, j)\). In addition, it often occurs that no solution to the mapping calculation exists, because after modifying the distances \(d^{weft}\) and \(d^{warp}\), they may be very small compared to their original distances. This method can be applied to both Cases (A) and (B).

The essence of the second method is the temporary merge of the NURBS patches that may contribute to the mapping calculation into a global patch. By merging the NURBS patches, the points \(p(i - 1, j), p(i, j - 1),\) and \(p(i, j)\) are guaranteed to be in the same (temporary) patch. As long as the number of NURBS patches to be merged is two (in this case, the two adjacent NURBS patches share a boundary curve), it is guaranteed that a global patch can be found. However, it is not always guaranteed that we can merge three or more NURBS patches into a global patch. For instance, if a singular point such as the pole of a sphere is shared by the four NURBS patches to be merged, there is no way to merge them into a single global patch. Therefore, the second method is applied to only Case (A). Note that the degree elevation may be necessary before merging two adjacent NURBS patches, because the degrees of the two adjacent NURBS patches must be equal in order to merge them into a temporary global NURBS patch.

The third method does not require the merge of the NURBS patches. It relies on the local differential geometry around the given two points \(p(i - 1, j)\) and \(p(i, j - 1)\). It first approximates the point \(p(i, j)\) by the intersection between the two spheres (whose centers are \(p(i - 1, j)\) and \(p(i, j - 1)\) and whose radii are \(d^{weft}\) and \(d^{warp}\)) and a plane. The plane is defined by the mid point between \(p(i - 1, j)\) and \(p(i, j - 1)\), and the normal vector given by the average of the surface normals at \(p(i - 1, j)\) and \(p(i, j - 1)\). This approximation is then used as the initial guess of another intersection calculation between the same two spheres and a ray, which
is defined by a point (the intersection in the previous step) and a direction vector (the normal to the plane in the previous step). This method works well as long as enough smoothness is assured around the points \( p(i - 1, j) \) and \( p(i, j - 1) \). It is relatively slow compared to the above two methods because it involves the numerical intersection calculations two times. This method can be applied to both Cases (A) and (B), since it does not depend on the UV space parametrization.

The second and the third methods are incorporated in our CAD system. Specifically, when the calculation of \( p(i, j) \) involves only two adjacent NURBS patches, the second method is applied. When the calculation of \( p(i, j) \) involves three or more adjacent NURBS patches, the third method is applied. This choice is based on the fact that the accuracy of the second and the third methods is better than the first method, the efficiency of the second method is better than the third method when only two adjacent NURBS patches are involved, and the robust nature of the third method when three or more NURBS patches are involved.

### 5.5 Plane Development

In this section, we will consider the problem of predicting “flattened” patterns that are used for fitting onto a specific portion of the surface. Because we know which mesh points have contributed to the fitting, it is rather straightforward to obtain the corresponding flattened patterns. No flattening actually needs to occur. Before focusing on our method for obtaining the flattened patterns, however, we describe previous approaches to the flattening, in order to grasp the idea of how the flattening of a 3D surface has been carried out so far in the related fields in which the surface is modeled as a deformable sheet.

#### 5.5.1 Previous Approaches to the Flattening

It is well-known that only developable surfaces such as a plane, a cylinder, and a cone, whose Gaussian curvature is everywhere zero, can be exactly developed onto a flat plane [27, 31, 39, 100, 101]. A sphere, for instance, cannot be exactly
developed onto a plane. This is why numerous methods for projecting a terrestrial
globe onto a plane have been devised [13, 101].

In computer-aided garment design and shoe making, it is crucial to obtain the
flattened patterns of the material draped on a human body and a foot. Wachi [109]
presented a method for cutting cloth into pieces of strips, such that each strip is
developable. A garment in 3D space is formed by gluing the pieces of strips together.
The disadvantage of this method is that too many narrow strips may be generated.
Therefore, it may not be feasible in terms of manufacturability.

In her investigation of garment fitted patterns, Heisey [36, 37] presented a
generic model represented by a set of quadrilaterals for a garment, and developed a
method for flattening the quadrilaterals onto a planar surface. Her basic assumption
on the flattening method is that the length of all four sides of a quadrilateral and the
proportional relationship at one angle are preserved when the garment is deformed
and fitted onto a human body. The biggest problem in her approach is that when the
surface is not developable, two adjacent quadrilaterals might either overlap or have
gaps between them. She cope with this problem by translating and/or rotating
the neighboring quadrilateral to the base quadrilateral until both the horizontal and
vertical sides are aligned with the neighboring quadrilateral.

Bennis et al. [12] described a method for flattening non-developable surfaces
by cutting the surface into strips and mapping each strip to a surface such that the
geodesic curvature and the arc length of the curve specified in the strip are preserved.
They applied the algorithm to non-distorted texture mapping and presented exam-
ple including a development of a sphere and a shoe shape. Their method suffers
from the same problem as in Wachi’s and Heisey’s methods in that it may produce
many narrow strips.

Shimada et al. [97] described the development of a curved surface based on the
finite element method [115]. They subdivided the surface into triangular elements,
derived the total strain energy by summing up the contribution of the strain en-
ergy for each triangle, and attempted to find the deformation with minimum strain
energy. Because their method may cause a large deformation of the material when seeking the deformation with minimum strain energy, it is doubtful that their linear approximations between strains and displacements would produce valid solutions.

Imaoka et al. [50] employed a similar energy-based technique to get a paper pattern for a garment. They also formulated the strain energy based on the finite element method. They plotted the distribution of strains on the flattened paper patterns, and attempted to reduce the strains by inserting one or more “darts” into highly strained portions, based on the strain distribution map.

5.5.2 A Method for Obtaining Flattened Patterns

The differences between our model of woven cloth composites and the models for garments and shoes mentioned above are as follows. First, in the models for garments and shoes, a wide variety of deformations including extension and shrinkage of the material must be considered, whereas in our model, these is no extension or shrinkage of any thread, and only shear deformation caused by changing the thread angles between wefts and warps should be taken into account. Second, in the models for garments and shoes, the deformed material need not completely fit to the given surface of an object, whereas in our model, the deformed material must fit to the given surface.

Considering these differences, our 2D plane development algorithm should be simpler and more efficient. Indeed, as will be mentioned below, it is very simple and efficient, primarily because we can utilize the result of the mapping calculations, instead of initiating another numeric computation. Moreover, as one of the features of our algorithm, the output is not a set of narrow strips, but a connected area in the given ply of woven cloth composites. It is desirable in terms of manufacturability, for ease in cutting out the necessary portion of the cloth.

The basic idea behind our algorithm is as follows. We first categorize the mesh points in the woven cloth ply as either “mapped” or “inactive” mesh points. Initially all the mesh points are classified into “inactive” mesh points.
When a mesh point is mapped to a 3D point inside the area of interest on the 3D surface, the mesh point is marked as “mapped.” The only remaining work to be done to produce a plane development is to define the boundary of the region containing the “mapped” mesh points. Traversing the plane development boundary is similar to traversing the boundary of bit map images represented by chain codes [88]. The only difference is that we have to deal with “fractional fragments” (See Figure 5.23) between adjacent crossings if the boundary curve crosses the thread between them.

The following algorithm calculates the boundary of the “mapped” mesh points in a woven cloth composite ply. Mesh points that contribute to the boundary are called boundary mesh points, and one of them is designated the starting point of the traversal. Recall that we implicitly assume the data structure for a mesh point (called “Grid”) as listed in Appendix C.

Algorithm 5.2 (BOUNDARY TRAVERSAL OF “MAPPED” MESH POINTS)

1. Set a pointer to a mesh point represented by $G$ to the starting mesh point of the fitting.

2. Trace the NORTH pointer of $G$ repeatedly until we reach the boundary mesh point on the warp thread. Let $G_0$ denote a pointer to the boundary mesh point.

3. Find the first neighboring boundary mesh point met from the current boundary mesh point in a clockwise direction. Note that a neighboring boundary mesh point should be in one of the following directions: WEST, EAST, SOUTH, NORTH, SOUTH-WEST, SOUTH-EAST, NORTH-WEST, and NORTH-EAST.

4. Repeat the Step 3 until we come back to the boundary mesh point pointed by $G_0$.

Some remarks on the above algorithm: First, in Step 2, we may go toward WEST, EAST, or SOUTH, instead of NORTH. The important thing is to find a
Figure 5.22: The result of the boundary traversal applied to the example of Figure 5.15.

A particular boundary mesh point served as a sentinel for the boundary traversal. Second, the fact that the neighboring boundary mesh points can always be found in one of eight directions as specified in Step 3 is a straightforward consequence that the fitted region is a connected region in the surface. Figure 5.22 shows the result of the boundary traversal applied to the example of Figure 5.15.

Figure 5.23 shows an example of traversing the boundary. The boundary mesh points are denoted by black circles, and one of them is designated the starting point of the traversal. We traverse the boundary in clockwise order, keeping track of the fractional fragments that are denoted by bold line segments in the figure. Figure 5.24 compares two results: (a) a fitting result and the flattened pattern without appending the fractional fragments, and (b) a fitting result and the flattened pattern with the fractional fragments.

We have developed a set of rules for appending the fractional fragments, which depends on the relationship between two adjacent boundary mesh points, as summarized in Appendix E.
Figure 5.23: An example of plane development. Black circles denote boundary mesh points, white circles denote “mapped” mesh points, and bold edges denote the fractional fragments of boundary mesh points. The boundary is obtained as the dotted line.

5.6 Formability

Our investigation of the conformability of a given 2D ply of woven cloth composites onto a 3D surface has been focused on the geometric aspects. In other words, the given ply is regarded as conforming to a given 3D surface, as long as the thread angles between wefts and warps lie between allowable minimum ($\gamma_{\text{lock}}^{\text{min}}$) and the maximum ($\gamma_{\text{lock}}^{\text{max}}$) limits.

It is important to consider how easily the ply is deformed onto the surface in terms of the work done, or the energy necessary to force the ply to fit into the final shape. We call this measure formability (or moldability) of a given ply of woven cloth composites. In garment design, a similar term called tailorability has been used for referring to the ease with which a fabric can be converted into the intended end product [9]. Most studies of tailorability in garment design have used objective measurements of the properties of woven cloth fabrics including fabric shear stiffness, fabric bending stiffness, compressibility in the fabric plane, thickness and weight, tensile extensibility, and buckling stability [9, 24, 58, 59].

Here we neglect that the potential energy due to the gravity of the ply as well
Figure 5.24: Fractional fragments appended to the boundary mesh points: (a) no fragments added to both the fitting and the plane development, and (b) fragments added to both the fitting and the plane development.
as the energy necessary to translate and rotate the ply without changing the shapes of the threads. Since we have assumed the inextensibility of the threads in the ply, the allowable deformation in 3D space is limited to shearing and bending. In the following, we describe each deformation energy in turn.

5.6.1 Shear Deformation Energy

Assuming elasticity in each thread of the ply and also assuming in-plane shear deformation [105], the shear deformation (strain) energy, $V_S(i, j)$, at a crossing $(i, j)$, is given by $V_S(i, j) = \frac{1}{2} \sigma(i, j) \epsilon(i, j)$, where $\sigma(i, j)$ is shear stress and $\epsilon(i, j)$ is shear strain. Since $\epsilon(i, j) = \frac{1}{G} \sigma(i, j) = r_x(i, j) \cdot r_y(i, j) = |r_x(i, j)||r_y(i, j)| \cos \gamma(i, j) = \cos \gamma(i, j)$, where $r_x$ and $r_y$ are Tchebychev net coordinate vector fields along a weft and a warp, respectively (See Section 7.1), the strain energy is reduced to:

$$V_S(i, j) = \frac{1}{2} G \epsilon(i, j)^2 = \frac{1}{2} G \cos^2 \gamma(i, j), \quad (5.17)$$

where $G$ is the shear modulus. Generally speaking, a greater deviation from the starting value ($\pi/2$) implies a greater accumulation of shear deformation energy.

5.6.2 Bending Deformation Energy

In order to discuss the bending deformation of a given ply of woven cloth composites, we have made the following assumptions:

- Each thread has constant width and thickness, and is regarded as an elastic beam.
- Only small strains and displacements occur by bending.
- Torsional rigidity is neglected.

From these assumptions, we can define the bending moment $M_x$ along a weft and $M_y$ along a warp at a crossing between a weft and a warp, just as in a finite-deformation
shell theory [21, 105] as follows:

\[
\begin{bmatrix}
M_x \\
M_y
\end{bmatrix} = \begin{bmatrix}
C_x & \nu_y C_x \\
\nu_x C_y & C_y
\end{bmatrix} \begin{bmatrix}
\kappa_x \\
\kappa_y
\end{bmatrix},
\]  

(5.18)

where \( \nu_x \) and \( \nu_y \) are Poisson’s ratios along a weft and a warp, \( C_x \) and \( C_y \) are the flextural rigidities along a weft and a warp, and \( \kappa_x \) and \( \kappa_y \) are normal curvatures at the crossing along a weft and a warp, respectively. If the deformation is within the range of linear elasticity and \( C = C_x = C_y \), then \( C \) is given by

\[C = \alpha EI,\]

where \( \alpha \) is a positive constant, \( E \) is Young’s modulus, and \( I \) is the moment of inertia of the normal section of the thread.

With the above assumptions, the bending deformation energy \( V_B(i, j) \) is expressed by the following formula [48, 49, 104]:

\[V_B(i, j) = \frac{1}{2} C [(\kappa_x(i, j) + \kappa_y(i, j))^2 - 2(1 - \nu)(\kappa_x(i, j)\kappa_y(i, j))], \]

(5.19)

where \( \nu = \nu_x = \nu_y \) is assumed. Note that \( \kappa_x \) and \( \kappa_y \) are further put into the following expressions:

\[
\begin{align*}
\kappa_x &= r_{xx} \cdot n \\
\kappa_y &= r_{yy} \cdot n
\end{align*}
\]

(5.20)

where \( n \) is the surface normal and \( r_{xx} \) and \( r_{yy} \) are the acceleration vectors of a weft and a warp, respectively.

Table 5.1 gives values for Young’s modulus \( E \) (\( E_{tensile} \) for a tensile direction and \( E_{trans} \) for a tranverse direction), shear modulus \( G \), and Poisson’s ratio \( \nu \), for several fiber-reinforced composite materials. It should be noted that the shear modulus, \( G \), is usually one or more orders of magnitude smaller than the modulus of elasticity in tension \( E_{tensile} \), and this is why we can regard the extension of threads as negligible compared to the shear deformation in most cases.
Table 5.1: Elastic constants for some fiber-reinforced composite materials with 60 % unidirectional fibers by volume. Based on Table 3.4 in [28]. The unit for \( E \) and \( G \) is \( GPa \) (giga pascal), where \( 1 \ Pa = 1 \ Nm^{-2} = 10 \ dyn \ cm^{-2} \).

<table>
<thead>
<tr>
<th>Material</th>
<th>( E_{\text{tensile}} )</th>
<th>( E_{\text{trans}} )</th>
<th>( G )</th>
<th>( \nu )</th>
</tr>
</thead>
<tbody>
<tr>
<td>E-glass</td>
<td>45</td>
<td>12</td>
<td>4.4</td>
<td>0.25</td>
</tr>
<tr>
<td>Kevlar49</td>
<td>76</td>
<td>5.5</td>
<td>2.1</td>
<td>0.34</td>
</tr>
<tr>
<td>Graphite (T-300)</td>
<td>132</td>
<td>10.3</td>
<td>6.5</td>
<td>0.25</td>
</tr>
<tr>
<td>Graphite (GY-70)</td>
<td>320</td>
<td>5.5</td>
<td>4.1</td>
<td>0.25</td>
</tr>
</tbody>
</table>

5.7 Examples of Fittings

In this section, we present several fitting examples. Throughout these examples, we emphasize how the initial conditions are specified, what the resulting 2D plane development looks like, and how two or more fittings with different initial conditions compare.

5.7.1 An Octant of a Sphere

Here we consider a fitting to an octant of a sphere whose radius is \( R \). We assume that the geometry is defined in the first octant \( (X \geq 0, Y \geq 0, Z \geq 0) \). A sphere is a typical surface that can be represented exactly by a NURBS surface. (See Figure 7.23 for more details of the control points and the associated weights.) In this example, we present the basic idea of how initial conditions are specified for the fitting, and how they affect the resulting configurations.

We have carried out simulation experiments with three different initial conditions, which are summarized in Table 5.2. Specifically, they have a common base path along the circle on the \( Y = 0 \) plane in 3D space, and the mesh size of the ply is assumed to be \( 40 \times 40 \). In 2D space, cases (a) and (b) have a base path parallel to the weft threads, and case (c) has a base path at a 45 degree angle to both the wefts.
and the warps. Since the base path coincides with the boundary of the octant of a sphere, only one sweeping direction is specified. Each sweeping direction is defined by a pair of vectors both in 2D and 3D spaces. The starting points for cases (a) and (c) are located at the end points of the base paths, and the starting point for case (b) is located at the midpoint of the base path.

The resulting fitting configurations and the associated 2D plane developments produced from the mappings are shown in Figure 5.25. By convention, we denote the starting point by \( P_s \), the sweeping directions by directed arrows incident from the starting point, and the base path by a sequence of solid circles. Note that the bold line segments in the 2D plane developments represent the area mapped to the surface.

Among the three cases, case (a) produces the same results as the previous method mentioned in Section 5.1.1 \([84, 85, 14, 107]\) when a thread line is aligned with the base path and the sweeping direction is perpendicular to the base path.

In order to compare the results, we calculated the following values, which are listed in Table 5.3:

1. the area of the 2D ply necessary for the mapping,
2. the maximum and minimum thread angles between vertical and horizontal threads,
3. the shear deformation energy, and
4. the bending deformation energy.

The area of the 2D ply necessary for the mapping is approximated by summing up the number of mesh points in the 2D plane development pattern. The maximum \((\gamma_{\text{max}})\) and minimum \((\gamma_{\text{min}})\) thread angles are simply calculated as \(\gamma_{\text{max}} = \max(\gamma_{i,j})\) and \(\gamma_{\text{min}} = \min(\gamma_{i,j})\), where \(\gamma_{i,j} = \arccos(r_i \cdot r_j)\), and \(r_i\) is the unit weft vector and \(r_j\) is the unit warp vector at mesh point \((i, j)\). The shear deformation energy is approximated by the value \(\sum_{i,j} \cos^2 \gamma_{i,j}\) as mentioned in Section 5.6.
Table 5.2: Initial conditions for the three fitting in Figure 5.25.

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>starting point</td>
<td>(0,0,(\frac{R}{2}))</td>
<td>(0,(\frac{R}{2}),0)</td>
<td>(0,0,(R))</td>
</tr>
<tr>
<td>sweeping direction</td>
<td>(0,1,0)</td>
<td>(0,1,0)</td>
<td>(0,1,0)</td>
</tr>
<tr>
<td>base path</td>
<td>(Y = 0)</td>
<td>(Y = 0)</td>
<td>(Y = 0)</td>
</tr>
</tbody>
</table>

|        |             |             |             |
| 2D     |             |             |             |
| starting point | (0,0) | (20,0) | (15,0) |
| sweeping direction | (0,1) | (0,1) | (-1,1) |
| base path | \(Y = 0\) | \(X = 20\) | \(Y = X - 15\) |

The bending deformation energy is approximated by the value \(\sum_{i,j} B(i,j)\), where \(B(i,j) = (\kappa_x(i,j) + \kappa_y(i,j))^2 - 1.5\kappa_x\kappa_y\). This is obtained from Equation (5.18) by setting \(v = 0.25\) (Poisson’s ratio for E-glass and graphites in Table 5.1.) and removing the coefficients. Normal curvature \(\kappa_x\), which was defined by Equation (5.20), can be computed by the following formula:

\[
\kappa_x(i,j) = \frac{L\lambda_x(i,j)^2 + 2M\lambda_x(i,j) + N}{E\lambda_x(i,j)^2 + 2F\lambda_x(i,j) + G},
\]

where \(\lambda_x(i,j) = \frac{du(i,j)}{dv(i,j)}\), \(\mu_x(i,j) = \frac{dv(i,j)}{du(i,j)}\), and \(E = r_u \cdot r_u\), \(F = r_u \cdot r_v\), \(G = r_v \cdot r_v\), \(L = r_{uu} \cdot n\), \(M = r_{uv} \cdot n\), and \(N = r_{vv} \cdot n\). Note that \(du(i,j)\) and \(dv(i,j)\) are obtained from the unit weft vector \(\tilde{v}_{\text{weft}}\) (See Section 5.2.3). Normal curvature \(\kappa_y\) is similarly obtained.

Given two or more fittings and the associated values as defined above, we may evaluate the quality of the fittings. The best fitting is the one that consumes the smallest ply area in 2D space, has the least variation of the thread angles (i.e. minimum \(\gamma_{\text{max}}\) and maximum \(\gamma_{\text{min}}\)), and has the minimum deformation energy. It should be noted, however, a “best” fitting may not be unique.

In our simulation experiments, the fitting that consumes the smallest ply area is case (b). Case (b) also has the least variation of the thread angles, and has the minimum deformation energy. We can therefore conclude that case (b) is the most
Figure 5.25: Fittings to an octant of a sphere and their plane developments.
Table 5.3: Comparison of results with different initial conditions.

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td>area</td>
<td>716</td>
<td>657</td>
<td>725</td>
</tr>
<tr>
<td>min(arccos $\gamma_{i,j}$)</td>
<td>0.707</td>
<td>1.167</td>
<td>0.353</td>
</tr>
<tr>
<td>max(arccos $\gamma_{i,j}$)</td>
<td>1.570</td>
<td>1.570</td>
<td>1.570</td>
</tr>
<tr>
<td>$\sum_{i,j}(\cos \gamma_{i,j})^2$</td>
<td>82.5</td>
<td>17.4</td>
<td>128.2</td>
</tr>
<tr>
<td>$100 \times B(i,j)$</td>
<td>0.119</td>
<td>0.091</td>
<td>0.099</td>
</tr>
</tbody>
</table>

desirable fitting among the three cases.

Figure 5.26 shows the results of the three different fittings to an octant of a sphere with an actual sheet of woven cloth fiberglass, corresponding to the fittings in Figure 5.25. It is remarkable that the theoretical deformation patterns that we have predicted match quite nicely with the real deformation patterns.

5.7.2 A Hemisphere

A hemisphere is a popular example that has been used for the fitting experiments by many researchers [10, 61, 62, 84, 106, 107]. Surprisingly, no previous efforts have been made to simulate the fittings by changing the initial conditions. In fact, all of the previous experiments (both simulation and physical experiments) have assumed the initial conditions in the same way as case (a) in Figure 5.25. The result of the fitting simulation with shading is shown in Figure 5.27 (a). If we assume the initial conditions as case (c) in Figure 5.25, the fitting result shown in Figure 5.27 (b) is obtained. Note that a fitting simulation with the initial conditions corresponding to case (b) in Figure 5.25 has not been done, because they produce an asymmetric fitting result for a hemispherical data.

Figure 5.28 shows the same fitting results, with colors assigned based on the thread angle values ($\gamma = T$ in Figure 5.28) as attached to the color bar. Refer to Appendix F for the color assignment method in detail. In this example, it is clear
Figure 5.26: Fitting results with an actual sheet of woven cloth fiberglass, corresponding to the theoretical fittings shown in Figure 5.25.
Figure 5.27: 3D fitting results with the starting point at the north pole; (a) the wefts take 90 degrees to the base path, and (b) both wefts and warps take 45 degrees to the base path.

that case (b) in Figure 5.28 produces the fitting result with greater shear deformation. Thus we can intuitively predict that it is more susceptible to anomalous events such as wrinkles and folds. Figure 5.29 shows the 2D plane development patterns with colors assigned in the same way as in Figure 5.28. It demonstrates where the shear deformation is most likely to occur on the 2D “flattened” patterns.

5.7.3 One Third of a Surface of Revolution

Here we present several fittings onto one third of a surface of revolution. Figure 5.30 presents three different views of the shape before the fitting: (a) an oblique view, (b) a side view, and (c) a top view. The purpose of this example is to compare the results of different fittings with the aid of the plots of the thread angle values.

Figure 5.31 presents the results of six different fittings. The first three cases
Figure 5.28: 3D fitting results with colors assigned, based on the thread angle values \(T\); (a) the wefts take 90 degrees to the base path, and (b) both wefts and warps take 45 degrees to the base path.
Figure 5.29: 2D plane development patterns corresponding to the fitting results of Figure 5.28; (a) the wefts take 90 degrees to the base path, and (b) both wefts and warps take 45 degrees to the base path.
Figure 5.30: Three views of a quadrant of a surface of revolution.
Table 5.4: Initial conditions for the fitting in Figure 5.31.

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3D</strong></td>
<td>starting point</td>
<td>(10,0,0)</td>
<td>(10,0,0)</td>
</tr>
<tr>
<td></td>
<td>sweeping directions</td>
<td>(0,1,0)</td>
<td>(0,1,0)</td>
</tr>
<tr>
<td></td>
<td>base path</td>
<td>Y = 0</td>
<td>Y = 0</td>
</tr>
<tr>
<td><strong>2D</strong></td>
<td>starting point</td>
<td>(0,0)</td>
<td>(0,30)</td>
</tr>
<tr>
<td></td>
<td>sweeping directions</td>
<td>(1,0)</td>
<td>(1,-1)</td>
</tr>
<tr>
<td></td>
<td>base path</td>
<td>X = 0</td>
<td>Y = X + 30</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>3D</strong></td>
<td>starting point</td>
<td>(10,10\sqrt{3},0)</td>
<td>((\frac{5\sqrt{3}}{2},\frac{5\sqrt{3}}{2},\frac{15}{2}))</td>
</tr>
<tr>
<td></td>
<td>sweeping directions</td>
<td>(1,-1,0)</td>
<td>(1,-1,0)</td>
</tr>
<tr>
<td></td>
<td>(-1,1,0)</td>
<td>(-1,1,0)</td>
<td>(-1,1,0)</td>
</tr>
<tr>
<td></td>
<td>base path</td>
<td>X = Y</td>
<td>X = Y</td>
</tr>
<tr>
<td><strong>2D</strong></td>
<td>starting point</td>
<td>(35,0)</td>
<td>(35,35)</td>
</tr>
<tr>
<td></td>
<td>sweeping directions</td>
<td>(-1,0)</td>
<td>(-1,1)</td>
</tr>
<tr>
<td></td>
<td>(1,0)</td>
<td>(1,-1)</td>
<td>(1,0)</td>
</tr>
<tr>
<td></td>
<td>base path</td>
<td>X = 35</td>
<td>Y = X</td>
</tr>
</tbody>
</table>

((a), (b), and (c)) have a common base path along the boundary curve on the Y = 0 plane, and the last three cases ((d), (e), and (f)) have a common base path along the curve on the X = Y plane. In 2D space, cases (a), (c), (d), and (f) have the base path parallel to the warps, and cases (b) and (e) have the base path at a 45 degree angle to the warps. The definition of the sweeping directions and the starting points is similar to the previous example, and the initial conditions are summarized in Table 5.4. We assume that the mesh size of the 2D ply in this example is 70 × 70. Figure 5.32 includes the 2D plane developments corresponding to the fittings of cases (a) to (f) in Figure 5.31.

Figures 5.31 and 5.32 clearly demonstrate the critical effect of initial conditions
Figure 5.31: Fittings to a third part of a surface of revolution with six different initial conditions.
Figure 5.32: Plane development patterns used for the fitting to a third of a surface of revolution.
Table 5.5: Comparison of results with different initial conditions.

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
<th>(c)</th>
<th>(d)</th>
<th>(e)</th>
<th>(f)</th>
</tr>
</thead>
<tbody>
<tr>
<td>area</td>
<td>1159</td>
<td>1246</td>
<td>1474</td>
<td>1023</td>
<td>1017</td>
<td>1010</td>
</tr>
<tr>
<td>min((\arccos \gamma_{i,j}))</td>
<td>0.719</td>
<td>1.266</td>
<td>1.347</td>
<td>1.157</td>
<td>1.252</td>
<td>1.347</td>
</tr>
<tr>
<td>max((\arccos \gamma_{i,j}))</td>
<td>2.832</td>
<td>2.764</td>
<td>3.135</td>
<td>2.182</td>
<td>2.317</td>
<td>2.429</td>
</tr>
<tr>
<td>(\sum_{i,j}(\cos \gamma_{i,j})^2)</td>
<td>211.5</td>
<td>385.9</td>
<td>588.6</td>
<td>36.7</td>
<td>50.5</td>
<td>40.9</td>
</tr>
<tr>
<td>(100 \times B(i,j))</td>
<td>0.052</td>
<td>0.068</td>
<td>0.042</td>
<td>0.055</td>
<td>0.024</td>
<td>0.048</td>
</tr>
</tbody>
</table>

on the final mappings. Table 5.5 summarizes the results of the six different cases. Case (f) requires the smallest ply area. Case (d) has the minimum \(\gamma_{max}\), and case (f) has the maximum \(\gamma_{min}\). Case (d) has the least shear deformation energy, and case (e) has the least bending deformation energy. We cannot conclude which is the best fitting, but we can clearly state that cases (d), (e), and (f) produce better results than cases (a), (b), and (c). In the six different results, case (c) is exceptional since it fails to cover the entire surface region with the given 2D ply.

Figure 5.33 shows the distributions of thread angles between wefts and warps at mesh points projected above the associated plane developments, in which the vertical axis (or height field) represents the value of the thread angle, ranging from 0 to \(\pi\).

The plot of thread angle values provides insight into the differences between two or more fittings onto a common surface. Generally speaking, a good fitting has little bend in the surface of the plot. It is clearly observed in Figure 5.33 that cases (d), (e), and (f) have less bend in the surface of the thread angle plot than cases (a), (b), and (c).
Figure 5.33: Distribution of thread angles between wefts and warps at mesh points projected above the associated plane developments.
5.7.4 A Bump in a Panel

The final example in this chapter is the shape shown in Figure 5.34. Two different initial conditions are given and the 3D fitting results are shown in Figure 5.35 (a) and (b). Both fittings have the same base path as depicted in bold lines. However, the orientations of the woven cloth with respect to the base path are different. Specifically, the first fitting result (Figure 5.35 (a)) is obtained by aligning weft threads perpendicular to the base path. The second fitting result (Figure 5.35 (b)) is obtained by aligning weft and warp threads with 45 degrees to the base path. Figure 5.35 (c) and (d) are the 2D plane developments corresponding to (a) and (b), respectively. Table 5.6 shows the result of the two fittings. No major difference is observed in the areas necessary to produce the two fittings. However, the minimum thread angle ($\gamma_{\text{min}}$) in the second fitting is much smaller than the first fitting, and therefore we can conclude that the second fitting is more susceptible to anomalous events.

Figure 5.34: A surface with a bump.
Figure 5.35: Two fittings to a surface with a bump; (a), (b) are the fitting results, and (c), (d) are the 2D plane development patterns corresponding to (a) and (b), respectively.

Table 5.6: Comparison of the two fitting results.

<table>
<thead>
<tr>
<th></th>
<th>(a)</th>
<th>(b)</th>
</tr>
</thead>
<tbody>
<tr>
<td>area</td>
<td>4534</td>
<td>4588</td>
</tr>
<tr>
<td>min((\arccos \gamma_{i,j}))</td>
<td>0.222</td>
<td>0.014</td>
</tr>
<tr>
<td>max((\arccos \gamma_{i,j}))</td>
<td>2.157</td>
<td>1.817</td>
</tr>
<tr>
<td>(\sum_{i,j}(\cos \gamma_{i,j})^2)</td>
<td>196.0</td>
<td>237.4</td>
</tr>
<tr>
<td>(\bar{B}(i,j))</td>
<td>12.22</td>
<td>10.92</td>
</tr>
</tbody>
</table>
CHAPTER 6
ANOMALOUS EVENTS AND THEIR PREVENTION BY INSERTING DARTS

There is a growing need to predict and prevent anomalous events, or manufacturing faults, during the lamination process of woven cloth composites. In this chapter, we focus primarily on what anomalous events typically occur in practice and how to prevent some of these anomalous events by inserting “darts.” Other methods for predicting and preventing anomalous events will be described in the next chapter.

6.1 Anomalous Events

During the lamination process of a woven cloth composite ply, there may occur many types of anomalous events, or manufacturing faults, as listed in Table 6.1. Superficial faults, dimension errors, and distortion are readily found by inspection. Extensive nondestructive inspection (NDI) measurements are performed to find internal flaws. The main techniques used to detect voids and delaminations are C-scan ultrasonics in a water bath, or a water jet, and radiography to detect foreign objects [44].

In this dissertation, we focus on wrinkles and voids that may occur during the fitting process of a given ply of woven cloth composites. The first set of events occurs due exceeding the deformable limits of threads (i.e. locking angle) during deformation. In other words, as long as the thread angle stays between a certain minimum and maximum, the ply behaves acceptably. Once the thread angle goes below the minimum locking angle \( \gamma_{\text{lock min}} \) or exceeds the maximum locking angle \( \gamma_{\text{lock max}} \), the ply may begin to behave in an unacceptable manner. For example, if the angle goes below the allowable minimum \( \gamma_{\text{lock min}} \), wrinkling occurs and eventually the ply may be torn apart. Similar events may happen when the angle exceeds the allowable maximum \( \gamma_{\text{lock max}} \). In terms of the mapping calculations, these events may
Table 6.1: Typical manufacturing faults in fiber composite laminates. Based on Table 6.2 in [44].

<table>
<thead>
<tr>
<th>Wrinkled layers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wavy fibers</td>
</tr>
<tr>
<td>Voids</td>
</tr>
<tr>
<td>Delaminations</td>
</tr>
<tr>
<td>Disbonds from metallic inserts</td>
</tr>
<tr>
<td>Fiber gaps</td>
</tr>
<tr>
<td>Incorrect fiber orientation</td>
</tr>
<tr>
<td>Incorrect ply sequence</td>
</tr>
<tr>
<td>Resin starved areas</td>
</tr>
<tr>
<td>Resin rich areas</td>
</tr>
<tr>
<td>Incomplete resin cure</td>
</tr>
<tr>
<td>Foreign body inclusion</td>
</tr>
<tr>
<td>Poor release from mold</td>
</tr>
<tr>
<td>Tolerance errors</td>
</tr>
</tbody>
</table>

occur even when a solution exists. Figure 6.1 (a) demonstrates the situation in which the thread angle goes below the allowable minimum, and Figure 6.1 (b) demonstrates the situation in which the thread angle exceeds the allowable maximum. Figure 6.2 illustrates a wrinkle.

Alternatively, there are situations in which a solution to the mapping calculations does not exist. This happens when the ply cannot fit to a given surface because of the local high convexity or concavity of the surface between two adjacent mesh points. If there is local high convexity, the resulting state is called “breaking” (Figure 6.3 (a)), and if there is local high concavity, it is called “bridging” (Figure 6.3 (b)).

In summary, we consider three anomalous events in the sections below:

Wrinkling

The state of ply not being in contact with the surface, either because of the too small thread angle ($\gamma < \gamma_{\text{lock}}^{\text{min}}$) or because of the too large thread angle ($\gamma > \gamma_{\text{lock}}^{\text{max}}$).
Figure 6.1: Anomalous events in terms of the thread angle at a mesh point between a weft and a warp: (a) the thread angle goes below the allowable minimum ($\gamma_{\text{lock min}}$) and (b) the thread angle exceeds the allowable maximum ($\gamma_{\text{lock max}}$).

Figure 6.2: An illustration of a wrinkle
Bridging

The state of ply not being in contact with the surface, because of the disability to deform enough to drape into a local concavity in the surface

Breaking (Tearing)

The state of ply being torn apart, (1) because of the excess of the thread angle maximum ($\gamma > \gamma_{lock}^{max}$) or minimum ($\gamma < \gamma_{lock}^{min}$), or (2) because of the disability to deform enough to drape on a local convexity in the surface

Obviously, the anomalous events stated above are undesirable and should be avoided if possible. In general, the difficulty of predicting and preventing anomalous events increases in proportion to the complexity of the given 3D surface shape to be fitted, the complexity of the material properties of the given 2D ply of woven cloth composites, and the complexity of the initial conditions for the fitting. However, if we limit our attention to the anomalous events caused by excessive shear deformation, the prediction of anomalous events becomes relatively straightforward since
only the thread angle limitations require consideration.

Here we propose several different approaches to preventing anomalous events. The first approach is to repeatedly attempt the process of fitting with different initial conditions and different ply materials until no anomalous events occur. Unfortunately, there is no guarantee that a solution can be found where no anomalous events occur. The second approach is to trim and split the ply into pieces whenever anomalous events may occur. For the wrinkling case, this splitting corresponds to “darting” the excessive portion; For the breaking and bridging cases, it corresponds to “cutting” the cloth and “patching” the necessary portion. The third approach is to automatically find a good initial path specification, given the input surface shapes and a ply material. This may be done by considering local surface differential geometry properties such as the distribution of Gaussian curvatures. The final approach is to split the surface region into small enough regions and fit a ply to each region separately. This method guarantees a solution without anomalous events (See Bieberbach’s theorem in the next chapter). However, this approach should be used only as a final resort because it is not practical to subdivide the surface area into many smaller regions and fit a separate piece of plies to each subdivided region. In this chapter, we will discuss the second approach, more specifically on dart insertion. The third approach will be covered in the next chapter.

6.2 Darting

Darting is one approach employed in practice for preventing anomalous events caused by excessive shear deformation. Specifically, during the fitting and the 2D pattern making process, designers press sheets onto the mold and sketch out the approximate shapes of the 2D patterns. The 2D plies are then deformed and “darted” in many cases in order to force them to fit onto the 3D surface. The insertion of “darts” is crucial to avoid anomalous events such as gaps and wrinkles, particularly when the ply is fitted to a highly curved surface. Technologies and tools that assist in automatically inserting darts during the design stage currently do not exist. In
this section, we deal with the problems related to dart insertion and present our solutions to them.

Quoted from a dictionary \(^3\), a dart is a *stitched tapering fold used especially in fitting garments to the curves of the body*. In the field of garment design, inserting darts has been a popular technique used for fitting a piece of cloth naturally around protruding surface regions [36, 37, 50, 64, 109]. The need for introducing darts in garment design has been based mainly on improving the fit of the garment, by reducing strains around highly deformed areas [50]. An important difference between this type of darting and the darting considered here is that it is not essential that the resulting garment patterns completely fit onto a surface.

In the following, we first define models for darts. We then describe dart insertion algorithms that are applied to the region where excessive shear deformation is predicted. As one of the important issues in dart insertion, we focus on overlap removal algorithms that allow us to automatically determine the shape of a dart to be inserted. We close this chapter with several examples of fittings a 2D ply of woven cloth composites with darts to a 3D surface mold.

### 6.2.1 Models for Darts

Darts are inserted in a given ply of woven cloth composites in order to avoid anomalous events such as gaps and wrinkles. We shall define two different models for darts. They are *stitched darts* and *trimmed darts*.

With stitched darts, excessive material is removed and the edges are stitched to produce a darting edge as shown in Figure 6.4. With trimmed darts, excessive material is simply removed without stitching. Trimmed darts are further classified into two types according to the shape of the trimmed materials. If the shape is given by an arbitrary polygon, the trimmed dart is called a *polygonal cut*. If the shape is given by a linear line segment, it is called a *linear cut*. Figure 6.5 shows two polygonal cuts and a linear cut.

\(^3\)Webster’s New International Dictionary
Figure 6.4: Stitched darts (before and after insertions).

Figure 6.5: Trimmed darts (before and after insertions).
Note that our underlying inextensibility assumption is kept valid under the insertion of darts (whether they are stitched or trimmed). The distance and topology between adjacent mesh points, however, may be affected by inserting a dart.

6.2.1.1 Stitched Darts

Stitched darts are defined by the following characteristics:

- The shape of a stitched dart is an isosceles triangle, which is called a darting triangle.
- Two of the vertices of a darting triangle must be located on the border of the cloth.
- Two or more darts may not overlap with each other.

With the above characteristics, the two sides of a darting triangle are guaranteed to exactly match up to form a single darting edge. This is important since a distortion may result if the two sides of a darting triangle do not match. Should this occur it may violate our inextensibility assumption for threads. Although the assumption that the shape must be an isosceles triangle is apparently restrictive, it provides a good first-order approximation to an arbitrary triangular stitched dart.

The advantage of stitched darts over trimmed darts is that phenomena like gaps and overlaps will never result (See Figure 6.12). The disadvantage of stitched darts is that they are less effective than trimmed darts at removing anomalous events (Compare Figure 6.22 with Figure 6.25).

A stitched dart is inserted with the following steps:

1. Specify three mesh points which correspond to the vertices of a darting triangle. Two of them must be on the border of the original rectangular woven cloth, and two interior edges of the triangle (called side edges) must be equal in length.
2. Scan the side edges of the darting triangle, and create mesh points called *darting mesh points* located at the intersection of the side edges and the threads.

3. Establish new topological and distance relationships between the mesh points on or adjacent to the darting edges.

These steps are repeated for each stitched dart. The darting information has to be maintained properly to allow for the insertion of a stitched dart specified in the above three steps. Since the vertices of a darting triangle are assumed to be coincident at the mesh points (crossings between wefts and warps), it is convenient to incorporate the darting information into the data structure of a mesh point, instead of making a special data structure for a dart. In order to incorporate the darting information into the mesh point data structure, several requirements have to be met:

1. Any mesh point must be categorized as an ordinary mesh point, a darting mesh point, or a mesh point not to be processed.

2. Topological rearrangements must be done when the darting information is specified.

3. Modifications to the metric data between neighboring mesh points must be done when the darting information is specified.

Regarding requirement 1, every mesh point is initially categorized as an ordinary mesh point. After the darting information is specified, darting mesh points are created at the intersections between the edges of the darting triangle and threads (wefts and warps). In addition, if an ordinary mesh point is inside the darting triangle, it is categorized as a mesh point not to be processed. Requirement 2 is satisfied by maintaining four pointers to the neighboring mesh points. For convenience, we call them WEST, EAST, SOUTH, and NORTH neighbors based on the initial 2D definition of the rectangular woven cloth composite ply. Specifically, the WEST neighboring mesh point is located on the left side of the same weft thread, and the EAST neighboring mesh point is located on the right side of the same weft thread.
Similarly, the SOUTH neighboring mesh point is located on the bottom side of the
same warp thread, and the NORTH neighboring mesh point is located on the top
side of the same warp thread. It should be noted that this static topological rela-
tionship is maintained after the insertion of one or more darts. Requirement 3 is
satisfied by maintaining the distances between the four neighbors mentioned above.
Refer to Appendix C for the (C language) data structure of a mesh point that
satisfy the above requirements.

We classify a stitched dart as either a side stitched dart or a corner stitched
dart. A side stitched dart is inserted at the side of a given rectangular sheet of
woven cloth composites, and a corner stitched dart is inserted at the corner. We
will describe them by using a simple example for each category.

**Side Stitched Darts**

Figure 6.6 shows an example of a stitched dart inserted at the side of the
given rectangular cloth. In Figure 6.6 (a), mesh points labeled from 1 to 16 are
affected by the insertion of the dart. Specifically, mesh points denoted by black
solid circles \{2,5,8,11,14,16\} represent the darting mesh points. Among them, mesh
points \{5,8,11,14\} are created during the dart insertion. Note that except mesh
point labeled 2, we have exactly two pairs for each darting mesh point. Mesh points
denoted by open circles \{1,3,4,6,7,9,10,12,13,15\} represent mesh points adjacent to
the darting edges that are affected by the insertion of the dart. The mesh point
labeled 2 and a pair of mesh points labeled 16 are coincident on the vertices of the
darting triangle. After dart insertion, the topological and distance relations must be
modified. For example, after the insertion of the dart, the mesh point labeled 4 has
a new neighboring mesh point labeled 5 and the distance between them is shorter
than the original distance between adjacent crossings. The result of this stitched
dart insertion is illustrated in Figure 6.6 (b).
Figure 6.6: An example of the insertion of a stitched dart ((a) before and (b) after the insertion).

Figure 6.7 illustrates the data structure corresponding to the side stitch dart shown in Figure 6.6. A square box with bold lines corresponds to a darting mesh point located on the edge of a darting triangle, and a square box with ordinary solid line corresponds to an ordinary mesh point. Each box has its mesh point ID written in the inner square. It has four pointers to its neighbors represented by small rectangles attached to the outside of the bigger square box, and the associated distances in each direction written in trapezoidal boxes. If a neighboring pointer is null, the corresponding box is filled in black. Note that the distance value 1.0 in this example implies the original distance between adjacent mesh points, and the values other than 1.0 implies the distances modified by inserting the dart. These values will eventually affect the mapping calculations.
Figure 6.7: The data structure for the example in Figure 6.6.
Corner Stitched Darts

A corner stitched dart is a stitched dart inserted at the corner of a rectangular sheet of woven cloth composites as shown in Figure 6.8. The shape is represented by an isosceles triangle as in the case of a side stitched dart. In terms of the data structure, there are slight differences. First of all, the adjacent darting mesh points on the edge of the darting triangle are not linked by pointers. In the case of a side stitched dart shown in Figure 6.6, the darting mesh points on the edge of the darting triangle are linked by the pointers from NORTH to SOUTH and vice versa. Secondly, the pointers of the mesh points adjacent to a darting mesh point have different thread directions. Let us demonstrate these by using an example.

Figure 6.8 illustrates a NORTH-EAST corner dart and Figure 6.9 illustrates the associated data structure. As mentioned, there is no link between adjacent darting mesh points on the edge of the darting triangle. The fact that the pointers of the mesh points adjacent to a darting mesh point have different thread directions can be demonstrated by the following example: darting mesh point 3 in Figure 6.9 has two adjacent ordinary mesh points 2 and 4. The direction of the pointer from
Figure 6.9: The data structure for the example in Figure 6.8.

Mesh point 3 to mesh point 4 is WEST to EAST along a weft thread, but the direction of the pointer from mesh point 3 to mesh point 2 is not EAST to WEST, but EAST to NORTH along a warp thread. The same applies to other darting mesh points. The important point is that the new mesh-point data structure represents the corner stitched dart precisely. Specifically, with this new data structure, we can calculate the mappings of mesh points (either ordinary mesh points or darting mesh points) along the darting edge as if it were a stitch. For example, suppose we scan the mesh points from WEST to EAST along each weft thread, and from SOUTH to NORTH along each warp thread. Then the scanning sequence for the mesh points labeled in Figure 6.9 is ordered as follows:
where the number with an asterisk denotes an ID attached to a darting mesh point.

### 6.2.1.2 Trimmed Darts

Trimmed darts (for both polygonal and linear cuts) have two advantages over stitched darts. With trimmed darts it is not necessary to match up threads along darting edges, and there is no restriction as to where they may be inserted and what shape they should be, as long as they are either polygons or line segments. On the other hand, the disadvantage of trimmed darts is that there is a possibility of gaps and overlaps in the fitting result.

### Polygonal Cuts

Polygonal cuts have the following characteristics:

- The shape of a polygonal cut is an arbitrarily-shaped polygon without holes.

- Vertices of a polygon may be located anywhere in the original rectangular woven cloth.

- Two or more polygonal cuts may not overlap with each other.

A polygonal cut is inserted with the following steps:

1. Specify all the vertices of a polygon in a clockwise order.
Figure 6.10: An example of the insertion of a polygonal cut.

2. Scan the polygonal area and set every mesh point inside the polygon to be OUT. Note that initially every mesh point is set to be IN, implying the mapping calculation should be performed on the mesh point.

3. Establish new topological and distance relationships between the mesh points located along the border of the polygon.

These steps are repeated for each polygonal cut. Figure 6.10 presents an example of inserting a polygonal cut. The mesh points denoted by black solid circles are removed by the insertion of the polygonal cut. The mesh points denoted by open circles labeled from 1 to 15 are affected by the insertion of the polygonal cut. Their topological and distance relationships with the neighboring mesh points must be modified appropriately. For example, the mesh point labeled 8 originally had four neighbors, but after the insertion of the polygonal cut, three of them are removed.

**Linear Cuts**

Linear cuts have the following characteristics:

- The shape of a linear cut is a straight line segment.
- End vertices of a linear cut may be located anywhere in the original rectangular woven cloth.

- Two or more linear cuts may not overlap with each other.

A linear cut is inserted with the following steps:

1. Specify two end points of a straight line segment.

2. Scan the line.

3. Establish new topological and distance relationships between every mesh point on or adjacent to the line.

These steps are repeated for each linear cut. As in the polygonal cut, the intersection calculation is performed between the cut line and the threads in the second step. The important difference between a linear cut and a polygonal cut is that no mesh points are removed during a linear cut, except when the mesh point is coincident on the cut line. Figure 6.11 provides an example of inserting a linear cut \( \overline{AB} \). Topological and distance relationships between mesh points denoted by open circles must be modified. Specifically, if there is an intersection between the linear cut and the edge between two adjacent crossings, the link is disconnected and the distance is changed to the distance from the crossing to the intersection.

### 6.2.2 Where Darts Should be Inserted

The solution to the problem of dart location becomes tractable if we limit our attention to the anomalous events caused by excessive shear deformation. With this limitation, the problem becomes that of identifying the region where excessive shear deformation exists. In essence, darts should be inserted where excessive shear deformation is likely to occur, under the given set of initial conditions. We hereafter refer to such a region as a critical region. A critical region is classified into two types: (1) a lower critical region, where the thread angle of every mesh point in the
Figure 6.11: An example of the insertion of a linear cut.

region is less than $\gamma_{\text{lock}}^{\text{min}}$ and (2) an upper critical region, where the thread angle of every mesh point in the region is greater than $\gamma_{\text{lock}}^{\text{max}}$.

In order to identify the existence of a critical region, we begin by generating an initial fitting, given appropriate initial conditions as mentioned in Chapter 5. Then we can either plot the distribution of thread angles between the weft and the warp at each mesh point, or shade each mesh point with a color based on the thread angle. When the thread angle at a mesh point goes below the minimum locking angle ($\gamma_{\text{lock}}^{\text{min}}$), we mark the mesh point, or shade with a suitable color, e.g. red. Similarly, when the thread angle exceeds the maximum locking angle ($\gamma_{\text{lock}}^{\text{max}}$), we mark the mesh point, or shade with a suitable color, e.g. blue. Otherwise we skip the mesh point, or mark it with another color, e.g. green. Of course, we can continuously color every mesh point by interpolating two extreme colors for the locking angles. Finally by tracing the marked mesh points, we can identify critical regions.

6.2.3 What Darts Should be Inserted

Once critical regions are identified, we must determine the type of darts that best fit into each region. If a critical region is completely inside the given rectangular ply, trimmed darts are the only possible choice. On the other hand, if a critical region is touching the boundary of the ply, either stitched or trimmed darts may be
applied. This restriction comes from the characteristics of our models of stitched and trimmed darts.

Next, the shape of each dart must be determined. For stitched darts, the shape of a dart is predetermined as a triangle by definition. For trimmed darts, we are allowed to define an arbitrary polygonal shape, including a concave polygon. The requirements for the shape of a trimmed dart include the following:

- A critical region should be removed.

- The shape of a trimmed dart should be simple for actual trimming. In other words, the number of edges of the polygon should be minimum.

Intuitively, it appears that the shape of a trimmed dart should match the shape of a critical region in the given 2D rectangular ply. Usually, however, a dart that is sufficient is one that is smaller than the critical region. In order to find the optimal shape, we have employed a strategy based on a relaxation technique. In other words, we start with a certain shape of a dart which includes a part (or the whole) of the critical region. Then, we successively modify the shape so that in each step we can decrease the critical region until none is left.

An important component of our relaxation technique is a good initial dart specification. If we start with an arbitrary-shaped polygonal cut, gaps and overlaps may occur at the same time, as shown in Figure 6.12. Similarly, if we start with an arbitrary-shaped stitched dart, a solution to the mapping calculations mentioned in the previous chapter may not always be found. On the other hand, if we start with a linear cut inserted into a critical region, the result of the fitting may produce either a gap or an overlap, but not both. This can be explained by the fact that a lower critical region is caused by a surplus of woven cloth material, while an upper critical region is caused by a shortage of material. This is an important property useful for determining an optimal shape for a dart. Stated formerly:
Property 6.1 If a linear cut is inserted into a lower critical region, the fitting result may produce an overlap. If a linear cut is inserted into an upper critical region, the fitting result may produce a gap.

By taking advantage of this property, we have developed an algorithm for determining an optimal shape for a dart when only “lower” critical regions exist. As illustrated in Figure 6.13, a linear cut is inserted into each lower critical region in the first stage of the algorithm. It should be noted that this algorithm does not cover the case where a linear cut is inserted into an “upper” critical region. As stated in Property 1, a gap may occur when a linear cut is inserted into an upper critical region. In this case, all that is needed is to prepare another ply and patch the gap with it. More details of the algorithm, especially on the removal of overlaps, will be described in the next section.

Before proceeding to the description of overlapping removal algorithms, another remark that should be made is that finding the best initial linear cut is still an open problem. As far as our experiments go, however, it works well when the
initial linear cut goes through a critical region and reaches slightly into the sur-
rounding normal region where the shear deformation is within the allowable limits.
See Figures 6.23 (b) and 6.28 (d) for our experimental cut insertions.

6.2.4 Overlapping Removal Algorithms

As described in the previous section, overlaps may occur in the fitting result when a linear cut is inserted into a lower critical region. In regard to fitting a 2D ply of woven cloth composites to a real 3D surface, minor overlaps are frequently observed and accepted. Major overlaps, however, cause serious problems since they may change the surface profile (called IML or Inside Mold Line) and alter the uniformity in terms of structural reinforcements. Our goal is to remove major overlaps and find the optimal shape for the dart that is to be inserted into a lower critical region.

As illustrated in Figure 6.13, our fundamental algorithm for removing overlaps includes two major steps: (1) finding overlaps and (2) updating the shape of the dart. The other two steps in Figure 6.13, “select an overlapping mesh point from OPL”\(^4\) and “remove the mesh point”, will be described in the second major step.

6.2.4.1 Finding Overlaps

The first major step toward determining a shape of the dart to be inserted into a lower critical region is to find overlaps. Assuming that both the 2D and the 3D deformed woven cloth composites are approximated by a collection of triangles as depicted in Figure 6.14, overlaps in the fitting result may be found with a point-triangle location algorithm in 3D space. Specifically, every mesh point is tested for inclusion in each triangle in 3D space. An overlap is identified if the projection of mesh point ‘P’ onto the plane formed by three mesh points A, B, and C is located inside triangle ABC as illustrated in Figure 6.15.

\(^4\)OPL stands for Overlapping Point List, and is defined later.
Figure 6.13: Algorithm for determining a shape of a dart to be inserted into a lower critical region.

Figure 6.14: Triangular mesh approximation for finding overlaps.

This simple algorithm works well as long as the distance between adjacent mesh points is sufficiently small and the thread segment between them can be regarded as a straight line, as we have assumed for our woven cloth model. In terms of the computational complexity, if we have $m \times n$ mesh points for the woven cloth composite, we need $O((mn)^2)$ time to find all the overlaps.

We have developed two methods for improving the efficiency of the algorithm. The first method takes the advantage of the properties of the overlapping mesh points. In other words, the overlapping mesh points are usually clustered, and in
Figure 6.15: A point-triangle inclusion test, where the projection of mesh point P is located inside triangle ABC.

each cluster one or more overlapping mesh points must be touching either the boundary of a given rectangular woven cloth ply, or the cut line generated by inserting a linear cut. The algorithm for finding the overlaps that utilizes these properties is given as follows:

1. Identify the mesh points that touch either the boundary of a given 2D rectangular woven cloth ply or the cut line generated by each linear cut. Let us refer to these mesh points as *edge-touching mesh points*.

2. Check if an edge-touching mesh point is included in a triangle formed by mesh points, where at least one of them must be an edge-touching mesh point other than the one currently being checked. If so, repeat the process by substituting the current edge-touching mesh point for its neighboring mesh points, until all of the neighboring mesh points are classified into non-overlapping mesh points.

3. Repeat Step 2 until all the edge-touching mesh points are processed.

Since each linear cut has at most $O(\text{max}(m, n))$ edge-touching mesh points and there are $O(m + n)$ boundary mesh points, the total number of edge-touching mesh points is $O(m + n + \text{max}(m, n)) = O(m + n)$. Similarly, the number of triangles that contain at least one edge-touching mesh points is $O(m + n)$. Thus, we need
$O((m + n)^2)$ time to find the overlaps that involve the edge-touching mesh points. The remaining work is to iteratively check overlaps for the mesh points adjacent to each edge-touching mesh point that is found to be overlapping. The efficiency of this work generally depends on the given surface shape to be fitted and the shape of each linear cut. In the worst case, $O(mn)$ such neighboring mesh points have to be checked for overlaps with $O(mn)$ triangles, whose computational complexity is equivalent to the original algorithm. However, assuming that overlaps are “local” and the number of overlapping mesh points is proportional to the number of edge-touching mesh points along each linear cut, only $O(m + n)$ neighboring mesh points have to be checked for overlaps with $O(m + n)$ triangles.

An alternative method, we have devised a triangle tree data structure similar to a quadtree [95] for representing the network of mesh points. The difference between them is that the leaf in a triangle tree is a triangle instead of a quadrangle and that the root of the tree of a triangle tree has two children instead of four. Figure 6.16 shows an example. Specifically, the original triangular mesh illustrated in Figure 6.16 (a) is compactly represented as shown in Figure 6.16 (b). The corresponding tree data structure is illustrated in Figure 6.16 (c). This data structure allows us to improve the efficiency of the algorithm for finding overlaps. The key idea is to check the overlap of each mesh point with a larger triangle, instead of the unit triangle formed by three adjacent mesh points.

The construction of a triangle tree goes as follows. Assume that we have $(2^k + 1) \times (2^k + 1)$ mesh points at the outset, where $k$ is a non-negative integer. Then, starting with $(2^k + 1) \times (2^k + 1)$ number of leaves for a triangle tree, we merge the four sibling triangles into a larger one, if all the vertices of the four triangles are included in the larger triangle. This assures that an overlap found in a smaller triangle is always detected in a larger triangle. On the other hand, if there is no overlap detected in a larger triangle, there is no overlap in smaller triangles descendant of the larger one.

This method significantly reduces the number of triangles for checking the
overlap with a mesh point. Suppose the number of the overlap checks for a given mesh point is proportional to the depth of the tree, we need $O(k) = O(\log_2 n)$ time to check for the overlap, where we assume $m = n = 2^k + 1$. In total, we need $O(n^2 \log_2 n)$ time to complete the overlap checks for all the mesh points. Of course, by combining the first method with the second method, we can further improve the computational efficiency.

6.2.4.2 Updating the Shape of a Dart

Once overlaps are found, we have to remove them and update the shape of the dart. Removing overlaps appears to be an easy matter. However, we cannot simply remove all of the overlapping mesh points. Suppose, for instance, we have a situation where overlaps occur as shown in Figure 6.17 (a). If we simply remove all the overlapping mesh points, the result looks like Figure 6.17 (b) and it opens a new gap. The removal of an overlapping mesh point is called safe, if it does not open a gap whose size is larger than the distance between adjacent mesh points. Therefore, if an overlapping mesh point lies on the border of the actual shape of the material, its removal is safe. Figure 6.17 (c) shows one of the valid solutions to the overlapping removal, by applying a sequence of safe removals. Note that there are
many valid answers.

The differences between valid answers come from the different orders of selecting mesh points when checking overlaps. It is necessary to resort to a particular optimizing strategy during overlap removal. Optimizing criteria which we utilize include the following:

1. Minimize shear deformation in the resulting woven cloth
2. Minimize raggedness in the darting edge
3. Increase symmetry along the cut

It may not be possible to implement an algorithm that satisfies all of the above optimizing criteria simultaneously. For this reason, we have implemented several different algorithms to correctly remove overlaps. For every method, we maintain a common data structure called overlapping point list (OPL), which is a list of mesh points that are found to overlap a part of the woven cloth fitted to a surface mold. The following is a list of the methods and characteristics that we have implemented:

• **Sequential Method:**
  The first mesh point in the overlapping point list (OPL) is selected and tested for safe removal. If the removal is safe, the mesh point is removed from the OPL and the shape of the dart together with the OPL is updated. This can
be implemented with a FIFO (First In First Out) or LIFO (Last In First Out) strategy. This method is easy to implement and Optimization Criterion 2 is usually guaranteed because of the coherence in the distribution of overlapping points. However, Optimization Criteria 1 and 3 are usually not assured.

- **Random Method:**
  With this method, we randomly select a mesh point from the OPL and test it for safe removal. On average, the result of this method assures Optimization Criterion 3, but it does not assure Criteria 1 and 2.

- **Cut Distance Priority Method:**
  With this method, the OPL is sorted by the distance from the cut inserted. The mesh point closest to the cut is first selected and tested for safe removal. This method guarantees Optimization Criteria 2 and 3, but does not necessarily guarantee Criterion 1.

- **Thread Angle Priority Method:**
  With this method, the OPL is sorted by the value of thread angles (γ). The mesh point whose cos γ is largest is first selected and tested for safe removal. This method guarantees Optimization Criterion 1, but does not necessarily guarantee Criteria 2 and 3.

The choice of a method depends on what optimization criteria is considered most important for the current configuration. For instance, if Criterion 1 is considered most important, the thread angle priority method is the best choice. If Criterion 2 is considered most important, the cut distance priority method or sequential method may be the best choice.

In the next section, we present several fitting examples where one or more darts are inserted. Particular attention is paid to what kind of methods for removing overlaps are used and what kind of optimizing criteria are sought when darts are inserted into lower critical regions.
6.2.5 Examples of Inserting Darts

Here we describe several examples of fitting a 2D ply of woven cloth composites to NURBS (Non-Uniform Rational B-Spline) surfaces [29], in which dart insertion has been applied.

6.2.5.1 Dart Insertion into a Ply Fitted to an Object with a Convex Corner

The first example is a fitting to an object with a convex corner, which is represented by a composite NURBS surface made up of seven different NURBS patches as shown in Figure 6.18. Three of them are planes, three of them are quadrants of a cylinder, and one of them is an octant of a sphere. \( C^1 \) continuity is assured at the boundaries between any two adjacent patches. Figure 6.19 shows a fitting of a woven cloth composite ply onto the object. Initial conditions for the fitting are specified by three parameters: a starting point \((P_s)\), a base path (or a guide line), and a sweeping direction. These are specified in both 2D and 3D spaces. The starting point in this example is on the \( Z \) axis, the base path is specified such that it is aligned with the curve on the \( XZ \) plane, and the sweeping direction is in the positive \( Y \) direction. As can be seen from the result, the thread angles become very small around the corner most distant from the starting point. Figure 6.20 (a) shows the plane development pattern corresponding to the fitting. Figure 6.20 (b) is the distribution of thread angles, projected above the associated plane development. It is clear from Figure 6.20 (b) that there are some mesh points whose thread angle is almost zero, and thus we cannot avoid anomalous events such as wrinkles in the critical region.

In this example, thread angles in the critical region become less than the allowable minimum \((\gamma_{\text{lock}})\), and therefore it is classified as a lower critical region. In order to intuitively identify the critical region, we shade each mesh point with a different color according to its thread angle value. Figure 6.21 (a) and (b) show the results corresponding to Figure 6.19 and Figure 6.20 (a), respectively. In this figure,
Figure 6.18: An object with a convex corner made up of seven NURBS patches: three planes, three cylindrical parts, and one spherical part.

Figure 6.19: Fitting to an object with a convex corner.
Figure 6.20: (a) Plane development pattern generated from the fitting for the surface with convex corner, and (b) distribution of thread angles at mesh points projected above the plane development.

the variable $T$ attached to the color bar represents thread angle $\gamma$, and the lower critical region is colored with red. In addition, the red lines in Figures 6.21 (a) and (b) represent the base path for the fitting.

Note that throughout the examples we have assigned colors as follows: red is assigned when $\gamma = 0$ degrees, green is assigned when $\gamma = 90$ degrees, blue is assigned when $\gamma = 180$ degrees, and an interpolated color is assigned in hue space [86] for intermediate values of $\gamma$. With this rule, a mesh point takes on a reddish color when $\gamma$ becomes lower than approximately 30 degrees, which we can be visually interpreted as a point in a lower critical region. Although we can shift the color distribution for fine tuning, the above rule is reasonable since most commercial fabrics “lock” or “jam” at angles less than 35 degrees [62]. See Appendix F for more details.

The important feature of this example is that the critical region has a symmetrical shape. Therefore it is possible to insert either a stitched dart or a polygonal cut. Figure 6.22 (a) is the result of fitting a woven cloth composite ply with a stitched dart, and Figure 6.22 (b) is the 2D plane development. It should be noted that there remains a small amount of critical region with red colors, after inserting
Figure 6.21: Fitting results with colors based on thread angle values; (a) 3D fitting and (b) 2D plane development pattern.
On the other hand, we can insert a polygonal cut. In this case, we apply the algorithm outlined in Figure 6.13. Specifically, we first insert a linear cut instead of a polygonal cut as shown in Figure 6.23 (b), and the fitting result shown in Figure 6.23 (a) is obtained. It is interesting that with the insertion of a linear cut, the critical region with red colors vanishes.

An overlap is observed in the fitting result in Figure 6.23 (a), as predicted by Property 1. Figure 6.24 shows the sequence of the removal of overlaps by applying the overlapping removal algorithm illustrated in Figure 6.13, based on the thread angle priority method. Figure 6.25 shows the fitting result obtained by inserting the polygonal cut, corresponding to the dart (loop = 9) in Figure 6.24. Unlike the insertion of a stitched dart, in Figure 6.25 (a) there is no critical region in the
Figure 6.23: Linear cut insertion: (a) 3D fitting and (b) 2D plane development pattern with a linear cut.
Figure 6.24: An example of the overlapping removal algorithm applied to a woven cloth ply, starting with a linear cut (loop = 1). A black circle denotes an edge-touching mesh point, and a white circle denotes an overlapping mesh point. The iteration stops when the OPL becomes empty as depicted in Figure 6.13. In other words, it stops when no more overlapping mesh points remain.
fitting result. Figure 6.25 (b) is the corresponding 2D plane development pattern. The minor overlap remaining in Figure 6.25 (a) does not cause serious problems, as long as there is no gap. Figure 6.26 shows a real woven cloth sheet (Kevlar) with the insertion of a polygonal cut specified with black contour lines, based on the simulation obtained as shown in Figure 6.25 (b). Figure 6.27 is the result of fitting the sheet after inserting the polygonal cut. Although there is still a minor overlap around the corner, major wrinkling as demonstrated in Figure 7.13 is removed by inserting the polygonal cut.
Figure 6.26: A woven cloth sheet (Kevlar), where a polygonal cut is specified with black lines.
Figure 6.27: The fitting result with a sheet of woven cloth Kevlar having a polygonal cut.
6.2.5.2 Linear Cut Insertion into a Ply Fitted to an Object with a Concave Corner

The second example is a fitting to an object with a concave corner. Figure 6.28 (a) shows the 3D fitting result with colors based on the thread angles, and Figure 6.28 (c) shows the 2D plane development. Unlike the previous example, the results indicate the existence of an upper critical region with blue colors. We can remove the upper critical region by inserting a linear cut as shown in Figure 6.28 (d) at the expense of producing a gap in the 3D fitting result as shown in Figure 6.28 (b). Property 1 also predicts this behavior. We therefore need two or more separate ply fragments to completely cover the surface, under the same initial conditions.
6.2.5.3 Dart Insertion into a Ply Fitted to a Shoe-like Object

The last example is a fitting to a shoe-like object. Refer to Section 7.3.5.1 for the definition of this object. Figures 6.29 (a) and (b) show the fitting result and the 2D plane development pattern, respectively. Clearly, two isolated critical regions are observed. Since both of the critical regions are not symmetrical, it is easier to apply trimmed darts to the example. First, two linear cuts are inserted as shown in Figure 6.30 (a) and the corresponding fitting result is shown in Figure 6.31 (a). As in the previous example, we have overlaps. We have tested three different methods for removing the overlaps; (1) sequential method (Figure 6.30 (b) for the final shape of the dart and Figure 6.31 (b) for the fitting result), (2) cut distance priority method (Figure 6.30 (c) for the final shape of the dart and Figure 6.31 (c) for the fitting result), and (3) thread angle priority method (Figure 6.30 (d) for the final shape of the dart and Figure 6.31 (d) for the fitting result). In terms of thread angle values, method (3) produces the best result. In terms of raggedness, methods (1) and (2) produce better results than method (3). In terms of symmetry with respect to the linear cuts, method (2) produces the best result.
Figure 6.29: (a) A fitting to a shoe-like object with colors based on the thread angle values at each mesh point and (b) the 2D plane development.
Figure 6.30: 2D plane development patterns with trimmed darts: (a) two linear cuts are inserted, (b) polygonal cuts obtained by sequential method, (c) polygonal cuts obtained by cut distance priority method, and (d) polygonal cuts obtained by thread angle priority method.
Figure 6.31: 3D fitting results with trimmed darts: (a) two linear cuts are inserted, (b) polygonal cuts obtained by sequential method, (c) polygonal cuts obtained by cut distance priority method, and (d) polygonal cuts obtained by thread angle priority method.
CHAPTER 7
ALGORITHMS DERIVED FROM GEOMETRIC PROPERTIES OF A TCHEBYCHEV NET

In this chapter, we first review geometric properties of a woven cloth model with a Tchebychev net assumption, and then present algorithms derived from the properties. Although the basic results described here are applied to a ply of woven cloth consisting of inextensible threads with “continuously” changing tangents, they can also be applied to a ply of woven cloth whose threads are approximated by piecewise linear line segments, as assumed in Chapter 4, as long as the length of each line segment is sufficiently small.

7.1 Geometric Properties of a Tchebychev Net

Here we present geometric properties of a 2D ply of woven cloth composites, based on the differential geometry derived from a Tchebychev net assumption. Our aim is to identify an important relationship between the thread angle at each crossing and the intrinsic properties of the surface when the ply is deformed. In the next section, we apply this relationship to the development of an algorithm for obtaining good initial conditions for a fitting, given a 3D surface shape, and an algorithm for predicting and preventing anomalous events such as wrinkling and tearing.

7.1.1 Fundamental Equation of a Tchebychev Net

Suppose \((x, y)\) represents a coordinate in a 2D ply of woven cloth composites. Consider a location \((x + dx, y)\) along a weft, and a location \((x, y + dy)\) along a warp, where \(dx\) and \(dy\) are sufficiently small positive values. We then consider mapping three points \(((x, y), (x + dx, y), (x, y + dy))\) onto a 3D surface. Suppose that \((x, y)\) is mapped to \((u, v)\), the coordinate of the point represented in a bivariate parametric space. Suppose also that the surface is \(C^\infty\) differentiable everywhere.
Then, by applying Taylor’s theorem to the first order partials around the point \((u, v)\),
the point \((x + dx, y)\) is mapped to \((u + \frac{\partial u}{\partial x} dx, v + \frac{\partial v}{\partial x} dx)\), and the point \((x, y + dy)\)
is mapped to \((u + \frac{\partial u}{\partial y} dy, v + \frac{\partial v}{\partial y} dy)\) as shown in Figure 7.1. From the inextensibility
assumption, the distance between \((u, v)\) and \((u + \frac{\partial u}{\partial x} dx, v + \frac{\partial v}{\partial x} dx)\) must be equal to
dx, and the distance between \((u, v)\) and \((u + \frac{\partial u}{\partial y} dy, v + \frac{\partial v}{\partial y} dy)\) must be equal to dy.
We therefore obtain the following pair of equations:

\[
\begin{align*}
\left( \frac{\partial u}{\partial x} \right)^2 + \left( \frac{\partial v}{\partial x} \right)^2 &= 1 \\
\left( \frac{\partial u}{\partial y} \right)^2 + \left( \frac{\partial v}{\partial y} \right)^2 &= 1
\end{align*}
\]

Equation (7.1) can be concisely put into the following: \(|r_x| = 1\) and \(|r_y| = 1\), where
\(r_x = (\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x})\) and \(r_y = (\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y})\).

Equation (7.1) represents a fundamental equation for a Tchebychev net composed of continuous inextensible threads. It is a system of non-linear hyperbolic partial differential equations. Formally, a deformed Tchebychev net is a surface
whose coordinate vector fields with respect to thread directions have unit magnitude [91]. When \(r_x\) depends only on \(x\) and \(r_y\) depends only on \(y\), Equation (7.1) is
integrable and the deformed Tchebychev net has the form [108]

\[ \mathbf{r}(x, y) = \mathbf{P}(x) + \mathbf{Q}(y), \]

where \( \mathbf{r}_x = \mathbf{P}'(x) \) and \( \mathbf{r}_y = \mathbf{Q}'(y) \). Surfaces of this kind are called translation surfaces [101, 71].

In terms of kinematic conformability, a surface that can be fitted with a single piece of a Tchebychev net must be sufficiently smooth and topologically equivalent to a simple sheet [102]. Mathematically, kinematic conformability is translated into the existence of a solution to the system of non-linear hyperbolic partial differential equations expressed in Equation (7.1). It is a necessary condition, but not a sufficient condition for “practical” conformability for the following two reasons: First, the surface obtained by deforming a 2D Tchebychev net must be enclosed by four threads (two wefts and two warps). In reality, however, a surface to be fitted with a woven cloth ply may not always be enclosed by four threads. Second, even if we have a solution to the partial differential equations, it may not be acceptable because thread angles in the solution may exceed the allowable shear limits (See Section 4.2).

As we will see, solutions to Equation (7.1) generally provide a broader class of fittings than the fittings obtained by the solutions to the mapping calculation mentioned in Section 5.2. This is because we do not assume “straightness” of a thread segment between adjacent crossings or “no slippage” at each crossing.

Similarly we can obtain the first fundamental form [99, 100, 101] for a 3D Tchebychev net. Here the arc length element \( ds \) is represented by \( dx \) and \( dy \) in the Tchebychev net coordinate.

\[
I = ds^2 \equiv Edx^2 + 2Fdx dy + Gdy^2
\]

\[
= |\mathbf{r}_x|^2 dx^2 + 2(\mathbf{r}_x \cdot \mathbf{r}_y) dx dy + |\mathbf{r}_y|^2 dy^2
\]

\[
= dx^2 + 2 \cos \gamma dx dy + dy^2, \quad (7.2)
\]

where \( \gamma \) represents the thread angle between a weft and a warp. From the above
equation, we have \( E = G = 1 \) and \( F = \cos \gamma \), which is another necessary condition for a surface to be a Tchebychev net.

### 7.1.2 The Relationship between the Intrinsic Geometry of a Surface and the Thread Angle of a Tchebychev Net

The first fundamental form of a Tchebychev net given in Equation (7.2) allows us to derive an interesting relationship between the Gaussian curvature (an intrinsic surface geometric property) and the thread angle \( \gamma \) as shown below. Incidentally, when the Gaussian curvature is constant, this equation is called \textit{sine Gordon equation}, widely known in mathematical physics [6, 89].

**Lemma 7.1** For a 3D surface where a Tchebychev net is fitted, there exists a relationship between the Gaussian curvature \( K \) of the surface and the thread angle \( \gamma \) between a weft and a warp, given as follows:

\[
K = -\frac{\gamma_{xy}}{\sin \gamma},
\]

(7.3)

where \( \gamma_{xy} = \frac{\partial^2 \gamma}{\partial x \partial y} \).

**proof:** From the theorem of Gauss, the Gaussian curvature of a surface can be expressed only by the first fundamental form [46, 100]:

\[
K = -\frac{1}{4W^4} \begin{vmatrix} E & E_x & E_y \\ F & F_x & F_y \\ G & G_x & G_y \end{vmatrix} = -\frac{1}{2W^4} \left( \frac{\partial}{\partial x} \frac{E_y - F_x}{W} - \frac{\partial}{\partial y} \frac{F_y - G_x}{W} \right),
\]

(7.4)

where \( W = \sqrt{EG - F^2} \). Since \( E = G = 1 \) for a 3D Tchebychev net, \( E_x = E_y = G_x = G_y = 0 \). Similarly from \( F = \cos \gamma \), we get \( F_x = -\gamma_x \sin \gamma \) and \( F_y = -\gamma_y \sin \gamma \). Substituting these equations into Equation (7.4), we obtain the equation in this lemma. \( \square \)

If the contribution from the mixed partial \( (\gamma_{xy}) \) is constant, we can infer an intuitive relationship between the Gaussian curvature of the surface and the thread
angle of a Tchebychev net at a mesh point. Specifically, the absolute of the Gaussian curvature at a mesh point must be large, if the angle is close to either zero or $\pi$. Conversely, the angle between a weft and a warp must be close to either zero or $\pi$, if the absolute of the Gaussian curvature at the point is large. On the other hand, the thread angle at a mesh point must be close to $\pi/2$, if the absolute of the Gaussian curvature at the point is very small.

The next lemma states a relationship between the angle of a weft and a warp, and the initial path along which a thread lies.

**Lemma 7.2** A thread of a Tchebychev net lies along a geodesic path, a curve of zero geodesic curvature, if and only if the thread angle between a weft and a warp is constant along the path.

**proof:** Let $\otimes$ denote a cross product operator, $\cdot$ denote an inner product operator, and $r_x$ and $r_y$ be the same as defined in Equation (7.1). Then, by definition, we have

$$r_x \cdot r_y = |r_x||r_y|\cos \gamma = \cos \gamma$$  \hspace{1cm} (7.5)

and

$$r_x \otimes r_y = (\sin \gamma)n,$$  \hspace{1cm} (7.6)

where $n$ denotes the unit surface normal. Taking the inner product with $n$ in Equation (7.6), we obtain

$$n \cdot r_x \otimes r_y = \sin \gamma.$$  \hspace{1cm} (7.7)

Meanwhile, by taking partial derivatives of both sides of Equation (7.5) with respect to $x$ and $y$, and by using the relations $r_{xy} = r_{yx} = \tau n$ ($\tau \in R$) and $n \cdot r_x = n \cdot r_y = 0$, we get

$$\begin{cases} r_{xx} \cdot r_y = -\gamma_x \sin \gamma \\ r_{yy} \cdot r_x = -\gamma_y \sin \gamma \end{cases}.$$  \hspace{1cm} (7.8)
Substituting Equation (7.7) into Equation (7.8) and considering \(|n \otimes r_x| = |n \otimes r_y| = 1\), we get

\[
\begin{align*}
\gamma_x &= -r_{xx} \cdot (n \otimes r_x) \\
\gamma_y &= r_{yy} \cdot (n \otimes r_y).
\end{align*}
\] (7.9)

By definition, geodesic curvatures along a weft (x-axis) \(\kappa^x_g\) and along a warp (y-axis) \(\kappa^y_g\) are represented by the following:

\[
\begin{align*}
\kappa^x_g &= |n \, r_x \, r_{xx}| = r_{xx} \cdot n \otimes r_x \\
\kappa^y_g &= |n \, r_y \, r_{yy}| = r_{yy} \cdot n \otimes r_y.
\end{align*}
\] (7.10)

By comparing Equation (7.9) with Equation (7.10), we finally obtain the relations \(\kappa^x_g = -\gamma_x\) and \(\kappa^y_g = \gamma_y\). The angle \(\gamma\) is constant if \(\kappa^x_g = 0\) or \(\kappa^y_g = 0\). Conversely, if either \(\gamma_x = 0\) or \(\gamma_y = 0\), then a thread lies along a geodesic represented by either \(\kappa^x_g = 0\) or \(\kappa^y_g = 0\). □

7.1.3 Global Properties of a Tchebychev Net

Another interesting result derived from Lemma 7.1 relates to the “global” conformability of a ply of woven cloth composites modeled with a Tchebychev net assumption [35]. By global conformability, we mean that a surface can be fitted with a single piece of woven cloth ply, assuming that the size of the ply is sufficiently large. Note that the phrase “clothing a surface globally” has the same meaning as “global conformability.”

**Theorem 7.1 (Hazzidakis 1880)** We cannot globally clothe a surface region enclosed by four threads, \(x = x_1, x = x_2, y = y_1,\) and \(y = y_2\) of a Tchebychev net, if

\[
|\int \int K \, dS| > 2\pi.
\]

**proof:** Let us use the notations in Figure 7.2, where \(\alpha_i (1 \leq i \leq 4)\) denote interior angles. Since the area element \(dS\) is expressed as \(dS = \sqrt{EG - F^2} \, dx \, dy = \)
Figure 7.2: A surface region enclosed by four threads: two wefts $x = x_1$ and $x = x_2$, and two warps $y = y_1$ and $y = y_2$. 

\[ \sin \gamma \, dxdy, \]

\[ - \int \int_S K \, dS = - \int \int K \sin \gamma dxdy \]

\[ = \int \int \frac{\partial^2 \gamma}{\partial x \partial y} \, dxdy \]

\[ = \int_{x_1}^{x_2} \int_{y_1}^{y_2} \frac{\partial^2 \gamma}{\partial x \partial y} \, dxdy \]

\[ = \gamma(x_2, y_2) - \gamma(x_1, y_2) - \gamma(x_2, y_1) + \gamma(x_1, y_1) \]

\[ = \alpha_3 - (\pi - \alpha_4) - (\pi - \alpha_2) + \alpha_1 \]

\[ = \sum_{i=1}^{4} \alpha_i - 2\pi \]

\[ < 2\pi. \]  

Hence, there is no set of interior angles $\alpha_i (1 \leq i \leq 4)$ that satisfy $|\int \int_S K \, dS| > 2\pi$ if $0 < \alpha_i < \pi$. \( \square \)

The theorem states that when we have a large integral Gaussian curvature of magnitude greater than $2\pi$, there is no chance of fitting to a given surface with a single ply of woven cloth composites, no matter what initial conditions are specified.
In contrast to the previous theorem, the next theorem assures the global conformability when we have a surface whose integral Gaussian curvature is less than $\frac{\pi}{2}$.

**Theorem 7.2 (Samelson 1989)** We can globally clothe a surface region with a Tchebychev net, if

$$|\int \int_{S} K \, dS| < \frac{\pi}{2}.$$  

Strictly speaking, this theorem is applied to a surface which is $C^\infty$-complete, open, simply-connected, two-dimensional Riemannian manifold with compactly supported Gaussian curvature. Samelson proved this theorem by showing the existence of a unique Tchebychev net of which initial data can be specified on the geodesic such that the threads make angles $\pi/4$ and $3\pi/4$ with the geodesic [91, 92]. An important consequence of this theorem together with Theorem 7.1 is that we can predict the global conformability of a given surface by computing the integral Gaussian curvature.

It should be noted, however, that even though the integral Gaussian curvature over a surface becomes less than $\frac{\pi}{2}$, it is not always guaranteed that a given ply of woven cloth composites can be fitted to the surface without wrinkling or tearing. In addition, there is an implicit assumption behind these theorems that the boundary threads must be aligned with geodesics. This can be confirmed by applying the Gauss-Bonnet theorem [100, 101] to this case, which is represented by the following formula:

$$\oint_{C} \kappa_g \, ds + \int \int_{S} K \, dS = 2\pi - \sum_{i=1}^{4} \alpha_i,$$  \hspace{1cm} (7.12)

where $\kappa_g$ represents the geodesic curvature of the boundary curves. In order for Equation (7.12) to hold for the surface region enclosed by threads of a Tchebychev net, $\kappa_g$ must be zero everywhere on the boundary curves (See Equation (7.11)). Thus, the boundary curves must be geodesics.
In Theorems 7.1 and 7.2, our main concern has been global conformability. The following theorem states that if we are allowed to use multiple pieces of woven cloth composites that can be joined, there is always an answer to the fitting problem.

**Theorem 7.3 (Bieberbach 1926)** *For any differentiable Riemannian manifold, there exists a local Tchebychev net.*

A “local” Tchebychev net here means a Tchebychev net with sufficiently small area. This result is due to Bieberbach [15]. The theorem is proved by showing the existence of a solution to the partial differential equations (Equation 7.1), in the vicinity of a point of interest.

### 7.2 Algorithms Derived from a Tchebychev Net Assumption

The purpose of this section is to describe algorithms derived from a Tchebychev net assumption. First, we present an algorithm for fitting a Tchebychev net to a curved surface with a finite difference method. Then, we describe an algorithm that allows us to predict the global conformability of a Tchebychev net, given an arbitrary 3D surface region. Finally, we present an algorithm that provides us with good initial conditions that lead to the least likelihood of anomalous events.

#### 7.2.1 An Algorithm for Fitting a Tchebychev Net to a Curved Surface

In Chapter 5, we investigated fitting a woven cloth model to a curved surface. Here we formulate an algorithm for fitting a Tchebychev net to a curved surface. As we have mentioned, a Tchebychev net is one of the assumptions for our woven cloth model. It is expected that solutions to the fitting problem with only the Tchebychev net assumption, should provide a broader class of fitting results under the same initial conditions and the same surface.

Let \( r(u, v) \) denote a parametric representation of a given surface. In principle, the fitting problem with a Tchebychev net assumption can be solved by using Equation (7.1). The goal we want to achieve with Equation (7.1) is to express \( u \) and \( v \)
in terms of the coordinates \( x \) and \( y \) in the Tchebychev net, or vice versa. For later discussion, we outline a derivation as Pipkin did in his paper in order to analyze equilibrium of Tchebychev nets [72].

First, from the chain rule of a partial derivative, we get

\[
\begin{align*}
\mathbf{r}_x &= \mathbf{r}_u u_x + \mathbf{r}_v v_x \\
\mathbf{r}_y &= \mathbf{r}_u u_y + \mathbf{r}_v v_y
\end{align*}
\]  

(7.13)

By substituting variables \( \mathbf{r}_x \) and \( \mathbf{r}_y \) in Equation (7.13) into Equation (7.2), and by considering \( d\mathbf{r} = \mathbf{r}_x dx \) with length \( ds = |\mathbf{r}_x| dx = dx \) and \( d\mathbf{r} = \mathbf{r}_y dy \) with length \( ds = |\mathbf{r}_y| dy = dy \), we obtain a system of non-linear partial differential equations:

\[
\begin{align*}
Eu_x^2 + 2Fu_xv_x + Gv_x^2 &= 1 \\
Eu_y^2 + 2Fu_yv_y + Gv_y^2 &= 1
\end{align*}
\]  

(7.14)

Ever since Tchebychev postulated his model for cloth [103], there have been several efforts as to solving this equation. Voss [108] was the first to consider expressing \( u \) and \( v \) as a function of \( x \) and \( y \), the Tchebychev net coordinates. Actually, he worked with Cartesian coordinate variables \( X, Y, \) and \( Z \) rather than bivariate UV coordinate variables \( u \) and \( v \). Servant [96] transformed Equation (7.14) into a system of second-order quasi-linear partial differential equations by differentiating the first equation of (7.14) with respect to \( y \) and the second equation with respect to \( x \). The resulting equations, called Servant’s equations, are expressed as follows:

\[
\begin{align*}
uxy &= -(ux_uy_1 + (ux_vy + uyvx)\Gamma^1_{11} + v_xvy_1\Gamma^1_{22}) \\
vxy &= -(ux_uy_1 + (ux_vy + uyvx)\Gamma^2_{11} + v_xvy_2\Gamma^2_{22})
\end{align*}
\]  

(7.15)

where \( \Gamma^i_{jk} \) \((i, j, k = 1, 2)\) are the Christoffel’s symbols of the second kind [46]. Specifically, they are represented by the quantities related to the first fundamental form
as below:

\[
\begin{align*}
\Gamma_{11}^1 &= (GE_u - 2FF_u + FE_v)/D \\
\Gamma_{12}^1 &= (GE_v - FG_u)/D \\
\Gamma_{22}^1 &= (2GF_v - GG_u - FG_v)/D \\
\Gamma_{11}^2 &= (2EF_u - EE_v - FE_u)/D \\
\Gamma_{12}^2 &= (EG_u - FE_v)/D \\
\Gamma_{22}^2 &= (EG_v - 2FF_v + FG_u)/D
\end{align*}
\]

(7.16)

where \( D = 2(EG - F^2) \). Servant’s equations form a quasi-linear hyperbolic system with threads of a Tchebychev net \((x = \text{const.} \text{ or } y = \text{const.})\) as characteristics.

Bieberbach [15] first proved the existence of a local solution to these equations as described in Theorem 7.3. Pipkin [72] gave a solution to Servant’s equations based on the Picard’s successive approximation, a well-known technique usually applied to an ordinary differential equation [4]. Let us suppose \( u(x, 0), u(0, y), v(x, 0), \text{ and } v(0, y) \) are known as initial conditions. The method starts with a rough approximation to \( u(x, y) \) and \( v(x, y) \) by integrating \( u_{xy} \) and \( v_{xy} \) with respect to both \( x \) and \( y \). For instance, the first approximation is given by

\[
\begin{align*}
u(x, y) &= u(x, 0) + u(0, y) - u(0, 0) \\
v(x, y) &= v(x, 0) + v(0, y) - v(0, 0)
\end{align*}
\]

(7.17)

This approximation is now used to evaluate the right-hand values of Equation (7.15), and the process of integration is repeated to obtain a new approximation. The problem with this approach is that, ignoring trivial cases, the integration of the right-hand side of Equation (7.15) has to be done numerically; numerical errors may be accumulated in successive approximations.

A better and simpler method that we propose is to apply a finite difference method [3] directly to the Servant’s equation (7.15). Let \( u_{x,y} \) and \( v_{x,y} \) denote \( u(x, y) \) and \( v(x, y) \), respectively. Then, finite difference approximations to \( u_x, u_y, \text{ and } u_{xy} \)
are expressed by

\[
\begin{align*}
  u_x &= \frac{u_{x+\delta x,y} - u_{x-\delta x,y}}{2\delta x} \\
  u_y &= \frac{u_{x,y+\delta y} - u_{x,y-\delta y}}{2\delta y} \\
  u_{xy} &= \frac{u_{x+\delta x,y+\delta y} + u_{x-\delta x,y-\delta y} - u_{x+\delta x,y-\delta y} - u_{x-\delta x,y+\delta y}}{4\delta x \delta y}
\end{align*}
\]

(7.18)

Similar expressions are obtained for \( v_x, v_y, \) and \( v_{xy} \) as well. By substituting these into Servant’s equations and by assuming \( \delta x \approx \delta y \approx 1 \), we obtain the following formula:

\[
\begin{align*}
  u_{x+1,y+1} &= u_{x+1,y-1} + u_{x-1,y+1} - u_{x-1,y-1} \\
 &+ \Gamma^1_{11}(u_{x+1,y} - u_{x-1,y})(u_{x,y+1} - u_{x,y-1}) \\
 &+ \Gamma^1_{12}(u_{x+1,y} - u_{x-1,y})(v_{x,y+1} - v_{x,y-1}) \\
 &+ \Gamma^1_{12}(v_{x+1,y} - v_{x-1,y})(u_{x,y+1} - u_{x,y-1}) \\
 &+ \Gamma^1_{22}(v_{x+1,y} - v_{x-1,y})(v_{x,y+1} - v_{x,y-1}). \\
 &\tag{7.19}
\end{align*}
\]

\[
\begin{align*}
  v_{x+1,y+1} &= v_{x+1,y-1} + v_{x-1,y+1} - v_{x-1,y-1} \\
 &+ \Gamma^2_{11}(u_{x+1,y} - u_{x-1,y})(u_{x,y+1} - u_{x,y-1}) \\
 &+ \Gamma^2_{12}(u_{x+1,y} - u_{x-1,y})(v_{x,y+1} - v_{x,y-1}) \\
 &+ \Gamma^2_{12}(v_{x+1,y} - v_{x-1,y})(u_{x,y+1} - u_{x,y-1}) \\
 &+ \Gamma^2_{22}(v_{x+1,y} - v_{x-1,y})(v_{x,y+1} - v_{x,y-1}). \\
 &\tag{7.20}
\end{align*}
\]

\( \Gamma^i_{jk} (i, j, k = 1, 2) \) in Equations (7.19) and (7.20) are functions of \( u_{x,y} \) and \( v_{x,y} \). In order to get \( (u_{x+1,y+1}, v_{x+1,y+1}) \) (where \( x > 0 \) and \( y > 0 \)) uniquely, we need initial data along a set of boundary curves \( u = u(x,0), u = u(0,y), v = v(x,0), \) and \( v = v(0,y) \). These curves can be defined explicitly by the given initial paths. However, they are not sufficient to uniquely determine \( (u_{x+1,y+1}, v_{x+1,y+1}) \) in Equations (7.19) and (7.20). For instance, we need additional initial data along a set of curves \( u = u(x,1), u = u(1,y), v = v(x,1), \) and \( v = v(1,y) \), for instance. These curves can be specified either explicitly as in the previous set of curves or indirectly with approximations similar to the techniques we mentioned in Section 5.3.3.
Note that it is not necessary to fix initial paths along the curves \( u = u(x, 0), \ u = u(0, y), \ v = v(x, 0), \) and \( v = v(0, y)\). In fact, they can be defined anywhere in the 2D Tchebychev net, as long as \( U \) curves and \( V \) curves are intersecting.

### 7.2.2 An Algorithm Based on Global Properties of a Tchebychev Net

From Theorems 7.1 and 7.2, we can construct an algorithm that predicts the conformability of fitting to a given surface with a single ply of woven cloth composites.

**Algorithm 7.1 (ROUGH APPLICABILITY TEST)**

1. Calculate the integral Gaussian curvature for a given surface region we want to fit.

2. If the value exceeds \( 2\pi \), then we conclude that we cannot fit to the surface with a single ply of woven cloth composites. If the value is less than \( \pi/2 \), we conclude that we can fit to the surface with a single ply of woven cloth composites, under a certain set of initial conditions.

Although this algorithm is a straightforward consequence obtained from Theorems 7.1 and 7.2, it provides a lamination designer with a useful measure because the algorithm only requires the analysis of a given surface. It should be noted that even if the integral Gaussian curvature value becomes less than \( \pi/2 \) and the initial conditions are specified exactly as mentioned in the paragraph of Theorem 7.2, the thread angles in the solution to the fitting may exceed the allowable maximum \( (\gamma_{\text{max}}) \) or the allowable minimum \( (\gamma_{\text{min}}) \).

**Example:** A simple example is an entire sphere with radius \( R \). Since the Gaussian curvature of a sphere with radius \( R \) is given by:

\[
K \equiv \frac{1}{R^2}
\]
Thus,
\[ \int_{S} K \, dS = K \int_{S} dS = K \times 4\pi R^2 = 4\pi > 2\pi. \]

It follows that we cannot globally fit a ply of woven cloth composites to an entire sphere, no matter what materials are used for the ply modeled by a Tchebychev net.

When the surface is simple enough as in the case of a sphere, we can evaluate the integral Gaussian curvature analytically. Generally speaking, this is not always the case, and we need a numerical approximation to the integral Gaussian curvature of a given surface. Suppose we subdivide a given surface into \( M \times N \) facets in UV parametric space. Figure 7.3 shows a facet enclosed by \( u = u_i, u = u_{i+1}, v = v_j, \) and \( v = v_{j+1}, \) where \( (0 \leq i < M) \) and \( (0 \leq j < N). \) The integral Gaussian curvature over the surface \((I_K)\) is approximated by the following formula:

\[
I_K = \int_{S} K \, dS \approx \sum_{i=0}^{M-1} \sum_{j=0}^{N-1} (K_{1,i,j}^i A_{1,i,j}^i + K_{2,i,j}^i A_{2,i,j}^i), \tag{7.21}
\]

where \( A_{i,j} = A_{1,i,j} + A_{2,i,j} \) denotes the area of the facet, \( K_{1,i,j}^i = (K_{i,j} + K_{i,j+1} + K_{i+1,j+1})/3 \) and \( K_{2,i,j}^i = (K_{i,j} + K_{i+1,j} + K_{i+1,j+1})/3 \) denote the average Gaussian curvatures that correspond to the areas \( A_{1,i,j} \) and \( A_{2,i,j}, \) respectively. As \( M \) and \( N \) become bigger, the approximation becomes more accurate.

The integral Gaussian curvature over an entire surface region provides an “intrinsic” quantity. In other words, it does not depend upon either the surface coordinate system or the Tchebychev net coordinate system.

On the other hand, as we have observed in Chapter 5, the fitting result depends heavily upon the initial conditions including the base path and sweeping directions. In order to capture this phenomenon, we define an intermediate Gaussian curvature integral \((I_{K_{i,j}})\) over a surface region as follows:

\[
I_{K_{i,j}} = \int_{x_0}^{x_1} \int_{y_0}^{y_j} K_{x,y} \, dx \, dy, \tag{7.22}
\]
The total integral Gaussian curvature of a surface is approximated by summing up the integral Gaussian curvature of the entire facets.

where the starting point is specified at \((x_0, y_0)\) in the Tchebychev net coordinate system. Clearly, \(I_{K_{i,j}}\) in Equation (7.22) depends upon how we have swept the ply so far from the starting point \((x_0, y_0)\) to the current point \((x_i, y_j)\) on the surface region.

In particular, if the base path is aligned with \(X = 0\) thread (or alternatively \(Y = 0\) thread), and the sweeping direction is in the positive \(Y\) direction (or in the positive \(X\) direction), \(I_{K_{i,j}}\) is approximated by the following formula:

\[
I_{K_{i,j}} \approx \sum_{p=1}^{i} \sum_{q=1}^{j} K_{p,q} \Delta x \Delta y, \tag{7.23}
\]

where \(\Delta x\) and \(\Delta y\) represent the distances between adjacent wefts and warps, respectively, and are constants due to the inextensibility for the threads of a Tchebychev net. From Equation (7.23), we can derive an interesting recursion formula regarding the intermediate Gaussian curvature integrals as follows:

\[
I_{K_{i,j}} = I_{K_{i-1,j}} + I_{K_{i,j-1}} - I_{K_{i-1,j-1}} + K_{i,j} \Delta x \Delta y, \tag{7.24}
\]

where \(I_{K_{0,j}} = I_{K_{i,0}} = 0\). This can be easily proved by induction on \(i\) and \(j\) for \(i \geq 1\).
and \( j \geq 1 \).

**Example:** Let \( \Delta x = \Delta y = d \). Consider a quadrant of a sphere as shown in Figure 7.4. Then, since the Gaussian curvature of a sphere is constant, \( I_{K_{m,n}} \) is represented by a closed formula as follows:

\[
I_{K_{m,n}} = mn\left(\frac{d}{R}\right)^2.
\]

Obviously, this is a monotonically increasing function with \( m \) and \( n \). From Equation (7.11) and notational conventions used in Figure 7.2, we have

\[
I_{K_{m,n}} = 2\pi - \sum_{i=1}^{4} \alpha_i
\]

\[
= 2\pi - \left(\frac{3}{2}\pi + \alpha_3\right)
\]

\[
= mn\left(\frac{d}{R}\right)^2.
\]

Hence, \( \alpha_3 = \frac{\pi}{2} - mn\left(\frac{d}{R}\right)^2 \). In order for \( \alpha_3 \) to be positive,

\[
mn < \frac{\pi}{2}\left(\frac{R}{d}\right)^2.
\]

In particular, if \( m = n, R = 10, \) and \( d = 0.7 \) as in Figure 7.4,

\[
m < \sqrt{\frac{\pi}{2}}\frac{100}{7} \approx 17.9.
\]

In other words, if the mesh size \( m \geq 18 \), we can predict that the thread angle at a mesh point \((m,m)\) becomes negative, and there is no chance to fit to the surface without wrinkling and tearing. The mesh size used in Figure 7.4 is \( 17 \times 17 \), and \( \alpha_3 \) at \((17,17)\) is evaluated as \( \frac{\pi}{2} - (17)^2\left(\frac{7}{100}\right)^2 \approx 0.1547 \) (radian).

In general, we have an arbitrary starting point, an arbitrary base path, and arbitrary sweeping directions. These initial conditions are specified in both 2D and 3D spaces. In order to cope with general initial conditions, we have developed the following algorithm which approximates the intermediate Gaussian curvature integral at an arbitrary mesh point.
Figure 7.4: Prediction of the thread angle $\alpha_3$ in a fitting to a quadrant of a sphere with radius $R = 10$ and the thread distance $d = 0.7$. The starting point is assumed to be at the north pole of the sphere.
Algorithm 7.2 (COMPUTATION OF INTERMEDIATE GAUSSIAN CURVATURE INTEGRAL)

Function TRACE(G,L): a list of pointers to a Grid;
if G in L then return NIL;
L = L ∪ G;
if G has two known adjacent mesh points G₁ and G₂ then
   L = L ∪ TRACE(G₁,L) ∪ TRACE(G₂,L)
else if G has one known adjacent mesh point G₁ then
   L = L ∪ TRACE(G₁,L)
endif
return L;

Procedure INT-GAUSS(G,dx,dy);
u = G→mapped_UV_pos.u;
v = G→mapped_UV_pos.v;
L = TRACE(G, φ);
G→gauss = Gaussian_curvature(u,v);
for m=1 to M do
   for n=1 to N do
      if _Grid[m][n] in L then
         G→int_gauss += G[m][n]→gauss * dx * dy;
      endif
   endfor
endfor

Note that, in the above algorithm, G represents a pointer to a Grid data structure, _Grid represents a global variable that is a pointer to a 2D array of Grid data structures, L represents a pointer to the function returning a list of pointers to a Grid data structure, and (dx,dy) represent distances between adjacent wefts and warps, respectively. A Grid data structure refers to the data structure listed in Appendix C. The function TRACE(G,L) is the same type as L. The output of Procedure INT-GAUSS is the intermediate Gaussian curvature integral that is set to the variable “int_gauss” pointed to by G. The value of the function Gaussian_curvature(u,v) is calculated by Equation (7.25). The key idea of the algorithm is to approximate the intermediate Gaussian curvature integral at a mesh point by summing up Gaussian curvature integrals defined at all the discrete mesh points that have been swept so
far, up to the current mesh point of interest. Let us suppose \( L(G) \) represents a list of all the mesh points that have been swept until the current mesh point pointed to by \( G \). Then, \( L(G) \) is calculated as follows: if two adjacent neighbors \( G_1 \) and \( G_2 \) are already known, perform a set union of \( L(G_1), L(G_2) \) and \( G \); if only one neighbor \( G_1 \) is known, perform a set union of \( L(G_1) \) and \( G \). The full algorithm is listed in Appendix D.

The intermediate Gaussian curvature integral computed by the above algorithm offers a useful tool to predict anomalous events for a fitting. After a fitting to a surface region is done, we can plot the absolute values of the intermediate Gaussian curvature integrals as a height field above the 2D plane development pattern. Then, the ply area including the mesh points with large intermediate Gaussian curvature integral values is generally susceptible to anomalous events.

Example 1: Figure 7.5 shows an example of the plot of intermediate Gaussian curvatures for the fitting examples to an octant of a sphere described in Section 5.7.1. It is clear that case (c) has the largest value for \( |I_{K_{i,j}}| = 1.42 \) at mesh point \((18, 30)\). Therefore, we can predict that an anomalous event may occur around there in case (c). On the other hand, case (b) has the largest \( |I_{K_{i,j}}| \) value 0.44, which is the smallest among the three cases (a), (b), and (c). Thus, we can reach the same conclusion as in Section 5.7.1 that case (b) is the best fitting among the three.

Example 2: Figure 7.6 shows the plot of intermediate Gaussian curvatures for the fitting examples to a surface of revolution described in Section 5.7.3. The results conform to those in Table 5.5. In other words, the ascending order of the value of shear deformation energy approximated by \( \sum_{i,j} (\cos \gamma_{i,j})^2 \) in Table 5.5 is same as the descending order of the largest value of intermediate Gaussian curvature integral \( |I_{K_{i,j}}| \) in Figure 7.6.

Remark: The intermediate Gaussian curvature integral should be employed as an indicator of anomalous events only if the base path is aligned with a “geodesic”; otherwise we cannot keep the thread angle constant along the base path (due to
Lemma 7.2), yielding unwanted shear deformations even if the surface is a developable surface (See Figures 7.8 and 7.9).

7.2.3 An Algorithm for Finding a Good Initial Path

From the discussions in Section 6.1, anomalous events such as wrinkling and tearing may occur when the thread angle becomes less than the allowable minimum ($\gamma_{\text{min}}$) or when it becomes greater than the allowable maximum ($\gamma_{\text{max}}$). These situations are very sensitive to the initial conditions of the fitting process. Good initial paths may greatly decrease the possibility of anomalous events.

Lemmas 7.1 and 7.2 provide us with the idea of an algorithm for finding a good initial path that produces an optimal fitting result, given a 3D surface shape. Thanks to the Lemma 7.1, if we start fitting at a point on the surface where the absolute of the Gaussian curvature is the largest, we can maximize the minimum thread angle or minimize the maximum thread angle in the fitting. This is because no location other than the starting point has a larger Gaussian curvature, and thus the increase or decrease of the thread angle may be less than the case in which we started fitting at other points. Similarly, thanks to the Lemma 7.2, if we align the initial path (e.g. base path) with a geodesic on the surface, we can keep the angular
Figure 7.6: Plots of the intermediate Gaussian curvature integrals projected above the plane developments for each different fitting of Figure 5.31.
relationship between wefts and warps along the path. By taking advantage of these Lemmas, we can potentially suppress the accumulation of the strain energy caused by shear deformation, if we judiciously set up the starting point and the initial base path. In addition, it is generally true that the starting point should be located around the center of the surface region if there is no unique point which has the maximum Gaussian curvature.

These considerations lead to an algorithm that provides a reasonably good initial path that has less probability of anomalous events, potentially suppressing the accumulation of deformation energy.

**Algorithm 7.3** (ALGORITHM FOR PROVIDING A GOOD INITIAL PATH FOR A FITTING)

1. Find a point in a given surface region where the absolute of the Gaussian curvature ($|K|$) is maximum. If there are two or more points with the same maximum, choose the one that is closer to the center of the surface region.

2. Calculate principal directions there and align the initial direction of the base path with one of the principal directions corresponding to $\kappa_n = \kappa_{\text{max}}$, where $\kappa_n$ denotes the normal curvature, and $\kappa_{\text{max}}$ denotes the maximum value of $\kappa_n$ at the point.

3. Set up the base path such that it is kept along a geodesic on the surface region, having the initial direction specified by $\kappa_n = \kappa_{\text{max}}$.

In Step 1, note that in order to determine the point closest to the center of a surface region, we need a certain measure of the distance. Mathematically, the distance between a point of interest in a surface region and a point on the boundary is defined as the arc length along the geodesic passing through the two points. However, it is not necessary to find the exact point closest to the center of a surface region since candidate points themselves are selected by “sampling” (or “approximation”). Assuming the surface region is convex in a bivariate parametric space of
the surface, we may select the point closest to the center by checking the distance between \((u, v)\) and \((\frac{u_{min} + u_{max}}{2}, \frac{v_{min} + v_{max}}{2})\), where \((u, v)\) is the coordinate of the point of interest and \((u_{min}, v_{min})\) and \((u_{max}, v_{max})\) are the minimum and maximum parametric values for the surface. In the following, we elaborate on computations of a Gaussian curvature, principal directions, and a geodesic.

### 7.2.3.1 Computation of a Gaussian Curvature

A Gaussian curvature \(K\) is calculated by the following equation.

\[
K \equiv \frac{L N - M^2}{E G - F^2} = \frac{(r_{uu} \cdot n)(r_{vv} \cdot n) - (r_{uv} \cdot n)^2}{|r_u \otimes r_v|^2},
\]

where \(L = r_{uu} \cdot n\), \(M = r_{uv} \cdot n\), \(N = r_{vv} \cdot n\), and \(n\) is the unit surface normal at the point.

Rigorously, the point in a surface region where \(|K|\) is maximum can be found at local extreme points inside the region satisfying the following conditions [19]:

\[
\begin{align*}
\frac{\partial K}{\partial u} &= \frac{\partial K}{\partial v} = 0 \\
\frac{\partial^2 K}{\partial u^2} \frac{\partial^2 K}{\partial v^2} - \frac{\partial^2 K}{\partial u \partial v} &< 0
\end{align*}
\]

There is a possibility that the above point is a purely local maximum and not a global maximum. In this case, the point where \(|K|\) is maximum should be located on the boundary.

A much simpler method, albeit an approximation, is to calculate the Gaussian curvatures at sample points and find the largest among them as follows:

\[
\begin{align*}
u_i &= u_{min} + \Delta h_u \times i \\
v_j &= v_{min} + \Delta h_v \times j
\end{align*}
\]

where \(\Delta h_u = \frac{u_{max} - u_{min}}{M}\), \(\Delta h_v = \frac{v_{max} - v_{min}}{N}\), \(0 \leq i < M\), and \(0 \leq j < N\).
7.2.3.2 Computation of a Principal Direction

Principal directions \((du : dv)\) required in Step 2 can be determined by solving the following differential equations.

\[
\begin{vmatrix}
Edu + Fdv \\ Ldu + Mdv
\end{vmatrix}
\begin{vmatrix}
Fdu + Gdv \\ Mdu + Ndv
\end{vmatrix} = 0 \quad (7.26)
\]

This is rewritten as a pair of differential equations referred to as Rodrigues’s formula.

\[
\begin{align*}
(L - \kappa E)du + (M - \kappa F)dv &= 0 \\
(M - \kappa F)du + (N - \kappa G)dv &= 0
\end{align*}
\]

\(\begin{array}{c}
\text{(7.27)}
\end{array}\)

where \(\kappa\) is an arbitrary variable.

The above equations are transformed into a quadratic formula of variable \(\kappa\) and its two real roots \((\kappa_{\text{max}} \text{ and } \kappa_{\text{min}})\) are called principal curvatures at the point on the surface. By solving it, we obtain

\[
\begin{align*}
\kappa_{\text{max}} &= H + \sqrt{H^2 - K} \\
\kappa_{\text{min}} &= H - \sqrt{H^2 - K}
\end{align*}
\]

\(\begin{array}{c}
\text{(7.28)}
\end{array}\)

where \(H(\equiv (\kappa_{\text{max}} + \kappa_{\text{min}})/2)\) is the mean curvature and \(K(\equiv \kappa_{\text{max}}\kappa_{\text{min}})\) is the Gaussian curvature.

From the first equation of (7.27), we obtain a principal direction, corresponding to \(\kappa = \kappa_{\text{max}}\) as follows:

\[
\begin{align*}
\frac{du}{dv} &= (M - \kappa_{\text{max}}F)/D \\
\frac{dv}{du} &= -(L - \kappa_{\text{max}}E)/D
\end{align*}
\]

\(\begin{array}{c}
\text{(7.29)}
\end{array}\)

where \(D = \sqrt{(L - \kappa_{\text{max}}E)^2 + (M - \kappa_{\text{max}}F)^2}\). We can obtain the same result from the second equation of (7.27) as well.

The important thing is that, except at an umbilic point, the principal direction given by the above formula is uniquely determined. At an umbilic point, an arbitrary direction can be specified because the normal curvature at the point is the same in every direction.
7.2.3.3 Computation of a Geodesic

A necessary and sufficient condition that a curve \( r(s) = (u(s), v(s)) \) on a surface \( r(u, v) \) is a geodesic is that it is defined as the solution to the following system of differential equations [46, 101].

\[
\begin{align*}
\frac{d^2 u}{ds^2} + \Gamma^1_{11}(\frac{du}{ds})^2 + 2\Gamma^1_{12}\frac{du}{ds}\frac{dv}{ds} + \Gamma^1_{22}(\frac{dv}{ds})^2 &= 0 \\
\frac{d^2 v}{ds^2} + \Gamma^2_{11}(\frac{du}{ds})^2 + 2\Gamma^2_{12}\frac{du}{ds}\frac{dv}{ds} + \Gamma^2_{22}(\frac{dv}{ds})^2 &= 0
\end{align*}
\]  

(7.30)

where \( \Gamma^i_{jk}(i, j, k = 1, 2) \) is the Christoffel’s symbol of the second kind.

Equation (7.30) is rewritten into the following system of linear differential equations with four variables \((u, v, u', v')\) [30]:

\[
\begin{align*}
\frac{du}{ds} &= u' \\
\frac{dv}{ds} &= v' \\
\frac{du'}{ds} &= -\Gamma^1_{11}u'^2 - 2\Gamma^1_{12}u'v' - \Gamma^1_{22}v'^2 \\
\frac{dv'}{ds} &= -\Gamma^2_{11}u'^2 - 2\Gamma^2_{12}u'v' - \Gamma^2_{22}v'^2
\end{align*}
\]  

(7.31)

Once the starting point \((u_0, v_0)\) and the initial direction \((\frac{du}{ds}, \frac{dv}{ds})\) are given, a sequence of points \((u, v)\) and \((u', v')\) on the geodesic can be obtained from Equation (7.31) by means of a fourth-order Runge-Kutta method [20].

Let \( s \) represent an arc length parameter and \( h \) represent a step size. Equation (7.31) can then be put into the following:

\[
\begin{align*}
\frac{du}{ds} &= u' \\
\frac{dv}{ds} &= v' \\
\frac{du'}{ds} &= F_1(s, u, v, u', v') \\
\frac{dv'}{ds} &= F_2(s, u, v, u', v')
\end{align*}
\]

where \( F_1 \) and \( F_2 \) denote the right hand side of corresponding equations in (7.31). The Runge-Kutta step is specified with variables \( k^i_j(1 \leq i, j \leq 4) \) computed in the following sequences:
\[
\begin{align*}
\{ & k_1^1 = hu' \\
& k_2^1 = hv' \\
& k_3^1 = hF_1(s, u, v, u', v') \\
& k_4^1 = hF_2(s, u, v, u', v') \}
\end{align*}
\]

\[
\begin{align*}
\{ & k_1^2 = h(u' + k_3^1/2) \\
& k_2^2 = h(v' + k_4^1/2) \\
& k_3^2 = hF_1(s + h/2, u + k_1^1/2, v + k_2^1/2, u' + k_3^1/2, v' + k_4^1/2) \\
& k_4^2 = hF_2(s + h/2, u + k_1^1/2, v + k_2^1/2, u' + k_3^1/2, v' + k_4^1/2) \}
\end{align*}
\]

\[
\begin{align*}
\{ & k_1^3 = h(u' + k_3^2/2) \\
& k_2^3 = h(v' + k_4^2/2) \\
& k_3^3 = hF_1(s + h/2, u + k_1^2/2, v + k_2^2/2, u' + k_3^2/2, v' + k_4^2/2) \\
& k_4^3 = hF_2(s + h/2, u + k_1^2/2, v + k_2^2/2, u' + k_3^2/2, v' + k_4^2/2) \}
\end{align*}
\]

\[
\begin{align*}
\{ & k_1^4 = h(u' + k_3^3) \\
& k_2^4 = h(v' + k_4^3) \\
& k_3^4 = hF_1(s + h, u + k_1^3, v + k_2^3, u' + k_3^3, v' + k_4^3) \\
& k_4^4 = hF_2(s + h, u + k_1^3, v + k_2^3, u' + k_3^3, v' + k_4^3) \}
\end{align*}
\]

It is clear from the above equation that four evaluations for each of the functions \( F_1 \) and \( F_2 \) are required. At each iteration, variables \( s, u, v, u', \) and \( v' \) are incremented as follows:

\[
\begin{align*}
s &= s + h \\
\{ & u = u + (k_1^1 + 2k_1^2 + 2k_1^3 + k_1^4)/6 \\
& v = v + (k_2^1 + 2k_2^2 + 2k_2^3 + k_2^4)/6 \}
\end{align*}
\]

\[
\begin{align*}
u' &= u' + (k_3^1 + 2k_3^2 + 2k_3^3 + k_3^4)/6 \\
v' &= v' + (k_4^1 + 2k_4^2 + 2k_4^3 + k_4^4)/6
\end{align*}
\]

The error term of Equation (7.32) is \( O(h^5) \) because of the nature of the fourth-order Runge-Kutta method [20].
This iteration is repeated until the new value \((u, v)\) extends the surface boundary. There are several things that we have to take into consideration when computing a geodesic with the above formulae. First, in applying the above Runge-Kutta method, a step size \(h\) must be properly chosen to produce enough accuracy. There are many approaches in the literature (e.g. [20, 43]) for choosing a step size adaptively in order to guarantee precise results. As long as the given surface is sufficiently smooth, a smaller step size normally makes the results more accurate.

More importantly, it should be noted that the step size \(h\) represents the distance in UV parametric space, not the distance in XYZ Cartesian space. Let us assume that we want to obtain a point on the geodesic with a specified distance \(d\) from a given point in XYZ space. In general, setting \(h = d\) does not produce the desired point because the distance in UV space depends upon the parametrization of the surface. To cope with this problem, we have developed a method, described below, assuming that \(\min(d_k) > h\) for every \(0 \leq k \leq M\), where \(\min(d_k)\) is a function returning the minimum value of \(d_k\).

Let \(x[i](0 \leq i \leq N)\) denote a sequence of points generated by Runge-Kutta method, assuming a constant step size \(h\). Let \(w[k](0 \leq k \leq M)\) denote a sequence of desired points and \(d_k(1 \leq k \leq M)\) denote a sequence of distances between \(w[k-1]\)
and $w[k]$ in XYZ space. These variables are illustrated in Figure 7.7. The algorithm for obtaining $w[k]$ is written in the following pseudocode:

**Algorithm 7.4** (COMPUTATION OF POINTS ON A GEODESIC)

```plaintext
procedure GENERATE-GEODESIC-POINTS(M,N,w)
    Calculate $x[i](0 \leq i \leq N)$ with Equation (7.32);
    $k = 0$;
    $i = 0$;
    while $k \leq M$ do
        $full = FALSE$;
        $d = 0$;
        while $(i < N)$ and (not full) do
            $t = |x[i+1] - x[i]|$;
            if $d + t > d_k$ then
                $s_1 = d_k - d$;
                $s_2 = d + t - d_k$;
                $w[k] = (s_1x[i+1] + s_2x[i])/t$;
                $full = TRUE$
            else
                $d = d + t$;
                $i = i + 1$
            endif
        endwhile
        $k = k + 1$
    endwhile
```

Once $(u, v)$ value is obtained by Equation (7.32), $x[i](0 \leq i \leq N)$ can be computed from $x[i] = r(u, v)$. The dotted lines in Figure 7.7 represent the first four mappings between $w[k](0 \leq k \leq 3)$ and the corresponding positions in UV space. Generally, $w[k]$ lies between $x[i]$ and $x[i+1]$ for a certain value $i$. The thrust of Algorithm 7.4 is to approximate the mapping of $w[k]$ with a linear interpolation between $x[i]$ and $x[i+1]$. The complexity of the algorithm is $O(MN)$. However, since we have assumed that $\min(d_k) > h$ for every $0 \leq k \leq M$, it follows that $M$ is usually much smaller than $N$. Thus, the constant in the expression $O(MN) \approx O(N^2)$ is usually very small, and the algorithm runs fast.

Finally, due to the approximation employed in the previous problem, it is desirable that each point represented in XYZ space should be examined to see if it
is sufficiently accurate. This can be done by checking if the calculated points satisfy the following equation:

$$\kappa_g = |\mathbf{n} \times \mathbf{\ddot{x}}| = 0,$$

(7.33)

where $\mathbf{n}$ denotes a surface normal at the point, $\mathbf{\dot{x}}$ denotes a derivative of the curve with respect to an arc length parameter $s$, and $\mathbf{\ddot{x}}$ denotes the second derivative. Equation (7.33) gives another formula for a geodesic, expressed by variables in XYZ Cartesian space.

**Example 1**: Figure 7.8 shows two fittings to a half cone. Figure 7.8 (a) has its base path on plane $Z = 0$ (i.e. a planar curve), which is clearly not a geodesic. Even though a cone is a developable surface, the fitting result produces shear deformations due to Lemma 7.2. On the other hand, Figure 7.8 (b) has its base path along a geodesic passing through two end points $(0, -R, 0)$ and $(0, R, 0)$, where $R$ denotes a radius of the circle on $Z = 0$ plane. Since the thread angles along the geodesic path are kept constant, there is no shear deformation in the fitting result in Figure 7.8 (b). This can be verified by plotting thread angles between wefts and warps at mesh points projected above the plane developments as shown in Figure 7.9.

In this example, we can calculate the explicit formula of the geodesic. As shown in Figure 7.10 (a), a half cone is parametrically represented as

$$
\begin{bmatrix}
x \\
y \\
z
\end{bmatrix}
= 
\begin{bmatrix}
u \sin \alpha \cos v \\
u \sin \alpha \sin v \\
\cot \alpha (R - u \sin \alpha)
\end{bmatrix},
$$

(7.34)

where $(0 \leq u \leq \frac{R}{\sin \alpha})$ and $(-\frac{\pi}{2} \leq v \leq \frac{\pi}{2})$. Figure 7.10 (b) represents the plane development pattern of the half cone, where $P_1$ and $P_2$ denote the two end points in the 2D plane development pattern. Note that the geodesic passing through the two end points $P_1$ and $P_2$ becomes a straight line. In this 2D space, the coordinates of $P_1$ and $P_2$ are represented by $P_1 = (\frac{R}{\sin \alpha}, 0)$ and $P_2 = (\frac{R \cos(\pi \sin \alpha)}{\sin \alpha}, \frac{R \sin(\pi \sin \alpha)}{\sin \alpha})$. 
Figure 7.8: Two fittings to a half cone; the fitting result of (a) produces shear deformations because the base path is not a geodesic, and (b) produces no shear deformation because the path is a geodesic.

Figure 7.9: Plots of thread angles at mesh points projected above the associated plane developments, corresponding to Figure 7.8 (a) and (b), respectively.
Figure 7.10: (a) A \((u, v)\) parametric representation of a cone, where \(R\) is the radius of a circle on \(Z = 0\) plane, and (b) the 2D plane development pattern of a cone, where \(0 \leq \theta \leq \pi \sin \alpha\).

Therefore, the geodesic passing through \(P_1\) and \(P_2\) is represented by the following equation:

\[
Y = \frac{\sin(\pi \sin \alpha)}{\cos(\pi \sin \alpha) - 1} (X - \frac{R}{\sin \alpha}). \tag{7.35}
\]

It should be noted that the geodesic curve represented by Equation (7.35) is not a planar curve in 3D space. Since \((X, Y)\) is represented as

\[
\begin{bmatrix}
X \\
Y
\end{bmatrix} = \begin{bmatrix}
u \cos \theta \\
u \sin \theta
\end{bmatrix} = \begin{bmatrix}
u \cos((\frac{\pi}{2} + v) \sin \alpha) \\
u \sin((\frac{\pi}{2} + v) \sin \alpha)
\end{bmatrix},
\]

Equation (7.35) may be written as

\[
u \sin((\frac{\pi}{2} + v) \sin \alpha) = \frac{\sin(\pi \sin \alpha)}{\cos(\pi \sin \alpha) - 1} (u \cos((\frac{\pi}{2} + v) \sin \alpha) - \frac{R}{\sin \alpha}).
\]

With some arithmetic, this equation may be put into the following:

\[
u = \frac{R \sin \tau}{\sin \alpha (\sin \tau \cos \beta + \sin \beta (1 - \cos \tau))^3},
\]

where \(\beta = (\frac{\pi}{2} + v) \sin \alpha\) and \(\tau = \pi \sin \alpha\). This is a parametric representation of the geodesic curve passing through \(P_1\) and \(P_2\). By varying \(v\) from \(-\frac{\pi}{2}\) to \(\frac{\pi}{2}\), the corresponding \(u\) values are obtained, and then by assigning \((u, v)\) values into
Equation (7.34), \((x, y, z)\) values on the geodesic are obtained. Note that this is an exceptional case, since a geodesic is usually obtained only numerically from Equation (7.32).

Example 2: The second example is an object with a convex corner shown in Figure 6.18. In order to apply Algorithm 7.3 to this example, the starting point must be chosen where the Gaussian curvature is maximum in the given surface region. In this example, the only surface region where the Gaussian curvature is non-zero is at the rounded corner itself, which is modeled by an octant of a sphere. We can therefore choose the starting point in the center of the octant of a sphere by conforming to Step 1 in Algorithm 7.3. Next the base path must be aligned with the geodesic passing through the starting point and having the initial direction in the principal direction. Since every point on the sphere is an umbilic point, the principal direction cannot be obtained uniquely. This does not cause problems, because any direction can be chosen as the initial direction for the geodesic, based on the fact that an umbilic point has the same normal curvature (i.e. \(\kappa_{\text{max}} = \kappa_{\text{min}}\)) in any direction.

Figure 7.11 (a) shows the fitting result with the initial conditions mentioned above. Figure 7.11 (b) is the plot of thread angles between wefts and warps at mesh points projected above the associated plane development, corresponding to the fitting in Figure 7.11 (a). The minimum thread angles observed in this example is 1.12 radian, which is approximately 64 degrees, shown in the ellipses in Figure 7.11 (b). Therefore, if the locking angle is less than 64 degrees, no anomalous event caused by shear deformation is expected. Figure 7.12 demonstrates the result with colors based on the thread angle values. It is generally true that finding a good initial path has more dramatic effects than inserting darts in order to reduce excessive shear deformation. Compare Figure 7.12 with Figures 6.22 and 6.25. It should be noted that there is no guarantee that we can always find a better initial path. In addition, the darting techniques mentioned in the previous chapter are still very
Figure 7.11: (a) Fitting to a cornered object with better initial conditions, and (b) the distribution of thread angles at mesh points projected above the plane development of (a).

effective when the base path is fixed as a guide line in practical applications.

Figure 7.13 shows two results of fitting actual woven cloth fabrics (fiberglass) to an object with a convex corner. The result on the left has the same initial conditions with the fitting result shown in Figure 6.19, and the result on the right has the same initial conditions with the fitting result in Figure 7.11 (a). The red thread attached to each result represents the base path. As we predicted in computer simulation, the result on the left exhibits “wrinkling” around the corner, while the result on the right exhibits no wrinkling.

Figure 7.14 is the graph of minimum thread angles ($\gamma_{\text{min}}$) by varying the distance between adjacent crossings for the fittings under the initial conditions corresponding to the Figure 7.11 (a). There is a tendency shown in the graph that the minimum thread angles ($\gamma_{\text{min}}$) converges to a certain value as the distance between adjacent crossing decreases. In Figure 7.13, the black circle in the graph represents the actual minimum thread angle measured in the fitting result on the right.
Figure 7.12: Fitting results with colors based on thread angle values: (a) 3D fitting with better initial conditions, and (b) the plane development pattern.
Figure 7.13: Results of fitting actual woven cloth fabrics (fiberglass) onto an object with a convex corner: the result on the left has its initial conditions corresponding to Figure 6.19 and the result on the right has its initial conditions corresponding to Figure 7.11 (a).
Figure 7.14: The graph of minimum thread angles ($\gamma_{\text{min}}$) for the fittings to a cornered object with the better initial conditions by varying the distance between adjacent crossing.
Example 3: The third example shown in Figure 7.15 is a free surface, where a number of geodesics, which pass through a point on the surface whose Gaussian curvature is maximum, are drawn. Figure 7.16 demonstrates the distribution of the Gaussian curvatures on the surface with colors. Specifically, red colors are assigned to the surface area where the Gaussian curvature is equal to or greater than 0.02, blue colors are assigned to the surface area where the Gaussian curvature is equal to or less than -0.02, green colors are assigned to the surface area where the Gaussian curvature is zero, and intermediate colors are assigned to the surface area whose Gaussian curvature is either between 0.02 and zero, or -0.02 and zero. In this example, we compare several different fitting results generated with different initial conditions. Specifically, the following four cases for initial conditions are compared:

1. The starting point is at $K = |K|_{max}$, the base path is aligned with a geodesic having the principal direction as the initial direction. (This exactly conforms to Algorithm 7.3.)

2. The starting point is not at $K = |K|_{max}$, but at the center of the surface patch, and the base path is aligned with a geodesic having the principal direction as the initial direction.

3. The starting point is at $K = |K|_{max}$, the base path is aligned with a geodesic, but the initial direction of the geodesic is not aligned with the principal direction.

4. The starting point is at $K = |K|_{max}$, but the base path is not aligned with a geodesic. Instead, the base path is defined as an intersection between the surface and a plane. (i.e. The base path is a planar curve.)

The fitting results and the associated 2D plane development patterns with colors are shown in Figures 7.17 and 7.18, respectively. Compared to the other cases, case (a) produces the best fitting result since there is no excessive shear deformation with red or blue colors. Therefore, case (a) has the least possibility of anomalous events.
Figure 7.15: A free surface with several geodesics passing through the point where the Gaussian curvature is maximum in the surface. One of the geodesics, denoted by $g^*$, is generated by aligning the initial direction with a principal direction at the point.
Figure 7.16: Gaussian curvatures with colors for the free surface shown in Figure 7.15.
Figure 7.17: 3D fitting results with four different initial conditions: (a) the starting point is at $K = |K_{\text{max}}|$, the base path is aligned with a geodesic having the initial direction as the principal direction at the point; (b) same as (a) except that the starting point is not at $K = |K_{\text{max}}|$, but at the center of the surface; (c) same as (a) except that the base path has the initial direction not in the principal direction; (d) same as (a) except that the base path is not a geodesic, but a planar curve.
Figure 7.18: 2D plane development patterns with four different initial conditions: (a) the starting point is at $K = |K_{max}|$, the base path is aligned with a geodesic having the initial direction as the principal direction at the point; (b) same as (a) except that the starting point is not at $K = |K_{max}|$, but at the center of the surface; (c) same as (a) except that the base path has the initial direction not in the principal direction; (d) same as (a) except that the base path is not a geodesic, but a planar curve.
Remark: Algorithm 7.3 works well in most cases as exemplified in the above three examples. The choice of the starting point where the magnitude of the Gaussian curvature is maximum, however, may not always produce the best fitting result. For instance, if the Gaussian curvatures of the four peaks in the last example (Refer to Figure 7.16) are nearly equal, we may not be able to find a single starting point. Choosing the base path as a geodesic, on the other hand, is proved to be a good choice since it guarantees that no shear deformation will result on the base path (See Lemma 7.2).

7.3 Algorithms for Shape Optimization

The integral Gaussian curvature, $I_K$, defined in Equation (7.21) provides a measure for global conformability of a woven cloth composite ply to a given surface. For example, if $|I_K|$ is less than $\frac{\pi}{2}$, we can globally clothe the surface due to Samelson’s theorem (Theorem 7.2), and if $|I_K|$ is greater than $2\pi$, we cannot globally clothe the surface due to Hazzidakis’ theorem (Theorem 7.1). Given two different surfaces $S_1$ and $S_2$ with their $|I_K|$ values denoted by $|I_{K_1}|$ and $|I_{K_2}|$, let us call the shape $S_1$ simpler with respect to global conformability if $|I_{K_1}| < |I_{K_2}|$. This is because the excessive shear deformation that leads to anomalous events is less likely to occur to the surface with a smaller $|I_K|$ value. In particular, if $I_K = 0$ or the surface is a developable surface, no shear deformation will result. Our aim of this section is to investigate the shape modification such that the new shape has a smaller $|I_K|$ value. Note that here we implicitly assume that there are some “tolerances” for modifying the original shape either globally or locally.

In the following, we first present requirements of the shape modification, then describe techniques that meet the requirements. The requirements are stated as follows:

1. The integral Gaussian curvature of the modified shape should become less than the original value.
2. The deviation of the modified shape from the original one should be kept minimum.

Let $I_{K'}$ denote the integral Gaussian curvature of the modified shape. Then, the first requirement is expressed as

$$|I_{K'}| < |I_K|.$$ 

Similarly, let $r'(u, v)$ denote a parametric representation of the modified shape. Then, the second requirement is expressed as minimizing the following value:

$$\int_u \int_v \|r(u, v) - r'(u, v)\|^2 dudv.$$

### 7.3.1 Historical Background

Before describing our methods, let us briefly review previous methods relevant to the shape modification in terms of intrinsic geometric quantities such as the Gaussian curvature. The relationship between a local surface shape and the Gaussian curvature has been well-known in differential geometry [27, 53, 65, 101]. In short, there is a relationship between a point of a surface and the Gaussian curvature ($K \equiv \kappa_{\text{max}}\kappa_{\text{min}}$) as follows: If $K > 0$, the point is called elliptic, if $K < 0$, the point is called hyperbolic, and if $K = 0$, the point is called parabolic. It is further broken down into the following cases, provided that $\kappa_{\text{max}}$ and $\kappa_{\text{min}}$ satisfy the conditions:

1. if $\kappa_{\text{max}} = \kappa_{\text{min}} = 0$, then the point is called planar,
2. if $\kappa_{\text{max}} = \kappa_{\text{min}} \neq 0$, then the point is called umbilic.

Note that $\kappa_{\text{max}}$ and $\kappa_{\text{min}}$ represent principal curvatures defined in Equation (7.28).

The so-called Gauss’ spherical map [27, 53] provides another way of describing the relationship between a surface shape and the intrinsic geometry of the surface. Roughly speaking, the Gauss’ spherical map is a mapping from a point on a surface to the corresponding point on the unit sphere whose center is at the origin. This correspondence is established by calculating the unit surface normal at a point on a surface, and interpreting it as a point on the unit sphere. The Gaussian curvature
is then defined as the ratio of the area of a point on a surface and the corresponding area on the unit sphere. In particular, if the surface is developable, the Gauss’ spherical map becomes a curve on the sphere. Redont [79] utilized this fact to interactively deform developable surfaces by editing the curve on the unit sphere. The Extended Gaussian image, in which the inverse of the Gaussian curvature at a point is mapped to the unit sphere, has also been used for representing the shapes of surfaces [42]. It has a fundamental limitation in that it can only uniquely represent “convex” shapes.

The explicit use of principal curvatures \((\kappa_{\text{max}} \text{ and } \kappa_{\text{min}})\), or equivalently the line of curvatures, provides another tool for controlling the surface shape. This has been extensively utilized in the field of image understanding [17, 53, 57]. Koenderink [53] advocated the use of two parameters \((R, S)\), where \(R\) controls the “curvedness” and \(S\) controls the “shape index” of a surface. These are defined as follows:

\[
\begin{align*}
R &= \sqrt{\frac{(\kappa_{\text{max}}^2 + \kappa_{\text{min}}^2)}{2}} \\
S &= -\frac{2}{\pi} \arctan \left( \frac{\kappa_{\text{max}} + \kappa_{\text{min}}}{\kappa_{\text{max}} - \kappa_{\text{min}}} \right).
\end{align*}
\]

Every distinct shape except for the planar points corresponds to a unique value of the shape index. Roughly speaking, if the value of \(S\) of a surface ranges from -1 to -0.5, then the shape is concave. If it ranges from 0.5 to 1.0, then the shape is convex. Otherwise, the shape is classified as “anticlastic.” On the other hand, as the value of \(R\) becomes smaller, the shape becomes flatter. Similarly, as it becomes larger, the shape becomes more singular.

Liang [56, 57] presented a method of reconstructing the surface shape by using sampling points and their principal curvatures. Once principal curvatures \(\kappa_{\text{max}}\) and \(\kappa_{\text{min}}\) at a point are available, the surface shape is approximated by

\[
z = \frac{\kappa_{\text{max}} x^2 + \kappa_{\text{min}} y^2}{2},
\]

where the local coordinate system \(XYZ\) is defined such that the point is set to be at the origin, the tangent plane at the point corresponds to the \(XY\) axes, and the surface normal corresponds to the \(Z\) axis [53, 65, 99]. Equation (7.36) provides
a local approximation of a small patch around the point. Liang also described a method of extending the approximation to an entire surface and applied the result in matching two developable surfaces.

To our knowledge, however, no previous research work has focused on the relation between the “integral” Gaussian curvature and the surface shape.

### 7.3.2 Modifying a Surface by Editing Principal Curvatures

Here we will focus on the first requirement, or $|I_{K'}| < |I_K|$. A simple idea of modifying a surface shape that meets this requirement is to edit principal curvatures. Initially, we plot $\kappa = \kappa_{\text{max}}(u, v)$ and $\kappa = \kappa_{\text{min}}(u, v)$ surfaces. Then we modify the $\kappa_{\text{max}}(u, v)$ and $\kappa_{\text{min}}(u, v)$ values at sampled $(u, v)$ values repeatedly until $|I_{K'}| < \delta$, where $\delta(< |I_K|)$ is a specific threshold value. Once new $\kappa_{\text{max}}(u, v)$ and $\kappa_{\text{min}}(u, v)$ values are obtained, local surface patches can be reconstructed from Equation (7.36). Finally, the parametric representation of the modified surface $r'(u, v)$ can be obtained by interpolating the sampling points [7]. The advantage of this method is that we can directly control curvatures to meet the requirement $|I_{K'}| < |I_K|$. The fundamental problem of this approach lies in the fact that with mere modification of principal curvatures, the resulting surface shape may be drastically altered. This is primarily because the surface shape is very sensitive to the change of the Gaussian curvature ($\equiv \kappa_{\text{max}}\kappa_{\text{min}}$) [29]. It is, therefore, difficult to meet the second requirement, that of minimizing the deviations. In addition, it may be inefficient because we have to employ Equation (7.36) repeatedly for local approximations of the new shape.

### 7.3.3 Modifying a Surface Shape by Editing Control Points

Since the surface model we have adopted is a NURBS surface, it is prudent to control the parameters that define the NURBS surface. The parameters include “control points”, “knot vectors”, and “weights” of the control points. Editing control points or knot vectors is a straightforward way of modifying the surface shape. Here
we describe the method of modifying a surface shape by editing control points. In order to meet the first requirement, we have to select some of the control points and modify them such that $|I_{K'}| < |I_K|$. This is not a trivial task, except when the surface shape is simple enough for us to easily infer the Gaussian curvature at every point on the surface.

Let us take an example of the surface shape shown in Figure 6.18 and attempt to modify the shape such that the new shape meets the first requirement, under the same initial conditions for the fitting shown in Figure 6.19. In order to decrease $|I_K|$ value of the original surface shape, we need to know the distribution of the Gaussian curvatures over the entire surface. Fortunately, the Gaussian curvatures of the surface are non-zero only in the spherical part at the rounded corner. Since the fitting result shown in Figure 6.19 has the critical region around the spherical part and beneath it, a solution may be to modify the surface shape such that the Gaussian curvature for the new shape, beneath the spherical part, becomes negative. Thus the contributions of the positive Gaussian curvature (around the spherical part) and the negative Gaussian curvature cancel out each other. Figure 7.19 demonstrates the distribution of the Gaussian curvatures of the original shape (Figure 7.19 (a)) and the modified shape (Figure 7.19 (b)), with colors based on the Gaussian curvature values. Color assignments are as shown in Figure 7.16.

Figure 7.20 shows the fitting result for the new shape under the same initial conditions with the example shown in Figure 6.21. It is obvious that the area of the lower critical region has decreased significantly. Figure 7.21 is the plot of thread angles between wefts and warps projected above the 2D plane development. If we compare this plot with Figure 6.20 (b), we can recognize that the new shape no longer causes the shear deformation such that the minimum thread angle becomes zero. Interestingly, if we plot the intermediate Gaussian curvature integrals, $I_{K_{i,j}}$, projected above the plane developments as shown in Figure 7.22 (a) and (b) (corresponding to the fitting of Figure 6.21 and the fitting of Figure 7.20), it becomes clear that as we fit the cloth toward the corner, the $I_{K_{i,j}}$ values for the original surface...
are monotonically increasing. On the other hand, the $I_{K_{i,j}}$ values for the new shape are first increasing, but decrease after the fitting process proceeds into the surface region where the Gaussian curvatures are negative.

Note that the above example does not consider the second requirement (i.e. deviation requirement). Generally, it is quite hard to meet both requirements by editing the control points directly. In the next section, we will provide a better alternative.

### 7.3.4 Modifying a Surface Shape by Finding Optimal Weights

A better method than the previous one is to modify the weights of the control points, while keeping the control points and knot vectors unchanged. This assures us that, as long as the new weights remain positive, the modified shape is within the
Figure 7.20: Fitting results for the new shape with colors assigned, based on the thread angle values: (a) 3D fitting and (b) 2D plane development pattern.
Figure 7.21: Plot of thread angles between wefts and warps at mesh points projected above the plane development, corresponding to the fitting of Figure 7.20.

Figure 7.22: Plots of the intermediate Gaussian curvature integrals (\(\mid I_{K_{i,j}}\mid\)), corresponding to the fittings of Figure 6.21 and Figure 7.20.
convex hull defined by the original control points [29]. Thus, the problem is put into an unconstrained optimization problem as follows: find a set of weights \( \mathbf{w} = \{w_{i,j}\} \) for a given NURBS surface \( \mathbf{r}(u, v) \) that satisfy

\[
G(\mathbf{w}) = |\int_u \int_v K(u, v; \mathbf{w}) \, dudv| - \delta = 0, \tag{7.37}
\]

where \( \delta \) denotes a threshold value, satisfying \( \delta \equiv |I_{K'}| < |I_K| \). Note that the solution to Equation (7.37) only meets the first requirement. Suppose there are \( N \) weights that we want to modify, Equation (7.37) represents a system of \( N \) simultaneous non-linear equations. Since the derivative of the function \( G(\mathbf{w}) \) cannot be obtained easily, we need to use a quasi-Newton method [26, 60] as a method that does not require the derivative of the function in order to solve Equation (7.37). The weights at the \( k \)-th iteration with a quasi-Newton method are represented by the following formula:

\[
w_{i,j}^{(k+1)} = w_{i,j}^{(k)} - \frac{G(w_{i,j}^{(k)})h}{G(w_{i,j}^{(k)} + h) - G(w_{i,j}^{(k)})}, \tag{7.38}
\]

where \( h \) represents a step distance.

The second requirement may be satisfied by adding the constraint that the deviation \( D(\mathbf{w}) \) should be minimum, where

\[
D(\mathbf{w}) = \int_u \int_v |\mathbf{r}(u, v; \mathbf{w}_0) - \mathbf{r}(u, v; \mathbf{w})|^2 \, dudv \quad \tag{7.39}
\]

and \( \mathbf{w}_0 \) denote the original weights for the NURBS surface.

From Equation (7.37) and the constraint given by Equation (7.39), the shape modification problem is translated into a constrained optimization problem [34] as follows: minimize \( D(\mathbf{w}) \) with the constraint that \( G(\mathbf{w}) = 0 \). Let us define a function \( F(\mathbf{w}, \lambda) \) as follows:

\[
F(\mathbf{w}, \lambda) = D(\mathbf{w}) + \lambda G(\mathbf{w}), \tag{7.40}
\]

where \( \lambda \) denotes an arbitrary variable. In order for \( F(\mathbf{w}, \lambda) \) to have a local minimum, its gradient vector must be zero [34]. Hence,

\[
\nabla F(\mathbf{w}, \lambda) = 0. \tag{7.41}
\]
This represents a system of \((N+1)\) non-linear simultaneous equations with \((N+1)\) variables. In particular, if \(w = \{w_1, w_2\}\), then

\[
F(w_1, w_2, \lambda) = D(w_1, w_2) + \lambda G(w_1, w_2)
\]

and Equation (7.41) is put into the following formula:

\[
\begin{bmatrix}
F_1 \\
F_2 \\
F_3
\end{bmatrix} = \begin{bmatrix}
F_{w_1} \\
F_{w_2} \\
F_{\lambda}
\end{bmatrix} = \begin{bmatrix}
D_{w_1}(w_1, w_2) + \lambda I_{w_1}(w_1, w_2) \\
D_{w_2}(w_1, w_2) + \lambda I_{w_2}(w_1, w_2) \\
I(w_1, w_2)
\end{bmatrix} = \begin{bmatrix}
0 \\
0 \\
0
\end{bmatrix}.
\]

Form this, \((w_1^{(n+1)}, w_2^{(n+1)}, \lambda^{(n+1)})\) is obtained iteratively as follows:

\[
\begin{bmatrix}
w_1^{(n+1)} \\
w_2^{(n+1)} \\
\lambda^{(n+1)}
\end{bmatrix} = \begin{bmatrix}
w_1^{(n)} \\
w_2^{(n)} \\
\lambda^{(n)}
\end{bmatrix} - \begin{bmatrix}
\frac{\partial F_1}{\partial w_1} & \frac{\partial F_1}{\partial w_2} & \frac{\partial F_1}{\partial \lambda} \\
\frac{\partial F_2}{\partial w_1} & \frac{\partial F_2}{\partial w_2} & \frac{\partial F_2}{\partial \lambda} \\
\frac{\partial F_3}{\partial w_1} & \frac{\partial F_3}{\partial w_2} & \frac{\partial F_3}{\partial \lambda}
\end{bmatrix}^{-1} \begin{bmatrix}
F_1(w_1^{(n)}, w_2^{(n)}, \lambda^{(n)}) \\
F_2(w_1^{(n)}, w_2^{(n)}, \lambda^{(n)}) \\
F_3(w_1^{(n)}, w_2^{(n)}, \lambda^{(n)})
\end{bmatrix}.
\]

In order to maintain the convex hull property of the given NURBS surface, all the weights in a solution to Equation (7.41) must be positive. The positiveness of weights can be put into additional constraints to Equation (7.40). In other words, the original problem is transformed into an “inequality” constrained problem [60] as follows: find a set of weights \(w\) that minimizes the function \(D(w)\) subject to the constraints that \(G(w) = 0\) and \(w > 0\). Recall that \(w = \{w_{i,j}\}\) is a vector consisting of \(N\) elements. Therefore \(w > 0\) corresponds to \(N\) distinct inequalities. For convenience, hereafter let us denote the element of \(w\) as \(w_1, \ldots, w_N\) hereafter.

By introducing additional \(N\) slack variables \(w_{N+1}, \ldots, w_{2N}\), the above inequality constrained problem is transformed into an equality constrained problem as follows: find a set of weights \(w\) that minimizes the function \(D(w)\) subject to the constraints that \(G(w) = 0\) and \(H(w_i, w_{N+i}) = 0\) for \(1 \leq i \leq N\), where \(H(w_i, w_{N+i}) = -w_i + w_{N+i}^2 + \epsilon\) and \(\epsilon\) is a positive constant. Let \(\omega = \{w_1, \ldots, w_N, w_{N+1}, \ldots, w_{2N}\}\). The function that we want to minimize is now represented by the following formula:

\[
F(\omega, \lambda, \mu) = D(\omega) + \lambda G(\omega) + \sum_{i=1}^{N} \mu_i H(w_i, w_{N+i}), \quad \text{(7.42)}
\]
where $\mu_i(1 \leq i \leq N)$ denote arbitrary variables. From this, a condition similar to Equation (7.41) is obtained as follows:

$$\nabla F(\mathbf{w}, \lambda, \mu) = 0.$$ (7.43)

This represents $3N + 1$ non-linear simultaneous equations with $3N + 1$ variables. In other words, Equation (7.43) can be written as $F_{w1} = 0, \ldots, F_{w2N} = 0, F_{\lambda} = 0, F_{\mu_1} = 0, \ldots, F_{\mu_N} = 0$, where $F_x \equiv \frac{\partial F}{\partial x}$ for all $x \in \{w_1, \ldots, w_{2N}, \lambda, \mu_1, \ldots, \mu_N\}$.

Example: Consider an octant of a sphere. Let $\delta = \pi/4$. From Equation (7.11), $\alpha_3 = 2\pi - (\alpha_1 + \alpha_2 + \alpha_4) - \int_S KdS$. Suppose $\alpha_1 = \alpha_2 = \alpha_4 \approx \pi/2$. Then, $\alpha_3 \approx \pi/2 - \int_S KdS \geq \pi/2 - \delta = \pi/4$. Thus, the minimum allowable thread angle ($\gamma_{\min}^{\text{lock}}$) is guaranteed to be larger than $\pi/4$. Since most of the practical woven cloth composites have larger $\gamma_{\min}^{\text{lock}}$ values than $\pi/4$ [62], it is reasonable to modify the surface shape such that $\int_S KdS = \pi/4$.

Figure 7.23 illustrates an octant of a sphere defined by a rational biquadratic B-spline surface. The control points and the associated weights are denoted $c_{i,j}$ and $w_{i,j}$, respectively, for $0 \leq i \leq 2$ and $0 \leq j \leq 2$. The knot vector in $u$ direction is $\{0, 0, 1, 1\}$ and the knot vector in $v$ direction is $\{0, 0, 1, 1\}$. Assuming the unit radius for the sphere, the coordinates of the control points and the associated weights are given as follows:

- Control Points ($c_{i,j}$):

<table>
<thead>
<tr>
<th>$c_{i,j}$</th>
<th>i=0</th>
<th>i=1</th>
<th>i=2</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j=0$</td>
<td>(0,0,1)</td>
<td>(1,0,1)</td>
<td>(1,0,0)</td>
</tr>
<tr>
<td>$j=1$</td>
<td>(0,0,1)</td>
<td>(1,1,1)</td>
<td>(1,1,0)</td>
</tr>
<tr>
<td>$j=2$</td>
<td>(0,0,1)</td>
<td>(0,1,1)</td>
<td>(0,1,0)</td>
</tr>
</tbody>
</table>
Figure 7.23: The original shape of an octant of a sphere defined by a rational biquadratic B-spline surface.

- Weights \((w_{i,j})\):

<table>
<thead>
<tr>
<th>(w'_{i,j})</th>
<th>(i=0)</th>
<th>(i=1)</th>
<th>(i=2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(j=0)</td>
<td>1</td>
<td>1/(\sqrt{2})</td>
<td>1</td>
</tr>
<tr>
<td>(j=1)</td>
<td>1/(\sqrt{2})</td>
<td>1/2</td>
<td>1/(\sqrt{2})</td>
</tr>
<tr>
<td>(j=2)</td>
<td>1</td>
<td>1/(\sqrt{2})</td>
<td>1</td>
</tr>
</tbody>
</table>

The first experiment that we have conducted with this example was to select one control point and to modify the weight that satisfies Equation (7.37) and \(w'_{i,j} > 0\), where \(w'_{i,j}\) represents the modified weight of \(w_{i,j}\). The following table shows the result, where N/S in the entry represents “No Solution.”

<table>
<thead>
<tr>
<th>(w'_{i,j})</th>
<th>(i=0)</th>
<th>(i=1)</th>
<th>(i=2)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(j=0)</td>
<td>N/S</td>
<td>11.43</td>
<td>N/S</td>
</tr>
<tr>
<td>(j=1)</td>
<td>N/S</td>
<td>0.14</td>
<td>4.43</td>
</tr>
<tr>
<td>(j=2)</td>
<td>N/S</td>
<td>11.43</td>
<td>N/S</td>
</tr>
</tbody>
</table>
Figure 7.24: The plot of $|I_K|$ as a function of $w_{1,1}$ and $w_{2,1}$.

In the second experiment, we have selected two weights ($w_{1,1}$ and $w_{2,1}$) and plotted the graph of the integral Gaussian curvature ($|I_K| = I(w_{1,1}, w_{2,1})$) by changing the two weights (See Figure 7.24) from 0.2 to 5.0. As opposed to the first experiment, the solution to Equation (7.37) is given by a curve instead of a point on the surface ($I(w_{1,1}, w_{2,1})$). In other words, a unique pair of weights $w_{1,1}$ and $w_{2,1}$ that satisfy Equation (7.37) do not exist. For example, the solution to $G(w) = \left| \int_u \int_v Kdudv \right| - 2.0 = 0$ is given by the curve $|I_K| = 2.0$ as shown in Figure 7.24. Therefore, an additional constraint, $D(w)$, defined by Equation (7.39), is necessary to obtain a unique solution.

Figure 7.25 (a) shows the original shape of the octant of a sphere. Figure 7.25 (b) corresponds to the result obtained by selecting the weight $w_{2,1}$ and modifying it into 4.43 as in the above table. Figure 7.25 (c) is obtained by modifying three weights $w_{1,0}, w_{1,2}$, and $w_{1,1}$ from Equation (7.37). In Figure 7.25 (d), two weights $w_{1,1}$ and $w_{2,1}$ are selected and the optimization of the deviation with Equation (7.42) is performed. Figure 7.26 shows the same result with control points and control polygon edges drawn with colors. Here, a green color is assigned to the
Figure 7.25: An example of shape modification; (a) is the original shape of the octant of a sphere, (b), (c), and (d) are modified shapes whose $I_K$ values are equal to $\pi/4$.

weight value 1.0. As the weight value becomes large, the color shifts toward red. Similarly, as the weight value becomes small, the color shifts toward blue.

To compare the results in terms of the deviation from the original shape, we have assigned colors based on the deviation value from the original shape at each $(u, v)$ point on the modified surface. Figure 7.27 shows the deviations with colors, corresponding to the example of Figures 7.25 and 7.26. Red colors are assigned to areas of large deviation values, and green colors are assigned to areas of no deviation values. Clearly, Figure 7.27 (d) has the minimum deviation among the three cases ((b), (c), and (d)). Figure 7.28 demonstrates the distribution of Gaussian curvatures with colors for the same example. Here, colors are assigned as shown in the color bar. Since the original spherical shape has a constant positive Gaussian curvature, Figure 7.28 (a) has a yellow color everywhere. The shapes corresponding to cases
Figure 7.26: An example of shape modification with colors assigned to the control points and control polygon edges of Figure 7.25, where \( W \), attached to the color bar, denotes the value of the weights of the control points.
Figure 7.27: Deviations with colors for the shapes shown in Figure 7.25, where $D$, attached to the color bar, denotes the deviation defined by Equation (7.39). Clearly, (d) has minimum deviation among the three ((b),(c), and (d)).

(b) and (c) have bumps on the surface where Gaussian curvatures take relatively large values. The optimal shape, case (d), looks similar to (a) in Figure 7.27, but it has a somewhat flatter region in the middle, as depicted by the green colors in Figure 7.28 (d). Figure 7.29 is the fitting results for each shape with colors assigned, based on the thread angle values. Obviously, all of the fitting results to the new shapes have no “reddish” (critical) regions.

7.3.5 Modifying a Surface Shape by Combining Control Polygon Adjustment and Optimal Weight Finding

The method provided in the previous section works well as long as a fitting under a given set of initial conditions produces a single critical region and the control
Figure 7.28: Gaussian curvatures with colors for the shapes shown in Figure 7.25.
Figure 7.29: 3D fitting results with colors for the shapes shown in Figure 7.25.
polygon that defines the convex hull of the given NURBS surface allows sufficient flexibility for shape modification. However, it may not be easy to find a solution to Equation (7.41) under the following conditions:

1. The given NURBS surface is fairly complex such that it has both positive and negative Gaussian curvature regions that appear repeatedly more than once.

2. The control polygon that defines the given NURBS surface is either too tight or too loose, where “tightness” is measured as the deviation between the given NURBS surface and the corresponding control polygon.

For each condition, we have associated problems as follows:

1. The fitting result may contain two or more critical regions. If this is the case, it is difficult to determine which control points should be selected for shape optimization.

2. If the control polygon is too tight, there is little flexibility in controlling the shape with the weight values. If the control polygon is too loose, a change of a weight value may cause a large deviation of the modified surface relative to its original shape.

Let us refer to the above problems as Problem 1 and Problem 2, respectively. In this section, we will focus on these problems: First, we will provide an example that makes the shape modification difficult with the optimal weight finding algorithm, and causes the above problems. Second, as a solution to Problem 1, we will present a method for identifying which control points affect the generation of a critical region. Third, as a solution to Problem 2, we will provide a method for alleviating the tightness by suitably adjusting control points, if the control polygon is too tight. We will also describe a method for alleviating the looseness by inserting knots or by elevating the degree of the polynomial that defines given NURBS surface, if the control polygon is too loose. Finally, we will present a flowchart for the shape modification that incorporates the methods mentioned so far.
7.3.5.1 An Example that Makes the Shape Change Difficult

The shoe-like object mentioned in Section 6.2.5.3 suffers from Problem 1. As shown in Figure 7.31, it has both positive and negative Gaussian curvature regions that appear more than once. The fitting result shown in Figure 6.29 has two critical regions shaded with red colors. Figure 7.30 demonstrates another view of the critical regions by plotting the distribution of thread angles between wefts and warps at mesh points projected above associated plane developments for the fitting. Note that one of the critical regions occurs around the tip of the shoe-like object, and the minimum thread angle in the region is 0.211 radian, and another critical region occurs around a side of the object, and the minimum thread angle in the region is 0.447 radian. Note that these minimum thread angles go below the locking angle, $\gamma_{\text{lock}}$, that is 0.523 radian (for the Kevlar sheet tabulated in Appendix F).

The critical regions can also be identified by plotting the intermediate Gaussian curvature integral, $I_{K_{i,j}}$, that is defined by Equation (7.22). Figure 7.32 shows the intermediate Gaussian curvature integrals for the shoe-like object under the same conditions.
Figure 7.31: Gaussian curvatures with colors for the shoe-like object.
set of initial conditions as those used in Figure 7.30. Two peaks in the intermediate Gaussian curvature integrals, which correspond to the two critical regions, are observed in this figure.

The shoe-like object also has Problem 2 in that its control polygon is very tight. In general, the tightness of the control polygon increases as the number of the control points that lie on the given NURBS surface increases. This is especially true if we define the NURBS surface with low degree polynomials. In particular, if the degree of the surface patch is one in both $U$ and $V$ parametric directions, all the control points lie on the surface (in this case, the plane).

For later discussion, let us call the control points that lie on the surface the on-surface control points and those that do not lie on the surface the off-surface control points. A control point is categorized as an on-surface control point if it is a Bézier control point in both $U$ and $V$ parametric directions and belongs to the end points of the corresponding parametric domain, or if it is one of the control points that have the same coordinate, defined consecutively as many times as the degree of the polynomial in one of the parametric directions. Note that in order for a control point of a NURBS surface to be a Bézier control point, the multiplicity of the knot
vectors in both $U$ and $V$ parametric directions must be equal to the degree of the polynomials in $U$ and $V$ domains, respectively [29].

The shoe-like object mentioned above is defined as a biquadratic NURBS surface whose knot vectors are given as $\{0, 0, \frac{1}{3}, \frac{1}{3}, \frac{2}{3}, \frac{2}{3}, 1, 1\}$ in both the $U$ and $V$ directions. The corresponding control points $c_{i,j}$, where $0 \leq i \leq 6$ and $0 \leq j \leq 6$, are shown in Figure 7.33. All the weight values $w_{i,j}$ at control points $c_{i,j}$ for the original shape are set to 1. The on-surface control points are marked with an asterisk in Figure 7.33. In this example, they are $c^{*}_{i,6}$ ($0 \leq i \leq 6$), and $c^{*}_{i,j}$ ($i \in \{0, 2, 4, 6\}$ and $j \in \{0, 2, 4\}$). Since there are 49 control points in total, more than one-third of the control points are classified as on-surface control points. This is why the control polygon of this object is too tight to flexibly modify the original shape with the optimal weight finding algorithm.

Here we will demonstrate how the original optimal weight finding algorithm, mentioned in the previous section, fails to remove all the critical regions that occur in the fitting result to the shoe-like object. As shown in Tables 7.1 and 7.2, of all the combinations of the weight modifications that involve one or two weights, $w_{1,5}$ and $w_{5,5}$ are found to be the best combination because the deviation $D(w_{1,5}, w_{5,5})$ (given by Equation (7.39)) is minimum. Note that we assume that $w_{1,5} = w_{5,5}$, in order to keep the modified shape symmetric as is the original shape. Note also that $\delta$ is set to 0.0 in Equation (7.37). As depicted in Figure 7.35 (b), it is clear that the original critical regions do not vanish.

7.3.5.2 A Method for Identifying a Set of Control Points that Affect the Generation of a Critical Region

Here we will focus on Problem 1 and provide a solution. Recall that Problem 1 is the difficulty in determining which control points should be selected for shape optimization with the optimal weight finding algorithm, when two or more critical regions result in a fitting. Selecting one or two control points randomly and
Figure 7.33: Control points for the shoe-like object. The control points with an asterisk represent the on-surface control points.
Table 7.1: The result of the optimal weight finding algorithm applied to the shoe-like object. For the weights \( w_{3,j} \) \((0 \leq j \leq 6)\), the optimal weight is sought such that \( G(w_{3,j}) = 0 \) with \( \delta = 0.0 \) in Equation (7.37). For the weights \( w_{i,j} \) \((i \in \{0,1,2\}, 0 \leq j \leq 6)\), the corresponding weights \( w_{6-i,j} \) are also changed into the same value to keep the symmetry of the resulting shape. For instance, if \( w_{0,j} \) is changed into a new value, \( w_{6,j} \) is also changed into the same new value, and the solution to \( G(w_{0,j}, w_{6,j}) = 0.0 \) with \( \delta = 0.0 \) is sought. See Figure 7.33 for identifying what control points constitute a pair that produces a symmetric result. Note that N/S in the entry represents No Solution, including the case where \( w'_{i,j} \) becomes negative.

<table>
<thead>
<tr>
<th>( w'_{i,j} )</th>
<th>i=0</th>
<th>i=1</th>
<th>i=2</th>
<th>i=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>j=0</td>
<td>N/S</td>
<td>N/S</td>
<td>N/S</td>
<td>169.77</td>
</tr>
<tr>
<td>j=1</td>
<td>1135.47</td>
<td>N/S</td>
<td>N/S</td>
<td>N/S</td>
</tr>
<tr>
<td>j=2</td>
<td>471.87</td>
<td>N/S</td>
<td>0.06</td>
<td>97.79</td>
</tr>
<tr>
<td>j=3</td>
<td>N/S</td>
<td>39.59</td>
<td>17.28</td>
<td>N/S</td>
</tr>
<tr>
<td>j=4</td>
<td>0.03</td>
<td>4.32</td>
<td>0.02</td>
<td>121.09</td>
</tr>
<tr>
<td>j=5</td>
<td>N/S</td>
<td>0.373</td>
<td>4.2</td>
<td>N/S</td>
</tr>
<tr>
<td>j=6</td>
<td>0.07</td>
<td>3.45</td>
<td>0.21</td>
<td>36.12</td>
</tr>
</tbody>
</table>

Table 7.2: The deviation value \( D(w) \) between the original surface shape and the modified shape by changing the one or two weights that correspond to \( w'_{i,j} \) in Table 7.1.

<table>
<thead>
<tr>
<th>( D(w'_{i,j}) )</th>
<th>i=0</th>
<th>i=1</th>
<th>i=2</th>
<th>i=3</th>
</tr>
</thead>
<tbody>
<tr>
<td>j=0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>8769.3</td>
</tr>
<tr>
<td>j=1</td>
<td>60578.1</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j=2</td>
<td>70762.6</td>
<td>-</td>
<td>3249.6</td>
<td>14082.6</td>
</tr>
<tr>
<td>j=3</td>
<td>-</td>
<td>6327.9</td>
<td>6744.9</td>
<td>-</td>
</tr>
<tr>
<td>j=4</td>
<td>1318.5</td>
<td>1732.4</td>
<td>1287.7</td>
<td>12690.0</td>
</tr>
<tr>
<td>j=5</td>
<td>-</td>
<td>75.3</td>
<td>1083.8</td>
<td>-</td>
</tr>
<tr>
<td>j=6</td>
<td>416.6</td>
<td>342.9</td>
<td>294.4</td>
<td>1243.3</td>
</tr>
</tbody>
</table>
modifying the associated weights may not remove all the critical regions, as demonstrated with the shoe-like object in the previous section. Generally speaking, as the number of critical regions that occur in the fitting result increases, the complexity of determining which control points should be selected for shape optimization also increases.

Given a surface position \((u, v)\), where \(u \in [u_I, u_{I+1}]\) and \(v \in [v_J, v_{J+1}]\), we can determine the control points that affect the shape of the surface at this position \[29\]. Note that \([u_I, u_{I+1}]\) and \([v_J, v_{J+1}]\) are certain parametric domains defined by the given knot vectors in \(U\) and \(V\) parametric directions. For the \(U\) parametric direction, the indices of the control points that affect the shape at \(u\) are \(I - m + 1, ..., I + 1\), where \(m\) is the degree of the polynomial in the \(U\) direction. Similarly, for the \(V\) parametric direction, the indices of the control points that affect the shape at \(v\) are \(J - n + 1, ..., J + 1\), where \(n\) is the degree of the polynomial in the \(V\) direction. Therefore, the control points \(c_{i,j}\), where \(I - m + 1 \leq i \leq I + 1\) and \(J - n + 1 \leq j \leq J + 1\), affect the shape of the surface at \((u, v)\). In total, there are \((m + 1) \times (n + 1)\) such control points.

In order to identify the control points that affect the occurrence of a critical region, we may trace all the mesh points that affect the mapping of a mesh point in the critical region. Since each mesh point keeps track of the surface position \((u, v)\) (See the variable “mapped_UV_pos” of a Grid data structure in \textbf{Appendix C}), we can calculate the control points that affect the generation of a critical region by taking the union of all the control points that can be obtained from a mesh point of interest in the critical region. Specifically, it is given by the following algorithm:

\textbf{Algorithm 7.5 (IDENTIFYING CONTROL POINTS THAT AFFECT THE SHAPE OF THE SURFACE AT A MESH POINT)}

\textbf{Procedure} IDENTIFY-CONTROL-POINTS \((G, S)\);
\[ \begin{align*}
R &= \emptyset; \\
L &= \text{TRACE}(G, \phi); \\
\text{for } m=1 \text{ to } M \text{ do } \\
\end{align*} \]
for n=1 to N do
  if _Grid[m][n] in L then
    G = _Grid[m][n];
    u = G→mapped_UV→pos.u;
    v = G→mapped_UV→pos.v;
    m = S→degree[0];
    n = S→degree[1];
    I = FIND-U-INDEX(S, u);
    J = FIND-V-INDEX(S, v);
    g = G→gauss;
    for i=I-m+1 to I+1 do
      for j=J-n+1 to J+1 do
        InsertSort-ControlPoint(R, c_{i,j}, g);
      endfor
    endfor
  endif
endfor

Note that G denotes a pointer to a Grid data structure, S denotes a pointer to
a NURBS_3S data structure, R denotes a pointer to a sorted list of control points,
and L denotes a pointer to a function that returns a list of pointers to a Grid data
structure. See Appendix B for the NURBS_3S data structure and Appendix
C for the Grid data structure. The function InsertSort-ControlPoint(R, c_{i,j}, g)
inserts the control points that affect the shape at a point (u, v) into a sorted list
pointed to by R. The sorting of the list is done by defining the sorting key as the
number of references of the control point multiplied by the Gaussian curvature at
(u, v) (denoted by g in the above algorithm). The functions FIND-U-INDEX() and
FIND-V-INDEX() return the indices of the control points along the U direction and
along the V direction, respectively. The function TRACE() is exactly the same as
described in Algorithm 7.2. The output of the procedure IDENTIFY-CONTROL-
POINT(G,S) is the sorted list of control points that affect the shape of the surface
at a point (u, v). Let us call the list a candidate control point list, hereafter.
Controlling the surface shape becomes difficult when the control polygon for a given NURBS surface is either too tight or too loose. In the following, we will first consider the “looseness” problem and then we will consider the “tightness” problem.

A simple idea of alleviating the looseness of the control polygon is to insert knots for the NURBS surface where the control polygon should be refined. A knot insertion is a technique that refines the control polygon by inserting a new knot into the knot vector without changing the surface shape [29]. A feature of knot insertion is that it does not change the degree of the given surface. However, if the multiplicity of each knot value is equal to the degree of the polynomial that defines the surface, nothing changes by inserting further knots. In such a case, we have to employ other techniques such as degree elevation. A degree elevation is a technique that refines the control polygon by increasing the degree of the polynomial without changing the surface shape [29]. It is known that by repeatedly elevating the degree of the polynomial, the control polygon eventually converges to the surface profile [29]. Figure 7.34 shows an example of degree elevation, that is applied to an octant of a sphere (See Figure 7.23). Note that the control polygon shown in Figure 7.34 (b) becomes more close to the surface profile.

Next, we will focus on the “tightness” problem. This is more difficult than the “looseness” problem, because there is no known algorithm that expands the control polygon without changing the original shape. There is a technique called degree reduction [29], but it only approximates the original shape, and the control polygon of the new shape does not necessarily contain the original control polygon. Our idea of alleviating the “tightness” problem is to move some of the control points such that the new control polygon expands the convex hull of the original control polygon. Specifically, we move some of the on-surface control points that may have vital effects on the occurrence of each critical region. The reason for selecting the on-surface control points instead of the off-surface control points is partly because it is more intuitive to move the on-surface control points due to the fact that the
Figure 7.34: An example of degree elevation applied to an octant of a sphere; (a) shows the original control polygon, and (b) shows the control polygon after degree elevation.

modified surface must pass through them, and partly because we can easily define the direction in which the convex hull expands.

For instance, we can move an on-surface control point $c_{i,j}$ along the surface normal at the point. Let $c'_{i,j}$ denote the new location for $c_{i,j}$. Then,

$$c'_{i,j} = c_{i,j} + \eta \tilde{n},$$

(7.44)

where $\tilde{n}$ denotes the unit surface normal, and $\eta$ is a small number (either positive or negative) that corresponds to the distance between $c'_{i,j}$ and $c_{i,j}$. Determining the parameter $\eta$ can be implemented with various methods. A straightforward method is to employ the equation similar to Equation (7.37) as follows:

$$G(C) = | \int_u \int_v K(u, v; C) du dv | - \delta = 0,$$

(7.45)

where $C$ denotes a set of on-surface control points that are selected. Similarly, the deviation of the modified surface $D(C)$ is given by

$$D(C) = \int_u \int_v ||r(u, v; C_0) - r(u, v; C)||^2 du dv,$$

(7.46)
where $C_0$ denotes a set of on-surface control points before modification. Then, the parameter $\eta$ can be obtained by $\nabla F(C, \lambda) = 0$, where $F(C, \lambda) = D(C) + \lambda G(C)$, that is analogous to Equations (7.40) and (7.41). This technique can be generalized such that both $C$ and $w$ are considered for the optimization problem. However, it is more difficult to find a solution that satisfies equations such as $G(C, w) = 0$ and $\nabla F(C, w, \lambda) = 0$.

A more practical method is to assign a small number to the parameter $\eta$ in advance, and to determine if the integral Gaussian curvature of the new surface, denoted by $|I'_K|$, increases or decreases. If it increases, we change the direction of the movement by switching the sign of $\eta$. If it decreases and satisfies the condition $D(C) < h_D$, where $h_D$ denotes a deviation threshold, we increment the $\eta$ value in that direction repeatedly. It is guaranteed that this method expands the convex hull of the control polygon and decreases the value of the integral Gaussian curvature. However, $|I'_K| = \delta$ is not yet assured. In order to assure $|I'_K| = \delta$, we may employ the optimal weight finding algorithm after expanding the convex hull of the original control polygon. The algorithm that seeks an optimal shape modification that is free of anomalous events will be stated more formally in the next section.

### 7.3.5.4 An Algorithm for Shape Modification that Combines Control Point Adjustment with Optimal Weight Finding

Here we will present an algorithm for modifying the shape of a given NURBS surface, assuming that the surface shape is fairly complex and the control polygon is very tight. The key idea of the algorithm is to combine the control point adjustment with the optimal weight finding. In the following algorithm, we maintain a data structure called a *candidate control point list*, together with a pointer to it, in order to select a candidate control point that will be adjusted next.

Algorithm 7.6 (ALGORITHM FOR SHAPE MODIFICATION BY CONTROL POINT ADJUSTMENT AND OPTIMAL WEIGHT FINDING)
1. Perform a fitting under a given set of initial conditions.

2. If the fitting produces no critical regions, quit. If it produces one or more critical regions, repeat the following for each critical region.

3. Select a mesh point that produces the minimum (or maximum) thread angle in the critical region.

4. Keep track of the control points that contribute to the mapping of the mesh point that is selected in Step 3 in a critical region, by inserting and sorting them into the candidate control point list.

5. Set the pointer of the candidate control point list at the first element.

6. Repeat the following control point adjustment until either the pointer of the candidate control point list becomes NULL or a solution is found in which no critical regions appear in the fitting.

   (a) Pick up the first on-surface control point from the candidate control point list.

   (b) Determine \(\eta\) such that the modified surface has a smaller integral Gaussian curvature value (i.e. \(|I'_K| < |I_K|\)) under the condition that \(D(c'_{i,j}) < h_D\), where \(h_D\) denotes a deviation threshold and \(c'_{i,j} = c_{i,j} + \eta \hat{n}\).

   (c) If there is no solution, go to Step 7; otherwise perform a fitting with the modified surface under the same set of initial conditions.

   (d) If the fitting produces no critical regions, quit with SUCCESS; otherwise increment the pointer of the candidate control point list and go back to 6-a.

7. Reset the pointer of the candidate control point list at the first element.

8. Repeat the following optimal weight finding process until either the pointer of the candidate control point list becomes NULL or a solution is found in which no critical regions appear in the fitting.
(a) Pick up the first control point (either an on-surface or an off-surface control point) from the candidate control point list.

(b) Apply the optimal weight finding algorithm for the selected control points.

(c) If there is no solution, go to Step 8-e; otherwise perform a fitting under the same set of initial conditions.

(d) If the fitting produces no critical regions, quit with SUCCESS; otherwise do the following.

(e) Increment the pointer of the candidate control point list, and go back to Step 8-a.

Note that Step 4 is given by Algorithm 7.5. The expansion of the convex hull of a given NURBS surface is done in Step 6 by adjusting some of the on-surface control points that may have vital effects on the generation of a critical region. Once the convex hull is expanded, we have greater flexibility in controlling the shape with the optimal weight finding process in Step 8.

Example: The shoe-like object shown in Figure 6.29 generates two critical regions. For convenience, let us refer to the critical regions as $S_1$ and $S_2$, respectively. Here, $S_1$ refers to the critical region that includes the mesh point whose thread angle is 0.211, and $S_2$ refers to the critical region that includes the mesh point whose thread angle is 0.447 (See Figure 7.30). Similarly, $S_1$ includes the mesh point whose intermediate Gaussian curvature integral is 1.59, and $S_2$ includes the mesh point whose intermediate Gaussian curvature integral is 0.91 (See Figure 7.32).

For the critical region $S_1$, the mesh point whose thread angle is 0.211 is chosen for the application of Algorithm 7.5, according to Step 3 in Algorithm 7.6. The resulting values are shown in Table 7.3, which represents how much each control point contributes to the shape at the selected mesh point. Similarly, for the critical region $S_2$, the mesh point whose thread angle is 0.447 is chosen for the application of Algorithm 7.5. The resulting values, which represent the contribution of each
Table 7.3: Contribution of control points in determining the shape at the mesh point whose thread angle is 0.211 in the critical region $S_1$. Note that the control points with larger values have a greater impact on the shape than those with smaller values.

<table>
<thead>
<tr>
<th></th>
<th>$i=0$</th>
<th>$i=1$</th>
<th>$i=2$</th>
<th>$i=3$</th>
<th>$i=4$</th>
<th>$i=5$</th>
<th>$i=6$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$j=0$</td>
<td>0.75</td>
<td>0.75</td>
<td>1.10</td>
<td>0.35</td>
<td>0.35</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$j=1$</td>
<td>0.75</td>
<td>0.75</td>
<td>1.10</td>
<td>0.35</td>
<td>0.35</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$j=2$</td>
<td>2.03</td>
<td>2.03</td>
<td>3.78</td>
<td>1.75</td>
<td>1.75</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$j=3$</td>
<td>1.28</td>
<td>1.28</td>
<td>2.67</td>
<td>1.40</td>
<td>1.40</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>$j=4$</td>
<td>5.04</td>
<td>5.04</td>
<td>7.68</td>
<td>2.64</td>
<td>6.59</td>
<td>3.94</td>
<td>3.94</td>
</tr>
<tr>
<td>$j=5$</td>
<td>3.76</td>
<td>3.76</td>
<td>5.00</td>
<td>1.25</td>
<td>5.19</td>
<td>3.94</td>
<td>3.94</td>
</tr>
<tr>
<td>$j=6$</td>
<td>3.76</td>
<td>3.76</td>
<td>5.00</td>
<td>1.25</td>
<td>5.19</td>
<td>3.94</td>
<td>3.94</td>
</tr>
</tbody>
</table>

control point to the shape at the selected mesh point, is shown in Table 7.4.

In order to reduce the number of combinations of the control points for adjustments, we are only interested in the shape modification that produces a symmetric shape. In this example, the control points $c_{i,j}$ are paired with $c_{6-i,j}$, and the summed values are compared to construct the candidate control point list. For the critical region $S_1$, after sorting the candidate control point list, the first few on-surface control points in the list are given by

$$\{c_{i,6}, < c_{2,4}, c_{4,4} >, < c_{0,4}, c_{6,4} >, \ldots\},$$

where the control points in angular brackets denote the paired on-surface control points, and $c_{i,6}$ ($0 \leq i \leq 6$) corresponds to a single point as shown in Figure 7.33. Unfortunately, the adjustment of these on-surface control points do not greatly improve the fitting result in that the critical region $S_1$ does not vanish with a small $h_D$ value. Thus, we proceed to the adjustment of control points for the critical region $S_2$. After sorting the candidate control point list, the first few on-surface control points in the list are:

$$\{< c_{2,2}, c_{4,2} >, < c_{2,4}, c_{4,4} >, < c_{0,2}, c_{6,2} >, \ldots\}.$$
Table 7.4: Contribution of control points in determining the shape at the mesh point whose thread angle is 0.447 in the critical region $S_2$.

<table>
<thead>
<tr>
<th></th>
<th>i=0</th>
<th>i=1</th>
<th>i=2</th>
<th>i=3</th>
<th>i=4</th>
<th>i=5</th>
<th>i=6</th>
</tr>
</thead>
<tbody>
<tr>
<td>j=0</td>
<td>0.73</td>
<td>0.73</td>
<td>0.95</td>
<td>0.22</td>
<td>0.22</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j=1</td>
<td>0.73</td>
<td>0.73</td>
<td>0.95</td>
<td>0.22</td>
<td>0.22</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j=2</td>
<td>1.54</td>
<td>1.54</td>
<td>2.65</td>
<td>1.11</td>
<td>1.11</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j=3</td>
<td>0.81</td>
<td>0.81</td>
<td>1.70</td>
<td>0.89</td>
<td>0.89</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j=4</td>
<td>0.81</td>
<td>0.81</td>
<td>1.70</td>
<td>0.89</td>
<td>0.89</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j=5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>j=6</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

By repeatedly applying Step 6 of Algorithm 7.6, the on-surface control points $< c_{0,2}, c_{6,2} >$ are finally selected for the adjustment. If we set $\eta = 1.0$ and $h_D = 100.0$, the critical region $S_2$ is removed where $|I_K'| = 1.20$ and $|I_K| = 1.75$ (Note $|I_K'| < |I_K|$).

Then, we apply the optimal weight finding algorithm in Step 8 in Algorithm 7.6 for the remaining critical region $S_1$. For the case that involves one or two weights, we obtain $w_{1,5} = w_{5,5} = 0.51$ from Equation (7.37), assuming that $\delta = 0.0$. With this modification, the critical region $S_1$ is removed. The fitting result with colors assigned to the thread angles is shown in Figure 7.35 (c). Figure 7.35 (d) is obtained by setting the deviation threshold $h_D$ to 400.0. Figure 7.36 shows the deviations of the modified surface, shaded with colors, corresponding to the examples in Figure 7.35.

7.3.6 Summary of the Methods for Shape Modification

Figure 7.37 summarizes the methods for shape modification described in the previous sections into a flowchart. First, the NURBS surface together with the convex hull defined by the control points are displayed. Second, the candidate control point list is constructed. Third, the threshold parameter $\delta$ for $|I_K'|$ value (See Equation (7.37)) and the deviation threshold $h_D$ (from Equation (7.39) or (7.46)) are
Figure 7.35: 3D fitting results to the shoe-like object where shape modifications are applied: (a) the original shape, (b) the modified shape obtained by the optimal weight finding algorithm alone, (c) the modified shape obtained by adjusting the on-surface control points $c_{0,2}$ and $c_{6,2}$, assuming the deviation threshold $h_D = 100.0$, and by applying the optimal weight finding algorithm for $w_{1,5}$ and $w_{5,5}$; (d) the same as (c) but with the deviation threshold $h_D = 400.0$. 
Figure 7.36: Deviations with colors for the shapes shown in Figure 7.35.
set. The convex hull defined by the control points is optionally refined by inserting knots or by elevating the degree of the polynomial of the given NURBS surface, if the convex hull is too loose. Similarly, the convex hull is optionally expanded by adjusting some of the on-surface control points of the given NURBS surface, if the convex hull is too tight. Fourth, one or more control points are selected from the candidate control point list, and the optimal weight finding algorithm is applied. If there is any solution and the resulting shape is satisfactory, then the surface shape is updated, and the fitting with the new shape is performed. Otherwise this entire process is repeated either from setting another threshold values $\delta$ and $h_D$ or from selecting another set of control points. Note that the procedure shown in this flowchart is initiated when one or more critical regions appear after a fitting under a given set of initial conditions.
Display the NURBS surface with the convex hull defined by the control points

Construct a candidate control point list and sort it

Set the thresholds $\delta$ and $\beta_D$

Is the convex hull ok?

Select control points from the candidate control point list

Apply the optimal weight finding algorithm

Any solution?

Modify the weights and display the new surface

Any solution?

Set the thresholds $\delta$ and $\beta_D$

Too loose?

Too tight?

Refine the convex hull by knot insertion or degree elevation

Expand the convex hull control points

Any solution?

Modify the weights and display the new surface

Any solution?

Set the thresholds $\delta$ and $\beta_D$

Too loose?

Too tight?

Refine the convex hull by knot insertion or degree elevation

Expand the convex hull control points

Any solution?

Modify the weights and display the new surface

Any solution?

Set the thresholds $\delta$ and $\beta_D$

Too loose?

Too tight?

Refine the convex hull by knot insertion or degree elevation

Expand the convex hull control points

Any solution?

Modify the weights and display the new surface

Any solution?

Set the thresholds $\delta$ and $\beta_D$

Too loose?

Too tight?

Refine the convex hull by knot insertion or degree elevation

Expand the convex hull control points

Any solution?

Figure 7.37: Flowchart for modifying a surface shape.
CHAPTER 8
CONCLUSION

This chapter summarizes the research described in the previous chapters, evaluates the achievement of objectives, and suggests topics for future research.

8.1 Summary

The objective of the research was to investigate the geometric aspects of the complex deformation mechanism of a woven cloth composite ply when it is fitted to a curved surface, and to provide a methodology for a computer-aided design system that supports the fitting process and the 2D “flattened” pattern making process. A particular attention was paid to the algorithms for predicting and preventing anomalous events by analyzing the geometric properties of a given surface and the fitting result, where anomalous events such as wrinkling and tearing are considered serious manufacturing faults that may occur during the process of forming woven cloth composites.

A mathematical model for a woven cloth composite ply was provided with several assumptions including inextensibility of threads (a Tchebychev net assumption), straightness between adjacent crossings, and no slippage at each crossing. Later, an enhanced model incorporating slippage at crossings was proposed. With a Tchebychev net assumption, only bending and shearing deformations are allowed for a given woven cloth composite ply, where shear deformation is caused by changing the thread angle between a weft and a warp. One of the goals that we want to achieve with this woven cloth model is to predict and prevent anomalous events such as wrinkling and tearing that are triggered by excessive shear deformation. In order to identify the beginning of such an anomalous event we have defined a locking angle as the angle that is determined by the geometric structure of a given woven cloth composite ply, including the width of each thread, the length between adjacent
crossings, and the weaving type. The locking angle allows us to predict the presence and location of anomalous events, particularly wrinkling. The object that a given woven cloth composite ply is fitted was modeled by a NURBS surface because of its flexibility in representing a wide variety of surfaces, and its local controllability of the surface shape.

Given the mathematical models for a woven cloth composite ply and a surface, we can simulate the fitting of a given woven cloth composite ply onto a surface. The fitting was shown to be equivalent to a mapping between a mesh point in 2D cloth and the corresponding point on the surface in 3D space. We have developed a flexible mapping method that made it possible to start with an arbitrarily-oriented base path and associated sweeping directions. This method provides greater flexibility than the previous methods that required the alignment of two perpendicular threads with the initial paths [14, 61, 84, 85, 106, 107]. We have also described a method for automatically calculating the mapping of mesh points on the base path by defining it as the intersection between a plane and the given surface, or by defining it as a geodesic on the surface. The "uncalculated surface region" problem, which was a serious problem in the previous methods, was avoided by introducing a heuristic by assuming that the local geometry around a mesh point is preserved. The method for obtaining the 2D flattened pattern was shown to be straightforward once we have done all the mapping calculations.

In order to prevent anomalous events, we developed several methods; inserting "darts", finding a set of good initial conditions for a fitting, and modifying the surface shape that minimizes the possibility of anomalous events.

For the method of inserting darts, we presented several models for darts and addressed the problems of where darts should be inserted and how to determine an optimal shape for the darts based on several optimizing criteria. A solution to these problems was given by the following steps; First, identify critical regions where excessive deformation may occur. Second, insert a linear cut into each critical region. Finally, traverse overlapping mesh points along the linear cut and obtain the dart
shape. This algorithm is applied to a *lower critical region* generated by a fitting. For an *upper critical region*, we need additional ply fragments to cover the surface completely.

For the method of finding a set of good initial conditions for a fitting and for the method of modifying the surface shape that minimizes the possibility of anomalous events, we took advantage of the differential geometric properties that are applied to a Tchebychev net. Specifically, the first method assumes that the shape of a given 3D surface is fixed, and it searches for a set of good initial conditions including the starting point where the magnitude of the Gaussian curvature is maximum, and the base path that is aligned with a geodesic whose initial direction is given by the principal direction at the starting point. The second method assumes that the initial conditions are fixed, and it searches for an optimal shape for the given 3D surface where a woven cloth composite ply is fitted.

Given two surface shapes where one is obtained by modifying another, we call the modified shape *simpler with respect to global conformability*, if the integral Gaussian curvature of the modified shape is smaller than the original shape. A better shape was defined as the shape that is simpler with respect to global conformability and whose deviation from the original surface shape is within a certain threshold value. It was shown that this problem was translated into a constrained optimization problem whose slack variables are the weights of the control points that define the given NURBS surface. As long as the modified weights are positive, it is guaranteed that the new shape is within the convex hull of the control polygon. However, there is a case in which the control polygon is either too tight or too loose to flexibly control the surface shape. To cope with this, we have developed a method that combines the adjustment of the control polygon with the constrained optimization of the weights.

Our computer simulations in Chapters 5 through 7 exhibit encouraging results, from which we believe that our fitting algorithms as well as the algorithm for inserting darts, the algorithm for finding a set of good initial conditions, and
the algorithm for modifying the surface shape, can be used to assist the design and manufacturing process for forming woven cloth composites.

Several physical fitting experiments, by using fiberglass and Kevlar as woven cloth composite materials and a sphere and a cornered object as 3D surfaces, were conducted to verify our computer simulations. A great similarity was observed between our computer simulation results and the physical fitting results, from which we can conclude that the model for woven cloth composite plies is sufficiently accurate and our fitting algorithms are applicable to predicting the deformation that may occur in forming woven cloth composites.

Our investigation on the fitting mechanism has been focused on the geometric aspects. However, we also addressed the physical aspects in terms of the “formability” of a 3D woven cloth composite, or the measure of how much energy is necessary to deform a given woven cloth composite ply into its final shape. Shear and bending deformation energies were formulated, based on the assumption that the deformation is within the range of linear elasticity. Given two or more fitting simulations, we can compare them and predict which one produces the best fitting result in terms of the formability, the presence of anomalous events, and the area in 2D cloth used for the fitting.

The development of the algorithms mentioned so far leads to a methodology for a computer-aided design system for forming woven cloth composites. We have stated necessary functions for constructing such a CAD system that include: (1) an accurate model of woven cloth composites, (2) a flexible model for 3D surfaces, (3) a mapping method between 2D cloth space and 3D surface space, (4) a method for 2D plane development, (5) a method for handling “darts”, and (6) a method for predicting and preventing anomalous events such as wrinkling and tearing. The models and methods are exactly what we have described in the previous chapters. In addition, the following graphic tools have been developed to support computer simulations:

- a tool to plot and shade the fitting result of a ply of woven cloth composites
to an arbitrary 3D NURBS surface under a given set of initial conditions

- a tool to plot and shade the 2D plane development used for producing the fitting result
- a tool to plot and shade the thread angles between wefts and warps as height fields projected above the 2D plane development
- a tool to shade a given NURBS surface with the Gaussian curvature at each point on the surface
- a tool to shade a modified NURBS surface with the deviation value at each point on the surface, corresponding to the point on the original surface
- a tool to plot the intermediate Gaussian curvature integrals as height fields projected above the 2D plane development
- a tool to interactively insert “trimmed darts” (polygonal cut and linear cut) into a 2D ply of woven cloth composites

These tools are incorporated into a prototype CAD system for forming woven cloth composites. Figure 8.1 summarizes an outline of the current system that allows designers to repeatedly simulate a wide variety of fitting configurations and the corresponding 2D flattened patterns, that incorporates the algorithms mentioned in the previous chapters.

### 8.2 Future Recommendations

There is still much work to be done in this area of research. The following are recommendations for future work and further study related to the problems we have mentioned so far.

- Extend the fitting simulation capability such that it can simulate the lay-up procedure involving multiple layers of plies piled up one on top of another.
Figure 8.1: Overall process flow of the CAD system for simulating the layout of woven cloth composites. The comment put in parentheses represent the sections and the algorithms that describe the process.
Introduce the thickness of each ply and consider how the shape of each layer is changed, and formulate the new shape as an offset surface of the surface one layer beneath.

- Implement an alternative fitting method based on the finite difference method (as discussed in Section 7.2), and compare the results with those obtained by the current method.

- Explore ways of inserting linear cuts automatically in order to find an optimal dart shape.

- Enhance the system that allows us to simulate the physical behavior of the deformed woven cloth. For instance, internal stresses at each mesh point can be predicted by considering the equilibrium between external forces including gravitational forces and (shear and bending) deformations developed in the cloth. Similarly, it may be possible to simulate how wrinkles are developed in a critical region.

There should be potentially more research problems for further study on the deformation mechanism of woven cloth composites and its applications. It is hoped that the problems and their solutions described in this thesis lay a groundwork for future investigations related to the geometric aspects for forming woven cloth composites.
LITERATURE CITED


[66] Product Definition and Exchange Specification (PDES), IGES Internal Report, NIST, Gaithersburg, MD (1987)


APPENDIX A
PARTIAL DERIVATIVES OF A NURBS SURFACE

In this appendix, partial derivatives of an arbitrary order of a NURBS surface \( r(u, v) \) with respect to parameters \( u \) and \( v \) are described. The derivatives of typical curves and surfaces in computer-aided geometric design such as Bezier curves, B-spline curves, and tensor product Bezier surfaces are described in ordinary textbooks for computer-aided geometric design [29, 31, 87], but the derivatives of a NURBS surface are not provided, partly because they are quite complicated and partly because they are not utilized in most of the algorithms. However, if we need to obtain the normal vector at an arbitrary point on a NURBS surface, we need the first partial derivatives. Similarly, if we need to obtain the Gaussian curvature at an arbitrary point on a NURBS surface, we need the second partials including the mixed partial with respect to parameters \( u \) and \( v \).

For the first order partial derivative, we provide only the formula with respect to parameter \( u \). The formula with respect to parameter \( v \) can be obtained similarly. Since \( r(u, v) \) can be expressed as \( r(u, v) = (x(u, v), y(u, v), z(u, v)) \), the first order partial derivative of \( r(u, v) \) with respect to parameter \( u \) is given as follows:

\[
\frac{\partial r}{\partial u}(u, v) \equiv \left( \frac{\partial x}{\partial u}, \frac{\partial y}{\partial u}, \frac{\partial z}{\partial u} \right) = \frac{\partial}{\partial u} \left( R(u, v) W(u, v) \right) = \frac{1}{W^2} \begin{vmatrix} \frac{\partial R}{\partial u} & R \\ \frac{\partial W}{\partial u} & W \end{vmatrix}, \quad (A.1)
\]

where

\[
R(u, v) = \sum_{i=0}^{C_u-1} \sum_{j=0}^{C_v-1} w_{i,j} c_{i,j} N_i^m(u) N_j^n(v), \quad (A.2)
\]

\[
W(u, v) = \sum_{i=0}^{C_u-1} \sum_{j=0}^{C_v-1} w_{i,j} N_i^m(u) N_j^n(v). \quad (A.3)
\]
Partial derivatives for \( R(u, v) \) and \( W(u, v) \) are expressed as follows:

For \( R \):

\[
\frac{\partial R}{\partial u} = m \sum_{i=0}^{C_u-1} \sum_{j=0}^{C_v-1} \frac{\Delta^{1,0} D_{i,j}}{u_{m+i+1}-u_{i+1}} N_i^{m-1}(u) N_j^n(v) \\
\frac{\partial R}{\partial v} = n \sum_{i=0}^{C_u-1} \sum_{j=0}^{C_v-1} \frac{\Delta^{0,1} D_{i,j}}{v_{n+j+1}-v_{j+1}} N_i^m(u) N_j^{n-1}(v)
\]

(A.4)

For \( W \):

\[
\frac{\partial W}{\partial u} = m \sum_{i=0}^{C_u-1} \sum_{j=0}^{C_v-1} \frac{\Delta^{1,0} w_{i-1,j}}{u_{m+i+1}-u_{i+1}} N_i^{m-1}(u) N_j^n(v) \\
\frac{\partial W}{\partial v} = n \sum_{i=0}^{C_u-1} \sum_{j=0}^{C_v-1} \frac{\Delta^{0,1} w_{i,j-1}}{v_{n+j+1}-v_{j+1}} N_i^m(u) N_j^{n-1}(v)
\]

(A.5)

where, the notation of \( \Delta^{r,s} \) is a backward difference operator defined as below:

\[
\Delta^{r,s} w_{i,j} = \Delta^{r-1,s} w_{i,j} - \Delta^{r-1,s} w_{i-1,j} = \Delta^{r,s-1} w_{i,j} - \Delta^{r,s-1} w_{i,j-1},
\]

where

\[
\Delta^{0,0} w_{i,j} = w_{i,j},
\]

and

\[
\Delta^{0,0} D_{i,j} = w_{i,j} c_{i,j}.
\]

Note that in Equations (A.4) and (A.5) we assume that \( u_{-1} = v_{-1} = 0 \).

Before describing the second partials, it is helpful to introduce a formula of a partial derivative of \( 2 \times 2 \) determinant whose components are functions of \( u \) and \( v \), as follows:

\[
\frac{\partial}{\partial u} \begin{vmatrix} f(u, v) & g(u, v) \\ h(u, v) & k(u, v) \end{vmatrix} = \frac{\partial f}{\partial u} - f \begin{vmatrix} \frac{\partial g}{\partial u} - g \\ \frac{\partial h}{\partial u} - h \end{vmatrix}.
\]

(A.6)

With the help of this formula, the second partials of a NURBS surface (\( \frac{\partial^2}{\partial u^2}(r(u, v)) \) and \( \frac{\partial^2}{\partial u \partial v}(r(u, v)) \)) are described as below:
\[
\frac{\partial^2}{\partial u^2} \left( \frac{R(u,v)}{W(u,v)} \right) = \frac{1}{W^2} \begin{vmatrix}
\frac{\partial^2 R}{\partial u^2} & \frac{\partial R}{\partial u} \\
\frac{\partial W}{\partial u} & W
\end{vmatrix}
- \frac{2}{W^3} \frac{\partial W}{\partial u} \begin{vmatrix}
\frac{\partial R}{\partial u} & R \\
\frac{\partial W}{\partial u} & W
\end{vmatrix},
\]

\[
\frac{\partial^2}{\partial u \partial v} \left( \frac{R(u,v)}{W(u,v)} \right) = \frac{1}{W^2} \begin{vmatrix}
\frac{\partial^2 R}{\partial u \partial v} & \frac{\partial R}{\partial u} \\
\frac{\partial W}{\partial u} & W
\end{vmatrix}
- \frac{2}{W^3} \frac{\partial W}{\partial v} \begin{vmatrix}
\frac{\partial R}{\partial v} & R \\
\frac{\partial W}{\partial v} & W
\end{vmatrix}.
\]

Let \( \Delta^{r,s} \) denote a generalized divided difference operator which is defined as:

\[
\Delta^{r,s} D_{ij}[u^m; v^n] = \Delta^{r-1,s} D_{i-1,j}[u^m_{p-1}; v^n_{q}] - \Delta^{r-1,s} D_{i,j-1}[u^m_{p}; v^n_{q-1}],
\]

\[
\frac{\partial^r \partial^s}{\partial u^r \partial v^s} D^{m,n}(u,v) = \frac{m!n!}{(m-r)!(n-s)!} \sum_{i=0}^{Cu-1} \sum_{j=0}^{Cv-1} \Delta^{r,s} D_{i,j}[u^m_{p}; v^n_{q}] N_{i}^{m-r}(u) N_{j}^{n-s}(v).
\]
APPENDIX B
DATA STRUCTURES FOR A COMPOSITE NURBS SURFACE

typedef struct Patch {
    /* W: u = umin boundary */
    /* E: u = umax boundary */
    /* S: v = vmin boundary */
    /* N: v = vmax boundary */
    struct NURBS_3S *nptr;
} Patch;

typedef struct NURBS_3S {
    /* rational bit field indicates whether the patch is really */
    /* */
    /* */
} NURBS_3S;
/* rational or not. If not, all weights are set 1.0. */
/*
/* There is a relationship among degree[i], numKNOT[i], and numCPNT[i]: numKNOT[i] - degree[i] + 1 = numCPNT[i]
/*
/* Actual valid surface domain extends from knot[i][n-1] to knot[i][L+n-1], where n = degree[i] and L = numCPNT[i] - degree[i].
/*
/* The Basis spline N(n,j) is nonnegative over (knot[i][j-1], knot[i][j+n]), where n = degree[i].
/*
/***************************************************************************/
typedef struct NURBS_3S {
        int nid;    /* NURBS Surface ID */
        struct Patch *pat;
        unsigned int rational; /* =0 (non-rational), =1(rational) */
        int degree[2]; /* degree of surface */
        /* [0]: along u-axis, [1]: along v-axis */
        int numKNOT[2]; /* number of knot vectors */
        /* [0]: along u-axis, [1]: along v-axis */
        REAL *knot[2]; /* knot vectors */
        /* [0]: along u-axis, [1]: along v-axis */
        REAL umin, umax, vmin, vmax; /* min-max knot values */
        int numCPNT[2]; /* number of control points */
        /* [0]: along u-axis, [1]: along v-axis */
        REAL **CPNTx; /* control points for x coord */
        REAL **CPNTy; /* control points for y coord */
        REAL **CPNTz; /* control points for z coord */
        REAL **weight; /* weights */
} NURBS_3S;

/***************************************************************************/
/* Function: PatchCode(surface, u, v) */
/* Description: This function returns a code that corresponds to */
/* a surface region where the given point (u,v) locates. */
/* If the point is inside the region, 0 is returned. */
/* Otherwise, it returns either WEST, EAST, SOUTH, NORTH,*/
/* or combination of them. */
/***************************************************************************/
int PatchCode(surface, u, v)
    NURBS_3S *surface;
    REAL u, v;
{
        int rc;

        rc = 0;
        if       (|u - surface->umin| > EPSILON && u < surface->umin)
rc |= WEST;
else if (|u - surface->umax| > EPSILON && u > surface->umax)
    rc |= EAST;
if (|v - surface->vmin| > EPSILON && v < surface->vmin)
    rc |= SOUTH;
else if (|v - surface->vmax| > EPSILON && v > surface->vmax)
    rc |= NORTH;
return(rc);
APPENDIX C
DATA STRUCTURES AND ALGORITHMS FOR SCANNING MESH POINTS NOT ON THE BASE PATH

typedef struct Grid {
    int sid; /* NURBS surface id for the mapped point */
    REAL Wd, Ed, Sd, Nd; /* distance between four adjacent grids */
    Vector_3 mapped_XYZ_pos; /* mapped XYZ coordinate */
    Vector_2 mapped_UV_pos; /* mapped UV coordinate */
    int ipos, jpos; /* original 2D position of the mesh point */
    REAL gauss; /* Gaussian curvature at the point */
    REAL int_gauss; /* intermediate Gaussian curvature integral */
    unsigned int done; /* calculation is done or not */
    unsigned int out; /* out of surface boundary or not */
    unsigned int dart; /* on the dart or not */
    unsigned int regular; /* a regular mesh point or not */
} Grid;

typedef struct Frontier_List {
    /* ...
    */
} Frontier_List;

248
/* A Frontier_List data structure defines a linked list of */
/* Grids that represents a sorted candidates for the mapping */
/* calculation. It also holds data that represents the number */
/* of known neighbors and their directions. */
/* *******************************************************************/

typedef struct Frontier_List {
    struct Grid *G; /* pointer to a Grid (on the path) */
    struct Frontier_List *next;
    unsigned int counter:2; /* the number of known neighbors */
    unsigned int known:4; /* known neighbor directions */
} Frontier_List;

/* Function: InsertFrontier(Head, Gold, code) */
/* Description: This function looks for the mesh point "Gold" in */
/* the frontier list. It it exists, then increments the*/
/* known neighbor counter; otherwise inserts a new */
/* element by insertion sorting. */
/* *******************************************************************/

void InsertFrontier(Frontier_List *Head, Grid *Gold, int code)
{
    Frontier_List *Frontier, *FrontierOld, *FrontierNew;
    Frontier = Head;
    while (Frontier != NULL){
        FrontierOld = Frontier;
        if (Frontier->G == Gold) {
            /* already in the frontier list */
            /* already in the frontier list */
            Frontier->count++;
            return;
        }
        Frontier = Frontier->next;
    }
    FrontierNew = AllocateFrontierElement();
    FrontierNew->G = Gold;
    FrontierNew->count = 1;
    FrontierNew->known = code;
    FrontierNew->next = NULL;
    InsertionSort(Head, FrontierNew);
}

/* Function: TraceFrontier(Head) */
/* Description: This function traces the frontier list from its */
/* head element and either performs an ordinary mapping*/
/* calculation (when known neighbors are two) or */
/* approximates the mapping (when known neighbor is */
/* one) until the frontier list becomes empty. */
/* *******************************************************************/

/* Function: InsertFrontier(Head, Gold, code) */
/* Description: This function looks for the mesh point "Gold" in */
/* the frontier list. It it exists, then increments the*/
/* known neighbor counter; otherwise inserts a new */
/* element by insertion sorting. */
/* *******************************************************************/

void InsertFrontier(Frontier_List *Head, Grid *Gold, int code)
{
    Frontier_List *Frontier, *FrontierOld, *FrontierNew;
    Frontier = Head;
    while (Frontier != NULL){
        FrontierOld = Frontier;
        if (Frontier->G == Gold) {
            /* already in the frontier list */
            /* already in the frontier list */
            Frontier->count++;
            return;
        }
        Frontier = Frontier->next;
    }
    FrontierNew = AllocateFrontierElement();
    FrontierNew->G = Gold;
    FrontierNew->count = 1;
    FrontierNew->known = code;
    FrontierNew->next = NULL;
    InsertionSort(Head, FrontierNew);
}

/* Function: TraceFrontier(Head) */
/* Description: This function traces the frontier list from its */
/* head element and either performs an ordinary mapping*/
/* calculation (when known neighbors are two) or */
/* approximates the mapping (when known neighbor is */
/* one) until the frontier list becomes empty. */
/* *******************************************************************/

/* Function: InsertFrontier(Head, Gold, code) */
/* Description: This function looks for the mesh point "Gold" in */
/* the frontier list. It it exists, then increments the*/
/* known neighbor counter; otherwise inserts a new */
/* element by insertion sorting. */
/* *******************************************************************/

void InsertFrontier(Frontier_List *Head, Grid *Gold, int code)
{
    Frontier_List *Frontier, *FrontierOld, *FrontierNew;
    Frontier = Head;
    while (Frontier != NULL){
        FrontierOld = Frontier;
        if (Frontier->G == Gold) {
            /* already in the frontier list */
            /* already in the frontier list */
            Frontier->count++;
            return;
        }
        Frontier = Frontier->next;
    }
    FrontierNew = AllocateFrontierElement();
    FrontierNew->G = Gold;
    FrontierNew->count = 1;
    FrontierNew->known = code;
    FrontierNew->next = NULL;
    InsertionSort(Head, FrontierNew);
}

/* Function: TraceFrontier(Head) */
/* Description: This function traces the frontier list from its */
/* head element and either performs an ordinary mapping*/
/* calculation (when known neighbors are two) or */
/* approximates the mapping (when known neighbor is */
/* one) until the frontier list becomes empty. */
/* *******************************************************************/

/* Function: InsertFrontier(Head, Gold, code) */
/* Description: This function looks for the mesh point "Gold" in */
/* the frontier list. It it exists, then increments the*/
/* known neighbor counter; otherwise inserts a new */
/* element by insertion sorting. */
/* *******************************************************************/

void InsertFrontier(Frontier_List *Head, Grid *Gold, int code)
{
    Frontier_List *Frontier, *FrontierOld, *FrontierNew;
    Frontier = Head;
    while (Frontier != NULL){
        FrontierOld = Frontier;
        if (Frontier->G == Gold) {
            /* already in the frontier list */
            /* already in the frontier list */
            Frontier->count++;
            return;
        }
        Frontier = Frontier->next;
    }
    FrontierNew = AllocateFrontierElement();
    FrontierNew->G = Gold;
    FrontierNew->count = 1;
    FrontierNew->known = code;
    FrontierNew->next = NULL;
    InsertionSort(Head, FrontierNew);
}

/* Function: TraceFrontier(Head) */
/* Description: This function traces the frontier list from its */
/* head element and either performs an ordinary mapping*/
/* calculation (when known neighbors are two) or */
/* approximates the mapping (when known neighbor is */
/* one) until the frontier list becomes empty. */
/* *******************************************************************/
void TraceFrontier(Frontier_List *Head) {
    Frontier_List *Frontier;
    Frontier = Head;
    START:
    while (Frontier != NULL) {
        G = Frontier->G;
        if (G == NULL || G->done) return;
        /* check if the head element in the frontier list 
        has two known neighbors. */
        if (Frontier->counter == 2) {
            MappingCalculation(G);
            RemoveElement(Frontier);
            if (Frontier->known == SOUTH_WEST) {
                GE = G->E; GN = G->N;
                if (GE != NULL && (!GE->done) && (!GE->out))
                    InsertFrontier(Frontier, GE, WEST);
                if (GN != NULL && (!GN->done) && (!GN->out))
                    InsertFrontier(Frontier, GN, SOUTH);
            }
            else if (Frontier->known == SOUTH_EAST) {
                GW = G->W; GN = G->N;
                if (GW != NULL && (!GW->done) && (!GW->out))
                    InsertFrontier(Frontier, GW, EAST);
                if (GN != NULL && (!GN->done) && (!GN->out))
                    InsertFrontier(Frontier, GN, SOUTH);
            }
            else if (Frontier->known == NORTH_WEST) {
                GE = G->E; GS = G->S;
                if (GE != NULL && (!GE->done) && (!GE->out))
                    InsertFrontier(Frontier, GE, WEST);
                if (GS != NULL && (!GS->done) && (!GS->out))
                    InsertFrontier(Frontier, GS, NORTH);
            }
            else if (Frontier->known == NORTH_EAST) {
                GW = G->W; GS = G->S;
                if (GW != NULL && (!GW->done) && (!GW->out))
                    InsertFrontier(Frontier, GW, EAST);
                if (GS != NULL && (!GS->done) && (!GS->out))
                    InsertFrontier(Frontier, GS, NORTH);
            }
            continue;
        }
        Frontier = Frontier->next;
    }
    /* Now, elements in the frontier list have one known neighbor. */
    while (Frontier != NULL) {
        G = Frontier->G;
        if (G == NULL || G->done) return;
        /* check if the head element in the frontier list 
        has two known neighbors. */
        if (Frontier->counter == 2) {
            MappingCalculation(G);
            RemoveElement(Frontier);
            if (Frontier->known == SOUTH_WEST) {
                GE = G->E; GN = G->N;
                if (GE != NULL && (!GE->done) && (!GE->out))
                    InsertFrontier(Frontier, GE, WEST);
                if (GN != NULL && (!GN->done) && (!GN->out))
                    InsertFrontier(Frontier, GN, SOUTH);
            }
            else if (Frontier->known == SOUTH_EAST) {
                GW = G->W; GN = G->N;
                if (GW != NULL && (!GW->done) && (!GW->out))
                    InsertFrontier(Frontier, GW, EAST);
                if (GN != NULL && (!GN->done) && (!GN->out))
                    InsertFrontier(Frontier, GN, SOUTH);
            }
            else if (Frontier->known == NORTH_WEST) {
                GE = G->E; GS = G->S;
                if (GE != NULL && (!GE->done) && (!GE->out))
                    InsertFrontier(Frontier, GE, WEST);
                if (GS != NULL && (!GS->done) && (!GS->out))
                    InsertFrontier(Frontier, GS, NORTH);
            }
            else if (Frontier->known == NORTH_EAST) {
                GW = G->W; GS = G->S;
                if (GW != NULL && (!GW->done) && (!GW->out))
                    InsertFrontier(Frontier, GW, EAST);
                if (GS != NULL && (!GS->done) && (!GS->out))
                    InsertFrontier(Frontier, GS, NORTH);
            }
            continue;
        }
        Frontier = Frontier->next;
    }
}
G = Frontier->G;
if (Frontier->counter == 1){
    ApproximateMapping(G);
    RemoveElement(Frontier);
    if (Frontier->known == WEST){
        GE = G->E; GS = G->S; GN = G->N;
        if (GE != NULL && (!GE->done) && (!GE->out))
            InsertFrontier(Frontier, GE, WEST);
        if (GS != NULL && (!GS->done) && (!GS->out))
            InsertFrontier(Frontier, GS, NORTH);
        if (GN != NULL && (!GN->done) && (!GN->out))
            InsertFrontier(Frontier, GN, SOUTH);
    }
    else if (Frontier->known == EAST){
        GW = G->W; GS = G->S; GN = G->N;
        if (GW != NULL && (!GW->done) && (!GW->out))
            InsertFrontier(Frontier, GW, EAST);
        if (GS != NULL && (!GS->done) && (!GS->out))
            InsertFrontier(Frontier, GS, NORTH);
        if (GN != NULL && (!GN->done) && (!GN->out))
            InsertFrontier(Frontier, GN, SOUTH);
    }
    else if (Frontier->known == SOUTH){
        GW = G->W; GE = G->E; GN = G->N;
        if (GW != NULL && (!GW->done) && (!GW->out))
            InsertFrontier(Frontier, GW, EAST);
        if (GE != NULL && (!GE->done) && (!GE->out))
            InsertFrontier(Frontier, GE, WEST);
        if (GN != NULL && (!GN->done) && (!GN->out))
            InsertFrontier(Frontier, GN, SOUTH);
    }
    else if (Frontier->known == NORTH){
        GW = G->W; GE = G->E; GS = G->S;
        if (GW != NULL && (!GW->done) && (!GW->out))
            InsertFrontier(Frontier, GW, EAST);
        if (GE != NULL && (!GE->done) && (!GE->out))
            InsertFrontier(Frontier, GE, WEST);
        if (GS != NULL && (!GS->done) && (!GS->out))
            InsertFrontier(Frontier, GS, NORTH);
    }
    goto START;
}
Frontier = Frontier->next;
}
Here we list the algorithm for obtaining an intermediate Gaussian curvature integral $I_{K_{i,j}}$ at $\text{Grid}[i][j]$, where $\text{Grid}[i][j]$ is a global variable representing a pointer to a mesh point at $(i,j)$ position in a given rectangular woven cloth composite ply.

```c
/* Global Variables: */
char **_Gauss; /* a two dimensional array of flags: if _Gauss[i][j]
is ON, the mesh point at (i,j) contributes to the calculation of the intermediate Gaussian curvature integral */
Grid ***_Grid; /* two dimensional array of pointers to Grid data structure */
REAL _Dx; /* distance between adjacent crossing along a weft */
REAL _Dy; /* distance between adjacent crossing along a warp */

/* Function: GaussLookup(Grid *G) */
/* Description: This function returns 1 if the mesh point is already summed up; otherwise it sets the global variable _Gauss[][] and returns 0. */
int GaussLookup(Grid *G)
{
    int i, j;

    i = G->ipos; j = G->jpos;
    if (_Gauss[i][j]) return(1);
    _Gauss[i][j] = ON;
    return(0);
}

/* Function: Trace(Grid *G) */
/* Description: This function calculates the list of mesh points that have been swept so far, which contribute to the mapping calculation of the current mesh point by taking set union recursively. */
void Trace(Grid *G)
{
}
```
int c;
if (GaussLookup(G)) return;
c = G->known;
if (G->neighbor == 1){
    if (c == WEST) Trace(G->W);
    else if (c == EAST) Trace(G->E);
    else if (c == SOUTH) Trace(G->S);
    else if (c == NORTH) Trace(G->N);
}
else if (G->neighbor == 2){
    if (c == SOUTH_WEST) {
        Trace(G->S); Trace(G->W);
    }
    else if (c == SOUTH_EAST){
        Trace(G->S); Trace(G->E);
    }
    else if (c == NORTH_WEST){
        Trace(G->N); Trace(G->W);
    }
    else if (c == NORTH_EAST){
        Trace(G->N); Trace(G->E);
    }
}

_REAL IntegralGauss(Grid *G) {
    int i, j;
    REAL sum;

    Trace(G);
    sum = 0.0;
    for (i = 0; i < numWEFT ; i++ )
        for (j = 0; j < numWARP ; j++ ){
            if (_Gauss[i][j]){  
                _Gauss[i][j] = OFF;
                sum += _Grid[i][j]->gauss * _Dx * _Dy;
            }
        }
    return(g);

}
APPENDIX E
THE RULE OF APPENDING FRACTIONAL FRAGMENTS TO
BOUNDARY MESH POINTS IN BOUNDARY TRAVERSAL

There are eight different cases for the rule of appending fractional fragments in boundary traversal. In the following, we sequentially list these eight cases, classified by the direction from the current mesh point to the next mesh point. For each case, we enumerate possible previous directions with respect to the current mesh point, the associated mesh diagrams, and the resulting sequences of the boundary traversal.

The meaning of the symbols in the mesh diagrams is as follows: The label ‘1’ is attached to the current mesh point, and the label ‘2’ to the next mesh point that we should traverse. There is one mesh point in each mesh diagram that has no label, and it represents the previous mesh point. All of the mesh points represent boundary mesh points. The direction specified in the ‘Previous Direction’ column corresponds to the direction from the previous mesh point to the mesh point labeled ‘1’. The solid arrow specifies the previous direction and the next direction. The line segments dangling from the current and the next mesh point represent the fractional fragments we will add in the boundary traversal. The dotted arrow represents the direction of the traverse, which corresponds to the sequence specified in the ‘Traced Sequence’ column.

The meaning of the symbols in the ‘Traced Sequence’ column is as follows: Symbols Wd, Ed, Sd, and Nd denote the values of the fractional fragments in WEST, EAST, SOUTH, and NORTH directions, respectively. The number in the parenthesis attached to each of these fractional fragment symbols represents the mesh label shown in the mesh diagram. For example, if the traced sequence is “Ed(1) → Sd(1) → Sd(2)”, it means that the resulting traversal starts with the position at the EAST tip of the fractional fragment of the mesh labeled ‘1’, then
goes to the SOUTH tip of the same mesh, and finally goes to the SOUTH tip of the fractional fragments of the mesh labeled ‘2’.
Case 1: boundary grids toward WEST.

<table>
<thead>
<tr>
<th>Previous Direction</th>
<th>Grid Diagram</th>
<th>Traced Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td>![Grid Diagram W]</td>
<td>Sd(1) -&gt; Sd(2)</td>
</tr>
<tr>
<td>NW</td>
<td>![Grid Diagram NW]</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>![Grid Diagram E]</td>
<td>Nd(1) ➞ Ed(1) ➞ Sd(1) ➞ Sd(2)</td>
</tr>
<tr>
<td>SE</td>
<td>![Grid Diagram SE]</td>
<td></td>
</tr>
<tr>
<td>S</td>
<td>![Grid Diagram S]</td>
<td>Ed(1) ➞ Sd(1) ➞ Sd(2)</td>
</tr>
<tr>
<td>SW</td>
<td>![Grid Diagram SW]</td>
<td></td>
</tr>
</tbody>
</table>

Case 2: boundary grids toward EAST.

<table>
<thead>
<tr>
<th>Previous Direction</th>
<th>Grid Diagram</th>
<th>Traced Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td>![Grid Diagram E]</td>
<td>Nd(1) ➞ Nd(2)</td>
</tr>
<tr>
<td>SE</td>
<td>![Grid Diagram SE]</td>
<td></td>
</tr>
<tr>
<td>W</td>
<td>![Grid Diagram W]</td>
<td>Sd(1) ➞ Wd(1) ➞ Nd(1) ➞ Nd(2)</td>
</tr>
<tr>
<td>NW</td>
<td>![Grid Diagram NW]</td>
<td></td>
</tr>
<tr>
<td>N</td>
<td>![Grid Diagram N]</td>
<td>Wd(1) ➞ Nd(1) ➞ Nd(2)</td>
</tr>
<tr>
<td>NE</td>
<td>![Grid Diagram NE]</td>
<td></td>
</tr>
</tbody>
</table>
**Case 3: boundary grids toward SOUTH.**

<table>
<thead>
<tr>
<th>Previous Direction</th>
<th>Grid Diagram</th>
<th>Traced Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td><img src="image" alt="Diagram" /></td>
<td>Ed(1) -&gt; Ed(2)</td>
</tr>
<tr>
<td>SW</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td><img src="image" alt="Diagram" /></td>
<td>Wd(1) -&gt; Nd(1) -&gt; Ed(1) -&gt; Ed(2)</td>
</tr>
<tr>
<td>NE</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td><img src="image" alt="Diagram" /></td>
<td>Nd(1) -&gt; Ed(1) -&gt; Ed(2)</td>
</tr>
<tr>
<td>SE</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
</tbody>
</table>

**Case 4: boundary grids toward NORTH.**

<table>
<thead>
<tr>
<th>Previous Direction</th>
<th>Grid Diagram</th>
<th>Traced Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td><img src="image" alt="Diagram" /></td>
<td>Wd(1) -&gt; Wd(2)</td>
</tr>
<tr>
<td>NE</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td><img src="image" alt="Diagram" /></td>
<td>Ed(1) -&gt; Sd(1) -&gt; Wd(1) -&gt; Wd(2)</td>
</tr>
<tr>
<td>SW</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td><img src="image" alt="Diagram" /></td>
<td>Sd(1) -&gt; Wd(1) -&gt; Wd(2)</td>
</tr>
<tr>
<td>NW</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
</tbody>
</table>
### Case 5: boundary grids toward SOUTH-WEST.

<table>
<thead>
<tr>
<th>Previous Direction</th>
<th>Grid Diagram</th>
<th>Traced Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>W</td>
<td><img src="image1" alt="W Grid Diagram" /></td>
<td>Nd(1) -&gt; Ed(1) -&gt; Sd(1) -&gt; Ed(2)</td>
</tr>
<tr>
<td>NW</td>
<td><img src="image2" alt="NW Grid Diagram" /></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td><img src="image3" alt="E Grid Diagram" /></td>
<td>Nd(1) -&gt; Ed(1) -&gt; Sd(1) -&gt; Ed(2)</td>
</tr>
<tr>
<td>SE</td>
<td><img src="image4" alt="SE Grid Diagram" /></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td><img src="image5" alt="S Grid Diagram" /></td>
<td>Ed(1) -&gt; Sd(1) -&gt; Ed(2)</td>
</tr>
<tr>
<td>SW</td>
<td><img src="image6" alt="SW Grid Diagram" /></td>
<td></td>
</tr>
</tbody>
</table>

### Case 6: boundary grids toward SOUTH-EAST.

<table>
<thead>
<tr>
<th>Previous Direction</th>
<th>Grid Diagram</th>
<th>Traced Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>S</td>
<td><img src="image7" alt="S Grid Diagram" /></td>
<td>Ed(1) -&gt; Nd(1) -&gt; Ed(2)</td>
</tr>
<tr>
<td>SW</td>
<td><img src="image8" alt="SW Grid Diagram" /></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td><img src="image9" alt="N Grid Diagram" /></td>
<td>Nd(1) -&gt; Ed(1) -&gt; Nd(1) -&gt; Ed(2)</td>
</tr>
<tr>
<td>NE</td>
<td><img src="image10" alt="NE Grid Diagram" /></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td><img src="image11" alt="E Grid Diagram" /></td>
<td>Nd(1) -&gt; Ed(1) -&gt; Nd(2)</td>
</tr>
<tr>
<td>SE</td>
<td><img src="image12" alt="SE Grid Diagram" /></td>
<td></td>
</tr>
</tbody>
</table>
### Case 7: boundary grids toward NORTH-WEST.

<table>
<thead>
<tr>
<th>Previous Direction</th>
<th>Grid Diagram</th>
<th>Traced Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>N</td>
<td><img src="image" alt="Diagram" /></td>
<td>Wd(1) -&gt; Sd(2)</td>
</tr>
<tr>
<td>NE</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td>S</td>
<td><img src="image" alt="Diagram" /></td>
<td>Ed(1) -&gt; Sd(1) -&gt; Wd(1) -&gt; Sd(2)</td>
</tr>
<tr>
<td>SW</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td><img src="image" alt="Diagram" /></td>
<td>Sd(1) -&gt; Wd(1) -&gt; Sd(2)</td>
</tr>
<tr>
<td>NW</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
</tbody>
</table>

### Case 8: boundary grids toward NORTH-EAST.

<table>
<thead>
<tr>
<th>Previous Direction</th>
<th>Grid Diagram</th>
<th>Traced Sequence</th>
</tr>
</thead>
<tbody>
<tr>
<td>E</td>
<td><img src="image" alt="Diagram" /></td>
<td>Nd(1) -&gt; Wd(2)</td>
</tr>
<tr>
<td>SE</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td>W</td>
<td><img src="image" alt="Diagram" /></td>
<td>Sd(1) -&gt; Wd(1) -&gt; Nd(1) -&gt; Wd(2)</td>
</tr>
<tr>
<td>NW</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td><img src="image" alt="Diagram" /></td>
<td>Wd(1) -&gt; Nd(1) -&gt; Wd(2)</td>
</tr>
<tr>
<td>NE</td>
<td><img src="image" alt="Diagram" /></td>
<td></td>
</tr>
</tbody>
</table>
APPENDIX F
COLOR ASSIGNMENT

Colors are used in some simulation experiments to indicate how a certain variable changes with the location of a given surface. The variables include the thread angle at a crossing between a weft and a warp, the Gaussian curvature, and the deviation of a modified surface relative to the original shape. The advantage of using colors over the wireframe plots is the ease of intuitive understanding of the simulation results.

In the following, let us take the thread angle ($\gamma$) as an example of the variable to which colors are assigned. The assignment of a color to a thread-angle value is represented by a mapping between an RGB color value ($r, g, b$) and a $\gamma$ value. There are potentially many ways of the mapping that provides a one-to-one correspondence between the two values.

Here we define the mapping in which ($r, g, b$) is first converted into a hue value ($\theta$) that ranges from 0° to 360°, and then the hue value $\theta$ is mapped to a thread-angle value $\gamma$. Note that hue is the pure “color” of a chromatic color [86]. Specifically, as shown in Figure F.1, a hue value is specified by the angle from “Red” in the hue hexagon [86]. Since the thread-angle value ranges from 0.0 radian to $\pi$ radian, we will use a part of the hue hexagon, ranging from “Red” (0°) to “Blue” (240°). We have implemented two types of the mapping functions that assign colors to the thread-angle values; linear function of a thread-angle value and non-linear function of a thread-angle value.

With the linear function of a thread angle, a hue value $\theta$ is represented by the following formula:

$$\theta = \frac{240}{\pi} \gamma. \tag{F.1}$$

Initially, $\gamma$ is set to $\frac{\pi}{2}$. The corresponding hue value is 120°, which is “Green.” As the thread-angle value decreases from $\frac{\pi}{2}$, the color is shifted toward “Red.” Similarly, as
the thread-angle value increases from $\frac{\pi}{2}$, the color is shifted toward “Blue.”

One of the useful applications of assigning colors to the thread-angle values is to identify the critical regions defined in Chapter 6. Assume that human eyes regard a color as “reddish” when the hue value goes below $30^\circ$ in the hue hexagon. With the linear function given by Equation (F.1), the beginning of “reddish” color is fixed ($\frac{\pi}{6}$). However, by suitably defining a non-linear function, we may adjust the beginning of “reddish” color such that it conforms to the locking angle ($\gamma_{lock}$) of a given material. In other words, we can assign “reddish” colors to the (lower) critical regions of a given material.

A simple idea of the non-linear function is to express the function as a cubic polynomial of a thread-angle value. In other words,

$$\theta = a(\gamma - \frac{\pi}{2})^3 + b(\gamma - \frac{\pi}{2})^2 + c(\gamma - \frac{\pi}{2}) + d,$$

where $a$, $b$, $c$, and $d$ are coefficients determined by some initial conditions. The initial conditions that we want to constrain to the cubic polynomial are the following:

- The function increases monotonically with the change of the thread-angle value.
from 0.0 radian to $\pi$ radian.

- The hue value becomes $120^\circ$ ("Green"), if the thread-angle value is $\frac{\pi}{2}$ radian, and the function is symmetric at this point.

- The hue value becomes $0^\circ$ ("Red"), if the thread-angle value is 0.0 radian.

From the first two conditions, we get $d = 120$ and $b = 0$. Therefore,

$$\theta = a(\gamma - \frac{\pi}{2})^3 + c(\gamma - \frac{\pi}{2}) + 120. \quad (F.2)$$

From the third condition, $a$ and $c$ in Equation (F.2) are related by

$$\pi^3 a + 4\pi c = 16. \quad (F.3)$$

In order to provide a one-to-one correspondence between a thread-angle value $\gamma$ and a hue value $\theta$, we need one additional condition. The material-dependent locking angle can be served as this additional condition. In other words, it is expressed as

$$\theta(\gamma_{\text{lock}}) = 30^\circ,$$

where the hue value $30^\circ$ is mapped to the locking angle. If we substitute this condition into Equation (F.2), we get

$$\lambda a + \mu c = -90, \quad (F.4)$$

where $\lambda = (\gamma_{\text{lock}} - \frac{\pi}{2})^3$ and $\mu = \gamma_{\text{lock}} - \frac{\pi}{2}$. Thus, from Equations (F.3) and (F.4), we can determine the coefficients $a$ and $c$.

It is interesting to see how critical regions vary as we change the ply material. Here we will compare two materials: a sheet of fiberglass of plain weave (used in Figures 5.26 and 7.13) and a sheet of Kevlar of 4-harness satin weave (used in Figures 6.26 and 6.27). The following table shows the parameters $d$ and $h$, defined in Figure 4.3.
<table>
<thead>
<tr>
<th>Material</th>
<th>weave type</th>
<th>$h$ (mm)</th>
<th>$d$ (mm)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fiberglass</td>
<td>plain</td>
<td>0.40</td>
<td>1.50</td>
</tr>
<tr>
<td>Kevlar</td>
<td>4-harness satin</td>
<td>0.75</td>
<td>1.60</td>
</tr>
</tbody>
</table>

Assuming that $\alpha = 0.5$ and $\beta = 0.0$ in Equation (4.2), we can determine the locking angles for the two materials. For the fiberglass sheet, $\gamma_{lock}^{geom} \approx 0.31$ radian, and for the Kevlar sheet, $\gamma_{lock}^{geom} \approx 0.52$ radian. These locking angles are finally added as the initial conditions for the non-linear function for the hue value. Specifically, for the fiberglass sheet, the condition that $\theta(0.31) = 30^\circ$ will be added, and for the Kevlar sheet, the condition that $\theta(0.52) = 30^\circ$ will be added. Figure F.2 compares the color assignments for fiberglass (Figure F.2 (a)) and Kevlar (Figure F.2 (b)) by using the fitting example of an octant of a sphere under the set of initial conditions shown in Table 5.2 (c). The result of the color assignments with the linear function is similar to that of the color assignments for the Kevlar sheet. This can be verified by assigning the hue value $30^\circ$ to Equation (F.1), by which we get the thread angle $\frac{\pi}{8} \approx 0.523$. From this figure, it is clear that no critical regions may result with the fiberglass sheet, but a critical region may appear at the corner of the octant of a sphere with the Kevlar sheet. This is why we employ fiberglass in our physical experiments, as demonstrated in Figure 5.26. Note that throughout the simulation examples in Chapters 5 through 7, where colors are assigned, we implicitly assume that the hue value is determined by the non-linear function whose initial conditions are set by the locking angle for the Kevlar sheet.
Figure F.2: Two different color assignments to the thread angles for a fitting to an octant of a sphere under the set of initial conditions given by Table 5.2 (c) in Section 5.7.1; (a) colors are assigned with the non-linear function for the fiberglass sheet, and (b) colors are assigned with the non-linear function for the Kevlar sheet.
In this appendix, we will describe the implementation of a CAD system for woven cloth composite part layout, will describe external file formats that the system can read, and will provide two examples of external data files.

G.1 Implementation of a CAD System

We have implemented a prototype CAD system for simulating the fittings of a woven cloth composite sheet onto an arbitrary NURBS surface on an IBM RS/6000 workstation. All of the figures related to fittings used in this dissertation are produced by this system. Specifically, they are first displayed on an X window, and then dumped into PostScript formats. Color image rendering capability can be optionally used to obtain realistic fitting simulations.

The following external files are necessary for a simulation of fitting a 2D sheet of woven cloth composites to a 3D surface. Detailed syntax and semantics of these files follow after the description for them.

**NURBS file:** a file that describes the geometry of a 3D surface, modeled by a set of NURBS (Non-Uniform Rational B-Spline) surface patches.

**TOPOLOGY file:** a file that describes the topology among NURBS patches necessary for defining a composite surface.

**CLOTH file:** a file that describes a 2D ply of woven cloth composites, and the initial conditions in 2D space.

---

5 The X Window System is a trademark of the Massachusetts Institute of Technology.

6 PostScript is a registered trademark of Adobe Systems, Inc.
**PATH file:** a file that describes the initial conditions in 3D space, including the base path.

**CAMERA file:** a file that describes the viewing data for 3D display.

**LIGHT file:** a file that describes the light data for 3D rendering display.

The system first reads a NURBS file and the associated TOPOLOGY file, and constructs a composite NURBS surface. It then reads a CLOTH file and creates a data structure for a given 2D ply of woven cloth composites to be used for the fitting. At this point the NURBS surface data structure and the 2D woven cloth data structure are independently defined. The relationship between them is established by reading a PATH file. The mapping calculation is initiated, once the initial conditions including the base path and the sweeping directions are set up. The 3D wireframe drawing of the mapping calculation is displayed with the aid of graPHIGS [32], which is a standard 3D graphics package conformed to the ISO graphic standard PHIGS (Programmer’s Hierarchical Interactive Graphics System) [45]. A CAMERA file must be fed into the system before the fitting result is displayed. During the display of the result onto a NURBS surface, we can modify the view interactively. Optionally, a LIGHT file is used for producing a rendered image of the fitting, based on the Phong’s shading algorithm [68, 86].

From the main panel of the mapping display, we can proceed to one of the following:

- display the 3D fitting result as a wireframe or in a rendered image
- display the 2D plane development of the ply of woven cloth composites used for the mapping onto the 3D surface
- display the thread angles between wefts and warps as height fields projected above the 2D plane development, which serves as an indicator of potential anomalous events, such as wrinkling and tearing
• display the thread angles with colors

• display the Gaussian curvatures with colors

• display the intermediate Gaussian curvatures integrals as height fields projected above the 2D plane development, which serves as another indicator of potential anomalous events

• insert linear and polygonal cuts interactively

G.2 External Data File Format

Here we list external data file formats available in the current CAD system for woven cloth composite part layout. As we mentioned above, there are six external data files: NURBS file, TOPOLOGY file, CLOTH file, PATH file, CAMERA file, and LIGHT file. Before describing each file format, we list several implicit rules commonly applied.

• Any line with the symbol “#” at the first column is regarded as a comment line.

• A real (floating point) value can take both “%e” (exponential) and “%f” (fractional) formats.

• Only a fixed number of data items can be specified on each line. If several data items are written on a same line, each must be separated by one or more blanks.
NURBS file format

<Total No. of NURBS Patches (M)>
/***************************************************************************/
/* The following lines are repeated M times. */
/***************************************************************************/
<NURBS Patch ID> <Singular Flag>
<Rational Flag> <Periodic Flag (U)> <Periodic Flag (V)>
<Degree along U-axis (du)> <Degree along V-axis (dv)>
<No. of Knot Vectors along U-axis (ku)>
<Knot 1> <Knot 2> ... <Knot ku>
<No. of Knot Vectors along V-axis (kv)>
<Knot 1> <Knot 2> ... <Knot kv>
<No. of Control Points in U (m)> <No. of Control Points in V (n)>
<X1,1> <Y1,1> <Z1,1> <Weight 1,1>
<X1,2> <Y1,2> <Z1,2> <Weight 1,2>
...<X1,n> <Y1,n> <Z1,n> <Weight 1,n>
...<Xm,n> <Ym,n> <Zm,n> <Weight m,n>
/******************************************************************************/
/* The following lines are valid only when <Singular Flag> is ON. Note the only one singular point is allowed for each */
/* NURBS patch. */
/******************************************************************************/
<X0> <Y0> <Z0>
<Xu> <Yu> <Zu>
<Xv> <Yv> <Zv>
<Xuu> <Yuu> <Zuu>
<Xvv> <Yvv> <Zvv>
<Xuv> <Yuv> <Zuv>
<Du> <Dv>

******************************************************************************
/* Description */
******************************************************************************
<Total No. of NURBS Patches (M)> --- integer (> 0)
<NURBS Patch ID> --- integer (> 0)
<Singular Flag> --- integer (0: Non-singular 1: Singular)
<Rational Flag> --- integer (0: Non-rational 1: Rational)
<Periodic Flag> --- integer (0: Non-periodic 1: Periodic)
<Degree> --- integer (> 0), degree of the surface
<No. of Knot Vectors> --- integer (>0)
<Knot> --- real, knot vector
<No. of Control Points> --- integer (>0)
<X> <Y> <Z> --- reals, coordinate of a control point
<Weight> --- real, weight of a control point
<XO> <YO> <ZO> --- reals, singular point
<Xu> <Yu> <Zu> --- reals, U partial at singular point
<Xv> <Yv> <Zv> --- reals, V partial at singular point
<Xuu> <Yuu> <Zuu> --- reals, U second partial at singular point
<Xvv> <Yvv> <Zvv> --- reals, V second partial at singular point
<Xuv> <Yuv> <Zuv> --- reals, UV mixed partial (twist) at singular point
<Du> <Dv> --- reals, direction vector in UV coordinate at singular point

TOPOLOGY file format

<Surface ID> <No. of NURBS Patches (M)>
/******************************************************************************/
/* The following lines are repeated M times. */
/*******************************************************************/
<NURBS Patch ID><Fitted Flag>
<WEST Patch ID><EAST Patch ID><SOUTH Patch ID><NORTH Patch ID>
<WEST Continuity><EAST Continuity><SOUTH Continuity><NORTH Continuity>
/*******************************************************************/
/* Description */
/*******************************************************************/
<Surface ID> --- integer (>= 0)
<No. of NURBS Patches (M)> --- integer (>0)
<NURBS Patch ID> --- integer (>= 0)
<Fitted Flag> --- integer (0: Not Fitted, 1: Fitted)
<WEST Patch ID> --- integer (-1: NULL, 0>=: WEST NURBS Patch ID)
<EAST Patch ID> --- integer (-1: NULL, 0>=: EAST NURBS Patch ID)
<SOUTH Patch ID> --- integer (-1: NULL, 0>=: SOUTH NURBS Patch ID)
<NORTH Patch ID> --- integer (-1: NULL, 0>=: NORTH NURBS Patch ID)
<WEST Continuity> --- integer (-1: External Boundary, 0: C0-Continuity, 1: C1-Continuity, etc.)
<EAST Continuity> --- integer (-1: External Boundary, 0: C0-Continuity, 1: C1-Continuity, etc.)
<SOUTH Continuity> --- integer (-1: External Boundary, 0: C0-Continuity, 1: C1-Continuity, etc.)
<NORTH Continuity> --- integer (-1: External Boundary, 0: C0-Continuity, 1: C1-Continuity, etc.)

CLOTH file format

<Weave Pattern Type><Material Type>
<Number of Edges for Given Cloth (N)>
<X1><Y1><Cut Flag 1><CX1><CY1>
<X2><Y2><Cut Flag 2><CX2><CY2>
...
<XN><YN><Cut Flag N><CXN><CYN>
<No. of Weft Threads (m)><No. of Warp Threads (n)>
<Distance between Adjacent Wefts><Distance between Adjacent Warps>
<Slippage Mode><Minimum Slippage Angle><Maximum Slippage Angle><Slip Percent>
<Reference Position (Weft)><Reference Position (Warp)>
<Base Direction Vector (Weft)><Base Direction Vector (Warp)>
<Left Sweeping Direction (Weft)><Left Sweeping Direction (Warp)>
<Right Sweeping Direction (Weft)><Right Sweeping Direction (Warp)>
<No. of Darting Triangles (M)>
<i1_1><j1_1> <i2_1><j2_1> <i3_1><j3_1>
<i1_2><j1_2> <i2_2><j2_2> <i3_2><j3_2>
...
<i1_M><j1_M> <i2_M><j2_M> <i3_M><j3_M>

/*******************************************************************/
/* Description */
/*******************************************************************/
<Weave Pattern Type> --- integer
  0: Plain weave, 1: Twill weave, 2: Satin weave
<Material Type> --- integer
  0: Fiberglass, 1: Kevlar, 2: Graphite, 3: Cotton,
  4: Wool, 5: Jute, 6: Silk, 7: Polyester
<No. of Edges for Given Cloth (N)> --- integer
<X1><Y1> --- real, vertex of given cloth (0<=i<=N, 0.0 <= Xi, Yi <= 1.0)
<Cut Flag i> --- integer (0: no cut, 1: cut vertex mode)
<CX1><CY1> --- real, cut vertex (meaningful only when <Cut Flag i> == 1)
<No. of Weft Threads (m)><No. of Warp Threads (n)> --- integer (>0)
<Distance between Adjacent Threads> --- real
<Slippage Mode> --- integer (0: no slippage, 1: slippage)
<Minimum Slippage Angle> --- real, minimum allowable thread angle in radian before slippage
<Maximum Slippage Angle> --- real, maximum allowable thread angle in radian before slippage
<Slip Percent> --- real, slippage percentage
<Reference Position> --- integers, reference position in 2D thread space
  The left bottom crossing is assumed to be (0,0) and the right top corner be (m-1,n-1).
<Base Direction Vector> --- reals, direction vector along the base path in 2D thread space.
<Sweeping Direction Vector> --- reals, direction vector for sweeping the woven cloth in each half space delimited by the base path.
<No. of Darting Triangles (M)> --- integer, (>= 0)
<i1><j1><i2><j2><i3><j3> --- integers, coordinates of the darting triangle in clockwise order, assuming that the (i2,j2) is the apex coord.

PATH file format

<Surface ID><NURBS Patch ID>
<Singular Flag>
<Reference Point (u0)><Reference Point (v0)>{<u00><v00>}
<Step Size in XYZ><Step Size in UV>
<Type of Base Path>
/**************************************************************************/
/* The next lines depend on the <Type of Base Path> */
**************************************************************************/
/ * <Type of Base Path> = 0 : planar curve */
<Base_Direction_X><Base_Direction_Y><Base_Direction_Z>
<Base_Normal_X><Base_Normal_Y><Base_Normal_Z>
/ * <Type of Base Path> = 1 : geodesic curve */
<Step Size for Geodesic><Principal Direction Flag>
<Initial_Base_Direction_X><Initial_Base_Direction_Y><Initial_Base_Direction_Z>

/**************************************************************************/
/* Description */
**************************************************************************/
<Surface ID> --- integer (>=0)
<NURBS Patch ID> --- integer (>=0), NURBS Patch ID having the start point
<Singular Flag> --- integer (0 or 1)
  0: Non-singular
  1: Singular (Two start points must be specified. One is for the ordinary direction along the base path, and the other is for the reverse direction along the base path.)
<Step Size in XYZ> --- real, distance between adjacent grids in XYZ space
<Step Size in UV> --- real, default step size in UV space
<Reference Point> --- reals, (u0,v0) in UV space
<u00><v00> --- reals, reference point in reverse direction along the base path specified optionally when <Singular> flag is ON.
<Type of Base Path> --- integer (0: planar curve, 1: geodesic curve)
<Base_Direction> --- reals, direction vector of the base path
<Base_Normal> --- reals, normal vector to the base plane
<Step Size for Geodesic> --- real, step size used in numerical generation of geodesic
<Principal Direction Flag> --- integer, (0: arbitrary, 1: initial direction of geodesic is aligned with a principal curvature direction)
<Initial_Base_Direction> --- reals, the initial direction vector for the geodesic bath path
CAMERA file format

<Total No. of Cameras (M) >
/*******************************************************************/
/* The following lines are repeated M times. */
/*******************************************************************/
<Camera ID>
<Viewing Type>
<Camera Input Type>
/*******************************************************************/
/* The following lines depend on <Camera Input Type> */
/*******************************************************************/
/* <Camera Input Type> = 1, UVN element input */
<VRP_X><VRP_Y><VRP_Z>
<VUP_X><VUP_Y><VUP_Z>
<VPN_X><VPN_Y><VPN_Z>
/* <Camera Input Type> = 2, view matrix input */
<Viewmat 1,1><Viewmat 1,2><Viewmat 1,3><Viewmat 1,4>
<Viewmat 2,1><Viewmat 2,2><Viewmat 2,3><Viewmat 2,4>
<Viewmat 3,1><Viewmat 3,2><Viewmat 3,3><Viewmat 3,4>
<Viewmat 4,1><Viewmat 4,2><Viewmat 4,3><Viewmat 4,4>
<PRP_U><PRP_V><PRP_N>
.Window Min_U><Window Min_V><Window Max_U><Window Max_V>
.Screen Distance><Front Clip><Rear Clip>
.Screen Clip Flag><Front Clip Flag><Rear Clip Flag>
<2D Window Min-X><2D Window Max-X><2D Window Min-Y><2D Window Max-Y>
/***********************************************************************/
/* Description */
/***********************************************************************/
<Total No. of Cameras (M)> --- integer (>0)
<Camera ID> --- integer (>=0)
<Viewing Type> --- integer (1:parallel projection, 2:perspective projection)
<Camera Input Type> --- integer (1:UVN element, 2:viewing matrix)
<VRP> --- reals, View Reference Point in XYZ space
<VPN> --- reals, View Plane Normal in XYZ space
<VUP> --- reals, View Up Vector in XYZ space
<Viewmat> --- reals, 4*4 viewing matrix
<PRP> --- reals, Projection Reference Point in UVN viewing space
<Window Min> --- reals, Left Bottom Corner of Screen Window in UV space
<Window Max> --- reals, Right Top Corner of Screen Window in UV space
.Screen Distance> --- real, screen distance along N-axis from VRP
<Front Clip> --- real, front clip position along N-axis from VRP
<Rear Clip> --- real, rear clip position along N-axis from VRP
.Screen Clip Flag> --- integer (0:no clip, 1:clip with screen window)
<Front Clip Flag> --- integer (0:no clip, 1:clip with front clip plane)
<Rear Clip Flag> --- integer (0:no clip, 1:clip with rear clip plane)
<2D Window> --- reals, Device Window in 2D Device Coordinate

LIGHT file format

<Total No. of Lights (M) >
/*******************************************************************/
/* The following lines are repeated M times. */
/*******************************************************************/
<Light ID>
<Light On/Off Flag>
<Intensity_RED><Intensity_GREEN><Intensity_BLUE>
<Light Type>
G.3 Examples of External Data File Format

Two sets of external data files are described as examples.

G.3.1 External Data Files for a Surface of Revolution

The following sample external data files are used to generate the fitting to a surface of revolution shown in Figure 5.31 (a).

```
/* NURBS FILE: Surface of revolution */
/* File = "surface_of_revolution.nurbs" */
# total number of nurbs patches to be defined
1
# NURBS ID and Singular flag
1 0
# Rational flag and Periodic Flag for U and V directions
1 0 0
# Degree along U and V axes
2 3
# the number of U knot vectors
4
# U knot vectors
0 0 1 1
# the number of U knot vectors
6
# V knot vector
0 0 0 1 1 1
# the number of control points along U and V directions
3 4
# Control points and Weights
10 0 0 1 /* c(0,0), w(0,0) */
10 0 5 1 /* c(0,1), w(0,1) */
3 0 10 1 /* c(0,2), w(0,2) */
6 0 15 1 /* c(0,3), w(0,3) */
```
10 17.3205 0 0.5 /* c(1,0), w(1,0) */
10 17.3205 5 0.5 /* c(1,1), w(1,1) */
3 5.1962 10 0.5 /* c(1,2), w(1,2) */
6 10.3923 15 0.5 /* c(1,3), w(1,3) */

-5 8.6603 0 1 /* c(2,0), w(2,0) */
-5 8.6603 5 1 /* c(2,1), w(2,1) */
-1.5 2.5981 10 1 /* c(2,2), w(2,2) */
-3 5.1962 15 1 /* c(2,3), w(2,3) */

/* File = "surface_of_revolution.topology" */
# Surface Body ID and the number of Patches
1 1
# NURBS ID (1) and Fitted Flag
1 1
# WEST, EAST, SOUTH, and NORTH Adjacent Patch
-1 -1 -1 -1
# WEST, EAST, SOUTH, and NORTH Patch Continuity
-1 -1 -1 -1

/* File = "surface_of_revolution_1.cloth" */
# Coordinates of the trimmed cloth in a normalized coordinate
0 0
0 1 0
1 1 0
1 0 0

/* File = "surface_of_revolution_1.path" */
# Surface ID and NURBS Patch ID
1 1
# Singular Flag (specified only when the starting point is singular.)
0
# Step Distance (step_xyz = 0.5, step_uv = 0.05)
0.5 0.05
G.3.2 External Data Files for an Object with a Convex Corner

The following sample external data files are used to generate the shading output for the fitting to an object with a convex corner shown in Figure 6.21 (a).
5 5 10 1
5 5 10 1

#### No. 2 Patch
2 0
0 0 0
1 2
0 1
2
0 1
2 2
5 10 -5 1
5 10 5 1
-5 10 -5 1
-5 10 5 1

#### No. 3 Patch
3 0
1 0 0
1 2
4
0 1
2 3
5 10 5 1
5 10 10 0.7071
5 5 10 1
-5 10 5 1
-5 10 10 0.7071
-5 5 10 1

#### No. 4 Patch
4 0
1 0 0
1 2
4
0 1
2 3
10 -5 5 1
10 -5 -10 0.7071
5 -5 10 1
10 5 5 1
10 5 10 0.7071
5 5 10 1

#### No. 5 Patch
5 0
1 0 0
1 2
4
0 1
2 3
10 5 5 1
10 10 5 0.7071
5 10 5 1
10 5 -5 1
10 10 -5 0.7071
5 10 -5 1

#### No. 6 Patch (octant of a sphere)
6 1
1 0 0
2 2
4
0 0 1 1
0 0 1 1
3 3
 5 5 10 1
 5 5 10 0.7071
 5 5 10 1
 10 5 10 0.7071
 10 10 10 0.5
 5 10 10 0.7071
 10 5 5 1
 10 10 5 0.7071
 5 10 5 1
# x0 y0 z0
 5 5 10
# xu0 yu0 zu0
 1 0 0
# xv0 yv0 zv0
 0 1 0
# xuu0 yuu0 zuu0
 0 0 0.2
# xvv0 yvv0 zvv0
 0 0 0.2
# xuv0 yuv0 zuv0
 0 0 0
# du0 dv0
 1 0
### No. 7 Patch
7 0
0 0 0
1 1
2
 0 1
2 2
 0 1
2 2
 10 -5 -5 1
 10 -5 5 1
 10 5 -5 1
 10 5 5 1

rehhkkikkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkkk kk
/* CLOTH FILE: Object with a convex corner */

# File = "convex_corner.cloth"
# Weave (plain weave) and material (fiberglass)
0 0
# Number of edges for trimming cloth
4
# Coordinates of the trimmed cloth in a normalized coordinate
0 0 0
0 1 0
1 1 0
1 0 0
# Number of wefts and warps
50 50
# Distance between adjacent wefts and warps
0.65 0.65
# Thread angle minimum and maximum
0.0 3.14
# Slip
0 0.0 3.14 0.0 0.0 0.0
# Reference point along weft and warp
0.0 0.0
# Base Direction Vector
1 0
# Left sweeping direction
0 1
# Right sweeping direction
0 -1
# Number of Stitched Darts
0

/* PATH FILE: Object with a convex corner */

# File = "convex_corner.path"
# Surface ID and NURBS Patch ID
1 1
# Singular Flag (specified only when the starting point is singular.)
0
# Step Distance (step_xyz = 0.65, step_uv = 0.065)
0.65 0.065
# Starting Position (u0,v0)
0.0 0.0
# Base Path Type
# (0-type: Base path is defined as a Planar Curve.)
0
# Base Path Direction (dx,dy,dz)
1 0 0
# Base Plane Normal (nx,ny,nz)
0 1 0

/* CAMERA FILE: Object with a convex corner */

# File = "convex_corner.camera"
# Camera ID
1
# view type (PERSPECTIVE)
1
# input type (VRP, VUP, VPN direct input)
# View Reference Point (VRP)
-0.6 -0.6 4.2
# View Up Vector (VUP)
0 0 1
# View Plane Normal (VPN)
1 1 0.75
# Projection Reference Point (PRP)
0 0 100
# View Window (Umin, Umax, Vmin, Vmax)
-13 13 -13 13
# Distance from the Screen, Near Clipping Plane, and Far Clipping Plane
20 40 -30
# Flags (Near Clipping, Far Clipping, and Window Clipping)
1 1 1
# 2D Device Coordinate (Xmin, Xmax, Ymin, Ymax)
100 600 100 600

/**********************************************************************/
/* LIGHT FILE: Object with a convex corner */
/**********************************************************************/
# File = "convex_corner.light"
# Number of Light Sources
3
# Light ID (1)
1
# Light On/Off Flag
1
# Intensity (RGB)
0.30 0.30 0.30
# Light Type (Point Light Source)
2
# Light Source
30 -5 30
# Light ID (2)
2
# Light On/Off Flag
1
# Intensity (RGB)
0.40 0.40 0.40
# Light Type (Point Light Source)
2
# Light Source
-5 20 25
# Light ID (3)
3
# Light On/Off Flag
1
# Intensity (RGB)
0.20 0.20 0.20
# Light Type (Point Light Source)
2
# Light Source
20 20 -5
APPENDIX H
GLOSSARY

applying a ply to a surface: See “fitting a ply to a surface.”

base path: The path used as a guide line to fit a woven cloth ply to a surface. In 2D space, the base path is defined as a straight line, but does not need to be aligned with a particular thread. In 3D space, the base path is a loop-free open curve on the given surface. Our current implementation allows two methods: (a) a planar curve and (b) a geodesic curve. See Section 5.1.2.

base plane: A plane on which the base path is defined.

Bézier control point: A control point that is used to defines a Bézier curve or a Bézier surface. See Section 7.3.5.1 for the condition that a NURBS control point becomes a Bézier control point.

boundary mesh point: A mesh point that contributes to the boundary in a given 2D ply after a mapping. See Figure 5.23. Same as “edge-touching mesh point.”

Chebychev net: See Tchebychev net.

chopped fiber composite: An isotropic composite production form in which short strands of fibers are randomly arranged. See fiber-reinforced composite.

clothing a surface with a ply: See “fitting a ply to a surface.”

composite material: A material that is manufactured; consists of two or more physically and/or chemically distinct, suitably arranged or distributed phases with an interface separating them; and has characteristics that are not depicted by any of the component in isolation. In this thesis, a composite implicitly refers to a “fiber-reinforced” composite material.

conforming a ply to a surface: See “fitting a ply to a surface.”

coordinate curve: A curve on a bivariate parametric surface $r(u, v)$ expressed by either $u = constant$ or $v = constant$. Same as iso-parametric curve.
critical region: A region in a deformed woven cloth composite ply where thread angles at crossings are predicted to exceed the locking angle. See lower critical region and upper critical region.

crossing: The intersection between a weft and a warp. Same as a crossover point.

DAG: An acronym for Directed Acyclic Graph.

dart: A stitched tapering fold (a stitched dart) or a trimmed cut (a trimmed dart) used in fitting a ply to a surface, especially when excessive shear deformation that may cause wrinkling or tearing is expected in the ply. A trimmed dart is further classified into a polygonal cut or a linear cut.

DDA algorithm: A Digital Differential Analyzer algorithm. Originally, the DDA algorithm was developed for drawing lines on a raster graphic display in the 1960’s. Also called the Bresenham algorithm.

degree elevation: To elevate the degree of a given (piecewise) polynomial curve or a surface. See Section 7.3.5.3.

dependency graph: A directed acyclic graph representing a dependency relation of the mapping calculations among mesh points.

developable surface: A surface with zero Gaussian curvature everywhere. Examples include plane, cone, and cylinder.

edge-touching mesh point: See boundary mesh point.

fabric: A piece of cloth assembled from threads.

fiber: A filament, or the minimum component of a thread.

fiberglass: A generic name to refer to a group of manufactured fibers. Common substances are silica based ($\approx 50-60\% \text{ SiO}_2$). It is quite inexpensive, and available in a variety of forms (e.g. woven fabrics or non-woven mats). Same as glass fiber.

fiber-reinforced composite: A composite made of fibers as a reinforcing material suspended in a “matrix” material that stabilizes the reinforcing material and bonds it to adjacent reinforcing materials.
fitting a ply to a surface: To deform a ply onto a surface such that the ply is in contact everywhere without gaps and wrinkles. Same as “clothing a surface with a ply”, “applying a ply to a surface”, and “conforming a ply to a surface.”

fitting method: See “mapping method.”

frontier: A sorted list of mesh points that contain candidates for the next mapping calculation, while the scanning algorithm is applied. Initially, a frontier consists of mesh points adjacent to the mesh points on the base path. A frontier list is updated every time a mapping calculation for the mesh point in the first entry is done. See Section 5.3.

Gauss-Bonnet theorem: A theorem that relates the integral Gaussian curvature for a given surface region, the integral geodesic curvature of the boundary curve of the surface region, and the interior angles defined at the corners of the boundary curve.

Gaussian curvature: The ratio of a surface patch area to the corresponding area on a unit sphere by mapping the surface normals of the given surface patch to the points on a unit sphere. This map is called Gauss’ spherical map. Mathematically, it is defined as $K = \kappa_{\text{min}}\kappa_{\text{max}}$, where $\kappa_{\text{min}}$ and $\kappa_{\text{max}}$ are the minimum and the maximum normal curvature at the point of interest on the surface. When $K \equiv 0$, the surface is called flat, or a developable surface (See developable surface).

geodesic curvature: The magnitude of a vector at a point of a curve on a surface, obtained by projecting the acceleration vector onto the tangent plane at the point.

geodesic (curve): A curve on a surface having no geodesic curvature. All straight lines on a surface are geodesics. A curve not a straight line is a geodesic if the principal normal along the curve coincides with the surface normal. See Equation (7.30).

indegree one node: A node in a dependency graph with only one incoming arc. See Section 5.3.
indegree two node: A node in a dependency graph with two incoming arcs. See Section 5.3.

initial conditions for our fitting method: Initial conditions for our fitting method include the reference point, the base path, and the sweeping directions. They must be specified in both 2D cloth space and 3D surface space. See base path.

intermediate Gaussian curvature integral: The integral Gaussian curvature at a mapped mesh point by taking its integral path that encloses the previously mapped mesh points that affect the mapping calculation of the current mesh point. This value depends on how we fit a given woven cloth composite ply to the surface region. It is actually the integral Gaussian curvature for a part of the given surface region. See Section 7.2.2.

intrinsic property: A geometric property of an object (a surface) that is not affected by the definition of its coordinate system and a rigid motion (i.e. translation and rotation) of the object. Gaussian curvature and a line of curvature are examples of intrinsic properties defined for a given surface, whereas surface normals and the height of the surface are examples of non-intrinsic properties.

jamming angle: See locking angle.

Kevlar: A kind of aramid fibers manufactured from synthetic polyaramide. It is substantially lighter, yet stiffer and stronger than steel and aluminum. Well-known applications include bulletproof jackets (Kevlar 29) and the frames of aircraft and boats (Kevlar 49). It is a trademark of Du Pont.

knot insertion: Inserting a scalar value (called “knot”) into a knot vector. See Section 7.3.5.3.

knot vector: A non-decreasing sequence of scalar values that defines the parametric domain (interval) for a NURBS curve or a NURBS surface.

known neighbor: A neighboring mesh point to a mesh point of interest whose mapping calculation is already finished. See neighboring mesh point.

laminator: A worker who is responsible for the lamination process during the manufacturing of fiber-reinforced composites.
line of a curvature: A curve on a surface whose tangent at each point is along a principal direction.

linear cut: A type of a trimmed dart that is created by simply cutting a given cloth material without removing any parts. See Section 6.2.

locking angle: The thread angle between a weft and a warp when a “locking” begins. No further shear deformation occurs beyond this angle. Same as jamming angle. See Section 4.2.

lower critical region: A region in a deformed woven cloth composite ply where thread angles at crossings are predicted to go below the minimum locking angle. See also upper critical region.

Mack & Taylor’s cloth model: A woven cloth model that was proposed by Mack & Taylor in 1956 for fitting application. It assumes that threads in cloth are inextensible, the thread segment between adjacent crossings is straight, no slippage occurs at the crossing when the cloth is fitted, and the smallest radius of curvature on any surface is much larger than the distance between adjacent crossings. Basically, their cloth model is similar to ours except that the assumption of no slippage is removed (Section 4.3).

mapping method: A method for establishing a correspondence between a point in 2D cloth space and a point in 3D surface space. Same as the fitting method.

matrix material: A material used to stabilize a reinforcing material to form a composite. Epoxy resin is a typical example of a matrix material.

mesh point: A sampled point in woven cloth. A regular mesh point is defined at the crossing between a weft and a warp. An auxiliary mesh point is defined at the intersection between a line segment (either a darting edge or a base path) and a weft or a warp.

neighboring mesh point: A mesh point directly linked by the mesh point of interest. At most, four neighbors (called WEST, EAST, SOUTH, and NORTH neighbors) are defined at an arbitrary mesh point. See Section 5.4 and Appendix B. Also called a “neighbor” or an adjacent mesh point.

normal curvature: The magnitude of the acceleration vector of a curve on a surface along the surface normal.

NURBS: An acronym for Non-Uniform Rational B-Spline. See Section 4.4.

NURBS control point: A control point that is used for defining a NURBS curve or a NURBS surface.

off-surface control point: A NURBS control point that is not on the surface. See Section 7.3.5.1.

on-surface control point: A NURBS control point that is on the surface. See Section 7.3.5.1.

osculating plane: A plane that is determined by a tangent vector and a surface normal at a point of a curve on a surface.

overlapping mesh point: A mesh point that is found to be overlapped with a part of woven cloth composite ply fitted to a surface. This is caused by inserting a trimmed dart into a ply.

overlapping point list (OPL): A list that keeps track of overlapping mesh points. See Section 6.2.4.

patch code: A code representing where a mapped mesh point belongs relative to the previous position of its known neighbor in the original surface patch. See Section 5.4, especially Figure 5.19 and Appendix B.

plane development: A 2D flattened pattern of a ply, showing the area that is used for producing a 3D fitting result.

ply: A thin sheet of aligned fibers impregnated with partially cured resin. Same as prepreg. It also refers to a preform sheet (plain woven cloth) without cured resin.

polygonal cut: A type of a trimmed dart that is created by trimming a polygonal shape from a given cloth material. See Section 6.2.

prepreg: A thin sheet ($\approx 1$ mm) of partially cured resin containing aligned fibers.

principal direction: A direction in which a normal curvature of a point on a surface takes either a minimum or a maximum. This is uniquely defined at an arbitrary point on a surface except an umbilic.

ruled surface: A surface obtained by linearly interpolating two space curves $\mathbf{p}(u) = \mathbf{p}_1(u)$ and $\mathbf{p}(u) = \mathbf{p}_2(u)$, having the same parameter $u$. It is thus represented by $\mathbf{r}(u, v) = (1 - v)\mathbf{p}_1(u) + v\mathbf{p}_2(u)$.

sine Gordon equation: The equation that a mixed derivative of a bivariate function $f(u, v)$ is proportional to $\sin f$. In other words, $f_{uv} = C \sin f$, where $C$ is a constant and $f_{uv} = \frac{\partial^2 f}{\partial u \partial v}$. This equation is often encountered in mathematical physics [6, 89]. See Equation (7.3).

slippage model: See Section 4.3.

SSI problem: Surface-Surface Intersection problem.

stitched dart: A type of a dart that has a stitch along the darting edge. See Section 6.2.

sweeping direction: A direction in which a woven cloth ply is swept at the outset of the fitting. This direction must not be parallel with the direction of the base path, but need not be always perpendicular to the base path. A sweeping direction is one of the necessary initial conditions for a fitting in both 2D and 3D spaces. It is specified as a vector in each space.

Tchebychev net: A mathematical woven cloth model in which cloth is treated as a continuum and threads are inextensible, which was proposed by Tchebychev in 1878. See Chapter 7.

thread: A group of short textile fibers spinned and twisted together into a continuous strand.

thread angle: An angle between a weft and a warp when a woven cloth ply is deformed. Initially, this angle is assumed to be $\pi/2$ at every mesh point. It is also referred to as “angle of shear.”
**topological sorting**: A sorting that is applied to a DAG for obtaining a linear ordering of all its vertices such that if the DAG contains an arc from \( v_1 \) to \( v_2 \), where \( v_1 \) and \( v_2 \) are vertices, then \( v_1 \) appears before \( v_2 \) in the ordering.

**translation surface**: A surface defined by moving the space curve \( r = P(u) \) parallel to itself in such a way that are part of the curve moves along the space curve \( r = Q(v) \). The curves \( P(u) \) and \( Q(v) \) may be interchanged and still yield the same surface. An example is a cylinder obtained by moving a straight line parallel to itself along a circle. The curves \( P(u) \) and \( Q(v) \) form a conjugate set of parametric lines. Formally, a translation surface \( r(u, v) \) is represented by \( r(u, v) = P(u) + Q(v) \).

**trimmed dart**: A type of a dart that is created by trimming excessive parts from a given cloth material. See *linear cut* and *polygonal cut*.

**umbilic**: A point on a surface where the normal curvature is constant in every direction. For instance, on a sphere all points are umbilics, and an ordinary ellipsoid has four umbilics. Same as umbilical point. See Section 7.3.1.

**uncalculated surface region**: The region in a surface where the mapping calculation cannot be reached. This is one of the problems in the previous method for the mapping calculations [14, 38, 61, 84, 85, 106, 107]. With our mapping method, this problem does not occur. See Section 5.1.1.

**unidirectional tape**: A composite production form in which fibers are running in one direction. It is therefore strongest in that direction.

**upper critical region**: A region in a deformed woven cloth composite ply where thread angles at crossings are predicted to exceed the maximum locking angle. See also *lower critical region*.

**UV parametrization**: A bivariate parametrization used for representing a (NURBS) surface.

**warp**: A vertical thread in woven cloth.

**warp gradient vector**: A vector approximating the tangent direction of a warp thread around a mesh point. See Equation (5.13).
**weave:** A geometric pattern determined by how weft and warp threads pass over and under one another. See Section 4.2.

**weft:** A horizontal thread in woven cloth. Same as woof.

**weft gradient vector:** A vector approximating the tangent direction of a weft thread around a mesh point. See Equation (5.13).

**woof:** Same as weft.