World Academy of Science, Engineering and Technology
International Journal of Fashion and Textile Engineering
Vol:3, No:9, 2009

CAD/CAM Algorithms for 3D Woven Multi-layer Textile Structures

Martin A. Smith, Xiaogang Chen

***Abstract*—**This paper proposes new algorithms for the computer-aided design and manufacture (CAD/CAM) of 3D woven multi-layer textile structures. Existing commercial CAD/CAM systems are often restricted to the design and manufacture of 2D weaves. Those CAD/CAM systems that do support the design and manufacture of 3D multi-layer weaves are often limited to manual editing of design paper grids on the computer display and weave retrieval from stored archives. This complex design activity is time-consuming, tedious and error-prone and requires considerable experience and skill of a technical weaver. Recent research reported in the literature has addressed some of the shortcomings of commercial 3D multi-layer weave CAD/CAM systems. However, earlier research results have shown the need for further work on weave specification, weave generation, yarn path editing and layer binding. Analysis of 3D multi-layer weaves in this research has led to the design and development of efficient and robust algorithms for the CAD/CAM of 3D woven multi-layer textile structures. The resulting algorithmically generated weave designs can be used as a basis for lifting plans that can be loaded onto looms equipped with electronic shedding mechanisms for the CAM of 3D woven multi-layer textile structures.

***Keywords*—**CAD/CAM, Multi-layer, Textile, Weave.

I. Introduction

COMPUTER-aided design and computer-aided manufacturing research and development is a multi-application, inter-disciplinary activity that has received much attention in the scientific, engineering and manufacturing communities since its inception in the late 1950s and early 1960s. Automation resulting from CAD/CAM has led to an increase in productivity and quality with a simultaneous reduction in time-to-market, material costs and manufacturing errors leading to wastage. Additionally, CAD/CAM technology affords valuable prototyping capabilities: two- and three-dimensional visualisation and spatial manipulation of geometric models allow continuous editing and refinement prior to object manufacture.

Woven composite materials have emerged in a diverse range of applications in many areas such as the aerospace and automotive industries due to their thickness, high strength and high stiffness properties. Furthermore, stitching used to reinforce 3D composites offers improved delamination resistance and impact damage tolerance in comparison with conventional 2D laminates [1].

Two-dimensional single-layer woven fabrics are constructed from yarns known as warp and weft yarns and consist of one series of warp yarns and one series of weft yarns. The warp yarns are parallel to one another and run lengthwise, whereas the weft yarns are perpendicular to the warp yarns and run crosswise. The rules that govern the interlacement of yarns are known as the weave of the fabric. The interlacing pattern of a weave affects fabric properties including mechanical performance and drapability. The weaving process itself results in an interlacing of warp and weft yarns to produce a woven fabric the simplest of which is known as the plain weave as shown in Fig. 1. Automatic production of woven fabrics is made possible with the application of computer-aided manufacturing technology [2]. The graphical illustration of interlacement between warp and weft yarns is marked on design paper with each column of squares representing a warp yarn and each row of squares representing a weft yarn. A coloured square indicates that the warp yarn is lifted over the weft yarn. Conversely, a white square indicates that the warp yarn is lowered under the weft yarn. The binary weave matrix (a key component of weave CAM) shown in Fig. 1 is the machine-readable equivalent of the human-readable design paper representation. All weave designs described in this paper are shown on design paper. A 'formula' or descriptor used to describe the plain weave is 1/1. This can be interpreted as two floats in the path of the warp, i.e. one up float and one down float in the case of a plain weave. A step number can also be added to the descriptor to further describe the relative movement of a corresponding float on the next warp.

In the general case, a regular weave descriptor is defined as $e_1/e_2/e_3/\ldots/e_n$ step s where $e_l \in \mathbb{N}_1$ and e_l, n are subject to the finite loom capacity and $s \in A$, $A = \{x \in \mathbb{Z}: x \neq 0, |x| < \sum e_l\}$. The length e_l of a warp up float or warp down float is when $l \in B$ or $l \in C$ respectively, $B = \{x \in \mathbb{N}_1: x = 2m - 1, m \in \mathbb{N}_1\}$ and $C = \{x \in \mathbb{N}_1: x = 2m, m \in \mathbb{N}_1\}$. Furthermore, each weave has a unit size (i.e. single repeat) where the number of weft yarns = $\sum e_l$ and the number of warp (and weft) yarns is computed using the algorithm Compute2DWeaveDimensions found in §III Automatic Generation of 3D Multi-layer Weaves.

Three-dimensional multi-layer woven fabrics are constructed from m stitched ($1 < m <$ *loom capacity*) fabric layers where m is the number of layers and relates to fabric thickness. Each layer consists of a single two-dimensional

Martin Smith and Xiaogang Chen are with the School of Materials, University of Manchester, Oxford Road, Manchester, England (corresponding author to provide phone: +44-161-3065736; fax: +44-161-9558164; e-mail: martinanthony.smith@btopenworld.com).

weave. An *m*-ply multi-layer fabric consists of *m* series of warp yarns and *m* series of weft yarns. Interlacing the i^{th} layer warp yarns with i^{th} layer weft yarns forms a distinct fabric layer. To form an integrated 3D multi-layer fabric, individual layers are attached together with stitches that use through the thickness yarns to bind together the distinct fabric layers.

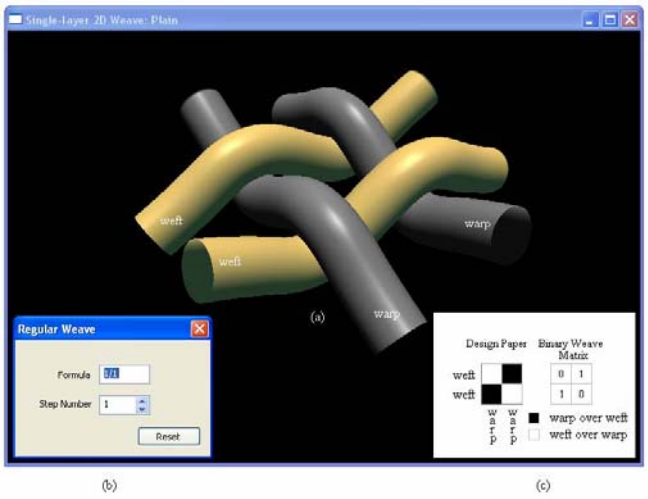

Fig. 1 Screenshot from WeaveStudio CAD/CAM software of (a) single-layer plain weave geometry, (b) weave specification and (c) weave design

CAD/CAM results for a range of 3D textile structures have been reported in the literature, including orthogonal and angle-interlock [3], hollow [4]–[5] and 3D multi-layer [6]-[9] weaves. Koltycheva and Grishanov [6] proposed two matrix representations of the same 3D multi-layer fabric structure: a binary weave matrix and a binary structure matrix together with linear transformations between the two representations. The structure matrix is divided into rectangular regions with each region containing either a distinct fabric layer weave or layer binding stitches. However, it is necessary to specify and automatically generate either the weave matrix or structure matrix before conversion between the two. Ping and Lixin [7] proposed the application of the Kronecker Product for the construction of double-layer weaves but the method does not extend to 3D multi-layer weaves containing more than two layers. Chen and Potiyaraj [8] proposed a non-interfering self-stitching approach to the formation of 3D multi-layer weaves. However, their solution supported limited editing of weave parameters post weave generation and restricted users to editing the design paper directly to apply layer binding stitches. Editing the design paper directly is a time-consuming, tedious and error-prone activity for all but the simplest 3D multi-layer weaves. Hewitt *et al.* [9] described the formation of 3D multi-layer weaves by assembling building blocks, each of which contains the interlacement of a warp and weft yarn. A block diagram was presented showing the inputs, processes and outputs of the system. However, details of weave dimension, assembly and binding were omitted.

Reproducible algorithms proposed in this paper for the CAD/CAM of 3D woven multi-layer textile structures address many of the deficiencies of previous works. The automatic generation of 3D multi-layer weaves has been implemented as part of the WeaveStudio CAD/CAM system [10]. The interface adopts intuitive and interactive spline manipulation for yarn path editing and stitching that is more suited to the formation of large and complex 3D multi-layer weaves. Application of layer binding stitches using the new spline based yarn editor triggers automatic updates to the design paper and binary matrix representations of the weave. Real-time rendering of 3D solid models of 3D multi-layer weaves form an integral component of the CAD interface. The technical weaver needs to insert layer binding stitches in precise locations and edit yarn paths according to end-user or application requirements. As a result, visualisation of 3D solid models is invaluable in the creation and editing of 3D multi-layer weaves and in addressing problem areas in tortuous yarn paths. Furthermore, the new approach allows the number of visualised yarn paths to include all yarns irrespective of individual layer weave size or total number of layers.

The remainder of this paper is organised as follows: in §II 3D multi-layer weaves are specified and described in detail, in §III algorithms are described and discussed for the automatic generation of 3D multi-layer weaves, in §IV layer binding stitch algorithms for layer attachment are described and discussed, in §V a resulting algorithmically generated weave design is presented and the paper concludes in §VI.

II. 3D MULTI-LAYER WEAVE DESCRIPTION AND SPECIFICATION

Without loss of generality and for the purpose of illustration, a simple minimal double-layer weave is introduced to describe and specify 3D multi-layer weaves. Note that the example double-layer weave introduced is easily designed by hand without the need for CAD software. However the need for designing larger and more complex 3D multi-layer weaves for real-world applications renders free-hand designing obsolete. Fig. 2 shows a unit size single-layer twill weave geometry characterized by diagonal lines and specified using the descriptor 2/2 step 1. The dimensions of the unit size weave are *warp* × *weft* = 4 × 4 and is a single repeat of the weave pattern. From inspection of Fig. 2 it can be seen that the pattern of the weave would cyclically repeat in either the warp or weft direction should extra warp or weft yarns be added to extend the size of the weave beyond the unit dimensions.

Fig. 3 (a) shows the design paper for a single-layer twill weave with warp and weft yarns labeled corresponding to the twill weave geometry in Fig. 2. Fig. 3 (a) also shows the unit size weave (i.e. single repeat) expanded to *warp* × *weft* = 8 × 8 dimensions to show the inherent repeating nature of woven structures.

As an example of yarn interlacement Fig. 2 shows that warp yarn 1 interlaces over weft yarns 1, 3 and under weft yarns 5, 7. The corresponding design paper weave of the yarn

interlacement is shown in Fig. 3 (a) at the intersection of column 1 and rows 1, 3, 5, 7. Fig. 3 (b) shows the single-layer twill weave extended to form a double-layer twill weave with warp and weft yarns labeled corresponding to the double-layer twill weave geometry in Fig. 5 and Fig. 6. In addition, Fig. 3 (b) is labeled with the layer number associated with individual warp and weft yarns. In effect, Fig. 2 and Fig. 3 (a) can be regarded as forming each layer of the double-layer weave structure. 3D multi-layer weave design paper notation introduced in Fig. 3 (b) and corresponding binary weave matrix entries are shown in Fig. 4.

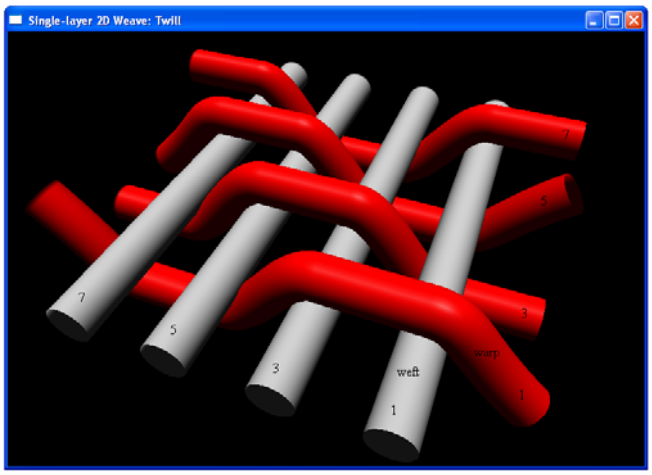

Fig. 2 Screenshot from WeaveStudio CAD/CAM software of single-layer twill weave geometry

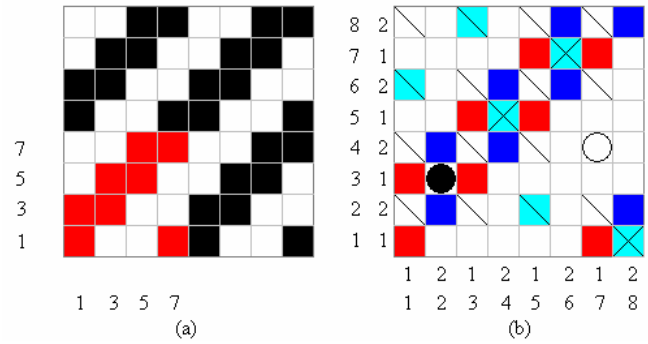

Fig. 3 Screenshot from WeaveStudio CAD/CAM software of (a) repeating single-layer twill weave and (b) single repeat double-layer twill weave

The application of stitching to binding distinct layers together is essential to form integrated 3D multi-layer weaves otherwise the individual layers are detached from one another. Self-stitching is used throughout this work and does not require the introduction of additional yarns to stitch layers together. Warp stitch up instructs the loom to allow the path of a warp yarn to interlace with a weft yarn from a layer above. In the general case, a warp yarn from level i stitches up to a weft yarn is at level j where $i > j$. Conversely, warp stitch down instructs the loom to allow the path of a warp yarn to interlace with a weft yarn from a layer below. In the general case, a warp yarn from level i stitches down to a weft yarn at level j where $i < j$. Where $i = j$ a warp yarn interlaces with a weft yarn in the same layer and no stitching occurs.

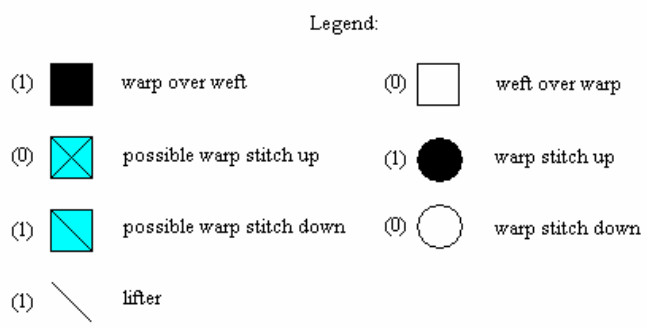

Fig. 4 Adapted screenshot from WeaveStudio CAD/CAM software of 3D multi-layer weave notation

The lifter notation signifies a warp yarn is lifted when a weft yarn from a lower layer is to be inserted. An example of lifter notation as applied to the paths of layer 1 warp yarns can be seen by inspection of Fig. 3 (b) and Fig. 5. All layer 1 warp yarns are lifted when the weft yarns from layer 2 are inserted during the weaving process. The only exception to the lifting of a layer 1 warp yarn is when a warp yarn stitches down to layer 2 as part of the layer binding process.

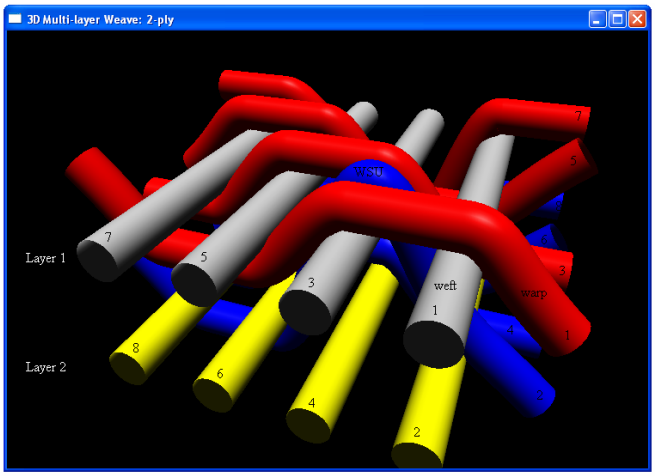

Fig. 5 Screenshot from WeaveStudio CAD/CAM software of double-layer twill weave geometry and warp stitch up layer binding

An example of warp stitch up notation as applied to warp yarn 2, layer = i = 2 can be seen by inspection of Fig. 3 (b) column 2 and Fig. 5. The warp yarn interlaces with weft yarn 3 from layer j = 1 ($i > j$). Fig. 5 shows the weave geometry of the yarn interlacement and in particular the warp stitch up labeled WSU. The warp yarn is shown passing over weft yarns 2, 4 (an up float over the two weft yarns), under weft yarns 6, 8 (a down float under the two weft yarns) and stitching up over layer 1 weft yarn 3. Note that Fig. 5 shows a non-interfering warp stitch up. Warp yarn 2 does not interfere with the fabric structure because the stitch is covered by adjacent layer 1 warp yarns 1,3.

An example of warp stitch down notation as applied to warp yarn 7, layer = i = 1 can be seen by inspection of Fig. 3

(b) column 7 and Fig. 6. Fig. 6 is an alternative viewing angle of the same double-layer weave in Fig. 5 and shows a perspective looking down at the underside of the weave. The warp yarn interlaces with weft yarn 4 from layer $j = 2$ ($i < j$). Fig. 6 shows the geometry of the yarn interlacement and in particular the warp stitch down labeled WSD. The warp yarn is shown stitching down under layer 2 weft yarn 4. In both examples of warp stitching it can be seen that the stitches do not interfere with the fabric structure because of the adjacent floats of the warp yarns that belong to the layer being stitched to.

Application requirements will determine whether fabric interference is permissible. If aesthetics are a consideration, such as for apparel use, then there may be a requirement for layer binding stitches not to interfere with visible layers. Fig. 3 (b) shows possible non-interfering warp stitch up and warp stitch down positions. For performance requirements, such as for composite applications, the aesthetics of stitching locations may not be a concern.

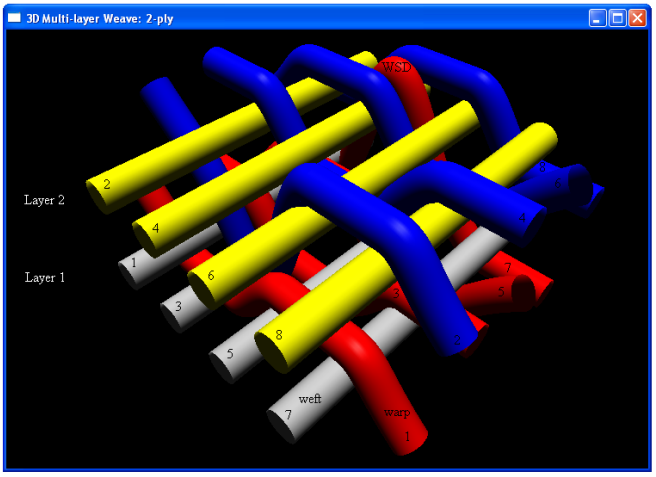

Fig. 6 Screenshot from WeaveStudio CAD/CAM software of double-layer twill weave geometry and warp stitch down layer binding

The double-layer weave introduced so far shows warp and weft yarns arranged in the ratio 1:1. Fig. 7 shows a double-layer twill weave with the warp yarns arranged in the ratio 2:3 and the weft yarns arranged in the ratio 1:1. Alteration of the warp yarn ratio for a double-layer weave from t:u to v:w changes the number of adjacent warp yarns grouped together in layer 1 from t to v and layer 2 from u to w. The weft yarn ratio similarly applies.

The parameters required for a compact specification of 3D multi-layer weaves are: number of layers, layer descriptors, layer step numbers and warp and weft yarn ratios. From the specification, the algorithms described in this paper automatically generate 3D multi-layer weaves and design paper updates for layer binding stitches. The 3D multi-layer weave parameters are specified in the CAD interface prior to automatic weave generation and subsequent layer binding as described in the following two sections.

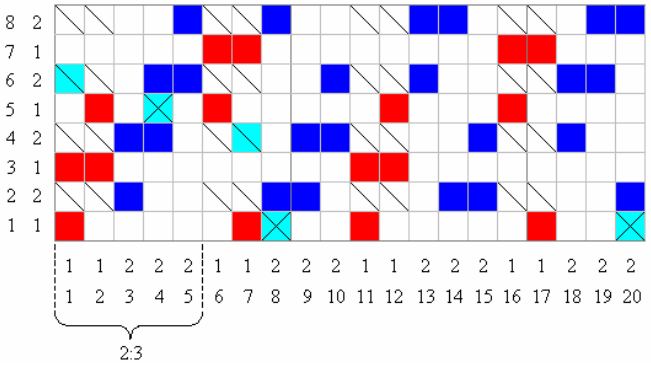

Fig. 7 Screenshot from WeaveStudio CAD/CAM software of double-layer twill weave in the warp ratio 2:3 and weft ratio 1:1

III. AUTOMATIC GENERATION OF 3D MULTI-LAYER WEAVES

At each stage of weave generation, it is essential to ensure correct intermediate weave dimensions are computed. This will ensure the final composite weave is of unit size and therefore guarantee there are no deficient or surplus yarns resulting in an erroneous weave. Fig. 8 shows the specification and overall process framework of key 3D multi-layer weave generation algorithms. Associated WeaveStudio CAD/CAM processes related to software interface, model rendering, weave archive, weave printing and loom interface are omitted for clarity.

Language independent pseudocode is used to express the algorithms to allow porting to a specific implementation. The WeaveStudio CAD/CAM system was implemented in C/C++ [11]. The $2D_{j,k}$ and $3D_{i,j,k}$ subscripted data structures corresponding to the weave schematics in Fig. 8 were implemented using the STL vector container [12] to take advantage of automatic memory management and dynamic resizing capabilities. Note that each array element corresponds to a coordinate location on the associated design paper.

Compute2DWeaveDimensions is responsible for computing the warp and weft dimensions of each of the specified weaves. Line 1 iterates over each constituent 2D layer (the i^{th} layer) of an m layer 3D multi-layer weave in order to compute each unit size weave warp and weft dimensions $warp_i \times weft_i$. Lines 2-5 initialise the $weft_i$ dimension to zero and use summation to compute the i^{th} layer weft dimension. The weave descriptor $e_{i,l}$ indexes into the l^{th} of n descriptor elements for the i^{th} layer where each descriptor element refers to a warp yarn float length. Lines 6-10 compute the minimum absolute step (*MAS*) value of the step number conditional on the sign of the step number $step_i$ of the i^{th} layer weave. The computation accounts for the equivalence of positive and negative step numbers. As an example, the 2/2 step 1 twill weave shown in Fig. 3 (a) can be equivalently specified as 2/2 step –3. Calls to min and abs functions return the minimum of two values and the absolute value of function argument(s) respectively. Lines 11-15 compute the i^{th} layer warp dimension. The computation is dependent on the conditional logic in line 11 that uses modulo arithmetic (MOD operator) to test if *MAS* can be integrally

divided into $weft_i$.

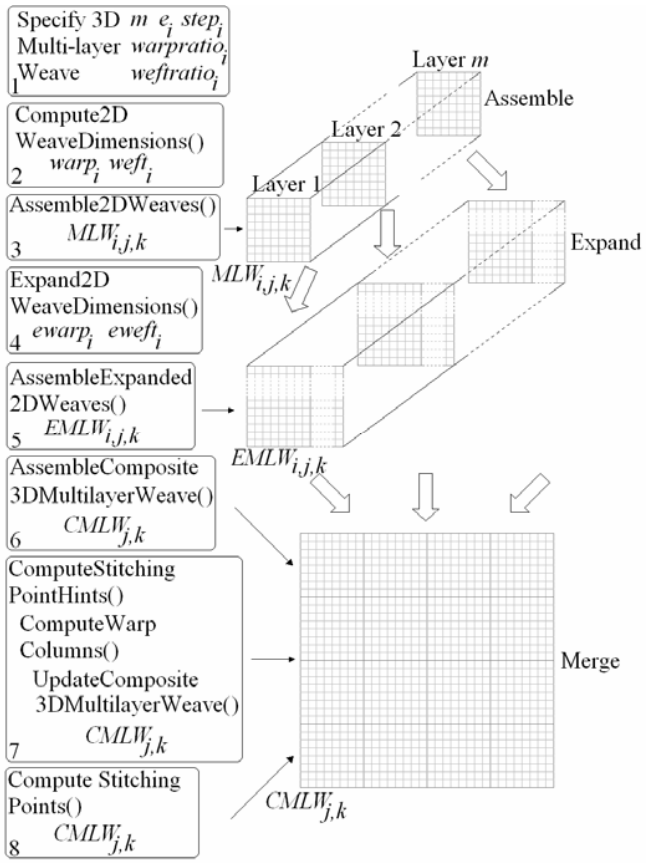

Fig. 8 Block diagram of specification and key process framework

```
Compute2DWeaveDimensions()
 1 for each layer_i i ∈ {1, 2, …, m}
 2    weft_i = 0
 3    for each e_i,l l ∈ {1, 2, …, n}
 4       weft_i = weft_i + e_i,l
 5    end for
 6    if (step_i > 0)
 7       MAS = min(abs(step_i), abs(step_i – weft_i))
 8    else
 9       MAS = min(abs(step_i), abs(step_i + weft_i))
10    end if
11    if (weft_i MOD MAS = 0)
12       warp_i = weft_i / MAS
13    else
14       warp_i = weft_i
15    end if
16 end for
```

```
Assemble2DWeaves()
 1 for each layer_i i ∈ {1, 2, …, m}
 2    for each e_i,l l ∈ {1, 2, …, n}
 3       if (l = 1)
 4          rowstart_i = 1
 5       else
 6          rowstart_i = rowend_i + 1
 7       end if
 8       rowend_i = rowstart_i + e_i,l – 1
 9       if (l AND 1)
10          interlacement = WARP_UP
11       else
12          interlacement = WARP_DOWN
13       end if
14       for each row_j j ∈ {rowstart_i, …, rowend_i}
15          MLW_i,j,1 = interlacement
16       end for
17    end for
18    if (step_i < 0)
19       step = step_i + weft_i
20    else
21       step = step_i
22    end if
23    for each col_k k ∈ {2, 3, …, warp_i}
24       for each row_j j ∈ {1, 2, …, weft_i}
25          if (j + step ≤ weft_i)
26             MLW_i,j+step,k = MLW_i,j,k-1
27          else
28             MLW_i,j+step–weft_i,k = MLW_i,j,k–1
29          end if
30       end for
31    end for
32 end for
```

Assemble2DWeaves is responsible for the formation of each of the specified 2D weaves. Line 1 iterates over each constituent 2D layer in order to compute each unit size weave. Line 2 iterates over each descriptor element (i.e. warp yarn float length) for the i^{th} layer. Lines 3-16 compute the weave for the 1st warp column of the i^{th} layer. Lines 3-7 compute the index into the i^{th} layer weft row start $rowstart_i$ conditional on whether the computation is for the first descriptor element or subsequent descriptor elements. Line 8 computes the index into the i^{th} layer weft row end $rowend_i$ as an offset from $rowstart_i$. Lines 9-13 compute the yarn interlacement status notational constant WARP_UP or WARP_DOWN using the bitwise AND operator in line 9 that returns the bitwise AND of the l subscript and 1 constant operands. Lines 14-16 populate a subset of the 3D multi-layer weave (MLW) array with the yarn interlacement status. The populated elements belong to the i^{th} layer, j^{th} weft row ($rowstart_i \le j \le rowend_i$) and 1st warp column. By convention, each element of the MLW array is referenced by the numerical subscripts i,j,k that denote layer, weft row and warp column respectively. Unless otherwise stated, the pseudocode uses one-based indexing and subscripting throughout to maintain consistency with the weaving convention that uses one-based yarn numeric identifiers. Lines 18-31 compute the remaining warp columns of the weave for the i^{th} layer. Lines 18-22 use conditional logic to compute the step number $step \in \mathbb{N}_1$ to account for the equivalence of positive and negative user specified step numbers. Lines 23-31 iterate over the remaining warp column and weft row MLW array elements to populate the MLW array to complete assembly of all 2D weaves. Line 26 and line 28 compute an array entry of the k^{th} MLW array column using the previously computed entry in the k-1th MLW array column. On

algorithm termination, the *MLW* array stores all *m* distinct weaves in readiness for weave expansion.

```
  Expand2DWeaveDimensions()
 1 i = 1
 2 multiplier = 1
 3 while (i ≤ m)
 4    if ((multiplier × warpratio_i) MOD warp_i = 0)
 5       i = i + 1
 6    else
 7       multiplier = multiplier + 1
 8       i = 1
 9    end if
10 end while
11 for each layer_i i ∈ {1, 2, …, m}
12    ewarp_i = multiplier × warpratio_i
13 end for
```

The original weave dimensions must be expanded to account for user-specified warp and weft yarn ratios. To perform the expansion Expand2DWeaveDimensions is a brute-force minimisation algorithm that finds the minimum *multiplier* value that guarantees an i^{th} layer expanded weave dimension is integrally divisible by the i^{th} layer unit size weave dimension and the final composite 3D multi-layer weave is of unit size. Specifically, the algorithm is responsible for computing the expanded warp and weft dimensions of each of the specified weaves. Lines 1-2 initialise the layer index *i* and *multiplier* at the start of 2D weave dimension expansion. Lines 3-10 compute *multiplier* used in the expansion product in line 12. The warp ratio ($warpratio_i$) in line 4 and line 12 refers to the number of adjacent warp yarns grouped together for the i^{th} layer. The key operation is in line 4 where the product of *multiplier* and $warpratio_i$ is the left operand and $warp_i$ is the right operand of the MOD operator. If the right operand is integrally divisible by the left operand the same operation is applied again for the next i^{th} layer. If the line 4 conditional evaluates to false, *multiplier* is incremented in line 7 and line 8 resets the layer index *i* to start again from the first layer. Lines 11-13 perform the computation for the expanded warp dimension $ewarp_i$ for all *m* layers. A similar algorithm is used to calculate $eweft_i$. The remainder of the Expand2DWeaveDimensions algorithm (not shown) for expansion of weft dimensions is similar to lines 1-13 of Expand2DWeaveDimensions. $warpratio_i$, $warp_i$ and $ewarp_i$ are substituted for $weftratio_i$, $weft_i$ and $eweft_i$ respectively.

```
  AssembleExpanded2DWeaves()
1 for each layer_i i ∈ {1, 2, …, m}
2    for each row_j j ∈ {1, 2, …, eweft_i}
3       for each col_k k ∈ {1, 3, …, ewarp_i}
4          EMLW_{i,j,k} = MLW_{i,(j MOD weft_i)+1,(k mod warp_i)+1}
5       end for
6    end for
7 end for
```

AssembleExpanded2DWeaves is responsible for taking the *MLW* array as input to assemble each of the expanded 2D weaves derived from the originally formed unit size 2D weaves. The algorithm iterates over each layer, expanded weft row and expanded warp column to compute all *m* expanded 2D weaves. The computation in line 4 uses modulo arithmetic to compute the expanded 3D multi-layer weave *EMLW* array derived from indexing into the *MLW* array. Each element of the 3D *EMLW* array is referenced by the numerical subscripts *i,j,k* using the same convention as for the *MLW* array. On algorithm termination, the *EMLW* array stores all *m* distinct expanded weaves in readiness for merging to assemble the composite 3D multi-layer weave.

```
  AssembleComposite3DMultilayerWeave()
 1 rows = ∑eweft_i
 2 columns = ∑ewarp_i
 3 warpRA = warpratio_1
 4 weftRA = weftratio_1
 5 warplayer = 0
 6 weftlayer = 0
 7 for each col_k k ∈ {1, 2, …, columns}
 8    if (k > warpRA)
 9       AdvanceWarpLayer()
10    end if
11    for each row_j j ∈ {1, 2, …, rows}
12       if (j > weftRA)
13          AdvanceWeftLayer()
14       end if
15       if (weftlayer > warplayer)
16          CMLW_{j,k} = LIFTER
17       else if (weftlayer < warplayer)
18          CMLW_{j,k} = WARP_DOWN
19       else
20          if ( EMLW_{warplayer+1, weftidx_{warplayer+1}, warpidx_{warplayer+1}} =
            WARP_UP)
21             CMLW_{j,k} = WARP_UP_{warplayer+1}
22          else
23             CMLW_{j,k} = WARP_DOWN
24       end else
25    end for
26    warpidx_{warplayer+1} = warpidx_{warplayer+1} + 1
27    weftidx_{warplayer+1} = 1
28    weftRA = weftratio_1
29    weftlayer = 0
30 end for
```

AssembleComposite3DMultilayerWeave is responsible for taking the *EMLW* array as input and merging distinct expanded 2D weaves into a single composite 3D multi-layer weave (*CMLW*) Lines 1-2 use summation to compute the total number of weft rows and warp columns respectively for the *CMLW* array. In this context, summation is understood to mean the sum of all *m* $eweft_i$ and $ewarp_i$ array elements for all *m* layers. Lines 3-4 initialise accumulated warp and weft ratios (*warpRA* and *weftRA* respectively). These accumulated values are used by the algorithm to determine if increments are required to the warp and weft layer numbers initialised in lines 5-6 respectively. The warp and weft layer numbers (*warplayer*

and *weftlayer*) are used to maintain the current zero-based weave layer number for the k^{th} warp column and j^{th} weft row respectively. Line 7 and line 11 iterate over all warp columns and weft rows respectively of the *CMLW* array to determine the yarn interlacement status at each j^{th}, k^{th} coordinate location. Each element of the *CMLW* array is referenced by the numerical subscripts *j,k* that denote row and column respectively. Lines 8-10 and 12-14 perform updates to *warplayer* and *weftlayer* and corresponding ratio accumulators respectively.

```
AdvanceWarpLayer()
1 warplayer = warplayer + 1
2 warplayer = warplayer MOD m
3 warpRA = warpRA + warpratio_warplayer+1
```

AdvanceWarpLayer called in line 9 of AssembleComposite3DMultilayerWeave uses modulo arithmetic to update the zero-based weave layer number for the k^{th} warp column of an *m* layer 3D multi-layer weave. A similar implementation is used to update the weave layer number for the j^{th} weft row. AdvanceWeftLayer substitutes *warplayer*, *warpRA* and *warpratio* found in AdvanceWarpLayer for *weftlayer*, *weftRA* and *weftratio* respectively. Lines 15-24 of AssembleComposite3DMultilayerWeave determine the yarn interlacement status for the j^{th} weft row and k^{th} warp column. For example and by inspection of Fig. 3 (b), it can be seen that $CMLW_{2,1}$ refers to the 2nd weft row and the 1st warp column. The binary relational operator in line 15 evaluates to true and $CMLW_{2,1}$ is assigned the lifter notation. Line 17 evaluates to true for $CMLW_{1,2}$ and the 1st weft row and the 2nd warp column is assigned the warp down notation. Line 20 indexes into the *EMLW* array if *warplayer* = *weftlayer*. From Fig. 3 (b) $CMLW_{1,1}$ refers to the 1st weft row and 1st warp column of the layer 1 weave. Line 20 indexes into the *EMLW* array, retrieves the yarn interlacement status and assigns the warp up notation for the 1st layer at $CMLW_{1,1}$. Similarly, $CMLW_{2,2}$ is assigned the warp up notation for the 2nd layer after indexing into the *EMLW* array. Line 20 indexes into the 3D *EMLW* array using the described layer, row and column convention. In particular, the weft row and warp column indices *weftidx* and *warpidx* are both 1D arrays that maintain the current layer (i.e. *warplayer*) weft and warp coordinate locations respectively into the *EMLW* 3D array. All elements of both 1D arrays are initialised to 1 before invocation of AssembleComposite3DMultilayerWeave to ensure indexing into the *EMLW* array commences at the 1st weft row and 1st warp column for each layer. Note $WARP\_UP_{warplayer+1}$ in line 21 signifies a notational constant unique coloured square to disambiguate between layers in the composite weave. At the end of populating the k^{th} column of the *CMLW* array, lines 26-29 perform the necessary bookkeeping prior to populating the $k+1^{th}$ column. Line 26 increments the *warpidx* for the specified *warplayer*. Line 27 resets the *weftidx* to start at the 1st weft row for the specified *warplayer*. Line 28 resets the *weftRA* to the weft ratio of the 1st layer ($weftratio_1$). Line 29 resets the zero-based *weftlayer* to the 1st layer. On algorithm termination, the *CMLW* array stores all *m* composited weaves in readiness for layer binding stitches.

IV. LAYER BINDING STITCHES

The formation of an integrated 3D woven multi-layer textile structure requires the introduction of layer binding stitches to attach the distinct fabric layers together. Further elaboration on stitching can be found in §II.

```
ComputeStitchingPointHints()
1 warpRA = warpratio_1
2 weftRA = weftratio_1
3 warplayer = 0
4 weftlayer = 0
5 for each col_k k ∈ {1, 2, …, columns}
6   if (k > warpRA)
7     AdvanceWarpLayer()
8   end if
9   for each row_j j ∈ {1, 2, …, rows}
10    if (j > weftRA)
11      AdvanceWeftLayer()
12    end if
13    if (weftlayer = warplayer)
14      cacheweftRA = weftRA
15      cacheweftlayer = weftlayer
16      south = weftRA + 1
17      for each layer_i i ∈ {1, 2, …, m - 1}
18        AdvanceWarpLayer()
19      end for
20      north = weftRA
21      if (CMLW_south-1,k = CMLW_(north MOD rows)+1,k)
22        ComputeWarpColumns()
23      end if
24      j = weftRA
25    end if
26  end for
27  weftRA = weftratio_1
28  weftlayer = 0
29 end for
```

ComputeStitchingPointHints is responsible for updating the composite 3D multi-layer weave with coordinate locations that can be used to stitch layers together without fabric interference. Accumulated warp and weft ratios and layers are initialised in lines 1-4. Line 5 and line 9 iterate over all warp columns and weft rows respectively of the *CMLW* array to determine the layer binding stitches at each j^{th}, k^{th} coordinate location. Lines 6-8 and 10-12 perform updates to *warplayer* and *weftlayer* and corresponding ratio accumulators respectively. Lines 13-25 determine if an up float or down float has occurred at the $CMLW_{j,k}$ coordinate location ($j \in A$, $A = \{south - 1, (north \text{ MOD } rows) + 1\}$) of the k^{th} column of the *CMLW* array. Both elements of set *A* will simultaneously assume the notational constant $WARP\_UP_i$ *or* WARP_DOWN in the event of an up float or down float respectively. Line 16 and line 20 compute *south* and *north*

respectively and these values identify $CMLW_{j,k}$ coordinate locations ($south \leq j \leq north$) that are candidates for layer binding stitches. The conditional in line 21 performs a comparison between two *CMLW* array elements to determine if an up float or down float has occurred. ComputeWarpColumns is called in line 22 if the conditional in line 21 evaluates to true. Line 24 updates the loop counter *j* to ensure that previously examined *CMLW* array elements for possible stitching are not re-examined. Lines 27-28 reset the weft ratio accumulation (*weftRA*) and weft layer (*weftlayer*) respectively between column iterations. On algorithm termination, the *CMLW* array stores all possible non-interfering layer binding stitch locations.

```
   ComputeWarpColumns()
 1 weftRA = cacheweftRA
 2 weftlayer = cacheweftlayer
 3 cachewarplayer = warplayer
 4 for each row_j j ∈ {south, south + 1, …, north}
 5    if (j > weftRA)
 6       AdvanceWeftLayer()
 7       warplayer = cachewarplayer
 8       if (weftlayer ≠ warplayer)
 9          east = warpRA + 1
10          warplayer = warplayer + 1
11          warplayer = warplayer MOD m
12          while (weftlayer ≠ warplayer)
13             east = east + warpratio_warplayer+1
14             warplayer = warplayer + 1
15             warplayer = warplayer MOD m
16          end while
17          if (east > columns)
18             east = east - columns
19          end if
20          west = east - 1
21          for each layer_i i ∈ {1, 2, …, m}
22             if (i ≠ warplayer + 1)
23                west = west - warpratio_i
24             end if
25          end for
26          if (west < 1)
27             west = west + columns
28          end if
29       end if
30    end if
31    j2 = j
32    if (j2 > rows)
33       j2 = j2 - rows
34    end if
35    UpdateComposite3DMultilayerWeave()
36 end for
37 warplayer = cachewarplayer
```

ComputeWarpColumns is responsible for computing the two warp column indices (*east* and *west*) located directly east and directly west of the $CMLW_{j,k}$ coordinate location under consideration. Specifically, the two warp column indices are the column positions in the *CMLW* array where the adjacent warp yarn floats belonging to the layer being stitched to are located. Lines 1-2 restore the cached accumulated weft ratio (*weftRA*) and weft layer (*weftlayer*) respectively that were stored in ComputeStitchingPointHints. Line 3 caches the warp layer (*cachewarplayer*) ahead of local updates to the same. Line 4 iterates over candidate coordinate locations ($CMLW_{j,k}$ $south \leq j \leq north$) for layer binding stitches. Changes to *CMLW* array column elements for warp stitch up or warp stitch down can only occur on change of weft layer (*weftlayer*) and when *weftlayer* and *warplayer* are unequal. If the conditional expressions in line 5 and line 8 evaluate to true, lines 9-28 compute *east* and *west* that identify the adjacent float *CMLW* array columns. Lines 17-19 and lines 26-28 guard against overflow that would result in indexing into the *CMLW* array beyond the array bounds. The overflow guards guarantee to wrap around to the first column (*east* lines 17-19) and last column (*west* lines 26-28). Similarly, lines 32-34 guard against out of bounds array row indexing prior to *CMLW* array update.

UpdateComposite3DMultilayerWeave updates the *CMLW* array with a location hint for possible application of warp stitch up (see Fig. 5) or warp stitch down (see Fig. 6). All four conditional expressions must be true to update the *CMLW* array with a hint that a layer binding stitch operation is possible. Note if the conditional expression in line 1 of UpdateComposite3DMultilayerWeave evaluates to true, an up float is guaranteed. This is as a result of the test for equality in line 21 of ComputeStitchingPointHints. The remainder of the UpdateComposite3DMultilayerWeave algorithm (not shown) for possible application of a warp stitch down is similar to lines 1-9 of UpdateComposite3DMultilayerWeave. $WARP\_UP_i$ and POSSIBLE_WSU are substituted for WARP_DOWN and POSSIBLE_WSD respectively. The less than operator in line 2 is changed to the greater than operator.

```
   UpdateComposite3DMultilayerWeave
1  if (CMLW_souh-1,k = WARP_UP_i)
2     if (weftlayer < cachewarplayer)
3        if (CMLW_j2,east = WARP_UP_i)
4           if (CMLW_j2,west = WARP_UP_i)
5              CMLW_j2,k = POSSIBLE_WSU
6           end if
7        end if
8     end if
9  end if
```

The ComputeStitchingPoints algorithm is critical to the formation of the complete and final composite 3D multi-layer weave. It allows the user to interactively edit yarn interlacement to stitch together individual layers transforming them into attached 3D multi-layer weaves. The ComputeStitchingPoints algorithm drives the logic to perform *CMLW* array lookup and updates to accurately compute yarn interlacement and layer binding stitches.

Note the concise bitwise left-shift (<<) and bitwise right-shift assignment (>>=) operators in the pseudocode borrow from C/C++ [11] to avoid verbose operator descriptions.

Lines 1-16 of ComputeStitchingPoints include iteration over specified rows ($rowstart \leq j \leq rowstart + m - 1$) to perform *CMLW* lookup to compute the bit set *warpbits*. For the quadruple-layer weave example shown in Fig. 9 and Fig. 10 *warpbits* post loop can compute to 1111_2, 111_2, 11_2, 1_2 or 0_2 depending on one of five possible positions of yarn interlacement/layer binding stitch. As an example and referring to Fig. 10 and given a user selection of weft yarn 11 using the mouse, *weftselected* = 11, number of layers $m = 4$ for a quadruple-layer weave, *rowstart* = 9, *warpbits* = 1111_2 pre loop, $j \in \{9, 10, 11, 12\}$, k(th column) = 7 (7th warp yarn pertaining to layer 3 as a result of user selection of weft yarn 11), *warpbits* = 1_2 post loop where all values are in base 10 unless explicitly stated to be in base 2 using the 2 subscript. As the user repeatedly selects weft yarn 11 in the yarn editor interface, the yarn interlacement/layer binding stitch notation for warp yarn 7, layer 3 is cycled repeatedly in the following sequence: '□□■\' '□□□\' '□□□○' '●●■\' '□●■\'.

In the quadruple-layer weave example in Fig. 9 and Fig. 10 the user has cycled the yarn interlacement/layer binding stitch notation from the default 2/2 twill quadruple-layer interlacement '□□■\' to '□●■\'.

The resulting weave update is shown in the highlighted stitched square region of design paper in Fig. 9 and highlighted stitched weft yarns 9-12 in Fig. 10. It can be seen from the design paper and yarn editor that warp yarn 7, layer 3 stitches up to layer 2. It can also be seen that warp yarn 6, layer 2 stitches down to layer 3. Lines 17-39 include iteration over specified rows ($rowstart \leq j \leq rowstart + m - 1$) and layer numbers ($1 \leq i \leq m$) as part of the final design paper update of yarn interlacement/layer binding stitch notation. Comparisons between the i^{th} layer number and the user selected layer number (*layerselected*) in addition to application of the bitwise AND operator between *warpbits* bit set and *warpmask* bit mask drive *CMLW* array update of interlacement and stitching notation. On algorithm termination, the *CMLW* array stores all layer binding stitching locations and the complete and final 3D composite 3D multi-layer weave in readiness for loom production.

A set of piecewise cubic splines [13] is used for internal representation and visualisation of each warp yarn path in the multi-layer weave yarn editor shown in Fig. 10. In addition to automatic design paper updates resulting from yarn editor manipulations, a 3D solid model of the weave is updated in real-time to show the layer binding stitches and tortuous yarn paths (see Fig. 5 and Fig. 6 for examples). Our earlier work describes geometric modeling algorithms for textile CAD [14].

```
ComputeStitchingPoints()
1 rowstart = weftselected – ((weftselected – 1) MOD m)
2 warpbits = (1 << m) - 1
3 for each row_j j ∈ {rowstart, rowstart + 1, …, rowstart + m - 1}
4     if (CMLW_j,k = WARP_DOWN)
5         exit for each
6     end if
7     if (CMLW_j,k = WSD)
8         exit for each
9     end if
10    warpbits >>= 1
11 end for
12 if (warpbits = 0)
13    warpbits = (1 << m) – 1
14 else
15    warpbits >>= 1
16 end if
17 warpmask = 1 << (m – 1)
18 for each row_j j ∈ {rowstart, rowstart + 1, …, rowstart + m - 1},
   layer_i i ∈ {1, 2, …, m}
19    if (i > layerselected)
20        if (warpbits AND warpmask)
21            CMLW_j,k = LIFTER
22        else
23            CMLW_j,k = WSD
24        end if
25    else if (i = layerselected)
26        if (warpbits AND warpmask)
27            CMLW_j,k = WARP_UP_i
28        else
29            CMLW_j,k = WARP_DOWN
30        end if
31    else if (i < layerselected)
32        if (warpbits AND warpmask)
33            CMLW_j,k = WSU
34        else
35            CMLW_j,k = WARP_DOWN
36        end if
37    end if
38    warpmask >>= 1
39 end for
```

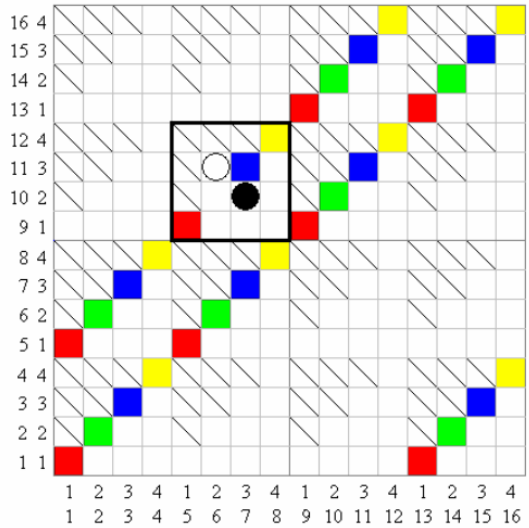

Fig. 9 Screenshot from WeaveStudio CAD/CAM software of a quadruple-layer twill weave

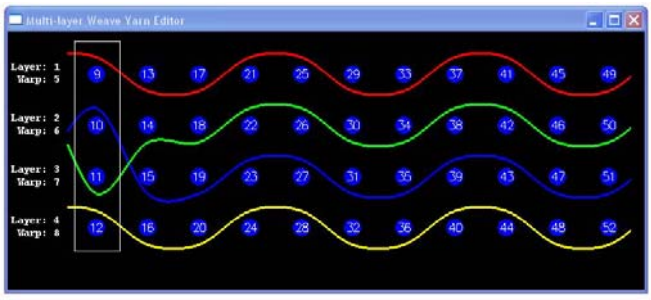

Fig. 10 Screenshot from WeaveStudio CAD/CAM software of warp yarn interlacement and stitching of a quadruple-layer twill weave

V. RESULTS

Table 1 shows the 3D multi-layer weave specification used to illustrate the algorithms described in the previous two sections:

TABLE I
3D MULTI-LAYER WEAVE SPECIFICATION

Layer $m = 3$	Descriptor	Step Number	Warp Yarn Ratio	Weft Yarn Ratio
1	$e_1 = 2/2$	$step_1 = 1$	$warpratio_1 = 2$	$weftratio_1 = 2$
2	$e_2 = 2/3$	$step_2 = 1$	$warpratio_2 = 3$	$weftratio_2 = 2$
3	$e_3 = 3/2$	$step_3 = -1$	$warpratio_3 = 4$	$weftratio_3 = 2$

The CAD interface in Fig. 11 shows the specification and properties windows for a 3D multi-layer weave. Fig. 11 (a) shows the weave specification parameters including the facility to navigate between layer specifications and edit parameters according to end-user or application requirements. Fig. 11 (b) shows editable properties of the weave geometry such as density, dimensions, crimp and cross-sectional shape [14].

The key process stages shown in Fig. 8 are followed in order to arrive at the complete and final composite 3D multi-layer weave. The initial weave dimensions are computed in stage 2:

$weft_1 = 2 + 2 = 4$
$weft_2 = 2 + 3 = 5$
$weft_3 = 3 + 2 = 5$
$warp_1 = weft_1 / MAS = 4 / 1 = 4$
$warp_2 = weft_2 / MAS = 5 / 1 = 5$
$warp_3 = weft_3 / MAS = 5 / 1 = 5$

The 2D weaves are assembled in stage 3 and are shown in Fig. 12.

The expanded weave dimensions are computed in stage 4:

$ewarp_1 = multiplier \times warpratio_1 = 10 \times 2 = 20$
$ewarp_2 = multiplier \times warpratio_2 = 10 \times 3 = 30$
$ewarp_3 = multiplier \times warpratio_3 = 10 \times 4 = 40$
$eweft_1 = multiplier \times weftratio_1 = 10 \times 2 = 20$
$eweft_2 = multiplier \times weftratio_2 = 10 \times 2 = 20$
$eweft_3 = multiplier \times weftratio_3 = 10 \times 2 = 20$

The expanded 2D weaves are assembled in stage 5 and are shown in Fig. 13.

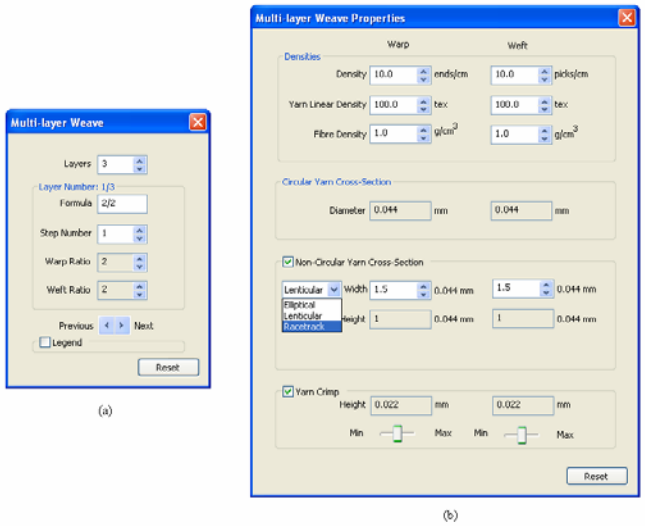

Fig. 11 Screenshot from WeaveStudio CAD/CAM software of (a) multi-layer weave specification and (b) multi-layer weave properties

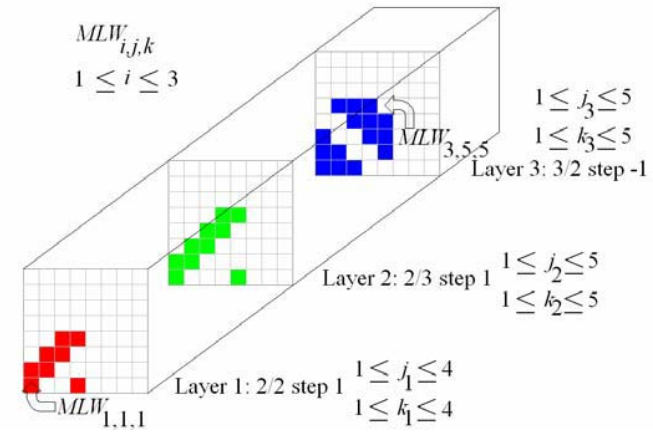

Fig.12 Adapted screenshot from WeaveStudio CAD/CAM software of 2D weave assembly

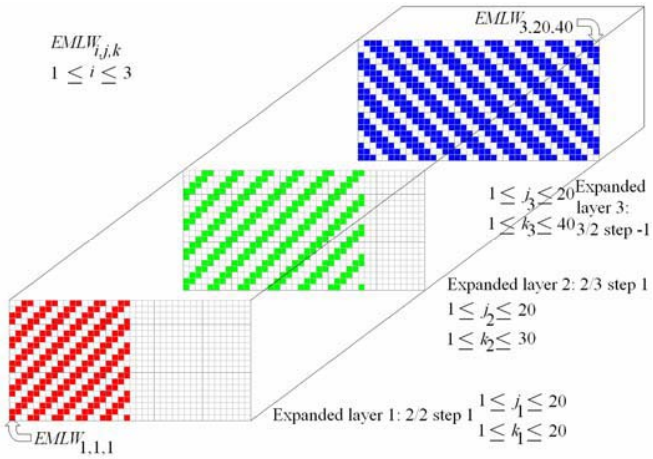

Fig.13 Adapted screenshot from WeaveStudio CAD/CAM software of expanded 2D weave assembly

The composite 3D multi-layer weave $CMLW_{j,k}$ ($1 \le j \le 60$, $1 \le k \le 90$) is assembled in stage 6 and is shown in Fig. 14.

The composite 3D multi-layer weave with hints for non-interfering layer binding stitches is assembled in stage 7 and is shown in Fig. 15.

The final composite 3D multi-layer weave with (optional non-interfering) layer binding stitches is assembled in stage 8 and a subset is shown in Fig. 16. Identification of $CMLW_{j,k}$ coordinate locations ($south \le j \le north$) for the k^{th} column that are candidates for layer binding stitches are shown in Fig. 16 (b) and Fig. 16 (c).

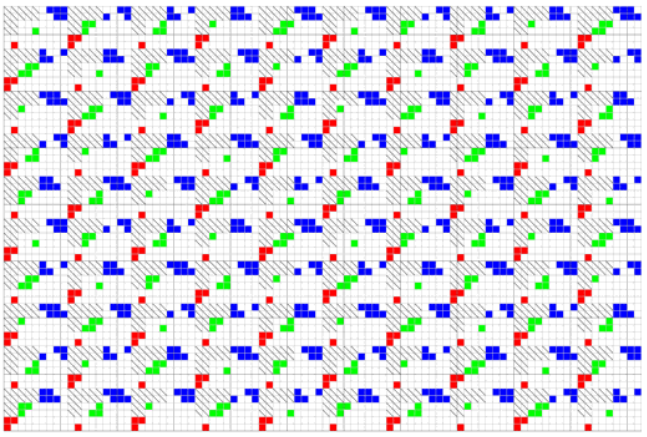

Fig. 14 Screenshot from WeaveStudio CAD/CAM software of the composite 3D multi-layer weave

The $CMLW_{j,k}$ coordinate locations shown in Fig. 16 (b) for potential warp stitch down are $9 \le j \le 12$, $k = 2$. The *east*, *west* $CMLW_{j,k}$ coordinate locations $CMLW_{9,3}$ and $CMLW_{9,86}$ respectively show layer 2 warp yarns pass under layer 2 weft yarns and therefore a non-interfering warp stitch down is possible at $CMLW_{9,2}$. Lines 26-28 of ComputeWarpColumns ensure wrap around to the last column for *west* computation. The $CMLW_{j,k}$ coordinate locations shown in Fig. 16 (c) for potential warp stitch up are $5 \le j \le 8$, $k = 4$. The *east*, *west* $CMLW_{j,k}$ coordinate locations $CMLW_{7,10}$ and $CMLW_{7,2}$ respectively show layer 1 warp yarns pass over layer 1 weft yarns and therefore a non-interfering warp stitch up is possible at $CMLW_{7,4}$.

VI. CONCLUSIONS

A key process framework has been presented together with detailed algorithms for CAD/CAM of 3D woven multi-layer textile structures. An assemble-expand-merge algorithmic approach in the formation of weave designs can relieve the technical weaver of many of the manual free-hand weave assembly operations. The resulting algorithmically generated weave designs show valid yarn paths for 3D multi-layer weaves. In addition, the weave design is updated with location hints to the user for possible application of non-interfering layer binding stitches. The proposed algorithms support comprehensive editing of all weave specification parameters, layer binding stitches and yarn paths to support the creation, modification and prototyping of a wide range of 3D multi-layer weaves according to end-use. Further work remains to partially or fully automate the layer binding stitch process to further reduce 3D multi-layer weave design times.

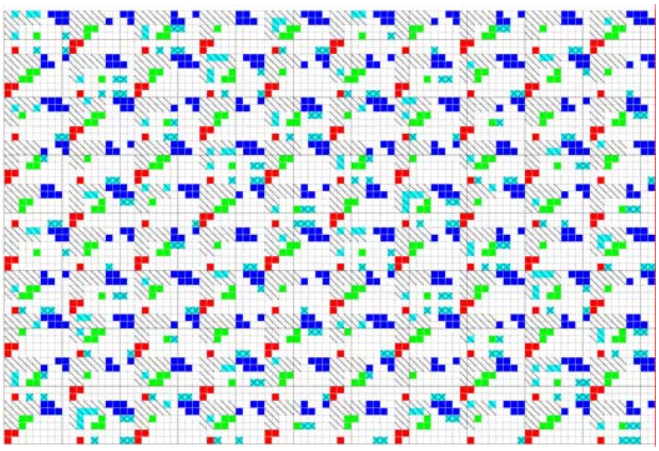

Fig. 15 Screenshot from WeaveStudio CAD/CAM software of the composite 3D multi-layer weave with layer binding stitch hints

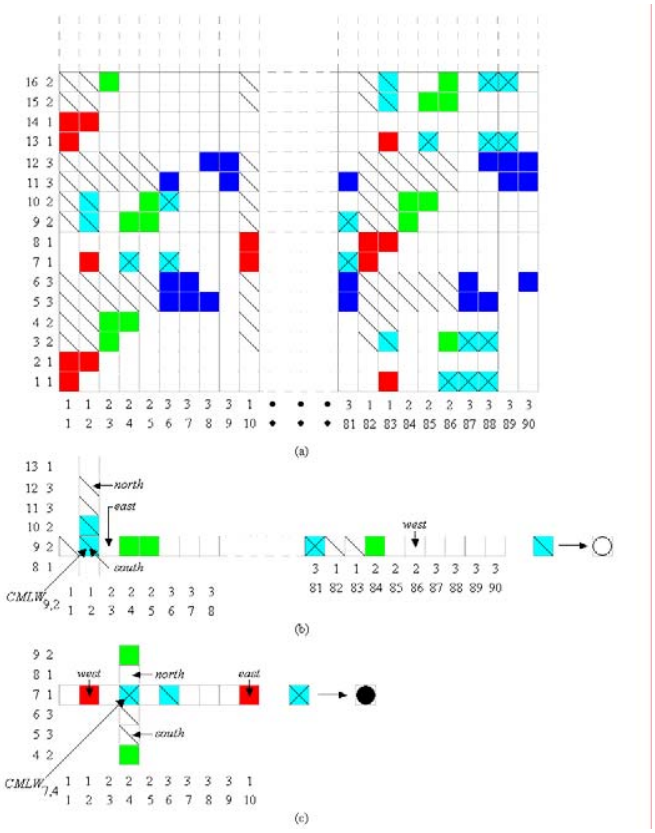

Fig. 16 Adapted Screenshot from WeaveStudio CAD/CAM software of (a) composite weave subset, (b) warp stitch down and (c) warp stitch up

REFERENCES

[1] L. Tong, A.P. Mouritz and M.K. Bannister, *3D fibre reinforced polymer composites*, Elsevier Science, 2002.

[2] X. Chen, R.T. Knox, D.F. McKenna and R.R. Mather, "Automatic generation of weaves for the CAM of 2D and 3D woven textile structures", *J. Text. Inst*., pt. 1, vol. 87, no. 2, pp. 356-370, 1996.
[3] X. Chen and P. Potiyaraj, "CAD/CAM of orthogonal and angle-interlock woven structures for industrial applications", *Text. Res. J*., vol. 69(9), pp. 648-655, 1999.
[4] X. Chen, Y.L. Ma and H.Zhang, "CAD/CAM for cellular woven structures", *J. Text. Inst*., vol. 95, nos. 1-6, pp. 229-241, 2004.
[5] X. Chen and H. Wang, "Modelling and computer-aided design of 3D hollow woven reinforcement for composites", *J. Text. Inst*., pt. 1, vol. 97, no. 1, pp. 79-87, 2006.
[6] N.G. Koltycheva and S.A. Grishanov, "A systematic approach towards the design of a multi-layered woven fabric: Modelling the structure of a multi-layered woven fabric", *J. Text. Inst*., pt. 1, vol. 97, no. 1, pp. 57-69, 2006.
[7] G. Ping and D. Lixin, "Algorithms for computer-aided construction of double weaves: application of the kronecker product", *J. Text. Inst*., pt. 1, vol. 90, no. 2, pp. 158-176, 1999.
[8] X. Chen and P. Potiyaraj, "CAD/CAM for complex woven fabrics part II: multi-layer fabrics", *J. Text. Inst*., pt. 1, vol. 90, no. 1, pp. 73-90, 1999.
[9] J.A. Hewitt, D. Brown and R.B. Clarke, "Modelling, evaluation and manufacture of woven composite materials", *Composites*, pt. *A* 27A, pp. 295-299, 1996.
[10] M. Smith and X.Chen. WeaveStudio CAD/CAM software for 3D woven structures, EPSRC funded project, University of Manchester, UK, 2005-2008.
[11] B. Stroustrup, *The C++ programming language: 3rd Ed.*, Addison Wesley, 1997.
[12] N. Josuttis, *The C++ standard library: A tutorial and reference*, Addison Wesley, 1999.
[13] M. Mortenson, *Geometric modeling: 3rd Ed.*, Industrial Press Inc., 2006.
[14] M.A. Smith and X.Chen, "CAD and constraint-based geometric modelling algorithms for 2D and 3D woven textile structures", *J. Inf. Comp. Sci.*, vol. 3, no. 3, pp. 199-214, 2008.