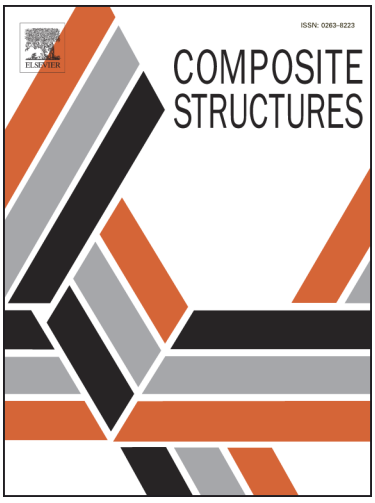
Journal Pre-proofs



Geometrical and Deep Learning Approaches for Instance Segmentation of CFRP Fiber Bundles in Textile Composites

Yuriy Sinchuk, Pierre Kibleur, Jan Aelterman, Matthieu N. Boone, Wim Van Paepegem

PII: S0263-8223(21)01087-4
DOI: https://doi.org/10.1016/j.compstruct.2021.114626
Reference: COST 114626

To appear in: *Composite Structures*

Received Date: 7 April 2021
Accepted Date: 30 August 2021

Please cite this article as: Sinchuk, Y., Kibleur, P., Aelterman, J., Boone, M.N., Paepegem, W.V., Geometrical and Deep Learning Approaches for Instance Segmentation of CFRP Fiber Bundles in Textile Composites, *Composite Structures* (2021), doi: https://doi.org/10.1016/j.compstruct.2021.114626

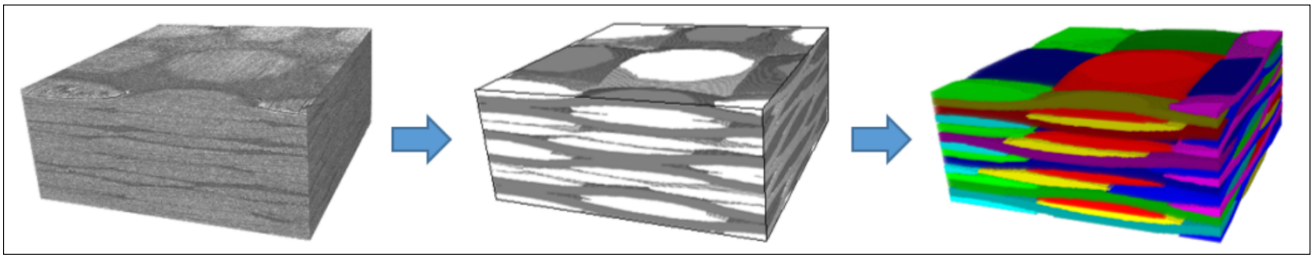
Geometrical and Deep Learning Approaches for Instance Segmentation of CFRP Fiber Bundles in Textile Composites

Yuriy Sinchuk [1,*], Pierre Kibleur [2,5], Jan Aelterman [3,4,5], Matthieu N. Boone [4,5] and Wim Van Paepegem [1]

[1] Department of Materials, Textiles and Chemical Engineering, Faculty of Engineering and Architecture, Ghent University, Technologiepark Zwijnaarde 46, 9052 Zwijnaarde, Belgium; E-Mail: Wim.VanPaepegem@UGent.be
[2] Department of Environment, Faculty of Bioscience Engineering, Ghent University, Coupure Links 653, 9000 Gent, Belgium; E-Mails: Pierre.Kibleur@UGent.be
[3] Department of Telecommunications and information processing - Image Processing and Interpretation Group, Faculty of Engineering and Architecture, Ghent University - imec, Sint-Pietersnieuwstraat 41. 9000 Gent, Belgium; E-Mails: Jan.Aelterman@UGent.be
[4] Department of Physics and astronomy, Faculty of Sciences, Ghent University, Proeftuinstraat 86, 9000 Gent, Belgium; E-Mails: Matthieu.Boone@UGent.be
[5] Ghent University Center for X-ray Tomography (UGCT), Proeftuinstraat 86, 9000 Gent, Belgium
* Correspondence: Yuriy.Sinchuk@UGent.be

Abstract

Segmenting micro-Computed Tomography (μCT) images of textile composites is a necessary step before modeling the material at the mesoscale. However, the accurate segmentation of fiber bundles (or tows) remains a challenge in carbon fiber reinforced textile composites. Segmentation approaches based on local fiber orientation perform well in recognizing individual tows only under ideal conditions, namely when the local fiber orientation bordering two tows' interface is different, or when the touching area is small relative to the thickness of a tow. Unfortunately, in many textile composite laminates used in the industry, these ideal conditions are not found. Such materials often consist of multiple plies, where each fiber is aligned in one of the two orthogonal directions, and where the touching area between similar-orientation tows is often much larger than the tow thickness. Therefore, we propose two new methodologies for splitting tow instances. One is based on the geometrical analysis of the material structure using conventional image analysis; the other is based on the deep learning prediction of ideal inputs for segmentation based on the watershed transform. The deep learning-based method is trained using randomly generated synthetic images of a woven composite material, which avoids an expensive human annotation step.



Keywords: *fabrics/textiles; carbon-fiber reinforced polymer; microstructure modeling; micro-CT based modeling; instance segmentation; deep learning*

1. Introduction

Fiber reinforced polymers (FRPs) are used as a high strength-to-weight ratio material in many different industries: construction, aerospace, marine, automotive, sports equipment, etc. Currently, the prediction of the properties and behavior of these composite materials by means of numerical modeling is one of the key elements of industrial developments. Many approaches to model woven composites' microstructural geometries are compared and reviewed in the cited works [1, 2]. The geometrical model of a layered textile composite is typically constructed for a given scale. It can be the level of a ply, the level of tows (also called unit cells level, or mesoscale), or the level of fibers (micro-scale). The ply-level and the fiber-level geometries are simple and can be generated relatively easily, based on fundamental geometrical parameters like ply thickness, fiber diameter, matrix volume fraction, etc. However, the tow-level geometry is complex to model, due to the need to account for randomly curved surfaces and sharp-cornered shapes. Generating a mesoscale model can reflect one of the following conceptions: (i) idealized – the geometry is characterized by a limited set of parameters, (ii) simulated – the geometry is the output of a simulation of the manufacturing process (e.g. fabric compaction), and (iii) realistic – the geometry is extracted from μCT images of a real material sample.

An idealized geometry model can be created by textile software such as TexGen [3] or WiseTex [4], using a set of characteristic parameters as the input. The relevant parameters for the model can be extracted from processing μCT images [5]. In the work [6], the initial geometry was generated using TexGen, and the resulting "simulated" geometry was obtained by modeling the compaction of the dry fabric by finite element modeling [7]. Another way to get "simulated" geometries of the textile structure is to simulate the (thermal) expansion, starting from an initial (idealized) geometry, until the desired matrix volume fraction

is reached [8]. The studies [9] propose an alternative approach to simulate the weaving and compaction of woven composites, based on the "digital element" technique that results in an accurate geometry at the mesoscale.

A realistic geometry of the tow structure can be reconstructed from a labeled 3D image, resulting from processing (segmenting) μCT images of the material. However, the μCT image segmentation can be very challenging due to low contrasts, low resolution, and noisy data [10, 11]. The structure tensor approach is a frequently used method for local fiber orientation analysis. The method results in images that are then used for an intensity-based segmentation [12-14]. However, a non-gradient based method to analyze the fiber texture is employed in work [15], where the Grey Level Co-occurrence Matrix was used to analyze the texture of μCT images of composites. The commercial software Avizo [16] offers many different segmentation solutions, which were also successfully adopted in the process of modeling the tows' geometry [17, 18]. Recently, deep learning methods were applied to segment challenging μCT images of fiber-reinforced composites [19, 20].

In the case of a multilayer textile composite with bi-directional tows alignment, a fully automatic (semantic) segmentation method usually results in only 3 labels: (i) the polymer matrix, and (ii) the warp and (iii) fill tows [11, 19]. Labeling an individual tow (i.e. instance segmentation) remains delicate, especially at high tow densities, typically when tows account for more than 80% of the volume. Generally, methodologies to split touching instances can be based on one of the following approaches or their combination: (i) boundaries geometry analysis, (ii) local features analysis, (iii) matching with an idealized geometry, and (iv) deep learning. Usually in the first approach, prior information about the object shapes is used to analyze their touching edges. In the works [21, 22], the tows' edge analysis is applied to individual tow labeling. This method is based on mapping the boundary points of the separated tow cross-sections in the current slice to the next slice. However, the first slice is segmented by a semi-automatic approach that ignores the incomplete border tows.

Local feature analysis results in a set of local characteristics (such as distance to tow boundary or centerline, local anisotropy, fiber orientation, etc.), which can be used by segmentation algorithms, such as

watershed, clustering, example-based classification [12], or lead to fiber tracking [17]. In the case of multilayer textile composites, two touching tows often exhibit analogous local characteristics in the contact area, making it challenging to find features robust enough to separate the tows. Therefore, the tow instance segmentation based on the fiber orientation can only be applied to high-resolution images, where individual fibers (or their gradient) are clearly visible. The classical watershed is also inaccurate, if the touching area of the tows is much larger than the tow thickness.

The idea of matching an idealized geometry with the real image or its semantic segmentation can be seen as a variation of the 3D active contour model [23]. In [24], individual tows are segmented by exploiting the result of an idealized geometric model optimization, which fitted the voxelization of the tows' surface to the real image. This approach guarantees a clean and topologically correct result, which can then be used for a finite element mesh generation. However, solving the corresponding optimization problem is intricate, due to its sensitivity to the local minima states.

Today, deep learning approaches with convolutional neural networks (CNN) are effectively used to solve challenging instance segmentations in the field of computer vision [25]. In [26], a deep watershed transform is proposed for the instance segmentation of a semantic segmentation output. This deep watershed approach is based on the prediction of an object's distance field with CNNs. The method's effectiveness is demonstrated on street scene photographies. Similarly, the idea of CNN-based distance field prediction was employed in [27, 28] on microscopic images, for the instance segmentation of touching cells without clear boundaries.

In this study, we continue to develop the segmentation methods proposed in our previous work [11]. To the best of our knowledge, there is no published work devoted to the tow instance segmentation of textile composites with multiple plies, based on deep learning. Furthermore, only a few studies exist where other approaches are developed, despite the relevance of such segmentation for the industry. Therefore, two new methods to segment tow instances are proposed. The first method is based on the interpolation of the inter-ply matrix layer. The second method uses deep learning with a 3D U-net architecture [29] to predict the result of a distance transform (similarly to [27]), before applying the watershed transform. However unlike

work [27], where the predicted splitting regions were used to find proper watershed markers, we focused on predicting the tow centerlines [30] to seed the watershed transform. These markers are expected to be more robust for the detection of tube-shape structures (like tows). The proposed deep learning approach furthermore uses a novel idealized unit cell geometry model, for the generation of training dataset samples. Semantic segmentation results of our previous study [11] are used as an input to test the proposed methods.

2. Material and µCT images

The material under investigation was a plain-weave CFRP laminate (TR3110 360GMP, Mitsubishi Rayon Co., Tokyo, Japan) with layup $[\#(0/90)]_8$. Some details on the composites were described in works [5, 11]. The ply thickness was about 0.23 mm and the count of tow ends per centimeter length was 4.92 (for both weaving directions). A tow contained 3000 carbon fibers, each having a diameter of about 7 µm. The material was periodic in-plane, with a computed periodic pattern size of 4.064x4.064 mm.

The µCT scan of the composite was acquired at the Centre for X-ray Tomography of the Ghent University (UGCT) using the custom-designed µCT system HECTOR. The 3D and 2D views of the material scan are presented in Figure 1a. The raw µCT image was cropped to the size of 357x861x862 voxels, where the voxel pitch v_s is about 5µm, i.e. a bit lower than the diameter of a single fiber. In our coordinate system, the first component corresponds to the out-of-plane (Z) axis. Thus, the physical size of the input image was a bit larger than the periodic pattern size. The image was rotated to align the material axes (out-of-plane, fill and warp directions) with the image's Cartesian system.

The semantic segmentation of these µCT images using variational and deep learning approaches was previously detailed [11]. 5 segmentations were presented: the manual ("ground-truth" image), the result of the variational approach, and 3 deep learning segmentations. However, instance segmentation was not treated in that study. These semantic segmentations are used in the current study as input data for testing the proposed methods. The same "ground-truth" image was also used to evaluate the accuracy of the instance segmentations. The following two sections describe the proposed instance segmentation methods, using the best deep learning semantic segmentation presented in [11] as the input image. Figure 1b shows 3D and 2D views of this image, where the red lines (in 2D view) indicate the touching tows' areas.

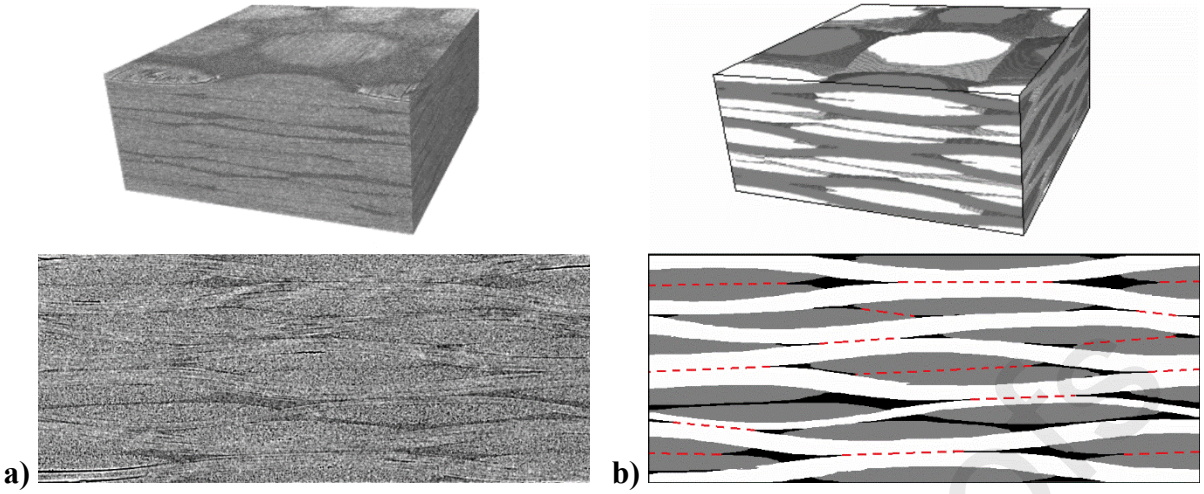


Figure 1: 3D and 2D views of the material images (357x861x862 voxels, voxel size ~5 µm): a) raw µCT data; b) deep learning segmentation result (red lines indicate the touching tows areas).

The proposed methodologies are designed to split tightly compacted tows. These are the cases where the touching area between tows (from two adjacent plies) is large compared to the tow thickness (Figure 1b). The low contrast and high noise in the raw µCT images (Figure 1a) can introduce additional obstacles for the splitting:

- the edges of touching areas may contain segments without concavity;
- false concavity segments can also occur next to the edges of the touching areas;
- noise and local distortion are possible at the tow/matrix interface of the semantic segmentation image.

3. Tow splitting by geometrical approach

Large tow touching areas can only occur between tows belonging to different plies. Therefore, the ply segmentation is way to the tows splitting. Essentially, two adjacent plies could be split by a single (curved) surface that interpolates the matrix layer between touching tows. Specifically, the following algorithm was able to separate plies:

Input: semantic (tow-vs-matrix) segmentation

Output: individual tow instance segmentation

1. *(optional:) Periodical image extension;*
2. *Split matrix layers by flat planes;*

3. *Find medial surface interpolation for each matrix volumetric layer;*
4. *Split plies by the medial surfaces from step 3;*
5. *Split tows within each ply;*
6. *Post-processing.*

The next subsections describe these workflow steps further.

3.1. Periodical image extension

The goal of the periodical image extension, or 4-sides in-plane padding, was to improve the splitting accuracy of tow pieces at the image boundaries. The padding could be especially important to preserve small pieces of tow. However, padding cannot be applied if the length of the image is less than the unit cell period in the extension direction. Also, the boundary accuracy remains limited, and the extension could be omitted if the sample is significantly larger than the unit cell. In our case, the input image was slightly larger than the unit cell size, so this step was essential to the segmentation accuracy. The extension uses the periodicity of the materials' architecture. The specified count of tow ends per centimeter length was 4.92, i.e. the period of the weaving pattern was 813 voxels ($P_f = P_w = 2 \times \frac{10000}{4.92 v_s} = 813$) in both weaving directions. Slices were taken from the input image, to extrapolate the extension. The extension span was set as one-third of the period (i.e. 813/3 ~= 243 voxels). Thus in one direction (3rd coordinate dimension of the image), the first and the last 243 slices were concatenated correspondently at the beginning and end of the stack, to pad the image. To smooth the transition between the input image and the padded regions, a median filter was iteratively applied (kernel size [3,3,7], 12 iterations). The kernel origin was shifted by one voxel in the direction of the region extension (e.g. [0,0,-1] and [0,0,1]), to further smooth the transition. However, this filtering process was only applied in a narrow region of the extended parts: it was 7 voxels in the first iteration, and was incremented by 1 voxel per iteration. This bi-directional extension algorithm was applied for both fill and warp directions, by permuting the second and the third components of the median filter kernel size and kernel origin vectors.

Figure 2 demonstrates a typical slice of the extended image, where red frames highlight the new regions. The extended image size was 357x1347x1348 voxels. Due to the smoothing, the perfect fitting of the extended region to the initial image edges was not required. Note that this step is optional and can be omitted if improving the accuracy of splitting tows at the image edges is not important. Besides, manually splitting tows at the image boundary could provide a more accurate segmentation result than this image extension.

Figure 2: A slice of the periodically extended image (sides with extended regions are within the red frames).

3.2. Matrix layer segmentation

The considered plain weave carbon/epoxy composites can be perceived as a stack of 8 plies of tows, which are separated by 7 inner matrix layers, plus 2 layers at the top and bottom of the stack. The matrix layer segmentation aims to separate these 9 matrix layers. Every pair of the neighboring layers was interconnected throughout the rich matrix pockets of the ply. This step thus consisted in cutting these matrix connections. The simple cutting planes in the middle of the ply could segment the matrix layer well, because the layers were almost parallel in the material plane. Thus, segmenting matrix layers consisted in: finding the middle thickness point of each ply; segmenting matrix layers by the flat planes passing through these points; and cleaning and smoothing the segmentation result. The top- and bottom-most layers, not interesting for the rest of our workflow, could be omitted.

Figure 3a shows the matrix volume fraction throughout the material thickness. The red markers indicating local minima of the matrix density were chosen as the middle thickness points that would be used for matrix segmentation. To detect these minima, the stack was split into 8 equally-thick sub-stacks, and the minimal value of the matrix volume fraction in each sub-stack was found. The 8 minima found would define 9 new sub-stacks, which split the matrix region into layers. Finally, the matrix segmentation

was cleaned with a median filter (size 3x5x5), and small regions (less than 5000 voxels) were removed. In case of noisy input segmentations, a binary opening operation can be applied before the median filter. The parameters for each of these operations require input-specific adjustments, while being sensitive to the quality of the input semantic segmentation. The result of the matrix layer segmentation is presented in Figure 3b. It consists of 7 disconnected layers (the first and last layers were removed), where holes correspond to the tow-touching areas.

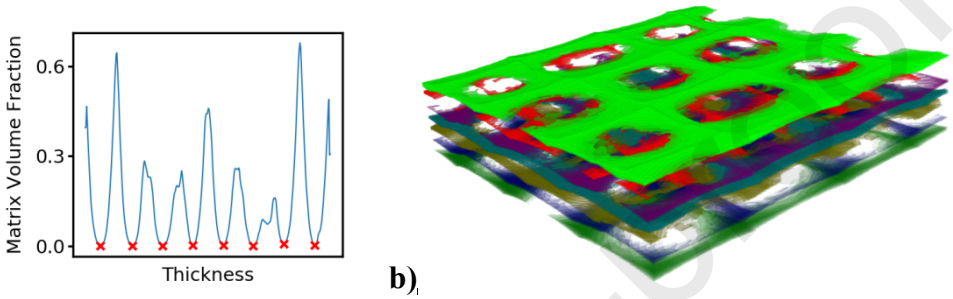


Figure 3: a) matrix volume fraction (red markers indicate ply middle planes); b) matrix segmentation (7 layers)

3.3. Matrix layer interpolation by medial surface

In this step, the assumption was made that plies can be perfectly split by the medial surfaces of the matrix layers. The surfaces are curved to fit to the waviness of the ply boundaries. One of the seven matrix volumetric layers is presented in Figure 4a. For such a layer, the following steps were used to interpolate its medial surface:

Input: matrix layer mask

Output: matrix layer interpolation

1. *Chamfer distance transform (CDT);*
2. *Medial points of the layer are extracted from CDT and stored as a 2D surface image;*
3. *This 2D image is smoothed by a median filter within the non-zero mask;*
4. *Holes (zero regions) are filled by piecewise-linear interpolation;*
5. *The result is smoothed by a Gaussian filter in 2D.*

Out-of-plane indices of the distance field maximal values were computed along the thickness axis. These indices form the Z coordinates of medial surfaces of the matrix layer, and can be presented as a 2D surface image (Figure 4b). The image had holes (zero values) at the ply touching regions (colored in black). A median filter using a disk structuring element (radius 15 voxels) was used to clean the image beside these touching regions (Figure 4c). Because a mask was used, the shape of these regions was untouched. Then, to interpolate matrix within these regions, we considered the planar image as 3 dimensional (the gray values were used as the 3rd coordinate), and used piecewise linear interpolation from the Python SciPy library [31]. The interpolation was able to fill these holes in the matrix interface (Figure 4d). Finally, a Gaussian filter (sigma 15 voxels) was applied to smooth the interpolation (Figure 4e).

Note that the final "surface" is defined only by pixel grey values in the 2D coordinate system of the image. Surface triangulation was not needed, because the sub-voxel resolution accuracy was not relevant to our purpose. The ply splitting ability is demonstrated in Figure 5a, where the interpolated surfaces are shown in red.

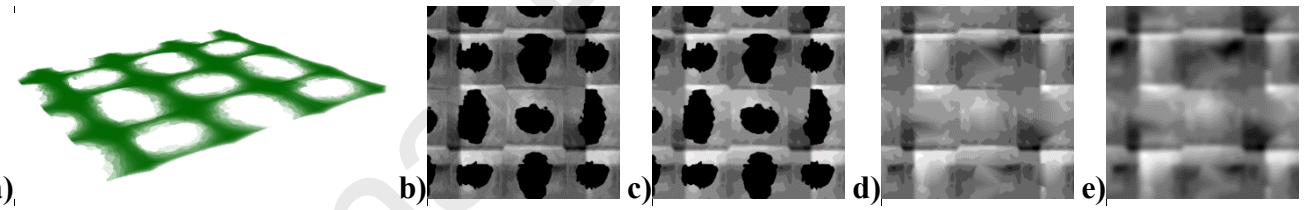


Figure 4: a) Single matrix layer (one of 7); b) medial surface image; c) median filter with the masked non-zero region; d) linear interpolation of holes; e) Gaussian filter

3.4. Ply segmentation

The medial surfaces of the matrix layers were then used to label plies with the following algorithm:

Input: tows mask (for fill or warp direction), matrix layer interpolations

Output: ply segmentation image (in the given direction)

1. *Initial ply segmentation;*
2. *Ply eroding and small objects removing;*
3. *Ply labeling;*
4. *Euclidian distance transform (EDT) for the region outside eroded plies;*

5. Segmentation by watershed for EDT image using the ply labels as seeds.

The algorithm was applied independently for fill and warp tow directions. Initially (step 1), the voxels were labeled according to their out-of-plane coordinates, i.e. the position between the matrix layer surfaces (Figure 5b). It resulted in a ply segmentation that could contain many incorrectly labeled voxels at the ply boundaries, because of the inaccuracy of the input semantic segmentation. An example of such incorrect labeling is highlighted by a yellow rectangle in Figure 5b and emphasized in Figure 5d (top) against the same region of resulting segmentation (Figure 5d bottom).

To improve this coarse ply labeling, steps 2-5 were applied. The ply regions were morphologically eroded and opened with a cubic structuring element (size 3x3x3 voxels). Then, all regions whose volume was larger than 10000 voxels were labeled by a value from 1 to 8 (according to the initial ply segmentation labels). This volume threshold should be adjusted to the input image, depending on the initial semantic segmentation quality. Then, the EDT was performed for the region outside of the ply labels. To properly recover the shape of the eroded tows with the watershed transform, the distance transform with an anisotropic sampling of [10, 1, 1] was used. Finally, the labeling result was used as the seeding for the watershed transform over the distance transform image, masked by the input tow regions. The result of the watershed splitting is shown in Figure 5c.

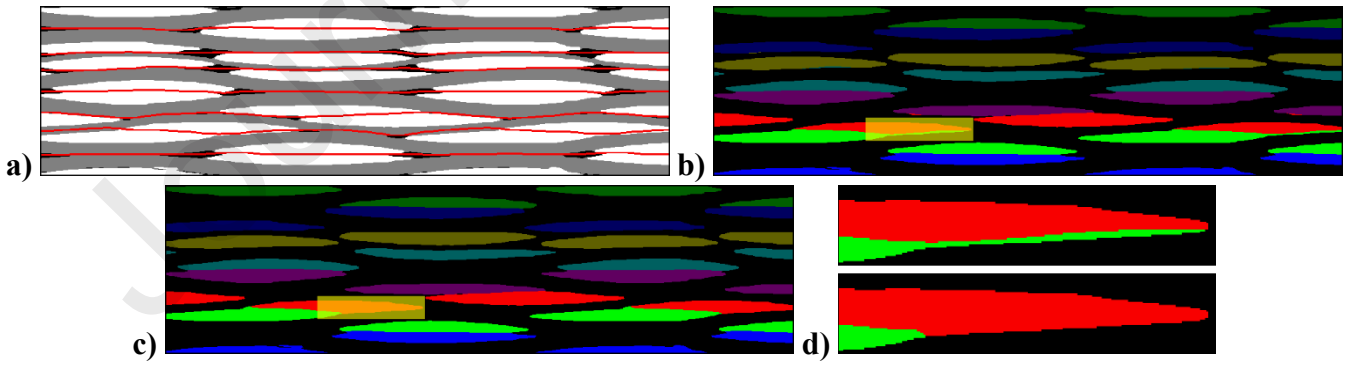


Figure 5: a) Interpolation of matrix layer medial surfaces (red curves); b) initial ply labels (an example of incorrect splitting is highlighted by the yellow rectangle); c) resulting ply labels; c) detail of the incorrectly split region: initial (top) and resulting (bottom) segmentations

3.5. Tow segmentation

The ply segmentation was not always able to differentiate touching tows within a same ply (Figure 6a). However, splitting these tows was straightforward, because their touching area was relatively small. The tow segmentation algorithm was similar to the ply segmentation described earlier. It was also performed independently for the fill and warp directions. Each ply label underwent the following steps:

Input: ply mask

Output: ply with labeled tows

1. *Group image slices by splitting tows in the thinnest area;*
2. *Morphological eroding;*
3. *For each image sub-stack remove small objects and assign tow label;*
4. *Splitting of the input object by watershed segmentation.*

At the first step, the slices of the ply image were divided into sub-stacks, providing the initial split of the tows in the thinnest area. The thinnest area slices were found as slices that have a locally minimal count of the material's voxels (local minima of volume fraction). Morphological eroding was performed iteratively (5 times) with a cubic structuring element (3x3x3 voxels). For each image sub-stack, all objects smaller than 10000 voxels were removed and a unique tow label was assigned for the remaining voxels, giving the seeds for the watershed transform. The watershed was computed on the negative chamfer distance transform of the input object region. The tow splitting result is shown in Figure 6b.

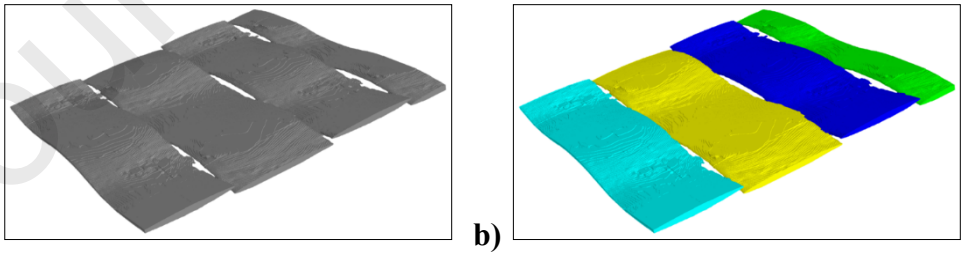


Figure 6: a) touching tows within a ply; b) tow splitting result

3.6. Post-processing

If the input was periodically extended at the first step, the fill and warp tow segmentation images were cropped back. Then, regions smaller than 25000 voxels were removed by watershed-filling their voxels with the nearest connected tow's label value, using corresponding fill or warp tows mask.

To compute the segmentation accuracy, the tow labels needed to be coherent with the "ground-truth". For this purpose, we iterated over all labels of the automatic segmentation, in decreasing volume order. For one given label, a mask was extracted from the segmentation result, and applied to the ground-truth. The most frequent ground-truth value within that mask could then be used to coherently label the segmentation output. Eventually, the algorithm equalized label numbers, so that each label of the segmentation output equaled the corresponding label of the "ground-truth", or equaled a new value which did not exist in the "ground-truth" image (in case no corresponding tow region was found). After equalizing the labels, the fill and warp tow images (results of tow segmentation step) were joined. The resulting tow splitting is presented in Figure 12a.

4. Tow splitting with deep learning

4.1. Methodology overview

This deep learning approach was inspired by the work [27], where the 2D U-net architecture was used for cell instance segmentation of microscopy images. The main idea of the method was to train a CNN to predict an input necessary for the watershed transform (EDT and watershed seeds). However unlike work [27], where the neighbor distance prediction was used to obtain the seeds, we extracted the seed based on the prediction of the tow centerlines. The deep learning approach proposed in [30] was modified in this study to extract the tow centerlines.

The variety of the training dataset is key to provide reliable CNN predictions, without the network overfitting on one type of example. However, acquiring a rich dataset can be a problem if many 3D X-ray images are needed, because of the lack of material sample variety, µCT scanning cost, and the image annotation efforts. Therefore, we instead used a set of random synthetic images to train the CNN. For this purpose, the parametrized geometry model of the repetitive pattern of the plain weave architecture was used.

This segmentation approach could be split into three parts: the generation of the training dataset, training the CNN, and applying the watershed transform from the CNN predictions. Note that the first and second parts would only need to be done once, after which the pre-trained CNN could be utilized multiple

times for the segmentation of similar images of the same material. In the next subsections, these three steps are detailed.

4.2. Idealized geometry model

To describe the repetitive pattern geometry (Figure 7a), the following parameters were used: material period, tow width, tow height at the crimp section, and tow height at the non-crimp section. Assuming that these characteristics could differ for fill and warp directions, we counted a total of 8 parameters. The average values of these parameters could easily be extracted from the material specification, or manually from the scan of the material. The pattern size was equal to a quarter of the in-plane period, and the height was the sum of fill and warp tow heights in the non-crimp section. To increase computational efficiency, the corresponding repetitive pattern was scaled in-plane with the coefficient $f_s = \frac{v_s}{64\mu m} = 0.078125$, where $v_s = 5\mu m$ is the scan's voxel size and the coefficient 64 was chosen empirically to suite well for different resolutions. The scaled material periods were computed as the original period multiplied by the scaling coefficient, plus a random shift of few voxels size: $p_f = P_f f_s + d_f$, $p_w = P_w f_s + d_w$, where d_f and $d_w \in [-5,\ 5]$. The uniform distribution was used throughout this work for the generation of random numbers. Each of the parameters for the geometry description is given in voxels, with decimal precision. Tow widths were computed as $W_f = 0.5 p_w w_f$ and $W_w = 0.5 p_f w_w$, where w_f is a random value in the range $[0.8, 1]$, w_w taken in $[1.8 - w_f, 1.9 - w_f]$, and $w_w = 1$ for all $w_w > 1$. The random pattern height $S_h \in [30, 36]$ was chosen empirically, based on the desired height of the stack sample and the plies nesting depth. The in-plane pattern widths were $S_f = 0.25 p_f$ and $S_w = 0.25 p_w$. The non-crimp tow heights were $h_f \in [0.45 S_h,\ 0.55 S_h]$ and $h_w = S_h - h_f$. The corresponding random crimp heights were bounded: $H_f \in [h_f,\ 1.25 h_f]$ and $H_w \in [h_w,\ 1.25 h_w]$. When all the required parameters were defined, the local coordinate system (green axes in Figure 7a) was chosen. The coordinates of the tow surface corners were then found as:

$F_1 = [h_w, S_f, 0]$; $F_2 = [F_2^0, S_f, 0.5 W_f]$; $F_3 = [S_h, S_f, 0]$; $F_4 = [0.5(S_h - H_f), 0, 0]$;

$F_5 = [0.5S_h, 0, 0.5W_f]$; $F_6 = [0.5(S_h + H_f), 0, 0]$;$F_7 = [F_7^0, S_f - 0.5W_w, 0]$; $F_8 = [0, S_f, 0]$;

$F_9 = [0.5(S_h + H_w), S_f, S_w]$; $F_{10} = [0.5S_h, S_f - 0.5W_w, S_w]$; $F_{11} = [0.5(S_h - H_w), S_f, S_w]$;

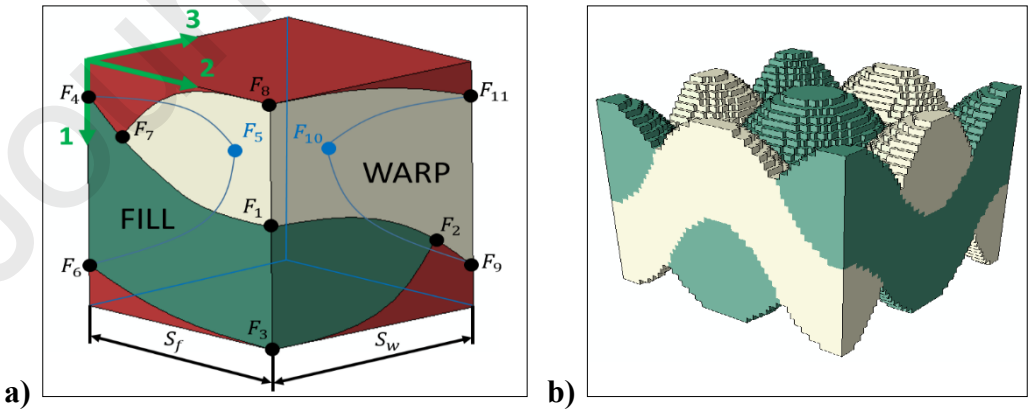
$F_2^0 = \frac{h_f - h_w + H_w}{2} {w_f}^2 + h_w$; $F_7^0 = \frac{h_f - h_w - H_f}{2} {w_w}^2 + h_w$,

where F_2^0 and F_7^0 were derived from the second-order interpolation edges F_1F_9 and F_1F_4 correspondently with condition zero derivatives in point F_1. The corner points are marked by a small black circle in Figure 7a.

Based on the points F_1,...,F_{11} the surface edges were interpolated by a second-order polynomial curve, under the assumption that each edge had zero derivatives in one of the endpoints, which is a natural condition due to the model symmetry. The model's 14 edges were thus found. For example, the edges that connect points F_1 and F_4, or points F_1 and F_9 could be expressed as:

$$F_1F_4(t) = \begin{bmatrix} \frac{h_f - h_w - H_f}{2} t^2 + h_w \\ S_f(1-t) \\ 0 \end{bmatrix}; F_1F_9(t) = \begin{bmatrix} \frac{h_f - h_w + H_w}{2} t^2 + h_w \\ S_f \\ S_w t \end{bmatrix}, t \in [0,1].$$

Finally, the tow surfaces were reconstructed by Coon's patch bilinear interpolation [32] for four edges input. Two surfaces for fill, and two for warp tows, were created. The resulting geometry was voxelized to obtain a synthetic image of the single-ply unit cell (Figure 7b). The unit cells assembling gave us the synthetic ply image with a periodic structure.



a) b)

Figure 7: a) Repetitive pattern model (backside edges and points are in blue); b) Unit cell voxelization result

4.3. Generation of the training dataset input

The unit cell model was used for the generation of 200 random full-stack samples (in our case 8 layers), using model parameters extracted from a downsampled semantic segmentation of the X-ray CT image. We propose that this dataset could be generated with the following algorithm:

Input: geometry model parameters

Output: a random synthetic image of the material

1. *Generate random ply;*
2. *Generate random stack of 8 plies;*
3. *Add random elastic deformation;*
4. *Remove small regions;*
5. *Add median filter smoothing (here applied only to every 2nd sample);*
6. *Binarize image with artificial matrix layers between fill and warp tows;*
7. *Add random "salt" noise (0.4% of the voxels set to zero).*

The downscaled size of the synthetic sample was [200, 64, 64] voxels. This sample size determines the computational cost of training the network, so it was chosen as small as possible without compromising the tow shapes. The downscaled unit cell size [64, 64] was equal to the rounded scaled period $[P_f f_s,\ P_w f_s]$. Due to the coefficient-based scaling, the input image was not constrained in terms of the resolution or in-plane cropping size. The height was resized (from 357 to 200 voxels) without using a scaling coefficient, because only an 8-ply input was expected.

The ply position in the stack was determined by two in-plane shifts and an out-of-plane (height) shift. The initial out-of-plane positions of plies were calculated to fill the stack height (200 voxels) with equal thicknesses of the plies overlapping (nesting) regions. The height shift for the top and bottom plies was zero, while for the internal plies it was a random value in [-3, 3] voxels. The in-plane ply shift was a vector of two components, which were chosen randomly in the ranges $[0, p_f]$ and $[0, p_w]$. The plies were joined into a single stack image according to the shift values, followed by the removal of tow intersections. A typical random stack image is presented in Figure 8a-b. To increase the variability of the samples, we applied a random elastic deformation of the images using the library [33]. Then, all small isolated regions

(less than 64 voxels) were removed. To decrease count of tows with sharp distortion due to the random deformation, the median filter (kernel size [3,3,3]) was applied to every second sample. The filter was used independently for fill and warp tow images, and followed by joining both direction images while removing the possible new tow intersections (see the result in Figure 8c). It was possible to equalize both (fill and warp) tow directions during the network training, by converting the stack images to binary (matrix/tows). To do that without losing the distinction between fill and warp tows, we added an (artificial) thin matrix layer to the region of touching tows. Finally, a random "pepper" (matrix) noise was added by making 0.4% of the voxels zero (Figure 8d). The noise can decrease sensitivity of the network for a noisy input. Practically, an input often can have false small matrix regions which are difficult to remove because they are connected with the main true matrix. The resulting training source sample, with random geometrical characteristics, is presented in Figure 8d-e.

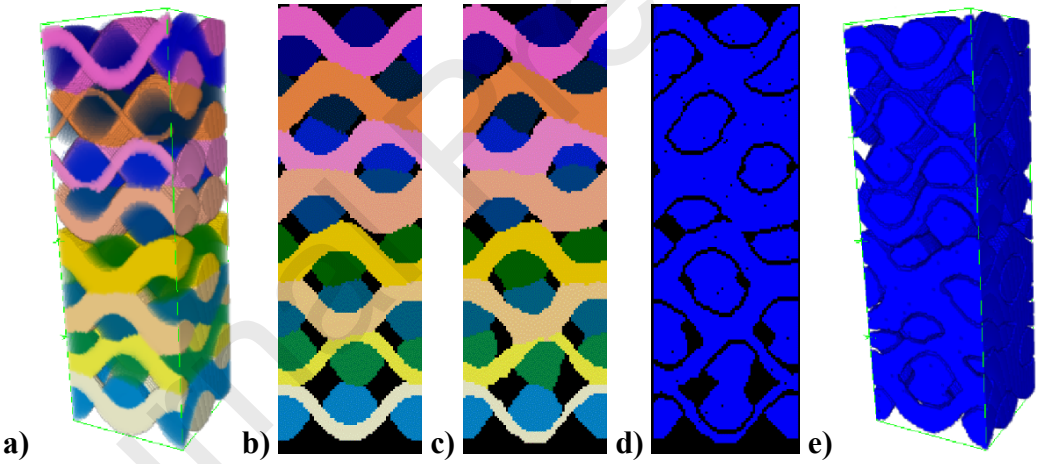


Figure 8: Initial synthetic stack (a) 3D view (b) 2D view; c) elastic random deformation and cleaning; final training source sample (d) 2D view (e) 3D view

4.4. Generation of the training dataset output

To split tows with the watershed transform, an image was required, that summed up the distance transform for each tow ("tow distances" image in Figure 9a). The watershed transform of this image with tow centerlines seeding allowed to split tows. Tows centerlines were extracted from the "centerline distances" images (Figure 9b), that is the sum of the modified distance transform for each tow centerline.

The works [27, 30] indeed showed that deep learning could successfully predict such distance maps. To generate them for each randomly generated synthetic input, we constructed the following algorithm:

Input: tow labels image, the matrix mask (step 5 and 6 results of the source image generation algorithm)

Output: "tow distances" and "centerline distances" images.

1. Initialize output images as zero values arrays;
2. For each tow's mask of the input labels image, do the steps 3-7;
    3. Compute Euclidean distance transform (EDT) for the tow's mask and add the result to the "tow distances" output image;
    4. Approximate start and end points of the tow's centerline;
    5. Find the centerline as the shortest path for the tow's EDT costs function;
    6. Compute EDT to the tow's centerline within the tow's mask;
    7. Logarithmic correction on the EDT centerline image, and addition of to the "centerlines distances" output;
8. Apply the matrix mask to the output images.

The tow centerline approximation was found as the shortest path between the two tow ends, with a cost defined by the "centerline distances" image. For this purpose, Dijkstra's algorithm for an image input (implemented in the Scikit-image Python library [34]) was used. The algorithm required a cost array, and start and end points as input. The cost array was computed as the negative EDT of a tow, scaled in the range [0, 1]. To guarantee that the optimal path did not grow out of the tow, out-of-tow voxels were penalized by a large cost (e.g. 255). To increase accuracy at the beginning and end of the tow, we used image padding (10 voxels width) for the tow axial direction (the direction coincides with one of the image axes). For the input tow mask, the start and end points were found on the first and last image slices, counted orthogonally to the tow axial direction. More precisely, the points were computed as central voxels of the corresponding tow sections.

For each centerline, the EDT with logarithmic correction was computed within the tow's mask (steps 6-7 of the algorithm). The logarithmic correction made the distance transform steeper at the centerline. It was computed as $-\log_2(1+I)$ where I is the corresponding image array normalized between 0 and 1. Eventually, all individual centerline distance transforms were normalized between 0 and 1 (within the tow's mask) and combined into a single image. Finally, the matrix region was set to zero in the output images, using the result of the input image generation algorithm's step 6 (section 4.3). The output images, corresponding to the input presented in Figure 8c, are presented in Figure 9.

Generating a training dataset of 200 samples (input and output sets of images) took about 95 minutes on a computer with an Intel Core i7 1.9 GHz CPU and 32 GB RAM.

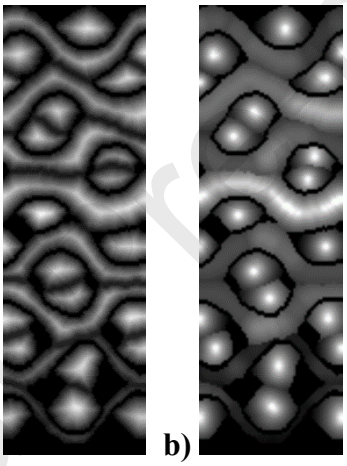


Figure 9: Target training images (matrix region in black): (a) "tow distances" (grayscale is proportional to distance to the tow's boundary), (b) "centerline distances" (grayscale is proportional to negative logarithmic distance to the tow's centerline)

4.5. CNN configuration and training

The U-Net architecture [29] was adapted to predict "tow distances" and "centerline distances" from a binarized tows input. Similarly to works [27, 30] the network had two decoder paths that correspond to two target image prediction tasks. The encoder and decoder paths had 3 blocks. The blocks contained a 3D convolution (deconvolution) with a rectified linear unit activation function and a max pooling. The expanding block concatenated the deconvolution layer with the corresponding max pooling result of the encoder path. The initial number of output filters in the convolution was 64. The output convolution layer had a linear activation function for both decoders.

The neural network was implemented in TensorFlow/Keras [35]. An Adam optimizer with a learning rate of 0.0003 was used to train the network. The other optimizer parameters were the default TensorFlow values. The logarithm of the hyperbolic cosine was used as the loss function to compute the prediction error. Because the network depth was 3, it was expected that the input image size was a multiple of 8 in each of the three dimensions (in our case, [200, 64, 64]). The verification dataset was limited to only one sample, generated from the manual segmentation ("ground-truth") image. It was the only real-image semantic segmentation that we had in pair with the tow instance segmentation. The network was trained for 200 epochs. This count of epochs was selected empirically based on the convergence of the loss function and the prediction accuracy to the manual segmentation image, which forms the test dataset. Similar to the U-net training in [11], the batch size was one, which is natural for the large size of a training dataset sample. The training was performed on UGent's High Performance Computing center's GPU cluster, using a single NVIDIA Volta V100 GPU (32GB memory) node. The total training time took about 57 minutes.

4.6. Tow instance segmentation

The watershed-based tow splitting algorithm for the pre-trained neural network can be summarized in the following steps:

Input: semantic image segmentation, pre-trained CNN

Output: tow instance segmentation

1. *Periodical image extension (identical to Subsection 3.1);*
2. *Image downscaling;*
3. *Prediction "tow distances" and "centerline distances";*
4. *Detect start and end points of the tow centerlines;*
5. *Extraction tow centerlines;*
6. *Splitting tows by seed-based watershed segmentation;*
7. *Image upscaling;*
8. *Post-processing steps (identical to Subsection 3.6).*

Firstly, the input semantic segmentation image (Figure 1b) was periodically extended as described in Subsection 3.1. This optional step improved the segmentation accuracy at the image boundaries. The extended image size was [357, 1433, 1434] voxels. Then, the extended image was downscaled to [200, 112, 112] voxels (see Figure 10a-b). The in-plane size was computed again with the scaling coefficient $f_s = 0.078125$, but with the preserving condition that the downscaled image's in each direction would be a multiple of 8. The stack height of 200 voxels was a fixed value and identical to the training dataset sample height. At the 3rd step of the algorithm, the pre-trained neural network model predicted "tow distances" and "centerline distances" for the downscaled input. The prediction results are presented in Figure 10c-d. At step 4, the start and end points of a tow's centerline were detected independently for fill and warp directions. Firstly, the tow central regions were obtained as the union of the thresholded "centerline distances" slices along the tow's direction (axis X in Figure 11a), followed by the removal of regions smaller than 10 voxels. Figure 11a-b shows the thresholded image (the threshold value was 0.55) and the tow central regions. Then, the tow's end was found as the point corresponding to the maximum value of the "centerline distances" computed within the corresponding tow's central region on the first or last slice of the image (with respect to the corresponding tow's axial direction).

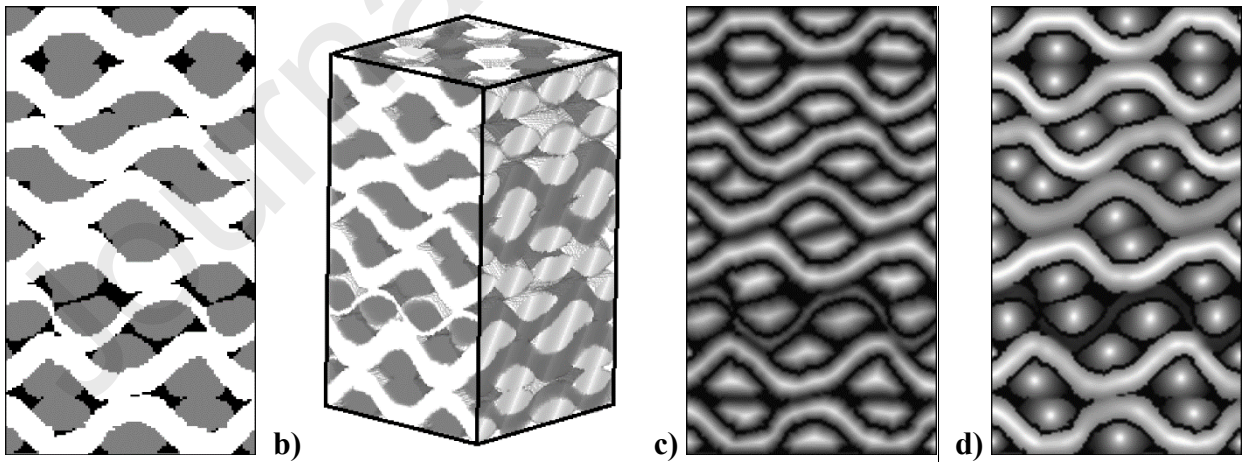


Figure 10: Downscaled segmentation (periodically extended image) in (a) 2D and (b) 3D view); c) "tow distances" (grayscale is proportional to distance to the boundary); d) "centerline distances" (grayscale is proportional to negative logarithmic distance to centerline)

The tow centerlines were calculated as the minimal cost path for the distance transform cost function using Dijkstra's algorithm, as in the generation of the training dataset (Subsection 4.4). The resulting

centerlines for one of the tow directions are presented in Figure 11c. For robustness, the watershed seeds (see Figure 11d) were computed as the union of the thresholded image and the centerlines image. Note that using only the thresholded image for seeding can lead to locally inaccurate watershed segmentation, as the thresholding cannot guarantee an unbroken path between the two tow ends. However, the unbroken tow's seeding path is important for the watershed segmentation of tube-shape structures. The result of the watershed segmentation of the "tow distances" with centerlines-based seeding is presented in Figure 11e.

The watershed segmentation's output was upscaled to the initial image resolution, and cropped to the size of the input semantic segmentation. The upscaled instance segmentation did not fit the original semantic segmentation, because of the interface pixelation. That was corrected by assigning zero to voxels outside of the original tows region and assigning the nearest tow label to zero voxels inside the region.

The final post-processing step has been described in Section 3.6. It includes cropping the periodical extension back, removing small tow pieces at the boundaries, joining fill and warp label images, and, finally, relabeling the segmentation result. The obtained segmentation is presented in Figure 12b.

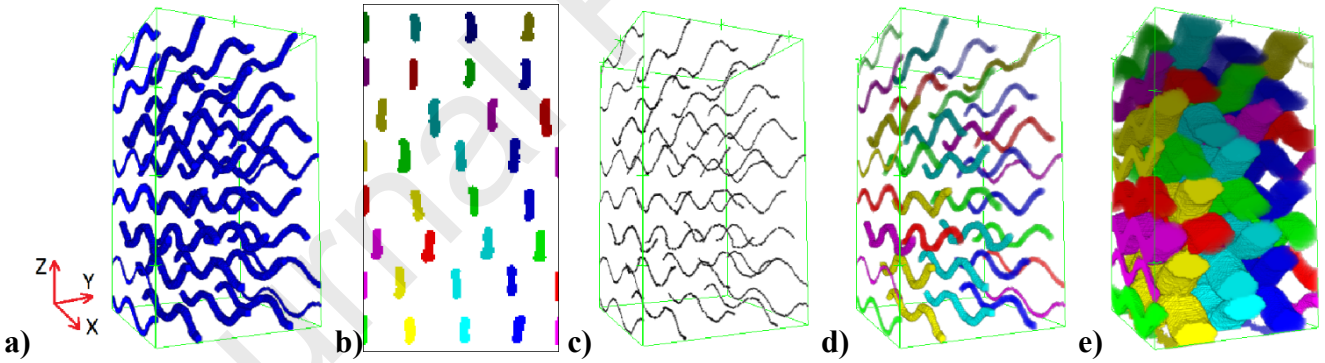


Figure 11: Watershed seeding and segmentation for one of the two tow directions: a) thresholded "centerline distances"; b) tow central regions (projection of image (a) on YZ plane); c) tow centerlines; d) watershed seeds; e) watershed segmentation

5. Results and discussion

The instance segmentation methods described above were applied to the five semantic segmentations [14] of the same µCT scan, encompassing 8-layers of textile composite presented in Figure 1a. These 3D input images consisted in one variational approach segmentation result, three different deep learning segmentations (referred to as "CNN_1", "CNN_2", "CNN_3"), plus one manual segmentation, in which all collinear tows had the same label. The manual segmentation was used as an ideal input to the tow

splitting. The visual representation of these input segmentations can also be found in the work [14]. Table 1 recalls that all automatic segmentations were very close to the ground-truth manual segmentation; The semantic segmentation accuracy is important as it feeds through to the accuracy of the tow splitting. Note that the same ground-truth image was used for the evaluation of both semantic and instance segmentation accuracies.

The error introduced due to instance segmentation is presented in Table 1 as the difference between the semantic and instance segmentation accuracies, where the accuracy was calculated as the percentage of correctly labeled voxels. Also, in Table 2 the segmentations were evaluated by adjusted Rand index (ARI), which is used as measure of two data clusterings similarity [36]. Both geometrical and deep learning methods yielded less than 0.3% error (percentage of wrongly segmented voxels) from this ideal input. However, even from non-ideal input, the maximal instance segmentation error was 1.12% for the geometrical approach, and 0.81% for the deep learning approach. For the most accurate automatic semantic segmentation ("CNN_3" image), both instance segmentation approaches preserved the high initial (semantic segmentation) accuracy. Figure 12d highlights the wrongly segmented voxels for the "CNN_3" input semantic segmentation (cyan color) and the corresponding result of the deep learning tow instance segmentation (magenta color). Remark that this particularly precise semantic segmentation was already used in Sections 3 and 4 for the description of the methods. Naturally, both instance segmentation methods were more accurate with more accurate input. For instance, in the case of the "CNN_3" image input, both approaches performed almost equally well. However, even in the case of the least accurate input (the "CNN_1" image), both methods reached low accuracy changes (~1%). Adjusted Rand index behaves similarly for both methods, only for the "CNN_1" and "CNN_2" tests, the deep learning result is slightly better. The index also decreases less for the higher quality of the input segmentation and shows almost perfect splitting for the manual segmentation input.

We can assume that deep learning was more tolerant to a noisy and inaccurate input than the geometrical method was. Theoretically, this could be improved further by expanding the training dataset, and fine-tuning the neural network architecture and learning parameters. Besides, the deep learning

approach can be generalized to other tow architectures and ply counts, while the geometrical approach remains strongly case-specific. Expectedly, a generalization of deep learning to segment raw X-ray µCT images (i.e. hiding the semantic segmentation step) could also be possible. Then, a synthetic training dataset that fully models the texture of real X-ray µCT images would be needed. Contrarily, the geometrical approach would always require a clean semantic segmentation input, which is not always available.

The running times of the instance segmentation algorithms (implemented in Python) on a computer with Intel Core i7 1.9GHz CPU and 32 GB RAM, was about 53 minutes for the geometrical approach, and about 14 minutes for the deep learning approach. Note that the training dataset generation and neural network training are not counted in the deep learning time.

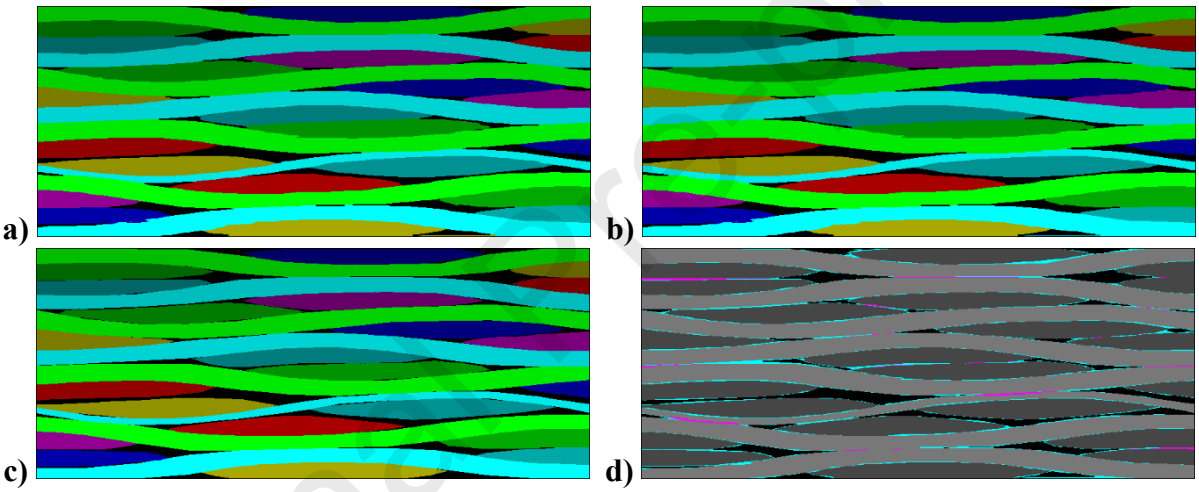


Figure 12: Tow splitting results using (a) geometrical approach and (b) deep learning approach; c) manual segmentation; d) incorrectly labeled voxels of deep learning segmentation (tow splitting error in magenta and semantic segmentation error in cyan colors)

Table 1: Instance segmentation accuracy for 5 test images

	Variational	**CNN_1**	**CNN_2**	**CNN_3**	**Manual**
	Initial (semantic) segmentation accuracy [%]				
	92.65	92.49	93.21	95.27	100
	Difference between semantic and instance segmentations accuracies [%]				
Matrix interpolation	0.89	1.12	0.32	0.33	0.30
Deep learning	0.52	0.81	0.24	0.31	0.21

Table 2: Adjusted Rand index [36] for 5 test images

	Variational	**CNN_1**	**CNN_2**	**CNN_3**	**Manual**
	Initial (semantic) segmentation				
	0.82	0.82	0.82	0.88	1
	Instance segmentations				
Matrix interpolation	0.72	0.72	0.80	0.84	1.00

Deep learning	0.73	0.73	0.80	0.84	1.00

6. Conclusions

Two instance segmentation approaches have been proposed to solve the challenging problem of constructing realistic μCT image-based models of textile laminate composites. One method separates tows based on morphological analysis of matrix layers, while the other uses deep learning to predict the seeds for a watershed transform. The input of both methods was a semantic segmentation, and the output was an image where each tow was independently labeled. To implement these two methods, several auxiliary algorithms were also proposed (e.g. the generation of random synthetic material images, finding the tow centerlines, interpolating matrix layers, etc.).

Both proposed methods demonstrate high accuracy, from different semantic segmentation and "ideal" (manually-annotated) inputs. In the latter case, the instance segmentation error was less than 0.3%. As is to be expected, the accuracy of instance segmentation methods largely depends on the quality of the input segmentation. The tow splitting result can be directly used for image-based finite element modeling with realistic material orientation field, or analyzing the tows' geometrical characteristics, before the generation of an idealized model of the material structure.

The deep learning tow splitting approach outperforms the method based on matrix layers interpolation in terms of accuracy and robustness, i.e. sensitivity to the quality of input data and sensitivity to adjustments of the method parameters. On the other hand, the deep learning method requires more powerful computing infrastructures. However, while the training of the proposed Neural Network is computationally expensive, the trained network can be used for the segmentation of other images with the same tow architecture. Furthermore, it can also be recycled in transfer learning, drastically reducing the training time of future networks dedicated to similar tasks.

In principle, the deep learning approach could alternatively be built to accept raw (instead of segmented) μCT images. However in that endeavor, the generation of synthetic texture material images (to form the training dataset) needs to be developed further.

Acknowledgement

This work was funded by the Research Foundation — Flanders in the Strategic Basic Research Programme (FWO-SBO), file number S003418N. The Special Research Fund of Ghent University is acknowledged for the financial support for M.N.B under project number BOF17-GOA-015.

Data availability

The raw/processed data required to reproduce these findings cannot be shared at this time as the data also forms part of an ongoing study.

CRediT authorship contribution statement

Yuriy Sinchuk: Methodology, Software, Formal analysis, Validation, Visualization, Investigation, Data curation, Writing - original draft, Writing - review & editing. **Pierre Kibleur**: Methodology, Writing - review & editing. **Jan Aelterman**: Project administration, Writing - review & editing. **Matthieu N. Boone**: Resources, Writing - review & editing. **Wim Van Paepegem**: Supervision, Funding acquisition, Conceptualization, Resources, Writing - review & editing.

References

[1] Ansar M, Xinwei W, Chouwei Z. Modeling strategies of 3D woven composites: a review. Composite Structures. 2011;93:1947-63.

[2] Isart N, El Said B, Ivanov D, Hallett S, Mayugo J, Blanco N. Internal geometric modelling of 3D woven composites: A comparison between different approaches. Composite Structures. 2015;132:1219-30.

[3] Sherburn M. Geometric and mechanical modelling of textiles: University of Nottingham, 2007.

[4] Verpoest I, Lomov SV. Virtual textile composites software WiseTex: Integration with micro-mechanical, permeability and structural analysis. Composites Science and Technology. 2005;65:2563-74.

[5] Sevenois R, Garoz D, Gilabert F, Spronk S, Fonteyn S, Heyndrickx M, et al. Avoiding interpenetrations and the importance of nesting in analytic geometry construction for representative unit cells of woven composite laminates. Composites Science and Technology. 2016;136:119-32.

[6] Doitrand A, Fagiano C, Chiaruttini V, Leroy F, Mavel A, Hirsekorn M. Experimental characterization and numerical modeling of damage at the mesoscopic scale of woven polymer matrix composites under quasi-static tensile loading. Composites Science and Technology. 2015;119:1-11.

[7] Potluri P, Sagar T. Compaction modelling of textile preforms for composite structures. Composite Structures. 2008;86:177-85.

[8] Stig F, Hallström S. Spatial modelling of 3D-woven textiles. Composite Structures. 2012;94:1495-502.

[9] Mahadik Y, Hallett S. Finite element modelling of tow geometry in 3D woven fabrics. Composites Part A: Applied Science and Manufacturing. 2010;41:1192-200.

[10] Huang W, Causse P, Brailovski V, Hu H, Trochu F. Reconstruction of mesostructural material twin models of engineering textiles based on micro-ct aided geometric modeling. Composites Part A: Applied Science and Manufacturing. 2019;124:105481.
[11] Sinchuk Y, Kibleur P, Aelterman J, Boone MN, Van Paepegem W. Variational and Deep Learning Segmentation of Very-Low-Contrast X-ray Computed Tomography Images of Carbon/Epoxy Woven Composites. Materials. 2020;13:936.
[12] Liu Y, Straumit I, Vasiukov D, Lomov SV, Panier S. Prediction of linear and non-linear behavior of 3D woven composite using mesoscopic voxel models reconstructed from X-ray micro-tomography. Composite Structures. 2017;179:568-79.
[13] Naouar N, Vidal-Sallé E, Schneider J, Maire E, Boisse P. Meso-scale FE analyses of textile composite reinforcement deformation based on X-ray computed tomography. Composite Structures. 2014;116:165-76.
[14] Wintiba B, Vasiukov D, Panier S, Lomov SV, Kamel KEM, Massart TJ. Automated reconstruction and conformal discretization of 3D woven composite CT scans with local fiber volume fraction control. Composite Structures. 2020:112438.
[15] Naouar N, Vidal-Salle E, Schneider J, Maire E, Boisse P. 3D composite reinforcement meso FE analyses based on X-ray computed tomography. Composite Structures. 2015;132:1094-104.
[16] User's guide avizo software. 2019.
[17] Auenhammer RM, Mikkelsen LP, Asp LE, Blinzler BJ. Automated X-ray computer tomography segmentation method for finite element analysis of non-crimp fabric reinforced composites. Composite Structures. 2020;256:113136.
[18] Vanaerschot A, Panerai F, Cassell A, Lomov SV, Vandepitte D, Mansour NN. Stochastic characterisation methodology for 3-D textiles based on micro-tomography. Composite Structures. 2017;173:44-52.
[19] Ali MA, Guan Q, Umer R, Cantwell WJ, Zhang T. Deep learning based semantic segmentation of μCT images for creating digital material twins of fibrous reinforcements. Composites Part A: Applied Science and Manufacturing. 2020;139:106131.
[20] Badran A, Marshall D, Legault Z, Makovetsky R, Provencher B, Piché N, et al. Automated segmentation of computed tomography images of fiber-reinforced composites by deep learning. Journal of Materials Science. 2020;55:16273-89.
[21] Wijaya W, Ali M, Umer R, Khan K, Kelly P, Bickerton S. An automatic methodology to CT-scans of 2D woven textile fabrics to structured finite element and voxel meshes. Composites Part A: Applied Science and Manufacturing. 2019;125:105561.

[22] Wijaya W, Kelly P, Bickerton S. A novel methodology to construct periodic multi-layer 2D woven unit cells with random nesting configurations directly from μCT-scans. Composites Science and Technology. 2020:108125.
[23] Kass M, Witkin A, Terzopoulos D. Snakes: Active contour models. International journal of computer vision. 1988;1:321-31.
[24] Bénézech J, Couégnat G. Variational segmentation of textile composite preforms from X-ray computed tomography. Composite Structures. 2019;230:111496.
[25] He K, Gkioxari G, Dollár P, Girshick R. Mask r-cnn. Proceedings of the IEEE international conference on computer vision2017. p. 2961-9.
[26] Bai M, Urtasun R. Deep watershed transform for instance segmentation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition2017. p. 5221-9.
[27] Scherr T, Löffler K, Böhland M, Mikut R. Cell segmentation and tracking using CNN-based distance predictions and a graph-based matching strategy. arXiv preprint arXiv:200401486. 2020.
[28] Wang W, Taft DA, Chen Y-J, Zhang J, Wallace CT, Xu M, et al. Learn to segment single cells with deep distance estimator and deep cell detector. Computers in biology and medicine. 2019;108:133-41.
[29] Çiçek Ö, Abdulkadir A, Lienkamp SS, Brox T, Ronneberger O. 3D U-Net: learning dense volumetric segmentation from sparse annotation. International conference on medical image computing and computer-assisted intervention: Springer; 2016. p. 424-32.
[30] Guo Z, Bai J, Lu Y, Wang X, Cao K, Song Q, et al. Deepcenterline: A multi-task fully convolutional network for centerline extraction. International Conference on Information Processing in Medical Imaging: Springer; 2019. p. 441-53.
[31] Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, et al. SciPy 1.0: fundamental algorithms for scientific computing in Python. Nature methods. 2020;17:261-72.
[32] Coons SA. Surfaces for computer-aided design of space forms. MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC; 1967.
[33] https://pypi.org/project/elasticdeform/.
[34] Van der Walt S, Schönberger JL, Nunez-Iglesias J, Boulogne F, Warner JD, Yager N, et al. scikit-image: image processing in Python. PeerJ. 2014;2:e453.
[35] Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. Tensorflow: A system for large-scale machine learning. 12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)2016. p. 265-83.
[36] Hubert L, Arabie P. Comparing partitions. Journal of classification. 1985;2:193-218.

Declaration of interests

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: