Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor

Paper

Jeppesen, N.; Dahl, V. A.; Christensen, A. N.; Dahl, A. B.; Mikkelsen, L. P.

Published in:
IOP Conference Series: Materials Science and Engineering

Link to article, DOI:
10.1088/1757-899X/942/1/012037

Publication date:
2020

Document Version
Publisher's PDF, also known as Version of record

Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor

IOP Conf. Series: Materials Science and Engineering **942** (2020) 012037 doi:10.1088/1757-899X/942/1/012037

Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor

N Jeppesen[1,3], V A Dahl[1], A N Christensen[1], A B Dahl[1] and L P Mikkelsen[2]

[1]Department of Applied Mathematics and Computer Science, Technical University of Denmark, Lyngby, Denmark
[2]Department of Wind Energy, Technical University of Denmark, Roskilde, Denmark

E-mail: [3]niejep@dtu.dk

Abstract. The mechanical properties of composite fiber materials are highly dependent on the orientation of the fibers. Micro-CT enables acquisition of high-resolution 3D images, where individual fibers are visible. However, manually extracting orientation information from the samples is impractical due to the size of the 3D images. In this paper, we use a Structure Tensor to extract orientation information from a large 3D image of non-crimp glass fiber fabric. We go through the process of segmenting the image and extracting the orientation distribution step-by-step using structure tensor and show the results of the analysis of the studied non-crimp fabric. The Jupyter notebooks and Python code used for the data-analysis are publicly available, detailing the process and allowing the reader to use the method on their own data. The results show that structure tensor analysis works well for determining fiber orientations, which has many useful applications.

1. Introduction

Fiber reinforced polymer matrix composites are used in many structural applications due to their excellent stiffness, strength, and fatigue properties and relatively low weight. Nevertheless, these properties are very sensitive to the fiber orientations, and therefore characterizing fiber orientations is an important part of quality control. However, the microscopic nature of the fibers makes it difficult to bridge the gap between the fiber/matrix scale on the micro-meter scale to the structural component on the meter scale and thereby examine representative samples, while maintaining adequate resolution. With modern micro-CT scanners, it is possible to capture and stitch together high-resolution data for larger samples, which results in large volumetric data-sets.

To extract orientation information about the fibers, one approach is to track every fiber individually [1]. This can provide detailed information about the fibers, but often requires some user interaction and can be very computationally demanding, making it less attractive for large data-sets. In addition, it is often not the location of the individual fibers but just the fiber orientation in each material point which is of relevance. An approach addressing this is the voxel-based Structure Tensor method [2, 3, 4]. This method is relatively simple to use as it requires only two scalar parameters, σ and ρ. Furthermore, distributing the structure tensor

calculations on modern multi-core CPU and GPU systems, allows the method to scale well, even with large volumetric data-sets.

In Sections 2 and 3 we introduce the Structure Tensor method and the data-set, respectively. Section 4 describes the procedure of using structure tensor analysis to extract orientation information from the volumetric data-set. A procedure which is demonstrated in the Jupyter notebook *StructureTensorFiberAnalysisDemo* which can be downloaded from [5]. In Section 5, we demonstrate the procedure on a realistic sized composite specimen, a full cross-section of a tension/tension fatigue test sample. The data and Jupyter notebook for this analysis can also be downloaded from [5].

2. Structure Tensor

In the context of volumetric (3D) image analysis, a structure tensor is a 3-by-3 matrix, which summarizes orientation in a small neighborhood around a point in space. The structure tensor is computed from the gradients of the 3D image. More precisely, if $\nabla V = [V_x \;\; V_y \;\; V_z]^T$ denotes the gradient of the 3D volume V, we can compute the structure tensor as

$$\mathbf{S} = \sum \nabla V \, (\nabla V)^T \,, \tag{1}$$

where summation runs in a certain neighborhood around the point. Nice properties of the structure tensor in respect to the scale of the analyzed structures are obtained if a Gaussian window is used for integration, and a Gaussian derivative for computation of the gradient [6], leading to the following formulation

$$\mathbf{S} = K_\rho * \left(\nabla V_\sigma \, (\nabla V_\sigma)^T \right) \,. \tag{2}$$

Here, the parameter σ is the standard deviation of the Gaussian derivative kernel used for computing the gradient ∇V_σ, while ρ is the standard derivation of the Gaussian kernel K_ρ used for integrating by convolution. The parameter σ is called the *noise scale*, and indicates the size of the structures filtered out while computing the gradient. The parameter ρ is called the *integration scale* and reflects the size of the window where the orientation is analyzed. Therefore, ρ should be chosen based on the size of the structures to be analyzed.

Given a unit vector $\mathbf{u}$, the product $\mathbf{u}^T \mathbf{S}\, \mathbf{u}$ gives the squared change in intensity for a small displacement in direction $\mathbf{u}$. Therefore, finding the predominant orientation amounts to minimizing $\mathbf{u}^T \mathbf{S}\, \mathbf{u}$, which is achieved through eigendecomposition of $\mathbf{S}$. Being symmetric and positive semi-definite, $\mathbf{S}$ yields three positive eigenvalues $\lambda_1 \leq \lambda_2 \leq \lambda_3$ and mutually orthogonal eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. The eigenvector $\mathbf{v}_1$, corresponding to the smallest eigenvalue, is an orientation leading to the smallest variation in intensities, which indicates a predominant orientation in the volume. In this work, we focus on the information obtained from $\mathbf{v}_1$.

3. Data-set

A stitched data-set based on three 3D X-ray CT scans is used in this demonstration of using the structure tensor method for a fiber orientation characterization. The specimen that is scanned is a so-called butterfly-shaped tensile test-sample see e.g. [7] with a cross-section of approximately $15 \times 4\,\mathrm{mm}^2$ and a uniform shaped gauge length of 60 mm. Three 16.5 mm field of view (FoV) binning 1 scans are reconstructed and stitched into a single Transmission X-ray Microscopy (TXM) file of 31 GB covering 32 mm of the sample length. Cropping away the air around the sample results in a 9 GB file, which is saved in the NIfTI format. Both files can be obtained from [5]. The laminate used in the test-sample contains four unidirectional non-crimp fabrics surround two biaxial non-crimp fabrics resulting in the following layup $[\mathrm{b_{biax}/biax, b_{UD}/UD, b_{UD}/UD]_s}$ where the "b" indicates the location of the backing layer, which is orientated in the transverse

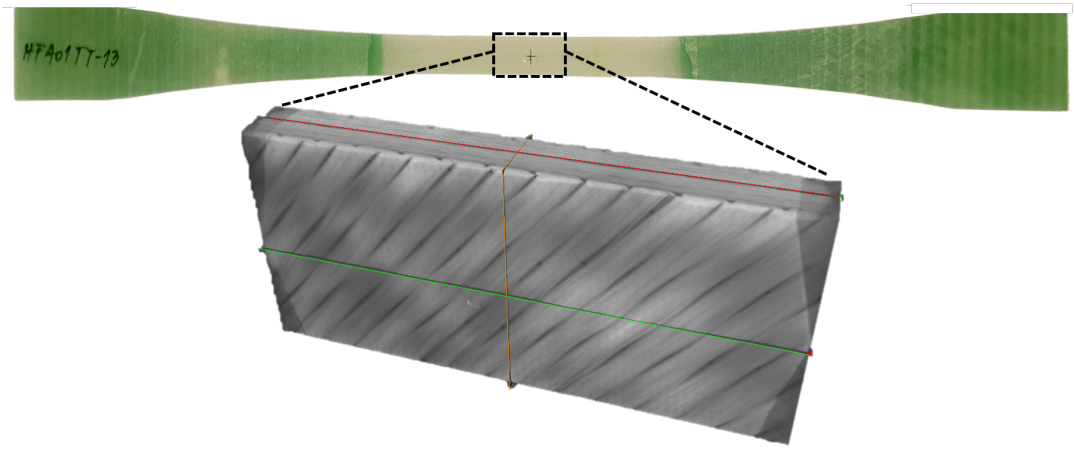


Figure 1. The test-sample and the scanned part of the gauge-section with the dimension $32 \times 15 \times 4.5\,\mathrm{mm}^3$.

direction. The UD and biax layers in the fabric have an area weight of $1134\,\mathrm{g/m^2}$ and $384\,\mathrm{g/m^2}$, respectively, and consist of 17 µm diameter glass fibers. The backing layers, $\mathrm{b_{UD}}$ and $\mathrm{b_{biax}}$, have an area weight of $36\,\mathrm{g/m^2}$ and $4\,\mathrm{g/m^2}$, respectively, with a glass fiber diameter of 9 µm. With a sample thickness of 4.4 mm in the scanned area and a glass fiber density on $\rho = 2600\,\mathrm{kg/m^3}$ this corresponds to an average fiber volume fraction in the composite laminate of $V_f = 54\%$. The fraction of fibers in the different directions is calculated later in table 2. The scan settings and resulting voxel size are listed in table 1.

Table 1. X-ray CT scan information (stitched size: $32\,\mathrm{mm} \times 16.5\,\mathrm{mm} \times 16.5\,\mathrm{mm}$)

Sample ID	Optical mag.	Voltage	Exposer time	Projections	Bins; Stitch	Scan Time	FoV	Voxel Size
HF401TT-13	0.4×	50 keV	24 s	3201	1;3	3×24h	$3 \times 16.5\,\mathrm{mm}^3$	8.08 µm

4. Procedure

We will now go through the steps needed to analyze the data using structure tensor in Python. With this paper, we release three Jupyter notebooks and a Python script file/module with helper functions. The first notebook, *StructureTensorFiberAnalysisDemo*, goes through the structure tensor analysis step by step, while the second, *StructureTensorFiberAnalysisAdvancedDemo*, offers a more advanced approach for speeding up computations on large volumes using multi-CPU or GPU systems. The third, *HF401TT-13_FoV16.5_Stitch*, can be used to recreate the results of the paper. The Python module file, *structure_tensor_workers.py*, contains various helper functions, including functions for parallel structure tensor computations across many CPUs and GPUs. The notebooks and helper functions can be obtained from [5].

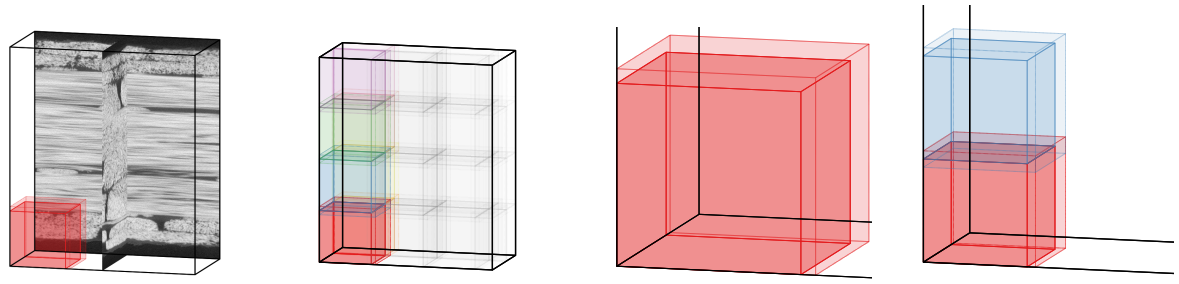
Step 0: Pre-process data. Depending on available tools and experience it may be useful to pre-process data. This includes cropping and rotating data, as well as converting it to a suitable

format for reading (see next step).

Step 1: Reading data. In our case the original data is stored in the TXM file format. To read this data in Python we can use the `dxchange` [8] package. However, we prefer either RAW format, which can be read directly with NumPy [9] `memmap`, or NIfTI, which can be read using the `nibabel` [10] package, due to the excellent functionality and performance of the `numpy` and `nibabel` packages. Another popular format is TIFF for which `scikit-image` and `tifffile` packages are useful. For large data-sets, using a memory-mapped file to access data is often more practical than reading all data into memory at once.

Step 2: Prepare data. Unless data has already been prepared before reading it, now is the time to crop, rotate, or otherwise transform the data. Depending on the size of the data-set and the amount of memory in our system, you can either keep all data in memory while doing this or use memory-mapped files. It is often convenient to save the prepared data to disk.

Step 2.5: Partition data. For large datasets, calculating the structure tensor for the full data-set at once is often infeasible due to hardware memory constraints. One way around this is to partition the data-set into blocks as shown in figure 2 and calculate the structure tensor, $\mathbf{S}_b$, for each block separately, before merging the results. Here b is the block index. This is not only more memory efficient but also allows for parallel computations. To ensure consistent results when calculating $\mathbf{S}$ in blocks, the blocks must be padded appropriately. The padding consists of actual voxels from the data-set, shown as the area between the inner and outer red box in figure 2(c). The values $\mathbf{S}_b$, which correspond to voxels located inside the padding, will not be correct and have to be discarded afterward. The size of the padding depends on the σ and ρ parameters, as they determine the size of the Gaussian filter kernels used to calculate $\mathbf{S}_b$. Because the kernel is discrete, the kernel radius is rounded to the nearest integer number. In our case the largest kernel radius is $\lfloor 4\rho \rceil = \lfloor 4 \cdot 2.96 \rceil = \lfloor 11.84 \rceil = 12$. Thus, for a value in $\mathbf{S}_b$ to be valid, the 12 neighboring voxels in all six directions must be included in the same block. Values in $\mathbf{S}_b$, where the block does not include the 12 neighboring voxels in each direction will vary depending on the partitioning of the blocks. Since $\mathbf{S}$ should not depend on the partitioning of blocks, these values have to be thrown away. The *structure_tensor_workers.py* file includes functions for creating and merging blocks (referred to as "crops" in the code) with proper padding. In our implementation,



(a) Block 1 (b) All blocks (c) Block 1 (zoomed) (d) Block 1 & 2

Figure 2. We partition the volume (black box) into blocks (colored boxes) as shown in (a) and (b). The outer red box, more clearly visible in (c), represents block 1, which contains the data required to calculate the structure tensor for block 1, $\mathbf{S}_1$. The inner red box represents the voxels for which the values of $\mathbf{S}_1$ are valid. The overlapping of the blocks needed to calculate a valid $\mathbf{S}$ for the complete volume is more clearly visible in (d).

the user specifies the size of the blocks using the `crop_size` parameter. The maximum size of a block is a cube with sides of length `crop_size` plus the kernel radius. Of course, blocks may be smaller to ensure they fit inside the volume.

Step 3: Compute structure tensor and eigendecomposition. Using the `structure-tensor` package, we can calculate $\mathbf{S}$ for a volume using the `structure_tensor_3d` function. Although $\mathbf{S}$ is in theory a 3-by-3 matrix, because it is symmetric, we only need to store six values per voxel. As a result, given a volume with the shape (X, Y, Z), the array returned by `structure_tensor_3d` function has the shape (6, X, Y, Z). To get the eigenvalues and vectors, we pass $\mathbf{S}$ to the `eig_special_3d` function. By default, given an input with shape (6, ...), the values and vectors returned by `eig_special_3d` will have shape (3, ...), where (...) is an arbitrary shape. The values returned are the three eigenvalues in ascending order. For the vectors, only the vector for the smallest eigenvalue is returned by default. If the other two vectors are needed, setting `full=True` will return vectors as a (3, 3, ...) array instead where the first dimension corresponds to the vectors for each of the three eigenvalues.

In our experiments we choose σ and ρ as

$$\sigma = \frac{r_f}{\sqrt{2}} = 0.74 \qquad \rho = 4\sigma = 2.96, \tag{3}$$

where $r_f = 1.05$ is the UD fiber radius in voxels. Decreasing σ much further can lead to numerical instability due to truncation, numerical precision used in the implementation, and the discrete nature of the data. For ρ, we want to include a large enough area to determine structural "direction", while keeping the integration area small enough to avoid to much "bleeding" between structures (fibers/bundles).

For computing $\mathbf{S}$, the `structure-tensor` package relies on the SciPy [11] `scipy.ndimage.gaussian_filter` function. To speed up computation using GPUs, the `structure-tensor` package uses the CuPy library [12], which implement GPU targeted versions of large parts of the NumPy and SciPy libraries.

Step 3.5: Saving results. Depending on the purpose of the analysis and workflow, it may be relevant to save the computed results. For instance, if we want to view the eigenvalues or vectors in another application, or if we want to save them for later analysis. Again, saving the data as NIfTI or RAW is simple with `niabel` or `numpy` and the formats are easily loaded in many applications.

Step 4: Orientation metrics. Since we focus on orientations, we will only be using the eigenvectors, $\mathbf{v}_1$. The eigenvalues have interesting use cases as well, but we will not discuss those in this paper.

Our analysis of orientations is based on the four directions, in which fiber bundles in our material are oriented. We represent each of the four directions as a class with a corresponding class unit vector, $\mathbf{c}_o$, where o is the counter-clockwise rotation of the bundle in the xy-plane from the x-axis in degrees. The four class vectors are $\mathbf{c}_0 = [1, 0, 0]$, $\mathbf{c}_{45} = [0.707, 0.707, 0]$, $\mathbf{c}_{-45} = [0.707, -0.707, 0]$ and $\mathbf{c}_{90} = [0, 1, 0]$. The class vectors can and should be specified to match material composition. At least one class vector must be set.

The first class vector is considered the primary vector, which means it is the direction in which the orientations are calculated relative to. The remaining class vectors are used solely to label the voxels, also known as segmentation. In our notebooks we use the `calculate_angles` function from the `structure_tensor_workers` module to get a label (class), stiffness (η_o), angle (θ), rotation in xy-plane and rotation out of the xy-plane for each eigenvector in $\mathbf{v}_1$.

The `calculate_angles` function works by first determining the class of each vector. Since each vector belongs to a specific voxel in the volume this results in a complete segmentation of the volume. The segmentation is done by calculating the angle between the eigenvectors, $\mathbf{v}_1$, and each class vector, $\mathbf{c}_o$. A vector is assigned the class of the class vector with which is it most aligned (has the smallest angle). The angle between $\mathbf{v}_1$ and the primary class vector, in our case $\mathbf{c}_0$, is the value θ, which is between 0 and 90 degrees. Based on θ we calculate the stiffness estimate, η_o, which is between 0 and 1. The last two metrics are the orientations in and out of the xy-plane. Both are between -90 and 90 degrees.

Step 5: Choosing fiber threshold. One issue with the segmentation from step 4 is that it labels all voxels as belonging to one of the fiber classes. However, the volume is not only fiber material but also includes epoxy, which have a lower density than the glass fiber but higher density than air. Segmenting epoxy (or air) into one of the four orientation classes or calculating the orientation of epoxy voxels does not make sense. Therefore, we create a background class and use a simple intensity threshold to separate the foreground (fibers) and background (non-fiber). Using a histogram, it is fairly easy to choose a sensible threshold value. Another option is to use a statistical method, such as Otsu's threshold, which we also demonstrate in our notebooks. When choosing our threshold, we make sure not to include values outside the sample or near the edge in our histogram. This is because we want to separate foreground and background inside the sample and intensities outside the sample may be significantly different.

Step 6: Plotting segmentation and orientations. Using the results of steps 5 and 6, we can plot segmentation and orientations on top of the data to verify the correctness of our calculations. Here, choosing appropriate color maps for orientations is important for meaningful figures. We demonstrate this in both our notebooks.

Step 7: Orientation distributions. After qualitatively assessing our results by plotting, we calculate the class fractions, which tell us how much of the fiber material belongs to each class. This is easily comparable to the material specifications. Afterward, we can use the segmentation labels to create histograms for each of the orientation metrics for each class, as well as combined for all classes. Along with the histograms, we can also calculate variables such as the median, mean, and standard deviation for the distributions. However, it is important to note that the orientation metrics are *not* normally distributed, so we have to be careful with treating them as such. A more appropriate distribution for describing orientation data would probably be the Bingham distribution [13], but that is outside the scope of this paper.

5. Results

The results shown here are also available in the *HF401TT-13_FoV16.5_Stitch* Jupyter notebook and can be reproduced by running the code in the notebook [5].

The composition of the fabric is critical to its mechanical properties. Table 2 show the expected fiber distribution across the four classes, as well as the segmented distribution in the scanned region, estimated using the previously described procedure. The expected fiber distribution is based on the reported fiber distribution for the individual non-crimp fabric used for the composite. The segmented fraction estimates are less than 2% from the expected fractions, indicating both that the material follows the specification closely and that the segmentation is accurate.

To qualitatively validate our results, we can look at the segmentation results for the analyzed region of the sample, shown in figure 3. We see that the segmentation appears accurate, although some voxels near the class interfaces may be mislabeled. This is not surprising, as orientations

Table 2. Volumetric fractions of the fiber orientations

$[\mathrm{b/biax, b/UD, b/UD}]_\mathrm{s}$	0°	45°	−45°	90°
Area weight $[g/m^2]$	4×1139	2×384	2×384	4×36+2×4
Calculated fractions	0.729	0.123	0.123	0.024
Segmented fractions	0.746	0.122	0.116	0.016

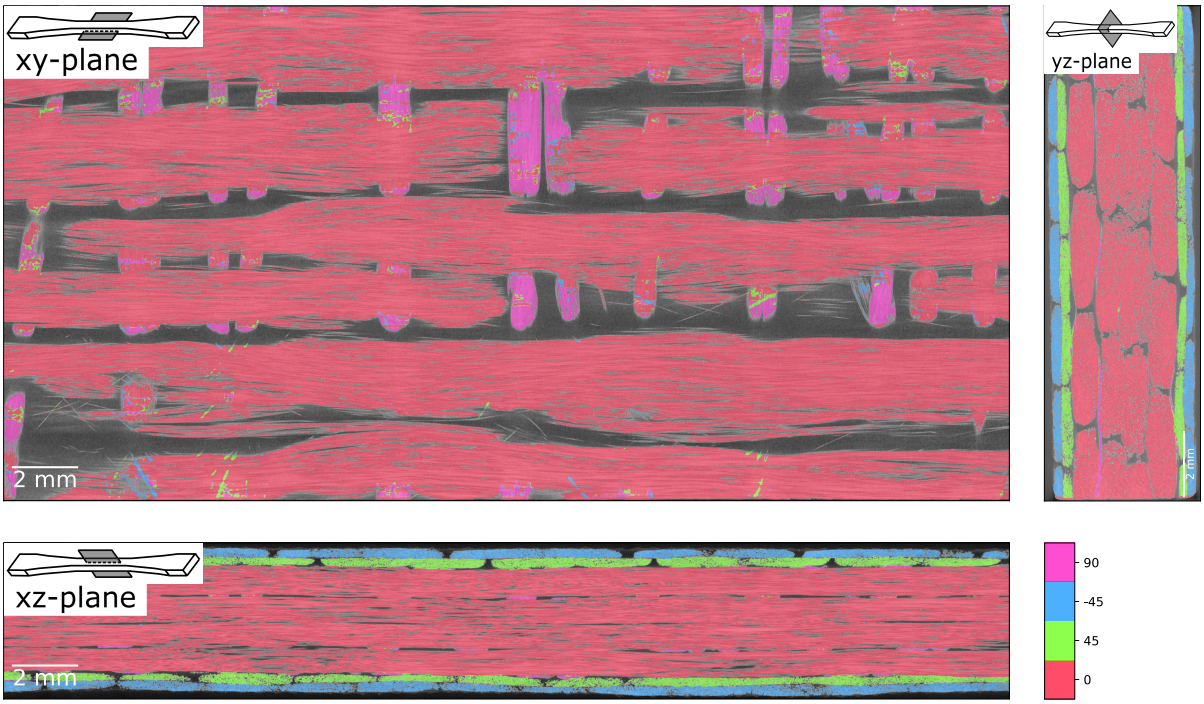


Figure 3. Segmentation based on fiber orientations of the non-crimp fabric shown in the center of the sample in the xz and yz-plane and in the xy-plane in a plane with backing bundles. The four different fiber classes are represented by four different colors, while the background is gray. The segmentation is shown on top of the original grayscale images.

near the interfaces are integrated over data from both sides of the interface. For instance, this could cause a voxel near the 0°/90°-interface to be classified as −45° or 45°. We see this in the figure 3 xy-plane image as blue or green "bleeding" between the red and pink areas. One way to correct for this is to remove/reassign small components, as done in [2]. However, as the number of mislabeled voxels is relatively small, it should not affect the orientation distributions much. One exception could be the least represented class, 90°, where the mislabel voxels near the interfaces could be contributing to the estimated fraction (1.6%) being smaller than the expected fraction (2.4%). In general, we suspect that the bleeding effect, which depends on the integration scale, ρ, could result in slightly underestimated fractions for the less frequent classes, which is exactly what we see. However, we do not know at this point, if this is actually the case here. Either way, the estimated fractions and visual inspection of segmented slices makes us quite confident in the accuracy of our segmentation and estimated orientations.

To distinguish the matrix-rich regions without a material orientation from the fiber-rich

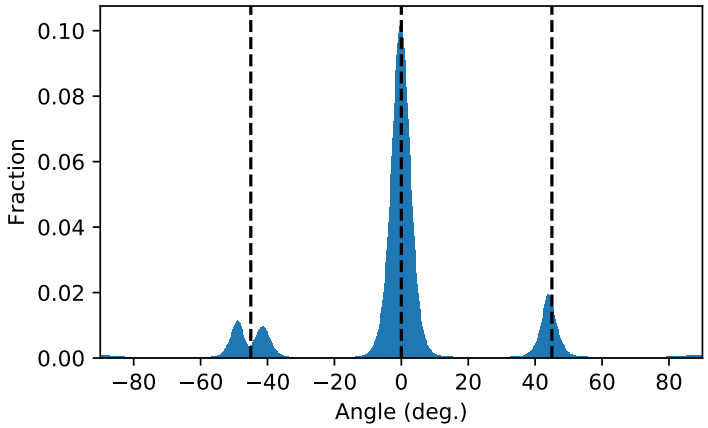


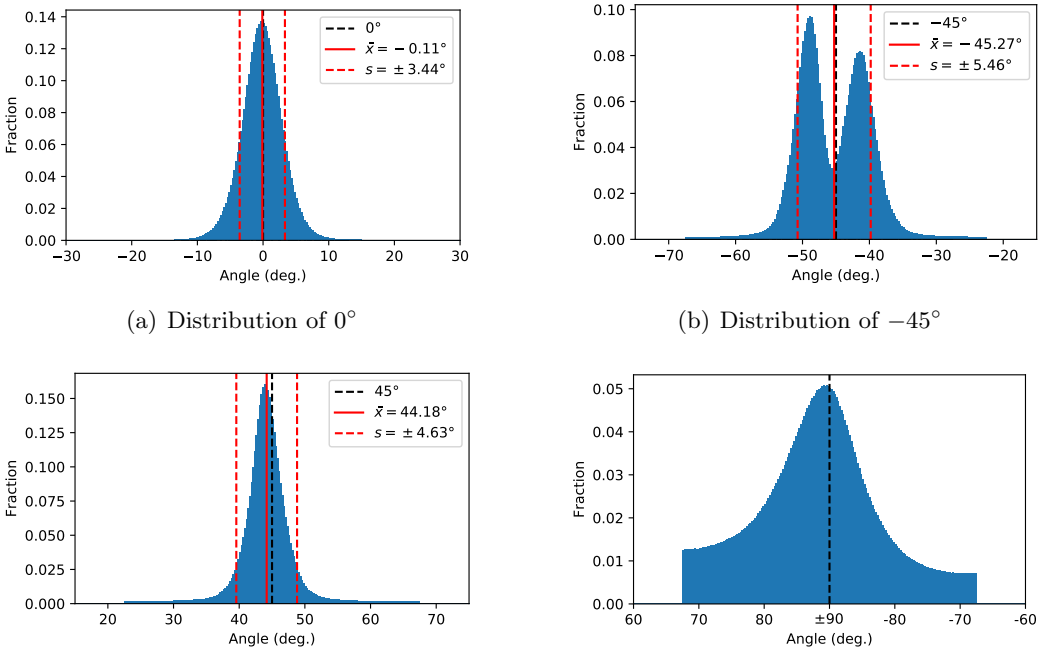
Figure 4. Distribution of the fiber orientations projected onto the xy-plane.

regions with a material orientation, the fiber regions are segmented using a threshold value corresponding to the non-black regions in figure 3. In our experiments we use a histogram to choose a reasonable threshold value. In theory, we can use the threshold to calculate the fiber volume fraction, V_f. However, with a voxel size of $8\,\mu m^3$ and a fiber diameter of $17\,\mu m$ for the dominating (UD) fibers, this is unlikely to give a good estimate of V_f. With this resolution, we cannot represent the small pockets of epoxy between the fibers inside the bundles, which means the majority of the voxels inside the bundles are counted as fiber material. Thus, in our experiment, the threshold value results in $V_f = 75\%$, which is much higher than the specified 54% for the material. We could increase the threshold value to get closer to the expected V_f, however inspecting the intensity distribution and the segmentation qualitatively, this does not seem like the right solution. For instance, a higher threshold value appears to favor the thicker UD fibers, thereby over-representing the 0° class. Although V_f is not accurate, we should still be able to trust the estimated ratio between the different fiber orientation classes, as long as the fiber volume fraction inside the bundles is approximately the same for all the orientation classes.

To quantitatively evaluate the fabric and bundles (fiber rowings) used for building the fabric, we can plot the in-plane fiber orientation distributions. Figure 4 shows the distribution orientations of the fibers projected onto the xy-plane for all fibers in the analyzed sample. In figure 5 the same distribution has been separated for the four classes: 0°, −45°, 45°, and 90°. From figures 5(b) and 5(d) we see that both 0° and 45° fibers appear to align with the expected orientation, with means being off by less than a degree and standard deviations of less than four degrees. However, the −45° class is another story. It appears that the fibers in this class follow two different distributions, shifted around four degrees in opposite directions.

To investigate the −45° class further, we have separated the orientation data for the two −45° surface biax-plies in the sample. Figure 6 clearly shows that the two plies correspond to the two peaks in 5(c). The two plies appear to be rotated about four degrees in opposite directions. It can, therefore, be concluded that there is a mismatch between the expected $+45°/-45°$ biax layup and the actual $+45°/(-45\pm4)°$. This difference could either come from the manufacturing of the non-crimp stitched fabric or have been introduced by shearing the fabric during the layup.

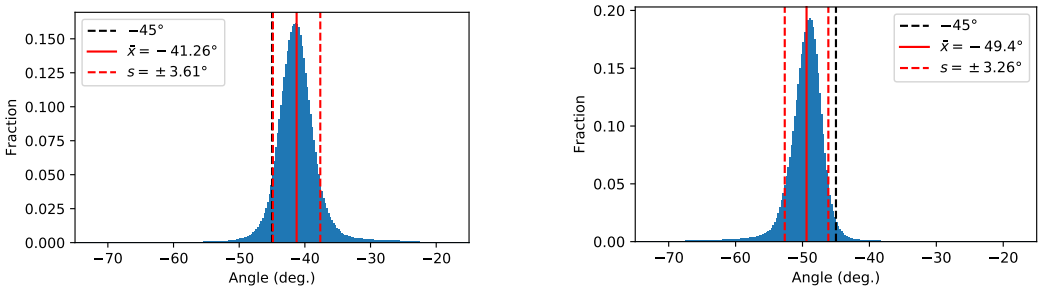
Figure 7 shows the out-of-plane fiber orientations for the xy-plane. Here, all classes should have the same 0° orientation, which also appears to be the case, with all means being within one degree of the expected orientation. Again the UD (0°) fibers appear to align almost perfectly, while the biax-ply fibers diverge slightly more. We also note that the −45° and 45° classes



(a) Distribution of 0° (b) Distribution of −45°

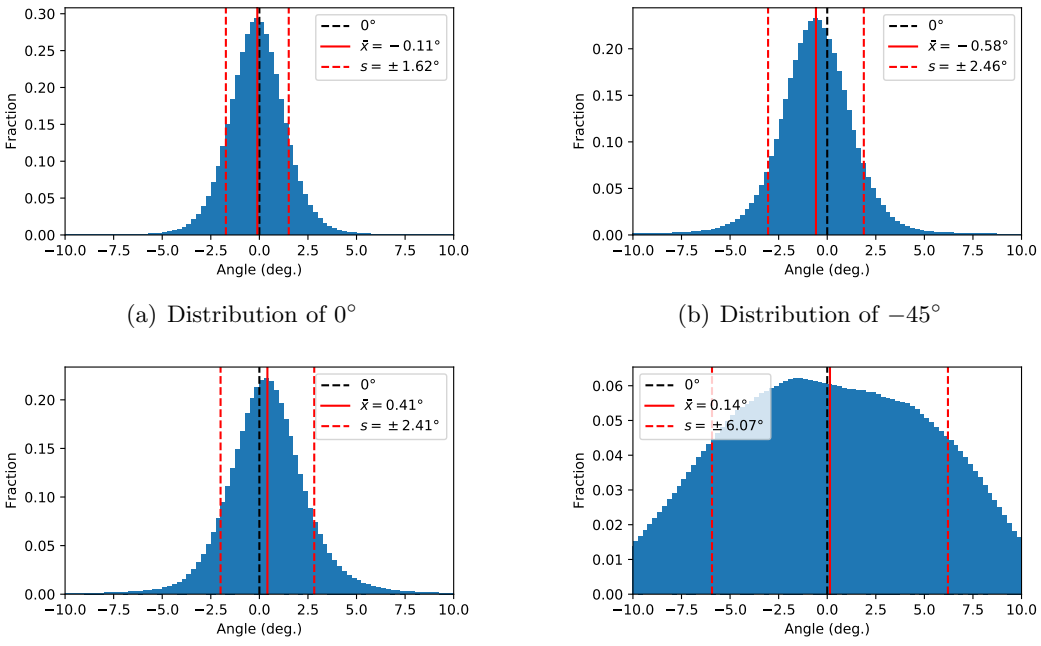
(c) Distribution of +45° (d) Distribution of 90°

Figure 5. Distribution of the in-plane fiber orientations for the xy-plane. The angle is the orientation of the eigenvectors projected onto the xy-plane. (a), (b), (c) and (d) shows the distribution for the four classes, 0°, −45°, 45°, and 90°, with corresponding class vectors, $\mathbf{c}_0$, $\mathbf{c}_{-45}$, $\mathbf{c}_{45}$ and $\mathbf{c}_{90}$. For the first three distribution the mean, $\bar{x}$, and standard deviation, s, are also included.



(a) Distribution of the upper ply (b) Distribution of the lower ply

Figure 6. Distribution of the in-plane fiber orientations for the xy-plane for the two −45° plies in the sample. Together these should be almost equal to the distribution in 5(c).



(a) Distribution of 0°

(b) Distribution of −45°

(c) Distribution of 45°

(d) Distribution of 90°

Figure 7. Distribution of the out-of-plane fiber orientations for the xy-plane. This is the angles between the eigenvectors and the xy-plane. (a), (b), (c) and (d) shows the distribution for each of the classes. For each distribution the mean, $\bar{x}$, and standard deviation, s, are also included.

appear to be oriented about half a degree in opposite directions. However, this small deviation could be an artifact from the structure tensor analysis, rather than the fibers actually being misoriented. While the 90° class is also oriented as expected, the orientation distribution for the class has a large variance. The most likely explanation is simply that it is by far the least represented class and thus the one most affected by noise and "bleeding" artifacts. Introducing a fifth class containing fibers, which fit poorly into all the four known classes could remove some of the noise from the distribution. Trying different values for ρ and σ could also help to reduce both noise and bleeding.

For a given load direction, the fiber orientation distribution can be quantified with a so-called fiber orientation efficiency factor [14],

$$\eta_o = \sum_i a_i \cos^4 \theta_i \tag{4}$$

where θ_i is the orientation with respect to the loading direction of the i-th fraction a_i. The loading direction is here given as the x-axis. A value of $\eta_o = 1$ corresponds to an ideal uni-directional composite with all fibers in the loading direction while $\eta_o = 0$ corresponds to a composite with all fibers orthogonal to the loading direction. For our sample, the fiber efficiency factor for fibers belonging to the 0° class is found to be $\eta_o(0°) = 0.991$, while the overall efficiency factor, including all fibers for all four classes, is $\eta_o = 0.802$.

6. Conclusion

We have shown that structure tensor analysis can be used to acquire important information about fiber orientations from micro-CT data of glass fiber fabrics. We have described how to

do this step by step and included three Jupyter notebooks and a Python module demonstrating the procedure.

The results show that structure tensor information can be used for volume segmentation, as well as to extract important metrics, such as fiber class fractions and orientations. Despite relativity low resolution with a voxel-size only half the dominating fiber diameter, a good agreement is obtained with the expected volumetric fractions for the four different orientation classes. In addition, the method has shown its value as a non-destructive quality control procedure, in this case, making it possible to identify the $\pm 4^\circ$ misorientation of the -45° class. Furthermore, the method can produce orientation distributions for each class, which amongst other things, can be used to calculate the fiber orientation efficiency factor. Such mechanical properties are valuable for quality control, and the scalability of structure tensor on modern hardware allows acquisition of these properties to be done quickly, even for large samples. In this paper, we have studied a glass fiber-based composite, but the method should work equally well for other fiber materials.

The presented segmentation method is simple and works well for cases, where the fiber classes are known beforehand. That said, there are many ways to improve the accuracy of the orientation distributions and segmentation. Firstly, methods for noise reduction, normalization, and image sharpening could enhance the data quality. This could make both threshold-based segmentation and estimated orientations more accurate. Secondly, using a connected component approach to eliminate isolated falsely classified voxels, as in [2], and/or introducing another class for "noisy" fiber voxels, could improve segmentation accuracy. However, we show that a very simple segmentation method can produce sufficiently accurate results when combined with structure tensor analysis.

Acknowledgments

N Jeppesen's work is supported by FORCE Technology. The material system used and L P Mikkelsen's work is supported by the Danish Energy Agency through the Energy Technology Development and Demonstration Program (EUDP), grant no. 64018-0068. The supported project is RELIABLADE: Improving Blade Reliability through Application of Digital Twins over Entire Life Cycle. This work is partly supported by The Center for Quantification of Imaging Data from MAX IV (QIM) funded by The Capital Region of Denmark.

References

[1] Emerson M J, Jespersen K M, Dahl A B, Conradsen K and Mikkelsen L P 2017 Individual fibre segmentation from 3D X-ray computed tomography to study the misalignment in unidirectional composite materials *Compos. Part A Appl. Sci. Manuf.* **97** 83–92 ISSN 1359835X URL `http://dx.doi.org/10.1016/j.compositesa.2016.12.028`

[2] Straumit I, Lomov S V and Wevers M 2015 Quantification of the internal structure and automatic generation of voxel models of textile composites from X-ray computed tomography data *Compos. Part A Appl. Sci. Manuf.* **69** 150–58 ISSN 1359835X URL `http://dx.doi.org/10.1016/j.compositesa.2014.11.016`

[3] Straumit I, Hahn C, Winterstein E, Plank B, Lomov S V and Wevers M 2016 Computation of permeability of a non-crimp carbon textile reinforcement based on X-ray computed tomography images *Compos. Part A Appl. Sci. Manuf.* **81** 289–95 ISSN 1359835X URL `http://dx.doi.org/10.1016/j.compositesa.2015.11.025`

[4] Nguyen N Q, Mehdikhani M, Straumit I, Gorbatikh L, Lessard L and Lomov S V 2018 Micro-CT measurement of fibre misalignment: Application to carbon/epoxy laminates manufactured in autoclave and by vacuum assisted resin transfer moulding *Compos. Part A Appl. Sci. Manuf.* **104** 14–23 ISSN 1359835X URL `https://doi.org/10.1016/j.compositesa.2017.10.018`

[5] Jeppesen N, Dahl V A, Christensen A N, Dahl A B and Mikkelsen L P 2020 Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor [data set] *Zenodo* URL `http://dx.doi.org/10.5281/zenodo.3877522`

[6] Weickert J 1998 *Anisotropic Diffusion in Image Processing* (Teubner Stuttgart) URL `https://www.mia.uni-saarland.de/weickert/Papers/book.pdf`

[7] Jespersen K M and Mikkelsen L P 2017 Three dimensional fatigue damage evolution in non-crimp glass fibre fabric based composites used for wind turbine blades *Compos Sci Technol* **153** 261–72 ISSN 02663538 URL http://linkinghub.elsevier.com/retrieve/pii/S0266353817301811
[8] De Carlo F *et al.* 2014 Scientific data exchange: a schema for HDF5-based storage of raw and analyzed data *J. Synchrotron Radiat.* **21** 1224–30 URL https://doi.org/10.1107/S160057751401604X
[9] van der Walt S, Colbert S C and Varoquaux G 2011 The numpy array: A structure for efficient numerical computation *Comput Sci Eng* **13** 22–30
[10] Brett M *et al.* 2020 nipy/nibabel: 3.1.0 URL https://doi.org/10.5281/zenodo.3757992
[11] Virtanen P *et al.* 2020 SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python *Nat. Methods* **17** 261–72
[12] Okuta R, Unno Y, Nishino D, Hido S and Loomis C 2017 *Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Thirty-first Annual Conference on Neural Information Processing Systems (NIPS)* URL http://learningsys.org/nips17/assets/papers/paper_16.pdf
[13] Bingham C 1974 An antipodally symmetric distribution on the sphere *Ann. Stat.* 1201–25
[14] Krenchel H 1964 *Fibre reinforcement. Theoretical and practical investigations of the elasticity and strength of fibre-reinforced materials* (Akademisk Forlag)