Downloaded from orbit.dtu.dk on: May 07, 2024

DTU Library

Image Analysis for Wind Turbine Blade Structures and Materials

Jeppesen, Niels

Publication date:
2021

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

Link back to DTU Orbit

Citation (APA):
Jeppesen, N. (2021). *Image Analysis for Wind Turbine Blade Structures and Materials*. Technical University of Denmark.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain
- You may freely distribute the URL identifying the publication in the public portal

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Ph.D. Thesis
Doctor of Philosophy

DTU Compute
Department of Applied Mathematics and Computer Science

Image Analysis for Wind Turbine Blade Structures and Materials

Niels Jeppesen

Kongens Lyngby 2021

DTU Compute
Department of Applied Mathematics and Computer Science
Technical University of Denmark

Richard Petersens Plads
Building 324
2800 Kongens Lyngby, Denmark
Phone +45 4525 3031
compute@compute.dtu.dk
www.compute.dtu.dk

"We demand rigidly defined areas of doubt and uncertainty!"

- Douglas Adams, The Hitchhiker's Guide to the Galaxy

Summary

The increasing size of wind turbine blades makes quality control of the blades increasingly challenging and costly. At the same time, tighter design tolerances require manufacturers of blades and blade materials to ensure that structural and material properties follow the specifications closely.

In this thesis, I use computer-based image analysis to address the challenges of extracting the structural and material properties needed for quality control of wind turbine blades. The goal is to present methods suitable for quality control solutions that can assist human evaluation or automate quality control of wind turbine blades.

For the evaluation of blade structures, I propose a method based on graph cut optimization for segmentation and surface detection in both 2D and 3D ultrasound images. To make this method scale to large 3D datasets, a new type of graph structure for multi-label segmentation has been developed along with new high-performance parallel and serial versions of state-of-the-art graph cut algorithms. The research contributions are generic and applicable to a range of different optimization and computer vision tasks. Furthermore, I discuss some of the challenges of analyzing ultrasound images of wind turbine blades and why the presented method is suitable for this purpose.

For the evaluation of blade materials, I present an approach for estimating fiber orientations in fiber-reinforced composites using structure tensor analysis. The method uses Gaussian kernels and analytical eigendecomposition that makes it tolerant to noise, resolution invariant, and fast. The implementation uses vector operations for even faster computations on modern hardware. Then, I demonstrate the use of structure tensor analysis for characterizing fiber orientations in unidirectional fiber-reinforced composites commonly used in wind turbine blades. Finally, I discuss some of the challenges and things to consider when dealing with orientation information in 3D.

The work presented in this thesis allows important structural properties to be extracted from large 3D images which form the basis for automated quantitative evaluation of wind turbine blades and fiber-reinforced composites.

Resumé

Kvalitetskontrol af vindmøllevinger bliver til stadighed mere udfordrende og omkostningsfuldt i takt med at vingerne bliver større og større. Samtidig medfører strengere designtolerancer, at producenter af vindmøllevinger og vingematerialer i endnu højere grad end tidligere har brug for at sikre, at struktur- og materialegenskaber følger specifikationerne.

I denne afhandling imødekommer jeg disse udfordringer ved at bruge computerbaseret billedanalyse til at udtrække struktur- og materialegenskaber, der er nødvendige for kvalitetskontrol af vindmøllevinger. Formålet er at præsentere metoder, som kan indgå i løsninger, der kan assistere eller erstatte menneskelig evaluering ved kvalitetskontrol af vindmøllevinger.

Til evaluering af vingestrukturer præsenterer jeg en metode baseret på grafoptimering, som kan bruges til billedsegmentering og lokalisering af overflader i både 2D- og 3D-ultralydsbilleder. For at metoden skal kunne skalere til store 3D-datasæt er der udviklet en ny grafstruktur, rettet mod billedsegmentering med mange etiketter, samt nye højtydende parallelle og serielle versioner af state-of-the-art algoritmer til grafoptimering. Forskningsbidragene er generiske og kan anvendes på en række forskellige optimerings- og datamatsynsopgaver. Desuden diskuterer jeg nogle af udfordringerne ved at analysere ultralydsbilleder af vindmøllevinger, samt hvorfor den præsenterede metode er velegnet til denne opgave.

Til evaluering af vingematerialer præsenterer jeg en metode til estimering af fiberorienteringer i fiberarmerede kompositter ved hjælp af struktur tensoranalyse. Metoden bruger gaussiske kerner og analytisk udledning af egenvektorer og egenværdier, hvilket gør den tolerant over for støj, uafhængig af opløsning og hurtigt. Implementeringen bruger vektoroperationer til at forøge beregningshastigheden yderligere på moderne hardware. Efterfølgende demonstrerer jeg brugen af struktur tensoranalyse til karakterisering af fiberorienteringer i fiberarmerede kompositter, som anvendes i vindmølleblade. Endelig diskuterer jeg nogle af de udfordringer og forhold, der skal tages stilling til, når man beskæftiger sig med orienteringsinformation i 3D.

Arbejdet, som indgår i denne afhandling kan bruges til at udtrække vigtige strukturelle egenskaber fra store 3D-billeder, som danner grundlag for automatiseret kvantitativ evaluering af vindmøllevinger og fiberarmerede kompositter.

Preface

This Ph.D thesis was prepared at the Department of Applied Mathematics and Computer Science at the Technical University of Denmark (DTU Compute) in fulfillment of the requirements for acquiring a Ph.D degree in computer science. The work presented in this thesis was funded by FORCE Technology. The project was carried out as a part of the Manufacturing Academy of Denmark (MADE) Digital, by Innovation Fund Denmark, and in collaboration with the ReliaBlade project, by The Energy Technology Development and Demonstration Programme (EUDP).

The thesis studies image analysis methods for extracting information from 3D images relevant to quality control of wind turbine blades. Six academic papers have been written as part of the Ph.D. project. Two of these papers are peer reviewed and published, three have been submitted and are awaiting peer review, and one is in preparation, but is expected to be submitted for peer review soon. An overview of the papers and published datasets can be found in the Publication List. Five of the papers are included in Chapters 3 and 4, while the sixth paper is included in Appendix B as my contribution to the paper does not differ significantly from the work presented in Chapter 4.

The project was supervised by Professor MSO Anders B. Dahl, Associate Professor Anders N. Christensen, and Associate Professor Vedrana A. Dahl from DTU Compute. CDO & COO Lars Vesth from FORCE Technology was the industrial supervisor. The research was carried out at the department for Digital Innovation at FORCE Technology, at the section for Visual Computing at DTU Compute, and during a short external stay at the University of Bristol under the supervision of Professor Robert A. Smith.

Kongens Lyngby, March 22, 2021

Niels Jeppesen

Acknowledgements

First and foremost, I would like to thank my industrial supervisor Lars Vesth for his continuous support and for allowing me to pursue a Ph.D. degree with support and funding from FORCE Technology.

Furthermore, I would like to thank my two supervisors, Anders B. Dahl and Anders N. Christensen, and my incredible bonus supervisor Vedrana A. Dahl, for their support, supervision, and late nights of writing and correcting papers. Their dedication to computer vision research and its practical application is an inspiration.

I would like to thank Lars P. Mikkelsen from DTU Wind Energy for the collaboration, which has contributed significantly to my thesis. Without his ideas and guidance, my work on fiber-reinforced composites would not have existed.

Additionally, I would like to thank all my amazing colleagues at DTU and FORCE Technology who have been a part of this journey. Whether it is for work or social activities, it is always a pleasure to be around you all. Especially, I would like to thank Patrick M. Jensen for the many interesting discussions and the hard work he has put into coding and writing for our collaborative work. Also, I would like to thank Hans Martin Kjer for keeping me company and bringing me snacks during my final days of writing.

I would also like to thank Robert A. Smith and the researchers at the University of Bristol for letting me visit and being so welcoming during my short stay.

Finally, I would like to thank my parents for all their support, great food, and for taking the time to proofread my entire thesis multiple times, as well as my girlfriend Mari for her support and for reminding me that there are more things to life than computers and wind turbine blades.

Publication List

Peer-Reviewed Conference Proceedings

Jeppesen, N., A. N. Christensen, V. A. Dahl, and A. B. Dahl (June 2020a). "Sparse Layered Graphs for Multi-Object Segmentation". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, pp. 12774–12782. ISBN: 978-1-7281-7168-5. DOI: 10.1109/CVPR42600.2020.01279.

Journal Articles

Jeppesen, N., V. A. Dahl, et al. (October 2020). "Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor". In: *IOP Conference Series: Materials Science and Engineering* 942, p. 012037. ISSN: 1757-899X. DOI: 10.1088/1757-899X/942/1/012037.

Submitted

Bo Salling, F. et al. (2021). "Individual Fibre Inclination Segmentation from X-ray Computed Tomography using Principal Component Analysis". In: *Composites Part A: Applied Science and Manufacturing.* Submitted.

Jeppesen, N., L. P. Mikkelsen, et al. (2021). "Quantifying effects of manufacturing methods on fiber orientation in unidirectional composites using structure tensor analysis". In: *Composites Part A: Applied Science and Manufacturing.* Submitted.

Jeppesen, N., P. M. Jensen, et al. (2021). "Faster Multi-Object Segmentation using Parallel Quadratic Pseudo-Boolean Optimization". In: *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*. Submitted.

In preparation

Jensen, P. M. et al. (2021). "Comparing Serial and Parallel Min-Cut Max-Flow Algorithms for Computer Vision". In: *IEEE Transactions on Image Processing.* In preparation.

Datasets

Jeppesen, N., V. Dahl, et al. (June 2020). *Characterization of the Fiber Orientations in Non- Crimp Glass Fiber Reinforced Composites using Structure Tensor*. Zenodo. DOI: 10.5281/zenodo.3877522.

Jeppesen, N., A. N. Christensen, V. A. Dahl, and A. B. Dahl (June 2020b). *Sparse Layered Graphs for Multi-Object Segmentation (notebooks)*. Technical University of Denmark. DOI: 10.11583/DTU.12016941.v1.

Jeppesen, N., A. N. Christensen, V. A. Dahl, A. B. Dahl, et al. (November 2020). *Sparse Layered Graphs for Multi-Object Segmentation (data)*. Technical University of Denmark. DOI: 10.11583/DTU.12462143.v2.

Published Software

maxflow

C++ library Modified version of Maxflow algorithm by Yuri Boykov and Vladimir Kolmogorov for very large graphs.
`https://github.com/Skielex/maxflow`

thinmaxflow

Python package Thin Python wrapper for a modified version of the Maxflow algorithm by Yuri Boykov and Vladimir Kolmogorov.
`https://github.com/Skielex/thinmaxflow`

QPBO

C++ library Modified version of QPBO algorithm by Vladimir Kolmogorov for very large graphs.
`https://github.com/Skielex/QPBO`

thinqpbo

Python package Thin Python wrapper for a modified version of the quadratic pseudo-Boolean optimization (QPBO) algorithm by Vladimir Kolmogorov.
`https://github.com/Skielex/thinqpbo`

slgbuilder

Python package This package allows building and solving Sparse Layered Graphs (SLG) using s-t graph cuts.
`https://github.com/Skielex/slgbuilder`

structure-tensor

Python package Fast and simple to use 2D and 3D structure tensor implementation for Python.
`https://github.com/Skielex/structure-tensor`

Contents

CHAPTER 1

Introduction

Renewable energy is vital, if we are to meet the goal set by the United Nations Paris Agreement of keeping global warming well below 2 °C, compared to pre-industrial levels. Our demand for electrical energy is one of the single largest contributors to human emissions of greenhouse gasses. Even though the Paris Agreement was signed only five years ago, in 2016, the yearly production of renewable energy has been steadily increasing over the past two decades. According to data from the International Energy Agency[1], the amount of renewable energy generated in 2018 was more than twice that of the year 2000. While hydro-electric power still made up the majority (69%) of the renewable energy generated in 2018, this is well below the 97% that it accounted for 18 years earlier. The reduced share of hydropower is not a result of reduced production. On the contrary, the production of hydropower actually increased by over 60% from 2000 to 2018. Instead, it is a result of an explosive growth in two other types of renewable energy: solar and wind.

Wind power production began ramping up in the late nineties in countries like Denmark and Germany. Still, by the year 2000, the total production of wind energy world-wide was only 31 348 GWh, corresponding to about 1% of renewable energy and a mere 0.2% of total energy production. However, from 2000 to 2018, the global production of wind energy increased over 40 times, increasing its share to 20% of the renewable and almost 5% of the total energy production world-wide. This makes wind energy the second largest source of renewable energy after hydropower, with more than twice the capacity of solar electric power, as of 2018. This incredible growth in wind energy production was made possible by an equally impressive growth in the size of the wind turbines, facilitated by rapid technological advancements. As the demand for cheap renewable energy increases, the pressure on wind turbine manufacturers to produce even bigger, cheaper, and more reliable wind turbines also increases.

In this thesis, I examine computer-based methods for assisted and automated quality control of wind turbine blades. Specifically, I propose a number of methods for quality control of composite blade structures and materials. These methods are based on computer-based image analysis and applied to ultrasound and X-ray computed tomography images. The goal is to improve the quality of the blades, while reducing the manufacturing costs. This combination of cheaper and better blades is necessary to sustain the growth required to meet the demand for wind energy world-wide.

[1] See Appendix A.

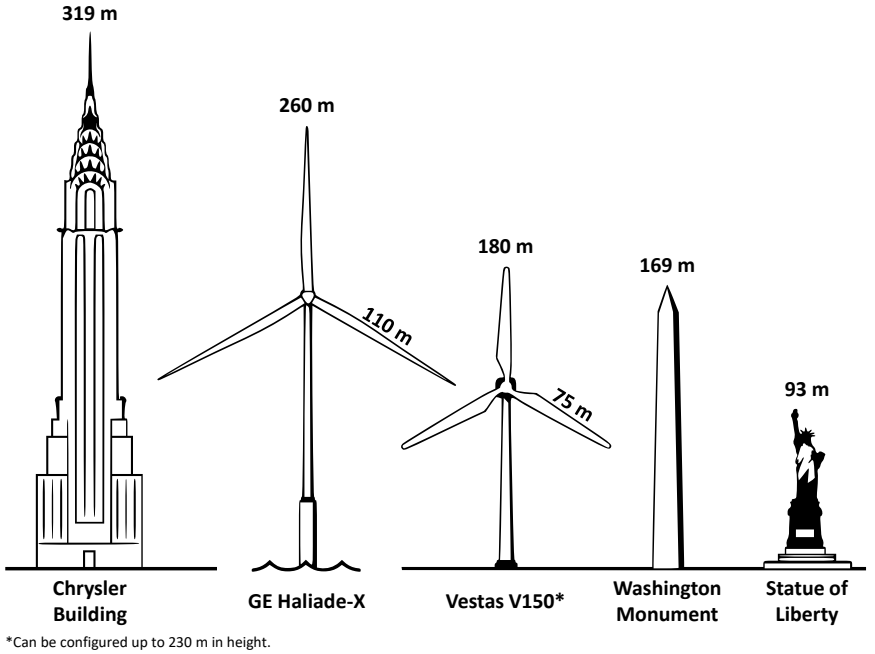


Figure 1.1 – Size comparison of some of the largest wind turbines and US monuments as of 2020. The GE Haliade-X is an offshore wind turbine prototype, while the Vestas V150 is a commercially available onshore wind turbine. Sources: Vestas, GE and Wikipedia. Graphic based on illustration by Bloomberg.

1.1 Quality control of wind turbine blades

Modern wind turbines are huge. The latest off-shore wind turbine prototypes are well over 250 meters tall from base to blade tip, with a blade length of over 100 meters and a power rating of over 10 MW. These enormous structures (depicted in Figure 1.1) are expected to operate under harsh off-shore weather conditions for at least 20 years, with as little downtime as possible. Downtime and repairs are costly due to the poor accessibility of the off-shore locations, harsh conditions, and not least, loss of uptime. Catastrophic failures are extremely costly and it is critical for off-shore wind turbine manufacturers and their customers that such failures are avoided. Therefore, quality control (QC) during the manufacturing of wind turbines, and in particular the blades, is essential to the profitability of the wind energy industry. Of course, human safety is also a major concern in places where turbines are located in populated areas.

The blades of modern wind turbines operate near the physical limits of the structure and the materials they are made from. Unlike the steel towers, which are often designed with wide tolerances, the composite blades are generally designed with narrow

tolerances. This is necessary to meet the demands for high stiffness and strength at low weight. Many blade designs use a combination of glass and carbon fiber composites, which allows the blades to deal with the extreme forces applied to them, while remaining relatively light. However, performing QC on these complex composite structures is both difficult and costly.

In this thesis, I examine and present methods for improving QC of wind turbine blades at two different scales. At the macroscopic scale, I present and discuss a number of methods that are useful for computer-based *automated* inspection at a structural level. The goal is to reduce the cost of QC during blade manufacturing, by reducing the time spent on data evaluation. Furthermore, automated data evaluation can also provide much more consistent, and often more accurate, results than human *manual* evaluation. Even in cases, where fully automated data evaluation cannot match the quality of human evaluation yet, *assisted* evaluation, where automated and manual evaluation are combined, can provide faster and more accurate evaluation. At the microscopic scale, I present a computer-based method for fast and accurate analysis of fiber orientations in composite materials. The goal is to improve the quality of the fiber-reinforced composite materials used in blades, by giving researchers and manufactures a fast, efficient, and simple method for extracting important material properties.

1.1.1 Structural inspection with ultrasound

Ultrasonic testing (UT), is the preferred method for inspecting the blade structure not visible from the surface. To my knowledge, all large blade manufacturers use UT during manufacturing to verify the structural integrity of the load-bearing parts of the blades. Most people are probably familiar with ultrasound from medicine, in particular fetal ultrasound, which is the common way to monitor the baby's growth and development during pregnancy. However, ultrasound is also a popular method for nondestructive testing (NDT) of structures in general. The reasons are that it is safe, flexible, and relatively cheap, when compared to techniques such as X-ray tomography and magnetic resonance imaging.

Traditionally, UT has mostly been carried out using small hand-held instruments. However, the large area of the blades subject to inspection makes the use of hand-held instruments impractical. To overcome this, highly specialized automated ultrasonic scanning systems have been developed by companies, such as FORCE Technology. Using this equipment, the several hundred meters of load-bearing blade structure can be scanned in high resolution in less than two hours – a task that would take days, or even weeks, to carry out with traditional UT equipment. While special scanning equipment has sped up data acquisition many times, this is not the case for data evaluation. Although evaluation of ultrasound data has, in some cases, been helped by new ways of presenting the data, or better data quality, the task has remained manual. As a result, evaluation has become the bottleneck of the structural inspection of blades.

Reducing the evaluation time of the ultrasonic inspection data from blades has become a priority for the blade manufacturers. A reduction in evaluation time will not just reduce the man-hours spent and thereby the cost of the blade. It will also reduce the time a blade sits on the manufacturing floor occupying valuable space in the production line. From the scanning is finished to the evaluation of the acquired data is complete, the blade usually remains in the production line, as potential repairs must be made before the blade is painted, which is usually the next step. Due to the size of the blades, most factories only have a limited number of production lines, so having a blade occupying a production line for many hours, while no work is being done on the blade, is costly. However, moving the large blades temporarily is also not viable in most cases. Thus, it is common for manufacturers to employ dedicated teams of data evaluators, who work around the clock to keep the evaluation time down.

Evaluating the ultrasonic data from blades is a very complex task. The combination of composite materials and complex structures makes the interpretation of data nontrivial, even for trained human experts. To further complicate things, design specifications often require evaluators to measure structural components to an accuracy that is close to, and sometimes beyond what is possible given the quality of the data. Although it is generally assumed that manual evaluation finds most critical defects, the difficulty of the task, coupled with the large amounts of data, easily leads to inconsistent, and sometimes incorrect, evaluation.

1.1.2 Fiber characterization using X-ray tomography

While UT is the preferred method for QC of blade structures, X-ray microtomography provides an excellent way of imaging the microstructures in blade materials. Material properties, such as stiffness and compression strength, are highly dependent on the fiber microstructures. Since these properties are vital to the structural integrity of the blades, knowledge about the microstructures is important to avoid structural failures and for designing stronger and lighter blades. Thus, we can use X-ray microtomography for QC of small samples of the fiber-reinforced composite materials used in blades.

1.2 Research objective

In this thesis, I investigate image analysis methods for automated QC of wind turbine blades. The analysis is targeted full blade inspection of load-bearing structures and inspection of blade composite materials at a micrometer scale. The aim is to improve the accuracy of the QC and reduce the time spent on QC.

1.3 Project evolution

This project started as an investigation into image analysis and machine learning methods suitable for automated evaluation of ultrasound data from wind turbine blades. At the beginning, most of my work focused on machine learning methods, in particular deep learning. However, as the labels needed for machine learning methods could not be obtained, my work shifted towards a more traditional image analysis technique, namely *graph cut*-based optimization, which turned out to be well suited for blade data. This work led to the development of a new scalable approach for solving very large computer vision tasks (e.g., 3D image segmentation) using graph cut optimization. Later, my co-authors and I also developed parallel versions of some of the most popular graph cut algorithms, allowing computations to be sped up significantly. These contributions are presented in this thesis.

Along the way, I assisted other researchers with their studies of fiber composites used in blades, by segmenting fiber bundles in 3D images using our graph cut-based method. This lead to a close collaboration with material scientists, whom I further assisted by creating a GPU-based scalable implementation of the *structure tensor* algorithm. With this implementation, we examined the fiber orientation distributions in several fiber-reinforced composites used in the load-carrying structures of wind turbine blades. Originally, the orientations were meant to be used with our graph cut method for segmentation, but the information also turned out to be very valuable by itself. The results of this work are also presented in this thesis. Although the work on microstructures was not originally planned, it aligns well with the goal of improving QC on wind turbine blades through image analysis and artificial intelligence. It also shows that the methods we developed can be applied to different types of problems at different scales.

1.3.1 Choice of method

The change of focus from machine learning and deep learning methods was primarily fueled by a lack of suitable training data. Although I have access to terabytes of ultrasound data from blades, labels of the types needed for tasks such as image segmentation, do not exist. Other types of labels, such as measurements of structurally important features, have not been recorded and organized with machine learning in mind. Moreover, trying to learn measurements from image data could easily result in a black box model, which would be difficult to validate and implement in production. Meanwhile, as the process of capturing the data is the same for every blade, we have a lot of prior knowledge at our disposal. Graph cut-based methods turned out to be well suited for incorporating this prior knowledge, while also being both fast and robust enough to handle large noisy 3D images.

1.4 Included contributions

The contributions to QC of wind turbine blades presented in this thesis can be separated into two different categories.

1.4.1 Structural segmentation using graph cuts

The majority of the work presented in this thesis is related to image analysis methods suitable for structural QC of blades based on UT. The primary methodological contribution consists of three papers about graph cut-based optimization methods with a focus on image segmentation. All three papers build on the so-called maxflow/min-cut algorithms and quadratic pseudo-Boolean optimization (QPBO). Below is a brief description of the three papers:

Paper A: Sparse Layered Graphs for Multi-Object Segmentation In this paper, we present a novel method for constructing highly efficient graphs for multi-object segmentation. The method enables graph cut-based instance segmentation of large 3D datasets, which could not feasibly be solved with previous methods due to the scale of the tasks. The work was presented as a poster at the Conference on Computer Vision and Pattern Recognition (CVPR) 2020 and published in the conference proceedings.

Paper B: Comparing Serial and Parallel Min-Cut/Max-Flow Algorithms for Computer Vision In this paper, we compare state-of-the-art serial and parallel graph cut algorithms in terms of performance on a set of different computer vision tasks. The comparison includes our own implementations and optimized versions of several well-known algorithms. The work is in preparation and will be submitted to the journal IEEE Transactions on Image Processing (TIP).

Paper C: Faster Multi-Object Segmentation using Parallel Quadratic Pseudo-Boolean Optimization In this paper, we present the first *parallel* QPBO algorithm and show experimentally that it outperforms the original serial algorithm by more than an order of magnitude for large segmentation tasks on modern hardware. The work has been submitted to the IEEE/CVF International Conference on Computer Vision (ICCV) 2021.

1.4.2 Fiber orientation analysis with structure tensor

A part of the work presented in this thesis concerns QC of blade materials, specifically composite fiber-reinforced materials. The primary contribution is two papers about characterizing and estimating the individual fiber orientations in glass fiber and carbon fiber composites. This is done at a microscopic scale, using volumetric data acquired with X-ray computed tomography (CT). The two papers are:

Paper D: Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor In this paper, we apply structure tensor analysis to X-ray CT volumetric data of a fiber composite material made from so-called non-crimp glass fiber fabric. Using this method, we estimate fiber orientations, which are important for the stiffness of the materials. We also segment different fiber bundles in the stitched material and estimate their fraction of the total fiber volume. The work was presented at the 41st Risø International Symposium on Materials Science and published in the IOP Conference Series: Materials Science and Engineering.

Paper E: Quantifying effects of manufacturing methods on fiber orientation in unidirectional composites using structure tensor analysis In this paper, we perform structure tensor analysis on three different glass and carbon fiber reinforced composite materials used in wind turbine blades. Apart from quantifying the global orientation distributions, we also investigate local patterns in the fiber orientations, which depend on the manufacturing process. We show that these variations can be quantified using our structure tensor-based approach, which is based on our own high-performance GPU implementation for calculating the structure tensor and eigendecomposition, allowing the analysis to be performed in just a few minutes. This allows fast quantitative QC of the materials, providing a way for manufacturers to tune process parameters to minimize unwanted local variations in the fiber orientations. The paper has been submitted to the journal Composites Part A: Applied Science and Manufacturing.

CHAPTER 2

Background

In this chapter, I give a brief introduction to the glass and carbon fiber composites used in wind turbine blades, as well as two of the imaging techniques, X-ray tomography and ultrasound, used for inspection of these materials. The goal is to provide a basic understanding of fiber composite materials used in blades, as well as the benefits and challenges of using the two imaging techniques on composites. First, I briefly introduce the materials. Second, I cover the basic concepts of X-ray tomography, data acquisition and data quality. Third, I explain the general concepts of UT and its application to composite materials. Both X-rays and ultrasound can be used in many ways. Only the techniques relevant for the contributions included in this thesis are discussed.

This chapter primarily serves to motivate some of the methodological choices made in Chapters 3 and 4. Most details related to blade inspection, and how to detect and characterize specific types of defects, are omitted as the information is not needed to understand the academic contributions presented in this thesis.

2.1 Fiber composites in wind turbine blades

Composite materials are made from two or more different materials with different chemical or physical properties, often referred to as constituent materials. Fiber composites, also known as fiber-reinforced polymers (FRP), consist of a *reinforcement* fiber material and a *matrix*, usually a resin, keeping the fibers together. The two types of fiber composites relevant to the contributions presented in this thesis are glass and carbon fiber composites. Both glass and carbon fiber composites are used in wind turbine blades due to their high strength-to-weight ratio and high durability.

2.1.1 Glass fiber composites

Glass fiber-reinforced composites (fiberglass) are used extensively in wind turbine blades. The blade surface and large parts of the load-bearing structure are usually made from fiberglass. Glass fibers are thin soft strands of glass, which can be processed similarly to textiles, as shown in Figure 2.1. The glass fibers used in blades are usually stitched or woven into large mattes (fabrics), which are layered to form the shape of the blade (see Figure 2.2). The thickness of the fiberglass depends on

the number of stacked glass fiber layers and varies for different parts of the blade. Once the appropriate glass fiber fabrics have been placed in the blade form, along with other materials, resin is added and the blade is baked to solidify the resin.

The glass fiber fabrics used in blades vary depending on where in the blade they are used. Both the diameter of the fibers themselves and the stitching patterns of the fabrics are chosen to give the material the desired properties needed at the specific location on the blade. An example of a commonly used fabric is the unidirectional (UD) non-crimp fabric (NCF), made from a mixture of 17 µm and 9 µm diameter fibers, shown in Figure 2.1. This type of fabric is examined in both Paper D and Paper E.

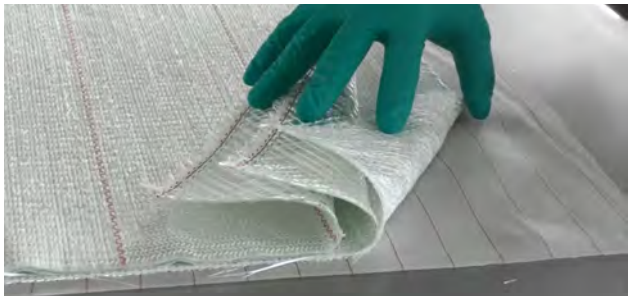
2.1.2 Carbon fiber composites

As wind turbine blades have increased in size, the need for stronger and lighter materials has led to the introduction of carbon fiber composites. The superior strength-to-weight ratio of carbon fiber composites, compared to fiberglass, makes them ideal for the load-bearing parts of the blade. Thus, despite the high cost of carbon fiber composites and the increased structural complexity, many, if not all, manufacturers of large off-shore blades have adopted carbon fiber composites in their designs.

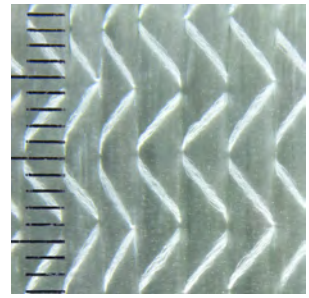
Unlike fiberglass, which is usually made during the manufacturing of the blade, the carbon fiber composites used in blades generally consist of premade beams, also known as "slabs". These are typically up to a couple of hundred millimeters wide and about 5 mm thick (see Figure 2.3). These carbon slabs are stacked to form large beams (see Figure 2.5) spanning the length of the blade. The stacks are known as pultrusion stacks, as the carbon slabs are usually manufactured using a manufacturing process known as pultrusion. The pultrusion process is ideal for creating long uniform slabs consisting of long unidirectional fibers held together by resin. Paper E examines two different types of carbon fiber composites used in blades, one of which is manufactured using pultrusion. The carbon fibers examined here have a diameter of about 7 µm, making them slightly thinner than the glass fibers described above.

2.2 Blade structure

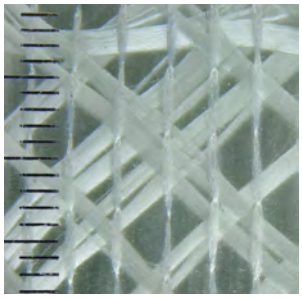
While the exact blade structure varies between blade models, most blades follow the same overall design principles. A schematic overview of a wind turbine blade is shown in Figure 2.4. Some of the most important parts of the blade are the *spar caps*, which are the primary load-bearing structures stretching from the root to the tip of the blade. The spar cap is the area of the shell, where the shear web is attached, and is usually reinforced with glass or carbon fiber composites ("laminate layer" in Figure 2.4). The blade shown in Figure 2.5 has a single shear web attached to the spar caps on either side. However, large blades often have two or three webs and four or six spar caps.



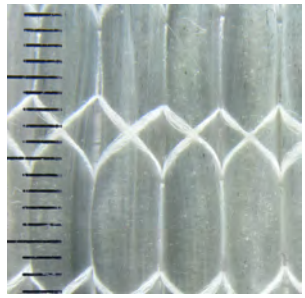
(a) Glass fiber fabric



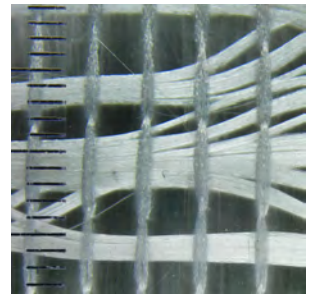
(b) Front side 0°



(c) Back side ±45°, 90°



(d) Front side 0°



(e) Back side ±80°

Figure 2.1 – Photos of unidirectional glass fiber fabrics used in wind turbine blades. The individual fibers are bundled and stitched together in layers to form the fabric. Photos by Lars P. Mikkelsen, DTU Wind Energy, 2018.

Figure 2.5 shows a cross-section of a blade near the tip. The leading edge of the blade is made from thick fiberglass to provide structural rigidity and allow it to withstand the pressure from the air. The spar caps are made from carbon fiber composites and glued to the shear web. This structure makes the blade strong and extremely stiff in the spanwise direction. The remaining part of the shell consists of a light filler material, such as foam or balsa wood, wrapped in a thin layer of fiberglass. On the outside of the blade, the fiberglass is coated and painted to make it smooth and protect it from the elements.

2.2.1 Spar cap defects

For larger blades, pultruded carbon stacks are often used in the spar cap due to their superior material properties compared to fiberglass. However, in many smaller blade models or older designs, the spar cap is made entirely from fiberglass. In either case, since this is the structurally most critical part of the blade, the spar cap and web adhesive region is subject to thorough ultrasound inspection during manufacturing. Although other parts of the blade may also be subject to UT, the vast majority of

Figure 2.2 – Blade technicians at Vestas Wind Systems manufacturing plant in Windsor lay fiber material in a shell of a wind turbine blade. Photo by BizWest/Joel Blocker, 2015. Published by BizWest, June 30 2017.

(a) **(b)**

Figure 2.3 – (a) Pultruded carbon during production. Photo by Zoltek, published by CompositesWorld, March 27 2018. (b) Carbon pultrusions manufactured by Fiberline Composites of the type used in wind turbine blades. Photo published by Environmental XPRT.

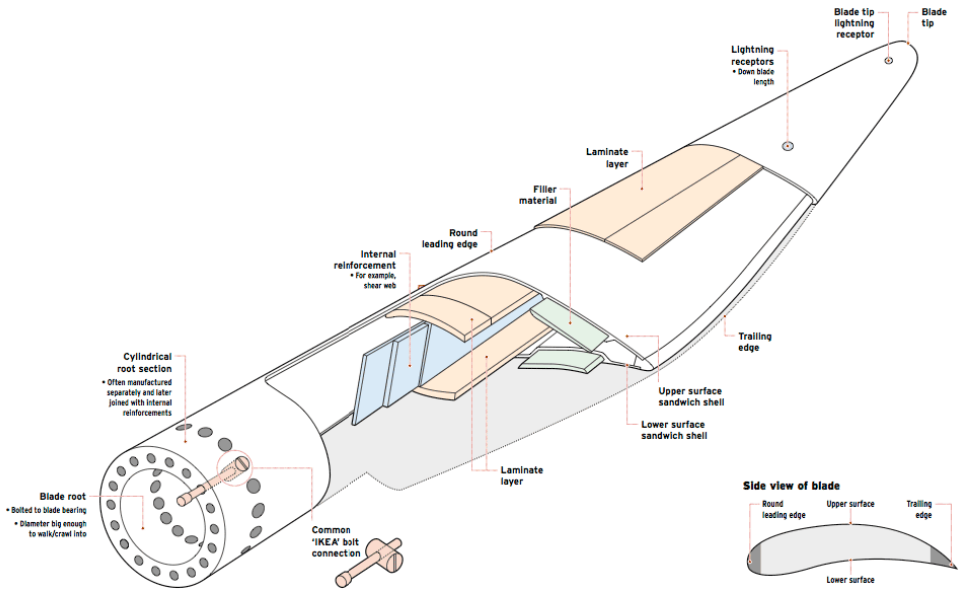


Figure 2.4 – Schematic overview of blade structure. The laminate (FRP) layers connected by the shear web are the primary load-bearing parts of the blade, known as the spar caps. Published by Windpower Monthly, 1 July 2012. Based on illustration by Gurit.

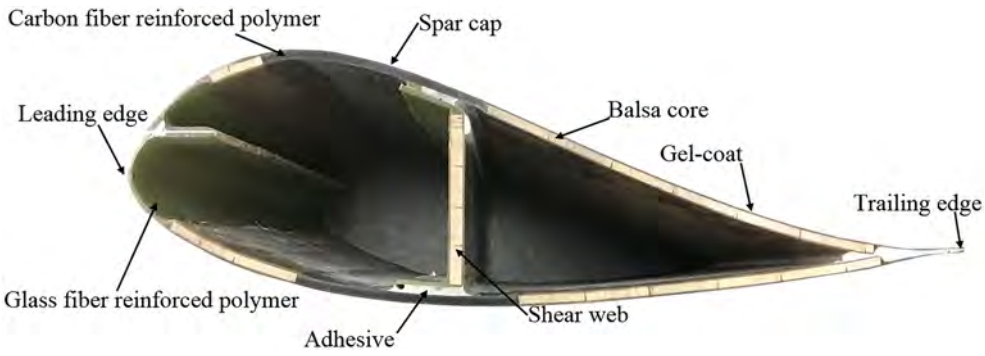


Figure 2.5 – Cross-section of a wind turbine blade near the tip. The different structural parts of the blade are highlighted. This blade uses carbon fiber composites for the spar cap. ©2018 Wiley. Used with permission from Martin et al. 2018.

the ultrasound data is collected from the spar cap and web region.

It is critical for the strength and stiffness of fiber-reinforced composites that all air between the fibers is replaced by resin. Air, or the lack of matrix material, is the primary cause of defects in fiber composites and thereby the primary cause of defects inside the spar cap. The related defect types include:

Porosity Microscopic air bubbles in the matrix material.

Voids Larger air bubbles in the matrix material.

Dry fibers Fibers (or parts of fibers) not in contact with matrix material.

Delamination Disbonds in the matrix material, often due to thin air pockets or fractures in the matrix.

All four defect types listed above may be critical, depending on the location and size of the defects. Because they are all variations of "lack of matrix material", distinguishing between the defect types can be difficult. A void may result in dry fibers or lead to a delamination. Thus, an observed defect may belong to several defect classes. However, in practice, human evaluators usually choose the defect type they think describes the defect most accurately. Due to overlapping defect definitions and imperfect data, the human evaluation is quite subjective. This means that both classification and measurements of defects may vary significantly depending on the person performing the evaluation.

Another defect type, which occurs more frequently in thicker fiberglass structures, is out-of-plane wrinkles. A wrinkle is essentially a fold in the fiber fabric, which may be a result of bad lay-up of the fabric or issues with the infusion process. The severity of the wrinkle is determined by its height, angle, and width. Wrinkles with high angles are often accompanied by pockets of air or resin, as shown in Figure 2.6.

The key to detecting these types of defects is being able to recognize structural features in the data and separate these from unexpected features, i.e. defects. Whether a significant feature in the ultrasound image is a defect depends very much on where in the structure the feature is located. In other words, to detect and characterize defects, we must be able to separate them from structural features and determine their position in the blade structure. However, defects often obscure the structural features we rely on for positioning, which along with high levels of noise makes evaluation very dependent on both prior knowledge and contextual information.

2.2.2 Bond line defects

The adhesive (glue) between the inner surface of the spar cap and the shear web is critical to the structural integrity of the blade. As a result, blade manufacturers

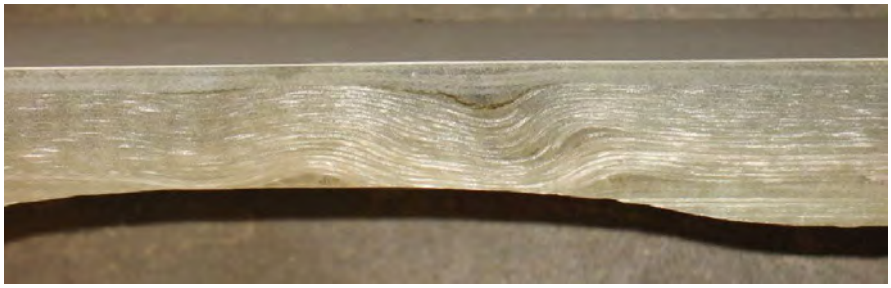


Figure 2.6 – Fiberglass cut-out with several out-of-plane wrinkles and pockets of resin. Photo by FORCE Technology.

inspect the glue area (bond line) looking for glue defects such as voids or delamination. Manufacturers often measure and record the glue width and glue thickness at intervals along the length of the blade to ensure they are within specification. Out-of-spec width or thickness is considered a defect and must, like other defects, be repaired before the blade can pass QC. Figure 3.16a illustrates how glue width and thickness may be measured on the cross section of the bond line.

Apart from inspection of the glue, bond line inspection may also include measuring the chordwise position of the web. If the web is not attached to the spar cap in the correct position, it could lead to a critical structural failure once significant load is applied to the blade.

The recording of measurements, such as glue width and thickness, are good examples of tasks, for which an automated method would be far superior to manual labor. Not only would it be feasible to record measurements at a much higher resolution, the consistency between the measurements should also be much better. Such measurements should be useful for blade designers and process managers to quantitatively compare blades produced using different methods, or for managers to compare blade quality over time at different factories.

2.3 X-ray computed tomography

X-ray computed tomography (CT) is a powerful nondestructive imaging technique widely used in science, medicine, and many other places. In tomography, X-rays are used to create a cross-section image of an object. The object is placed between the X-ray source and a detector/camera. X-rays penetrate the material but are absorbed according to the material density, so the image records the projected material density. For CT, the object that is being scanned is typically rotated relative to the source and detector to acquire multiple images taken at different angles, which are combined using a reconstruction algorithm to create a 3D image (volume) of the object. For laboratory X-ray CT scanners, this is typically achieved by placing the sample on a

rotating stand between the source and detector as shown in Figure 2.7. Volumetric CT data was used for experiments in all five papers presented in this thesis.

2.3.1 Micro-CT of composites

Micro-CT (µCT) is CT, where the image resolution (pixel size) is in the micrometer range. As will be shown in Chapter 4, µCT works well for acquiring high-resolution volumes of both glass and carbon fiber composites. Due to the difference in density between the fiber and matrix materials, modern laboratory µCT scanners can create 3D images, where individual fibers are clearly visible, which allow detailed analysis of the composite material's microstructure. However, this is a slow process, which can easily take days, even for samples only a few cubic centimeters in size.

2.4 Ultrasound

Ultrasound is sound at frequencies beyond what is audible to humans. This means that all sound from around 20 kHz and up is considered ultrasound. Apart from frequency, ultrasound is no different from regular audible sound in any physical way. It is simply mechanical pressure waves moving through matter at the speed of sound, which is approximately 343 m/s for air but much higher for solid materials such as steel (approx. 5960 m/s). The speed of sound in composites varies depending on the composition of the material, but for the composites used in blades the speed is often estimated at 2000 m/s to 3000 m/s for sound travelling perpendicular to the fiber

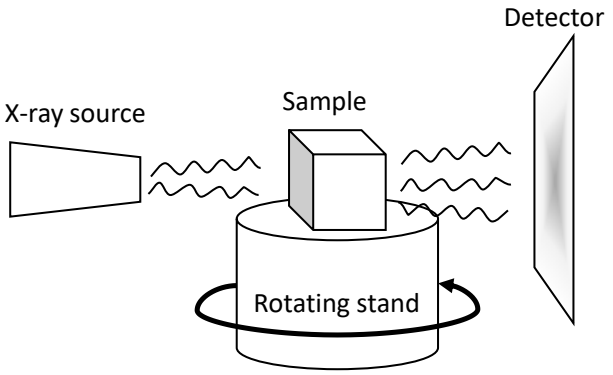


Figure 2.7 – Typical X-ray CT setup with a fixed X-ray source and detector on either side of a rotating stand.

orientation. It is the large difference between the speed of the sound in the air and in the composites that allows UT to detect air trapped inside the blade structure.

Similarly to X-ray tomography, ultrasound transmission tomography (UTT) can be used to measure the attenuation of an object by placing a transmitter and a receiver on either side of the object. However, the pulse-echo method, using a transceiver that both acts as a transmitter and receiver, is more commonly used. Pulse-echo ultrasound turns out to work well for detection of defects in many types of materials for a number of reasons.

The primary practical advantage of pulse-echo compared to tomography, is that it allows one-sided inspection. In many real-world scenarios, being able to image the internal structure of an object from one side is a huge advantage. This is especially true for large structures. Instead of measuring attenuation, pulse-echo tells us how much of the signal was reflected at a certain time. Thus, by measuring the time between the initial pulse and the reflected signal, and knowing the speed of sound in the material, it is possible to calculate the depth at which the signal was reflected.

Both sound and light behave like waves and follow the same physical laws of refraction and reflection. The reflected part of the wave is reflected at the angle of incidence, while the angle of the refracted wave can be calculated using Snell's Law. However, light moves much faster than sound and the X-rays have a much shorter wavelength than the ultrasound used for UT. And as noted above, the speed of sound is many times higher in composites than in air. As a result, X-rays practically pass straight through the composite blade structure including any of the potential defects described previously. Sound on the other hand, is reflected and refracted whenever it moves between materials with different sound velocities. How much of the signal is reflected versus refracted depends on material densities and differences in the sound velocities of the materials. Thus, interfaces between solid materials and air causes the majority of the signal to be reflected. This makes pulse-echo ultrasound well suited for locating imperfections, in particular air, inside solid structures.

2.4.1 UT of blade composites

As previously stated, UT is used to detect a variety of defects in wind turbine blades, particularly in the spar cap region. Figure 2.8 shows most of the defect types inspected for using pulse-echo ultrasound. This includes location and size estimation of delamination, dry areas, porosity, gelcoat disbonds, kissing bond, as well as measurement of shell thickness, glue thickness, and glue width. However, using ultrasound in composite materials is actually difficult for the same reason that ultrasound is effective for detecting certain defects.

The sound is reflected and refracted whenever it passes from one material to another, and because the blade composites consist of layers of resin, carbon fibers, and glass fibers, the sound is constantly reflected and refracted in different directions. This makes the reflected signal received by the probe incredibly noisy. To remove some of

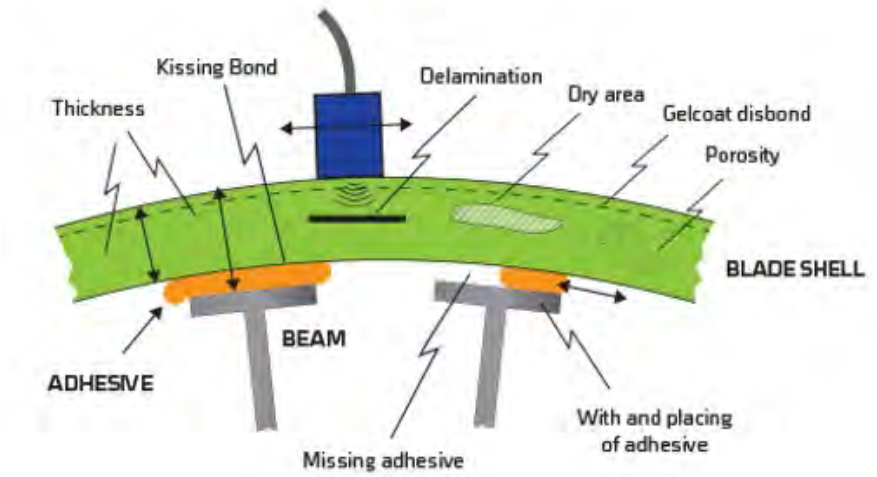


Figure 2.8 – Common defects in spar cap and bond line (web adhesive). One or more pulse-echo ultrasound probes (blue box) are moved across the surfaced transmitting ultrasound into the structure and listening for echoes. Illustration by FORCE Technology.

the noise, a band-pass filter around the probe frequency is used. Furthermore, because of the high attenuation of the composite materials and the thickness of the shell, it is necessary to use a relatively low probe frequency (0.5 MHz to 1 MHz). This puts a limit on the resolution in the direction of transmission, as the resolution depends on the signal frequency/wavelength.

Figure 2.9 shows an ultrasound image of the fiberglass sample from Figure 2.6. The type of image shown here is known as a B-scan, and consists of a series of A-scans stacked together and colored based on the amplitude. An A-scan is simply a series of sound/pressure wave amplitudes sampled over time. Figure 2.10 shows data from the same fiberglass sample, with a single A-scan plotted left of the B-scan. In this specific case, each A-scan contains 3702 amplitude values, sampled at 100 MHz. The A-scans are collected by moving the probe over the surface while tracking its position. Every time the probe has moved a certain distance it transmits a wavelet with a mean frequency of 0.5 MHz and then starts sampling at 100 MHz for a short amount of time (37.02 µs in this case).

The ultrasonic wavelet is created by applying a small electric current to the piezoelectric crystal in the probe. The reverse process is used for sampling, where the sound wave is converted to an electric current by the piezoelectric crystal. Thus, the raw sampling values are usually measured in mV. However, for easier interpretation, these values are usually converted to dB. Note that dB is a logarithmic scale, which

changes the shape of the signal. Figure 2.10a shows the full signal (with positive and negative values), while Figure 2.10a contains absolute values of the signal. It is common practice for human experts to use the absolute values for evaluation.

A common misconception in evaluation of ultrasound images of composites is that the horizontal lines in the B-scans (see Figure 2.10) correspond to glass or carbon fiber layers in the scanned composite. This is not the case. The pattern is actually the oscillation of the 0.5 MHz wavelet that was transmitted from our probe in the first place. Remember, what we are seeing are reflections of our transmitted signal.

2.4.1.1 Instantaneous amplitude

The A-scan oscillation form (phase) can be valuable for analysis, for instance, when looking for wrinkles. However, when measuring sizes or positions, it may complicate things. For example, imagine that we want to measure the thickness of the fiberglass from Figure 2.9. For simplicity, we assume a fixed sound velocity, which allows us to easily convert time to distance. Thus, all we need to do is to measure the time from the front surface reflection to the back surface reflection. We assume that the samples with the highest amplitudes are the ones closest to the surfaces. However, as the signal changes while passing through the composite material, it may become unclear which local maximum should be chosen. For instance, the back wall reflection for the A-scan in Figure 2.10b is somewhere around sample 2500, but it is not clear which of the three peaks is closer to the surface. In fact, the surface may reside between two of the peaks. A convenient way to overcome this issue is to calculate the instantaneous amplitude of the signal. This is done by using the Hilbert transform to determine the complex analytical signal, where the imaginary part of the signal corresponds to the phase and the real part is the instantaneous amplitude. Figure 2.11 shows the instantaneous amplitude for the data shown in Figure 2.10.

As shown in Figure 2.11, the instantaneous amplitude acts as an envelope around the absolute valued signal. In most cases, this simplifies the task of detecting surface positions based on the peak signal amplitude. From an image analysis point of view, the instantaneous amplitude data is much more approachable for surface detection as well as image segmentation, than the data that includes the phase as shown in Figure 2.10. For this reason, I use the instantaneous amplitude data for my work on automatic evaluation of blade structures.

The instantaneous amplitude data in Figure 2.11 contains some minor artifacts due to the data type used to store the data. It is likely that some high-frequency variations may be a result of this compression.

2.5 Summary

In this chapter, I have given a brief introduction to glass and carbon fiber-reinforced composites used in the spar caps of wind turbine blades. Furthermore, I have outlined

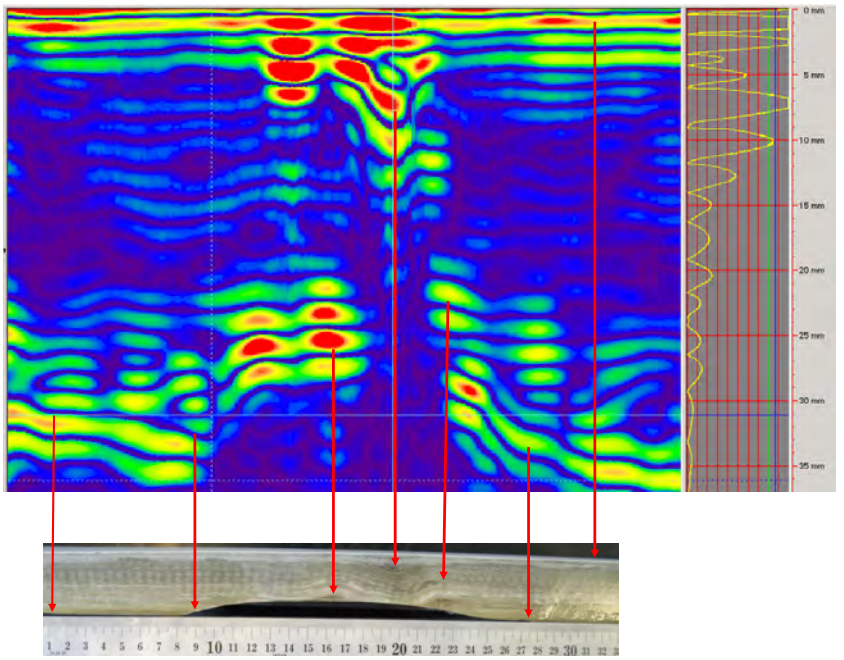
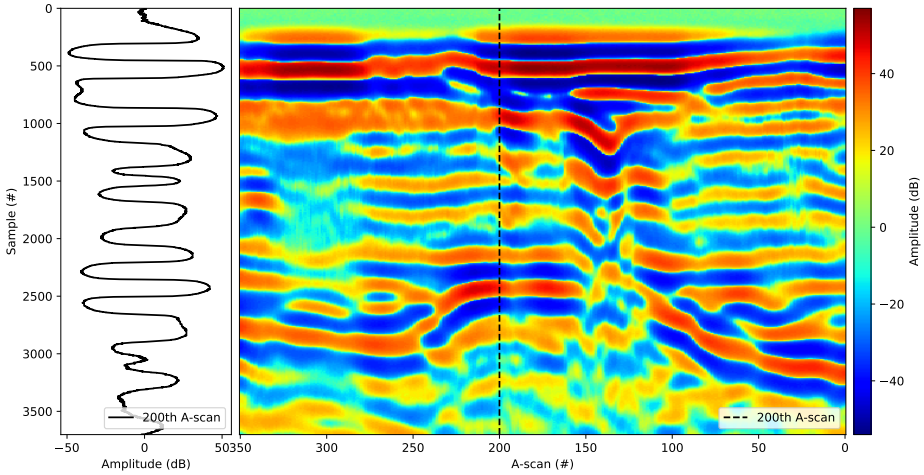


Figure 2.9 – Ultrasound B-scan of fiberglass containing multiple resin pockets and wrinkles. Markers highlight approximate positions of various prominent reflections. The top image is a screenshot from the P-scan 4 software. The signal shown is a rectified signal. Red indicates high amplitudes and blue low amplitudes. The curve to the right show the rectified A-scan at the cursor (dotted line) position. Data and photo by FORCE Technology.

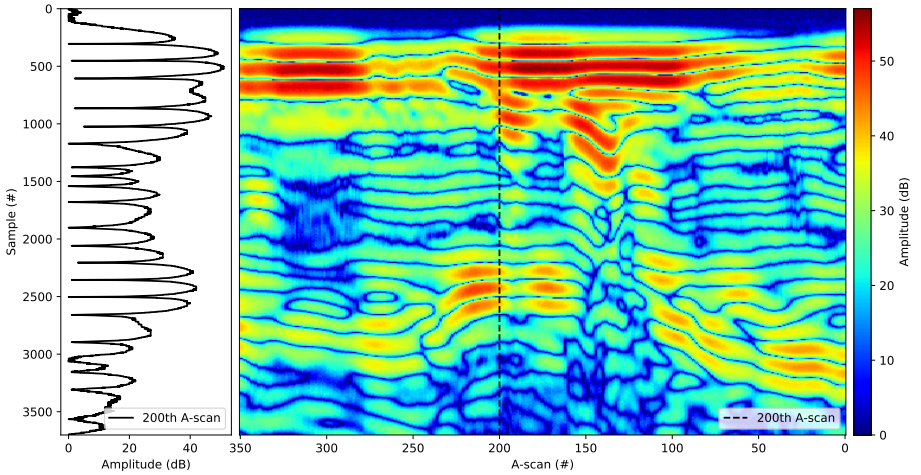
the structure of the spar caps, some of the defects commonly found within the spar cap region, and some of the challenges related to finding and characterizing these defects. Finally, I have covered the most fundamental aspects of the two imaging techniques used to acquire data in my work, namely X-ray CT and pulse-echo ultrasound.

The blade materials and structure play a significant role in the choice of the two imaging techniques presented in this chapter. X-ray CT is used for analysis of fiber composite microstructures due to its high data quality, allowing separate fibers to be clearly visible in the data, while ultrasound is used for inspection of blade composite structures due to its flexibility and sensitivity to changes in sound velocity.

The interpretation of X-ray CT data is fairly intuitive to most people as long as the densities of the imaged structures are different enough to allow good contrast. The fiber structures that we see in the datasets presented in Chapter 4 correspond to the actual fibers. Thus, we can estimate the actual fiber orientations by estimating the



(a) A- and B-scan



(b) A- and B-scan (absolute)

Figure 2.10 – Ultrasound data from the fiberglass sample shown in Figures 2.6 and 2.9. The B-scans (right) are made up of a series of A-scans (left) stacked together and colored using the amplitude.

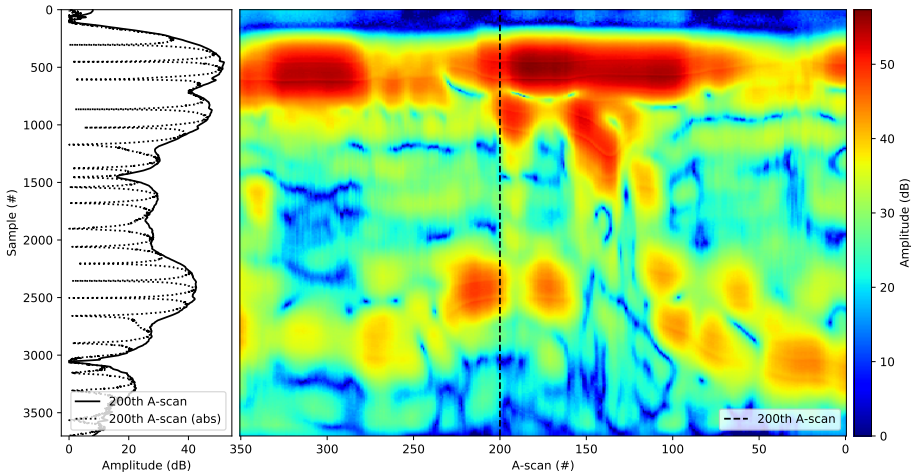


Figure 2.11 – Instantaneous amplitude B-scan for the data shown in Figure 2.10. The dotted line is the absolute valued A-scan signal shown in Figure 2.10b.

orientations of the fiber-like structures in the CT scans. This is inherently different from the pulse-echo ultrasound, used for blade inspection, where we are measuring reflections instead of attenuation. While transmission methods, such as X-ray CT can tell us something about the density of the materials based on attenuation, pulse-echo ultrasound signals carry information about changes in the materials. As a result, peaks in the signal amplitude appear at the interface between different materials, i.e. the material surfaces. Thus, measuring the blade structures often depends on accurate surface detection, which often relies heavily on prior knowledge due to the low signal to noise ratio of the data.

CHAPTER 3

Graph Cut Segmentation

In this chapter, I introduce the methods related to the research contributions of my Ph.D. project. Apart from introducing the methods, the goal of this chapter is to relate the contributions to each other and to quality control of wind turbine blades. I give an introduction to some popular *graph cut*-based optimization methods used in image analysis and discuss state-of-the-art within the field.

Afterwards, I present our three contributions. The first paper introduces a new method for creating efficient graph structures when solving multi-label segmentation tasks, while two other papers focus on parallel maxflow/mincut optimization algorithms. The papers included in this chapter do not discuss the application of the methods to wind turbine blade structures. The primary reason is data confidentiality, which means we are not able to share blade data publicly. However, in Section 3.8 I discuss the interpretation of ultrasound images of the blade spar cap and the motivation for using graph cut methods for analyzing the spar cap structure.

3.1 Maxflow/mincut algorithms

Maxflow/mincut optimization algorithms for s-t graph cut have long been popular in computer vision due to their excellent performance on many classical vision problems. This includes image restoration, stereo and motion, image synthesis, image segmentation, and more [Kolmogorov and Zabih 2004]. Common to all of these problems is that they can be solved by minimizing an energy function of the form

$$E(\mathbf{x}) = \sum_{p \in \mathcal{V}} \theta_p(x_p) + \sum_{p,q \in \mathcal{V}} \theta_{pq}(x_p, x_q) \ . \tag{3.1}$$

Here, $\mathcal{V}$ is a set of nodes, usually corresponding to the pixels in an image and $\mathbf{x} = \{x_p \in \{0,1\} \mid p \in \mathcal{V}\}$ is the binary labeling of the nodes. The energies θ_p and θ_{pq} usually encode data terms and interaction terms, respectively. In other words, minimizing E corresponds to assigning a binary label to each node, such that the sum of all energy terms is as small as possible. Using s-t graph cut algorithms, it

is possible to find a global optimal solution ($\mathbf{x}$ minimizing E) in polynomial time as long as E is submodular [Kolmogorov and Zabih 2004]. The energy function E is submodular when

$$\theta_{pq}(0,0) + \theta_{pq}(1,1) \leq \theta_{pq}(0,1) + \theta_{pq}(1,0) \tag{3.2}$$

for all pairs of nodes $p, q \in \mathcal{V}$. In Section 3.3, I introduce an algorithm for solving non-submodular problems. However, in this section, I will focus on submodular problems.

To minimize submodular energy functions using s-t graph cut, we must first construct a directed graph, $\mathcal{G} = (\{\mathcal{V}, s, t\}, \mathcal{E})$, with non-negative edge capacities, where $\mathcal{V}$ are the nodes and $\mathcal{E}$ are the graph edges. The set of nodes, $\mathcal{V}$, in the graph correspond exactly to the nodes in the energy function. However, the graph also has two special *terminal* nodes, s (source) and t (sink). The edges, $\mathcal{E}$, are then created from the energies as shown in Table 3.1.

Energy term	Corresponding edge	Edge capacity (cap)
$\theta_p(0)$	$(p \to t)$	$\theta_p(0)$
$\theta_p(1)$	$(s \to p)$	$\theta_p(1)$
$\theta_{pq}(0,1)$	$(p \to q)$	$\theta_{pq}(0,1)$
$\theta_{pq}(1,0)$	$(q \to p)$	$\theta_{pq}(1,0)$

Table 3.1 – Mapping from submodular energies to edges [Kolmogorov and Rother 2007].

Once the graph has been constructed, the optimal labeling can be found by solving a so-called s-t *mincut* problem: For $\mathcal{G}$, find a cut with a partition of the nodes $\{\mathcal{V}, s, t\}$ into two disjoint sets, S and T, where $s \in S$ and $t \in T$. This is a s-t-cut. The cost of the s-t-cut is the sum of the capacities of all edges from S to T

$$\text{cost}(S,T) = \sum_{u \in S, v \in T} \text{cap}(u,v) \ , \tag{3.3}$$

where $\text{cap}(u,v)$ is the capacity of the edge $(u \to v)$.

The partition with the smallest cost is the mincut. The labeling of the nodes is determined based on the partition, such that $x_p = 0 \ \forall \ p \in S$ and $x_q = 1 \ \forall \ q \in T$. The representation of the original energy function, E, as an s-t graph, $\mathcal{G}$, for which we can compute a minimum cut, is key to solving the problem fast. The reason for this is the Ford-Fulkerson theorem [Ford Jr and Fulkerson 1962], which states that finding the minimum cut is equivalent to computing the maximum flow from s to t. And there are a number of fast so-called *maxflow/mincut* algorithms for computing the maximum flow/minimum cut between two nodes in a graph.

Maxflow algorithms generally fall into one of three categories: augmenting path algorithms, push-relabel algorithms, or pseudoflow algorithms, with the latter being a combination of the first two. A few of the most notable algorithms are listed in Table 3.2. The Excess Incremental Breadth First Search (EIBFS) algorithm [Goldberg,

Hed, Kaplan, Kohli, et al. 2015] is considered state of the art, as it performs the best overall for a variety of different graph cut problems. However, for certain types of problems, the Hochbaum Pseudoflow (HPF) algorithm [Hochbaum 2008] performs significantly better. The Boykov-Kolmogorov (BK) algorithm [Boykov and Kolmogorov 2004] also remains popular due to its good performance and flexibility. There are different versions and implementations of all of these algorithms, in particular the BK algorithm. Benchmarks comparing popular maxflow algorithms can be found in Goldberg, Hed, Kaplan, Kohli, et al. 2015; Fishbain, Hochbaum, and Mueller 2016, and Paper B. It should be noted that no single algorithm performs the best for all problems. Their performance depends on the concrete energy function/graph, as well as implementation details such as data structures, compiler options, and even system details such as CPU cache size, memory bandwidth, etc.

Algorithm	Type	Reference
Ford-Fulkerson	Aug. path	Ford Jr and Fulkerson 1962
Dinic's	Aug. path	Dinic 1970
Push-relabel	Push-relabel	Goldberg and Tarjan 1988
BK	Aug. path	Boykov and Kolmogorov 2004
HPF	Psudoflow	Hochbaum 2008
IBFS	Push-relabel	Goldberg, Hed, Kaplan, Tarjan, et al. 2011
EIBFS	Psudoflow	Goldberg, Hed, Kaplan, Kohli, et al. 2015

Table 3.2 – A few notable maxflow algorithms.

It is important to understand that all these maxflow algorithms are interchangeable in the sense that they all find the global optimal solution. Although, if there are several global optimal solutions, the results may vary depending on implementation details. The fact that these algorithms are guaranteed to find a global optimal solution, even for large complex problems, is really quite impressive. Many, if not most, algorithms used in image analysis and computer vision operate locally (e.g., using kernels) and/or are limited to finding good locally optimal solutions (e.g., gradient descent). However, with maxflow algorithms, we can formulate and solve problems with complex non-local interactions in reasonable time with guaranteed optimality, as long as the energy function is submodular.

In summary, many vision tasks can be formulated as submodular binary optimization problems, which can be solved optimally, and usually fast, using maxflow/mincut algorithms. However, as we will see, by reformulating the problem slightly, it becomes possible to formulate, and in many cases solve, non-submodular problems as well. For a more in-depth explanation of the use of maxflow algorithms for image segmentation, I recommend the paper by Boykov and Funka-Lea 2006. The article contains good explanations of the theory, discusses how to formulate energies, and contains examples of image segmentation in both 2D and 3D using graph cuts.

3.2 Graph structures

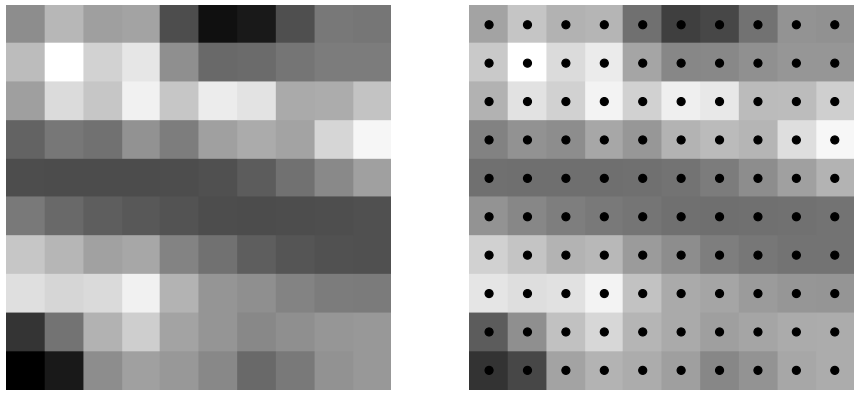
In this section, I will discuss a few common graph structures used when formulating energies for image segmentation. I will not discuss the formulation of energies based on actual data (e.g., pixel values), as the exact formulation will usually depend on the task at hand. Boykov and Funka-Lea 2006 offer some suggestions regarding the formulation of energies for image segmentation for those interested.

3.2.1 Grid graph

The most common way to structure the energies/graph is creating a node for each image pixel (or volume voxel) as shown in Figure 3.1. Then unary energies (terminal edges) are added as data terms, as shown in Figure 3.2, and pairwise terms (non-terminal edges) are added between neighboring pixel nodes as shown in Figure 3.3a. Typically a four or eight-neighborhood structure is used in 2D, and a six or 26-neighborhood structure is used in 3D. Figure 3.3b shows the segmentation found by solving the maxflow/mincut. Here, the graph has been cut into five regions marked by orange lines. The cut non-terminal edges are marked with orange and the cut terminal edges are marked with orange borders on their respective nodes.

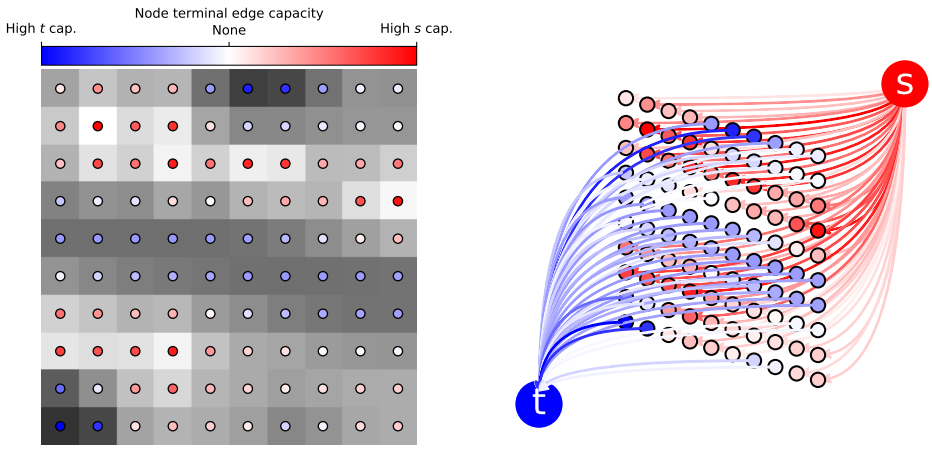
This grid structure is very popular, likely due to its simplicity, which has resulted in a number of specialized implementations of the BK algorithm (e.g., Grid-Cut by Jamriska, Sykora, and Hornung 2012) for cutting graphs with a fixed neighborhood or a fixed neighborhood size. For instance, in a fixed four-neighborhood structure, each node would contain an array of four edge capacities, one for each of the four neighbors in the grid. This is very efficient, as we do not have to store information about which node the edge is pointing to. Furthermore, because the information is stored in the node structure, performance is often improved due to improved cache efficiency in the hardware. Similarly, fixed size neighborhoods allow us to store edge information inside the node structures for better cache efficiency.

While a fixed neighborhood allows for more efficient implementation, as shown with both Grid-Cut and UBK by Goldberg, Hed, Kaplan, Tarjan, et al. 2011, it drastically limits the flexibility of the implementation, as only graphs with a specific predetermined graph structure can be cut. A hybrid approach, which has a small fixed size neighborhood, but allows further edges to be added dynamically, could perhaps provide the best of both worlds. However, to my knowledge, this approach has not yet been attempted. From an algorithmic point of view, implementation details, such as how edges are stored, may seem unimportant. Nevertheless, given the relatively small difference in performance between the fastest maxflow algorithms [Goldberg, Hed, Kaplan, Kohli, et al. 2015], implementation details often have a significant impact.



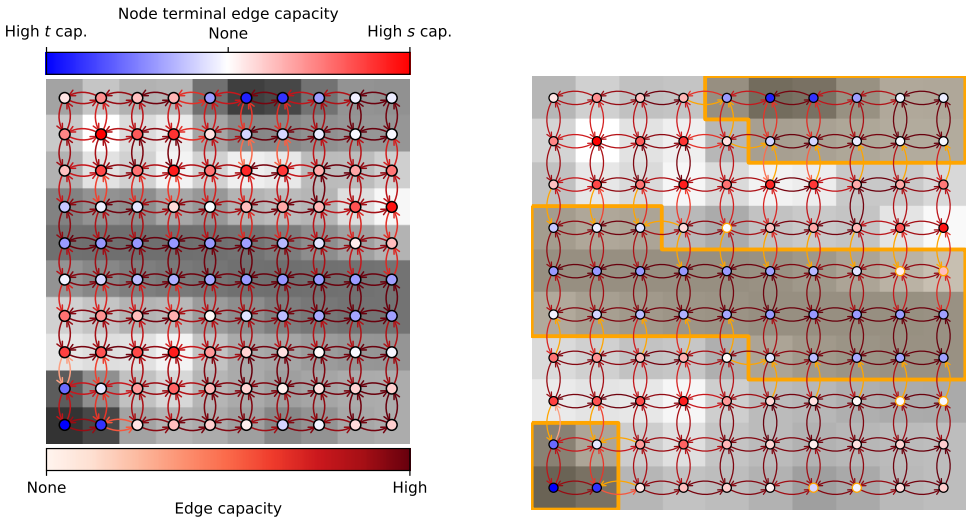
(a) Image **(b)** Image with nodes

Figure 3.1 – Small sample image with one node per pixel.



(a) Terminal capacities **(b)** Terminal edges

Figure 3.2 – (a) Sample image with nodes, where the node color indicates the terminal edge capacity for the given nodes. As shown in (b) the red edges go from the source to the nodes, while blue edges go from the nodes to the sink. In this example, the terminal capacities depend on the pixel intensities.



(a) Grid edges **(b)** Cut/segmentation

Figure 3.3 – (a) 4-neighborhood grid graph. Here, the terminal edge capacities depend on the pixel intensities and the non-terminal edge capacities depend on the difference in pixel intensities. (b) The mincut segmentation result. The orange lines show the segmentation boundary/cut separating the nodes connected to s and t. The orange edges are non-terminal edges that were cut and the nodes with orange border mark terminal edges that were cut.

3.2.2 Multi-label graph

Being able to solve binary optimization problems, such as binary segmentation is very useful. However, many segmentation tasks are not binary but rather multi-label/multi-object problems. I will use multi-label and multi-object segmentation interchangeably, depending on the context, but in practice, it is the same thing. One well-known approach for solving multi-label problems using a binary labeling is α-expansion [Boykov, Veksler, and Zabih 2001], which iteratively computes the maxflow using different energies for each label. However, this approach does not guarantee an optimal solution and often gets stuck in bad local minima [Isack et al. 2017].

Another approach is to construct a layered graph using the approach by Ishikawa 2003. In its simplest form, the Ishikawa layered graph is created by replicating all graph nodes for each label. Thus, if we want to segment an image with two different labels (plus a background label), instead of creating one node for each pixel as shown in Figure 3.1b, we create a graph with two layers, each containing one node for each pixel as shown in Figure 3.4a. For each layer, the binary node labeling will indicate whether the node, and thereby the corresponding pixel, belongs to the layer label or not. If the reader is not familiar with the Ishikawa technique for multi-label segmentation,

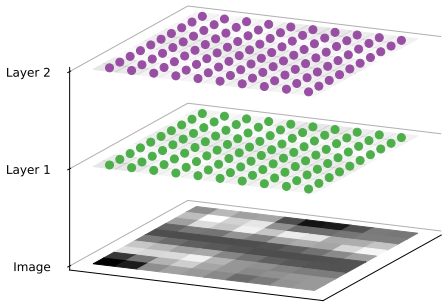
I recommend the paper by Delong and Boykov 2008. It does an excellent job of explaining how to use the Ishikawa layers for multi-label/multi-object segmentation on grid graphs with so-called interaction constraints.

Interactions are represented by pairwise energy terms between nodes in different layers and are essential to the Ishikawa approach. In Figure 3.4 we create *hard containment* interaction constraints between the two layers, by adding infinite capacity edges, corresponding to $\theta_{pq}(0,1) = \infty$, between nodes in layer 1 (L_1) and layer 2 (L_2). Figure 3.4d shows a small region of the image and nodes, where five containment edges are highlighted. Other edges are opaque to avoid clutter. The five edges enforce containment with a margin of one. This means that node $q \in L_2$ can be labeled $x_q = 1$ only if all five nodes $p \in L_1$ are also labeled $x_p = 1$. Effectively, this enforces the segmentation of L_2 to be "inside" that of L_1, with a margin of one pixel as shown in Figure 3.4c-f. The margin can be increased by adding more terms/edges and reduced to zero by using only a single term/edge. As discussed by Delong and Boykov 2008, *hard exclusion* can be enforced in a similar way. However, exclusion between more than two layers cannot be formulated using submodular energies.

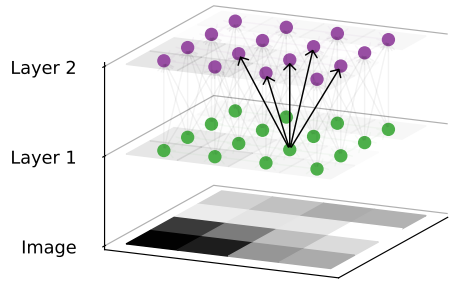
The primary advantage of the Ishikawa technique, compared to other multi-label approaches, such as α-expansion and HINTS [Isack et al. 2017], is that it keeps the guarantee of optimality. Another advantage is that the solution is found using a single cut. This also means that any runtime properties of the maxflow algorithm used still apply. In contrast, this is often not the case for iterative approaches, which may have to cut the graph many times before converging to a solution.

The most important downside to the Ishikawa technique is arguably the growth in the size of the graph, which affects runtime and, perhaps more importantly, the memory footprint. While this is less of a concern than it was a decade or two ago, due to the increase in system memory and processing power, it is still a concern for 3D segmentation tasks, and even 2D segmentation with many labels. In Paper A, I will introduce a method for overcoming this issue. Another limitation of the Ishikawa technique, as well as α-expansion and HINTS, is that the number of labels must be known before segmenting the image. Lastly, we are still limited to submodular energies, which means we can use containment/inclusion interactions, but we cannot enforce mutual exclusion interactions between more than two labels/objects.

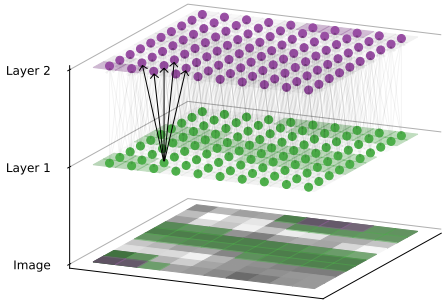
These geometric interactions are extremely powerful tools for multi-object segmentation, but without being able to exclude multiple objects, we have no way of guaranteeing non-overlapping segments. I will return to this issue in Section 3.3, but before that, I will move on to a special type of graph structure known as ordered multi-column graph. As it turns out, this graph structure applies containment in a slightly different way than grid-based containment. It could even be argued that the entire structure is based on geometric containment.



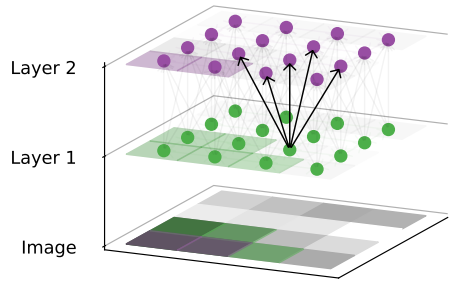
(a) Ishikawa layers



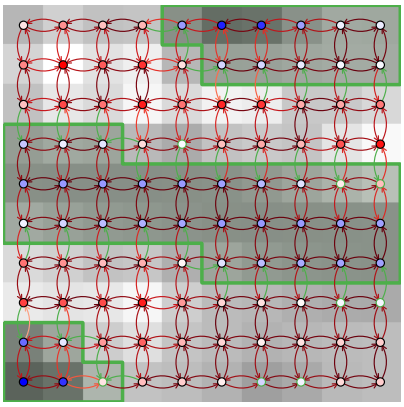
(b) Containment edges (zoomed)



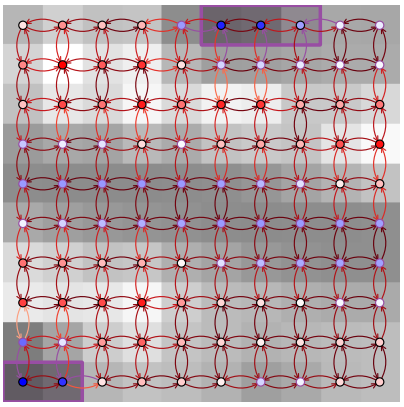
(c) Segmentation



(d) Segmentation (zoomed)



(e) Layer 1



(f) Layer 2

Figure 3.4 – Ishikawa graph with two layers used to segment two interacting labels/objects (green and purple). Each layer is a 4-neighborhood grid subgraph with one node per image pixel. The layers are connected with infinite capacity edges (black), which enforce containment with a margin of one. (b) and (d) show the lower left corner of the image and graph only.

3.2.3 Ordered multi-column graph

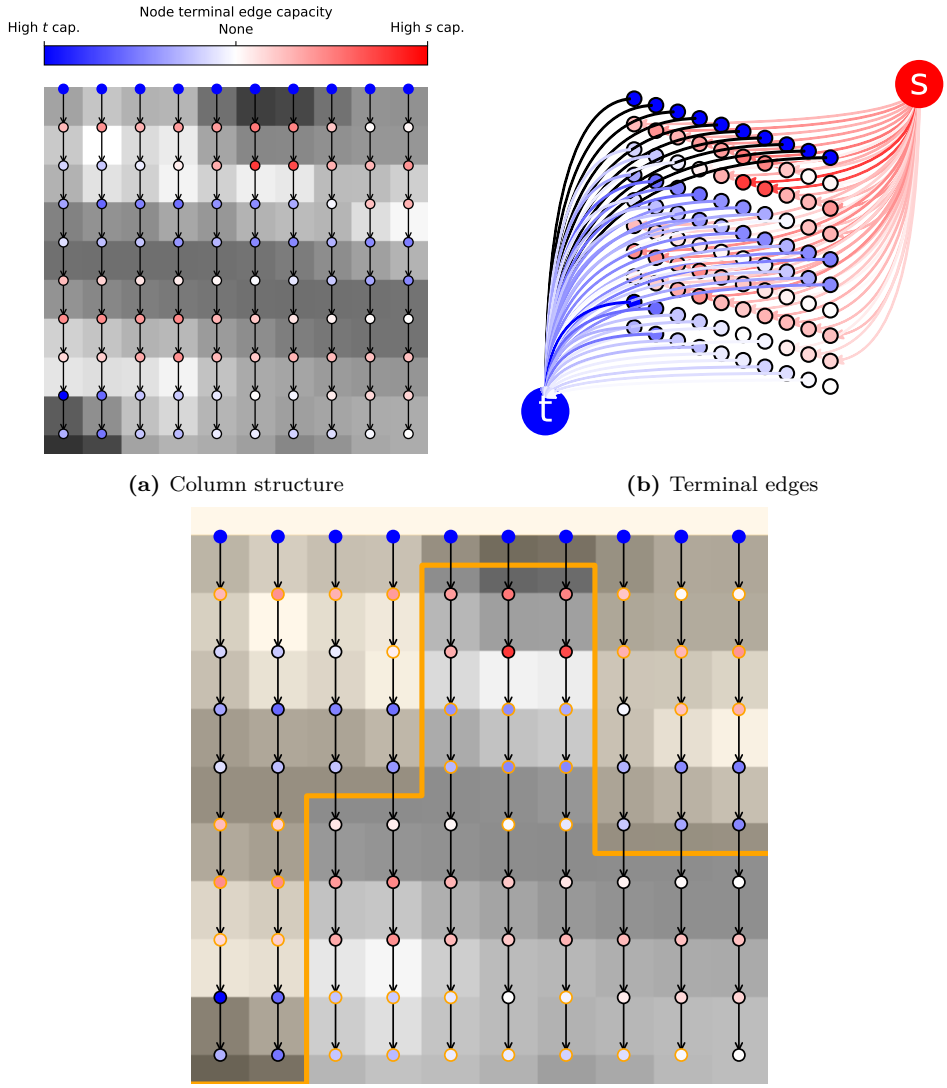
The *proper ordered multi-column graph* [Wu and Chen 2002] is a special graph structure, popularized by Kang Li et al. 2006, who demonstrated its usefulness for surface detection, as well as object segmentation, in both 2D and 3D. If the reader is not familiar with Li's paper, I encourage the reader to have a look at it. The structure used by Li relies heavily on hard containment with infinite energies, corresponding to infinite edge capacities in the graph.

For surface detection, the simplest approach is to start by creating one node per pixel, just as we did with the grid graphs. In this case, the nodes inherit the row and column neighborhood structure from the image pixels. The ordered columns are created by adding an edge corresponding to $\theta_{pq}(0,1) = \infty$ between each pair of nodes p and q, where p is "above" q in the same column (see Figure 3.5a). This hard constraint means that we can never have a node, p, with label $x_p = 1$ below a node, q, with label $x_q = 0$. As a result, there can be at most one change in labeling of the column nodes, which can occur only if $x_p = 1$ and $x_q = 0$ where p is above q in the column. This change in labeling corresponds to the position of the surface marked with the orange line in Figure 3.5c.

As shown in Figure 3.5, instead of having a one-to-one correspondence between pixels and nodes, on the vertical axis we instead position the nodes on the pixel borders. This is mostly a matter of interpretation, and is done because unary energies are formulated using the difference in pixel intensity along the vertical axis. Positive differences result in (red) edges from the source and negative differences result in (blue) edges to the sink. A top row is added to keep the original shape of the data. All nodes in the top row are connected with infinite capacity sink edges as shown in Figure 3.5b. The result of this energy formulation is that the surface will pass through the pixels with the lowest intensity. Clearly, this formulation by itself is not particularly useful as finding the pixel with the lowest value in each column is trivial. However, when combined with smoothness and interaction constraints, this formulation of unary energy terms is useful for many segmentation tasks.

To ensure smoothness, we may add $\theta_{pq}(0,1) = \infty$ terms between nodes in different columns, where the smoothness is determined by the integer value. Li refers to this value as Δx. In Figure 3.6, the hard smoothness constraints have been added with $\Delta \mathrm{x} = 2$. As a result, the segmented surface can move no more than two nodes/pixels up and down between adjacent columns. This completely changes the optimal solution and thus the segmentation result, as the segmentation is now required to be relatively smooth along the horizontal axis. As shown in Figure 3.6b, the new optimal surface is indeed smooth, compared to the unconstrained surface from Figure 3.5c.

For detection of multiple surfaces, Li also uses the Ishikawa layered technique. Li shows how geometric containment constraints can be efficiently applied to enforce both minimum and maximum margins between surfaces, even allowing negative margins. Figure 3.7 shows two interacting surfaces and Figure 3.5b shows the interactions

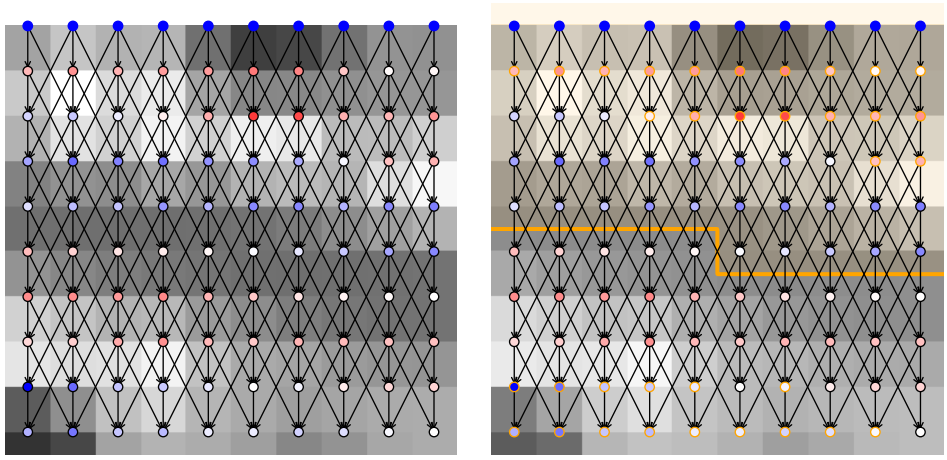


(a) Column structure

(b) Terminal edges

(c) Segmentation

Figure 3.5 – Column graph structure with infinite capacity (black) between column nodes. Terminal edge capacities are based on the difference in pixel values along the vertical axis. The top row of nodes are connected with infinite capacity edges to the sink. The orange line indicates the detected surface and nodes with orange border indicates nodes where terminal edges were cut.

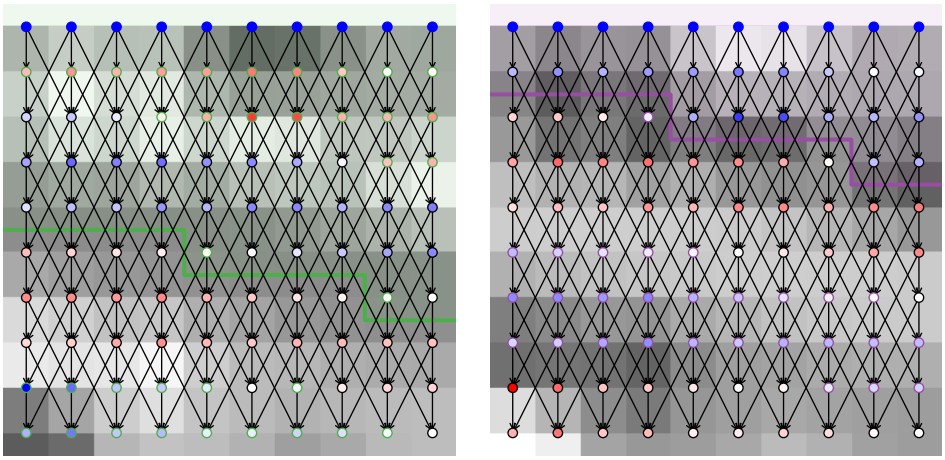


(a) Column structure with smoothness **(b)** Segmentation

Figure 3.6 – Column graph with smoothness constraints, $\Delta x = 2$. These constraints are enforced with infinite capacity edges between nodes in different columns. The constraint means that the optimal surface cannot move more than two nodes up or down between adjacent columns, thus resulting in a smooth surface.

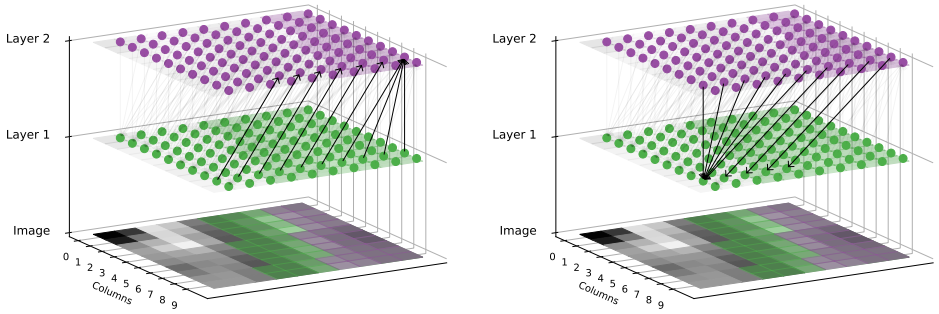
between these surfaces. The inner surface (purple) is forced inside (above) the outer (green) surface with a minimum margin of three and a maximum margin of five. The smoothness of both surfaces is two. Note that the values of the inner surface unary data terms have been negated, so that the surface it tracks is bright in the original image.

The principles behind the interacting surfaces, used by Kang Li et al. 2006 for ordered multi-column graphs, are fundamentally the same as for interacting objects, as used by Delong and Boykov 2008. In both cases the $\theta_{pq}(0,1) = \infty$ containment term is used to restrict the interacting Ishikawa layers. However, the maximum and negative margin constraints can only be applied to the ordered multi-column structure used by Li, not the grid structure used by Delong. Furthermore, for minimum margins larger than zero, the column structure allows much more efficient enforcement of constraints, as fewer terms/edges are needed than for the grid structure. With Li's approach, only a single edge per node is needed to enforce containment between two layers no matter the size of the margin. Figure 3.8a shows edges enforcing a minimum margin of three between the two surfaces using a single edge per node in layer 1. Meanwhile, for the grid structure, the number of edges required for containment depends polynomially on the size of the margin. For instance for a margin of size one, five edges are needed per node as shown in Figure 3.4d.



(a) Outer surface **(b)** Inner surface

Figure 3.7 – Cut layered ordered multi-column graph with surface smoothness. The surface smoothness is two. As for previous figures, node colors indicate s, t-edge capacities. The colored lines indicate the surface, and the node borders of similar color indicate cut terminal edges. The unary terms for the inner surface are negated, so that the surface tracks bright pixels in the original image. For details in label-interactions see Figure 3.8.



(a) Minimum margin edges **(b)** Maximum margin edges

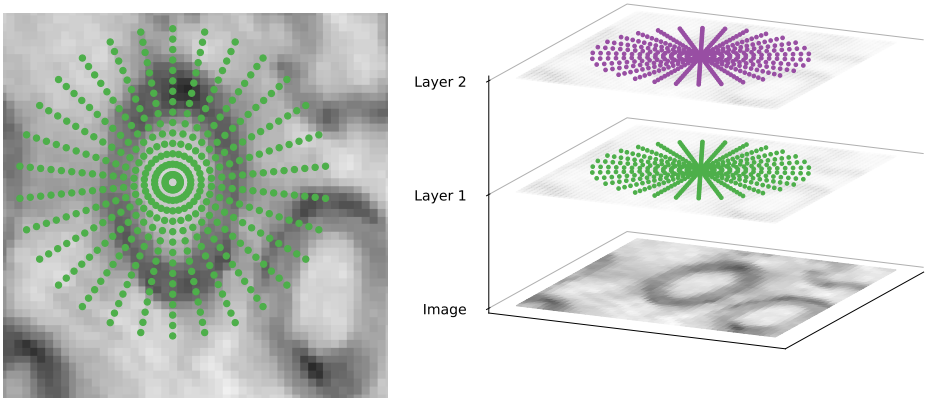
Figure 3.8 – Cut layered ordered multi-column graph with surface smoothness. Edges between two layers are infinite capacity interaction constraints, forcing layer 2 to be inside/above layer one. In (a) edges enforcing a minimum margin for one column are highlighted, while (b) highlights edges enforcing a maximum margin of five.

3.2.4 Object segmentation with ordered multi-column graphs

As shown by Kang Li et al. 2006, the ordered multi-column structure is useful, not just for surface detection, but also for object segmentation. For object segmentation, the approach relies on resampling the data, e.g., using radial sampling, to "unfold" the object. First, we sample radially around some known position inside the object we want to segment as shown in Figure 3.9a. In this case we would like to segment both the bright inner part of the object and the dark ring, so we create two layers of nodes as shown in Figure 3.9b. Now that we have the sample positions, we sample the image intensities at the positions, as shown in Figure 3.10a, which we can then unfold to a new images as shown in Figure 3.10b.

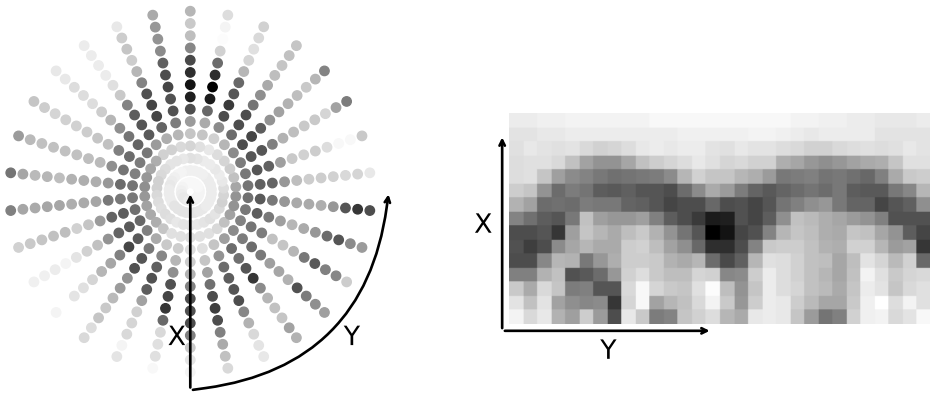
In this concrete example, we want to detect the edge of the dark ring rather than the ring itself. Thus, instead of using the difference along the vertical (X-)axis in the unfolded image for unary energy terms, as we did for the previous example, we use the second-order difference. Specifically, we use the negated second-order difference for the outer surface (layer 1) and the regular second-order difference for the inner object (layer 2). Just as before, we construct an ordered column graph with smoothness constraints in each of the two layers (see Figure 3.11). The only difference is that we also add smoothness constraints between the two outermost as we want the object to be smooth all the way around. Another, in my opinion better, way of illustrating these graphs is to draw the nodes at their original sample positions, rather than the unfolded image pixel positions. The graphs in Figure 3.12 and 3.13 are identical to those in Figure 3.11, except the nodes are drawn at the sampling positions rather than the unfolded image pixel positions. As shown in Figure 3.14, this simple approach allows accurate detection of object surfaces, which can either be interpreted as mesh vertices or converted to a segmentation of the pixels in the original image.

As long as we have the information needed to create an ordered multi-column graph along the surface(s) of the object we want to segment, the approach used by Li is very useful. The ability to enforce hard constraints on the problem, while still being able to compute the optimal solution fast makes it possible to incorporate prior knowledge very effectively. One thing, not touched upon by Kang Li et al. 2006, is the use of exclusion constraints between non-overlapping objects. This interaction constraint is extremely useful for segmentation of non-overlapping objects. For instance, we know the rings in Figure 3.9a cannot overlap. So, if we wanted to segment several of them, it would be useful to constrain the results to avoid overlapping outer surfaces. However, as noted by Delong and Boykov 2008, exclusion between more than two objects leads to non-submodular energies that we cannot represent using the mapping from energies to edges shown in Table 3.1. One way around this limitation is using so-called quadratic pseudo-Boolean optimization (QPBO).



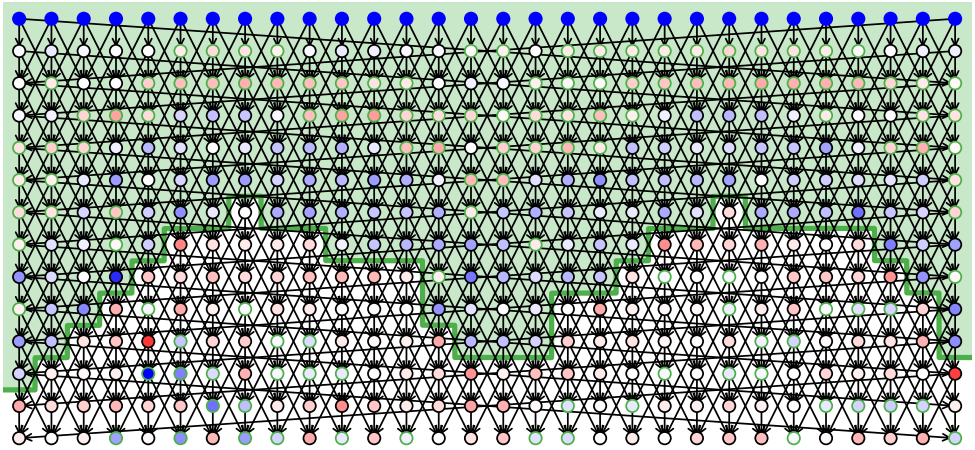
(a) Sampling positions **(b)** Two layers

Figure 3.9 – Radial sampling positions. The same positions are used for both layers.

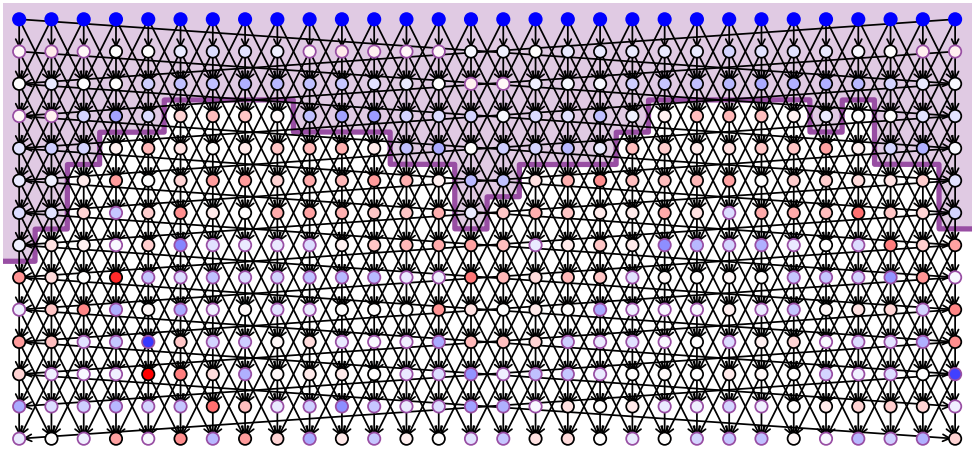


(a) Sampled intensities **(b)** Unfolded samples

Figure 3.10 – Sampled values from the image at their sampled positions and unfolded to a new image. The unfolded dark ring is clearly visible in the unfolded image.



(a) Outer object surface (unfolded)



(b) Inner object surface (unfolded)

Figure 3.11 – Cut layered ordered multi-column graph with surface smoothness for outer and inner object, respectively. Node colors indicate terminal edge capacities and node border color (green or purple) indicate cut terminal edges. Black edges have infinite capacity. The green and purple lines indicate the segmented surfaces of the two objects. Interacting edges between outer and inner object are not shown, but use the same structure as that of Figure 3.8, but with a minimum margin of three and maximum margin of six. The graphs are identical to those in Figure 3.12 and 3.13.

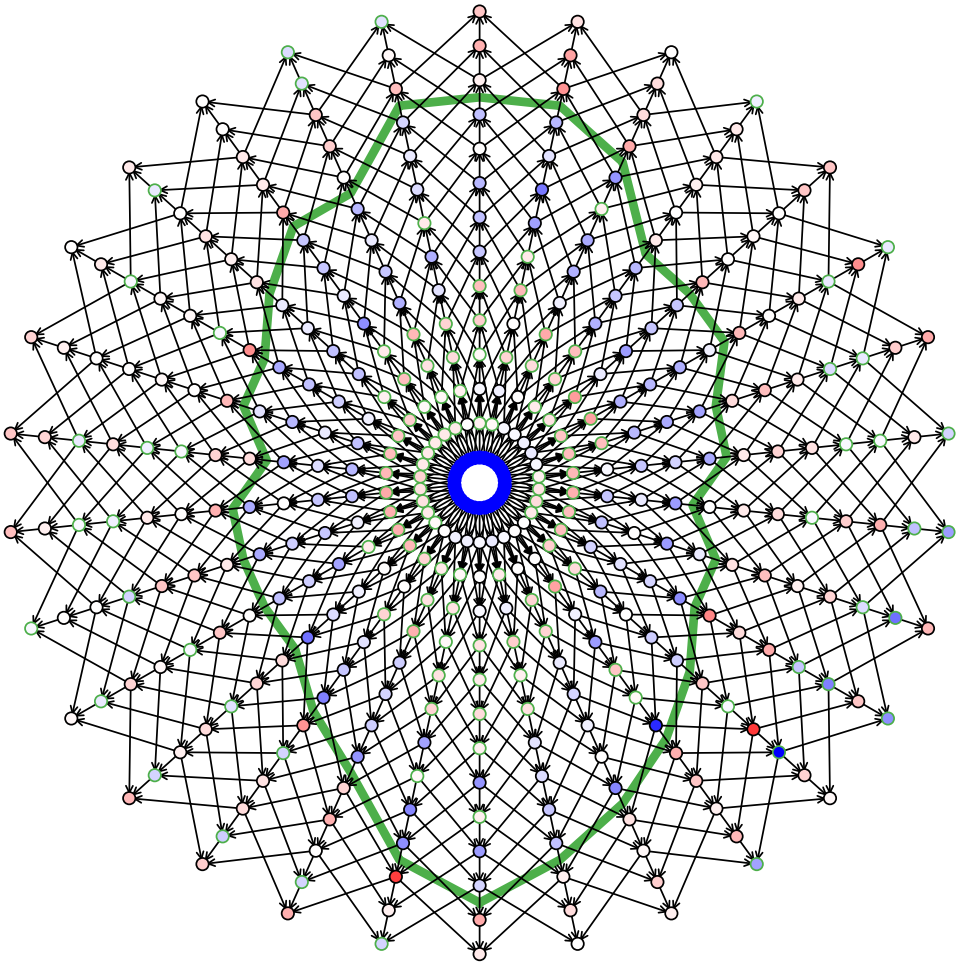


Figure 3.12 – Cut layered ordered multi-column graph with surface smoothness for outer object (layer 1). Node colors indicate terminal edge capacities and nodes with green border indicate cut terminal edges. Black edges have infinite capacity. The green line indicates the segmented surface. Interacting edges between outer and inner object are not shown, but use the same structure as that of Figure 3.8, but with a minimum margin of three and maximum margin of six. The graph is identical to the one in Figure 3.11a.

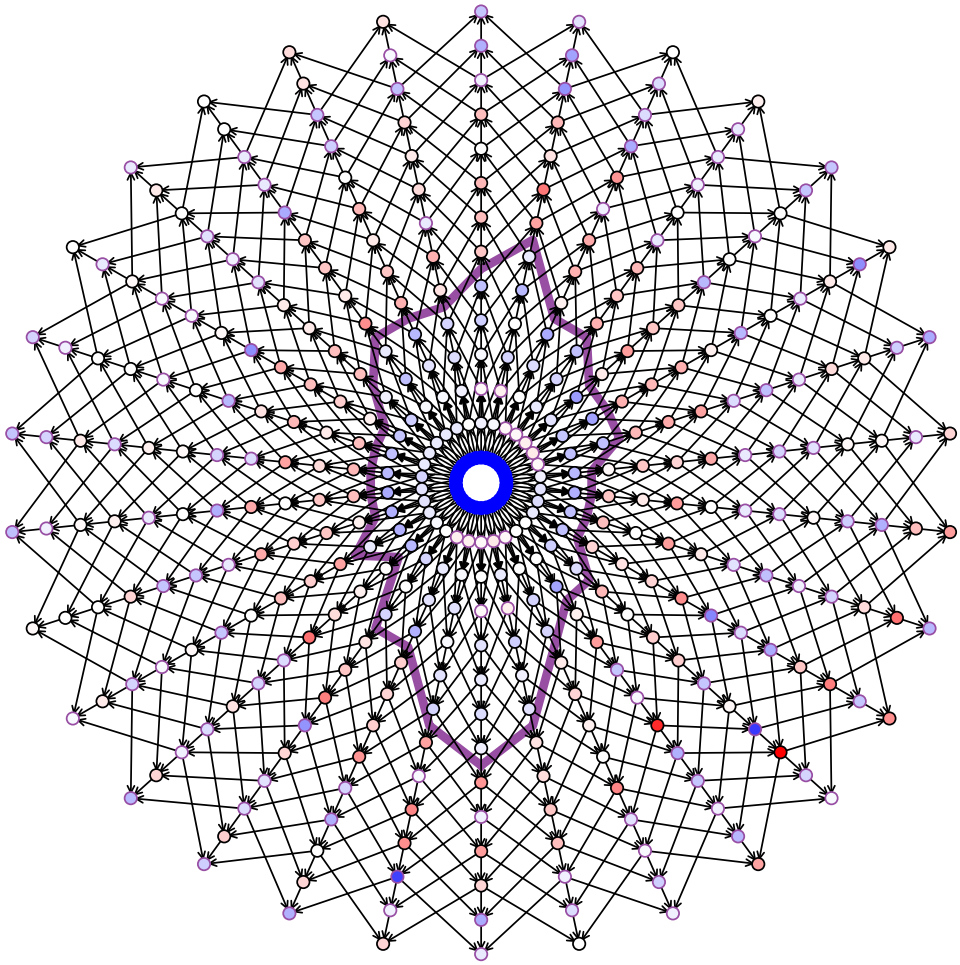
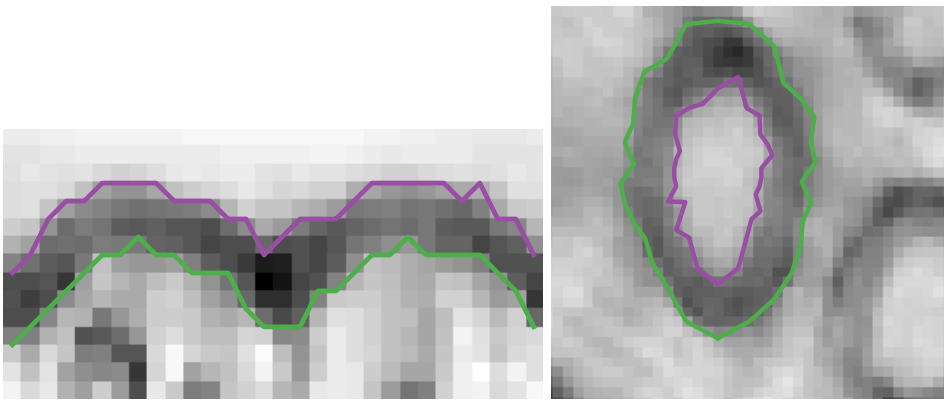


Figure 3.13 – Cut layered ordered multi-column graph with surface smoothness for inner object (layer 2). Node colors indicate terminal edge capacities and nodes with purple border indicate cut terminal edges. Black edges have infinite capacity. The purple line indicates the segmented surface. Interacting edges between outer and inner object are not shown, but use the same structure as that of Figure 3.8, but with a minimum margin of three and maximum margin of six. The graph is identical to the one in Figure 3.11b.



(a) Object surfaces (unfolded) **(b)** Object surfaces

Figure 3.14 – Detected surfaces drawn on top of the data.

3.3 Quadratic pseudo-Boolean optimization

While many problems in image segmentation can be formulated using submodular energy functions (3.2), there are cases where non-submodular energies are extremely useful. One such case is instance segmentation of multiple non-overlapping objects. Of course, there are many other uses of non-submodular energies, but here I will focus on exclusion constraints as this is a very powerful geometric interaction that has been important for my work on multi-object graph-based segmentation.

The foundation of the QPBO was laid by Hammer, Hansen, and Simeone 1984, and Boros and Hammer 2002. Later, Kolmogorov and Rother 2007 implemented a highly efficient version of the QPBO algorithm based on the BK Maxflow algorithm, which is 400-700 times faster than the previous QPBO implementation [Rother et al. 2007]. As a result, non-submodular problems can often be solved almost as fast as submodular problems. QPBO solves non-submodular optimization problems by creating a *dual* graph that contains a copy of each node in the original *primary* graph. Then, instead of creating one edge per energy term using Table 3.1, we use two edges for each term as shown in Table 3.3. Where we for submodular problems have one graph node per node in the energy function and one graph edge per energy term, we use two graph nodes and two edges per node and energy term, respectively, when using QPBO for non-submodular problems.

The strength of QPBO is that it allows us to represent the non-submodular energy terms $\theta_{pq}(0,0)$ and $\theta_{pq}(1,1)$ as edges in a graph, for which we can use the maxflow algorithms to find a maxflow/mincut solution. The publicly available QPBO implementation by Kolmogorov uses the BK Maxflow algorithm, but in principle any maxflow algorithm can be used. The QPBO implementation uses a few "tricks" to speed up computations, but at its core it relies on the maxflow algorithm to cut the

Energy term	Corresponding edge	Edge capacity (cap)
$\theta_p(0)$	$(p \to t), (s \to \bar{p})$	$\theta_p(0)$
$\theta_p(1)$	$(s \to p), (\bar{p} \to t)$	$\theta_p(1)$
$\theta_{pq}(0,1)$	$(p \to q), (\bar{q} \to \bar{p})$	$\theta_{pq}(0,1)$
$\theta_{pq}(1,0)$	$(q \to p), (\bar{p} \to \bar{q})$	$\theta_{pq}(1,0)$
$\theta_{pq}(0,0)$	$(p \to \bar{q}), (q \to \bar{p})$	$\theta_{pq}(0,0)$
$\theta_{pq}(1,1)$	$(\bar{q} \to p), (\bar{p} \to q)$	$\theta_{pq}(0,0)$

Table 3.3 – Mapping from submodular and non-submodular energies to edges [Kolmogorov and Rother 2007]. Using these capacities, the total minimum energy/maximum flow will be twice that of the non-dual solution. To get the same energy, the edge capacities may be halved, but this approach is not practical when integer capacities are used.

graph. As such, QPBO is simply a way of reformulating the problem using a special graph structure, rather than a competing graph cut algorithm.

The primary drawback of QPBO is that for non-submodular energy functions, the guarantee of optimality is replaced by one of partial optimality [Kolmogorov and Rother 2007]. In other words, we are no longer guaranteed a complete solution and may end up with unlabeled nodes. However, the nodes which are labeled, are labeled optimally. Thus, if the solution is complete, it is also optimal. The second downside of using QPBO is that it doubles the size of the graph. This affects performance negatively as runtime increases and the memory footprint is doubled. For smaller tasks, this is generally not an issue, especially since the Kolmogorov implementation uses a clever two-stage approach to re-use calculations between the primary and dual graphs and avoids creating the dual graph entirely if the problem is submodular. Papers A and C contain more details on QPBO. For now, the important thing to note is that QPBO allows us to solve non-submodular optimization problems using maxflow/mincut algorithms, at least partially.

3.4 Paper A: Sparse Layered Graphs for Multi-Object Segmentation

The first contribution in this thesis concerns a method for multi-label segmentation with geometric interactions, using a flexible layered graph structure, which we refer to as a sparse layered graph (SLG). The contribution was presented as a poster at the Computer Vision and Pattern Recognition (CVPR) 2020. The one-minute video for the presentation is available at https://youtu.be/CFUYuL1J85k. The poster, which was *not* presented due to a change in the presentation format as a result of the COVID-19 pandemic, can be found at http://doi.org/10.5281/zenodo.4575261. Links to code, notebooks, and data can be found in the paper.

The version of the paper included below is the postprint. The published version can be found at DOI: 10.1109/CVPR42600.2020.01279. The Open Access version made available by CVF can be found at https://openaccess.thecvf.com.

Sparse Layered Graphs for Multi-Object Segmentation

Niels Jeppesen, Anders N. Christensen, Vedrana A. Dahl, Anders B. Dahl
Department of Applied Mathematics and Computer Science, Technical University of Denmark
Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby, Denmark
niejep,anym,vand,abda@dtu.dk

Abstract

We introduce the novel concept of a Sparse Layered Graph (SLG) for s-t graph cut segmentation of image data. The concept is based on the widely used Ishikawa layered technique for multi-object segmentation, which allows explicit object interactions, such as containment and exclusion with margins. However, the spatial complexity of the Ishikawa technique limits its use for many segmentation problems. To solve this issue, we formulate a general method for adding containment and exclusion interaction constraints to layered graphs. Given some prior knowledge, we can create a SLG, which is often orders of magnitude smaller than traditional Ishikawa graphs, with identical segmentation results. This allows us to solve many problems that could previously not be solved using general graph cut algorithms. We then propose three algorithms for further reducing the spatial complexity of SLGs, by using ordered multi-column graphs. In our experiments, we show that SLGs, and in particular ordered multi-column SLGs, can produce high-quality segmentation results using extremely simple data terms. We also show the scalability of ordered multi-column SLGs, by segmenting a high-resolution volume with several hundred interacting objects.

1. Introduction

Most image segmentation research using graph cuts is demonstrated on problems with less than ten labels. This is enough for high-level segmentation tasks like organ, brain, and bone segmentation. However, many segmentation tasks, such as microscopy imaging in medicine and materials science, involve hundreds or more objects. Segmentation tasks with this number of labels have previously been difficult, or even impossible to solve using *s-t* graph cuts. With our method for constructing graphs, well-known graph cut algorithms can efficiently solve segmentation tasks with hundreds of labels.

Computational speed is essential for the practical use of segmentation. Much of the success of graph cuts is owed to the Boykov-Kolmogorov (BK) implementation of the Ford-Fulkerson maxflow/mincut algorithm, which performs well for many image-related optimization problems and gives a globally optimal solution for submodular problems [1]. More recently, the Incremental Breadth-First Search (IBFS) algorithm by Goldberg *et al*. [6] has shown even better performance and run-time guarantees. Another way to speed up the computations is to use a parallel algorithm [4, 18]. However, to our knowledge, these algorithms only work on regular grid-based graphs.

By definition, *s-t* graph cuts provide a binary labeling. For multi-label segmentation with graph cuts, one option is to use the iterative α-expansion method [2]. However, it often gets stuck in weak local minima. Another common approach is to use the Ishikawa layered graph construction [10], where each layer corresponds to one label. Using this technique, it is possible to solve multi-label problems, while enforcing label interaction constraints, such as *containment* and *two-label exclusion* [5]. These interaction constraints are often necessary to ensure that an object is inside another object or that objects do not overlap. However, because the exclusion term is non-submodular, the approach of [5] does not work for more than two exclusive objects.

To enable multi-object exclusion, one approach is to use the QPBO algorithm [11, 16], which can incorporate non-submodular terms, at the cost of completeness. The algorithm guarantees partial optimality, but may not find a complete solution, *i.e*. there may be unlabelled nodes. A higher-level alternative is the Path-Moves algorithm (HINTS) [8], which is also able to incorporate non-submodular terms. Unlike QPBO, it is an iterative algorithm that always provides a complete labeling and has been shown to find good solutions, although they are not guaranteed to be optimal.

The number of objects that can be segmented using the Ishikawa technique is in practice limited by the size of the layered graph. If Ω is a set of nodes, usually corresponding to the pixels of an image, and $\mathcal{L}$ is the set of labels/objects, then the spatial complexity of the nodes in the layered graph is $O(|\Omega||\mathcal{L}|)$. However, the number of objects is not the

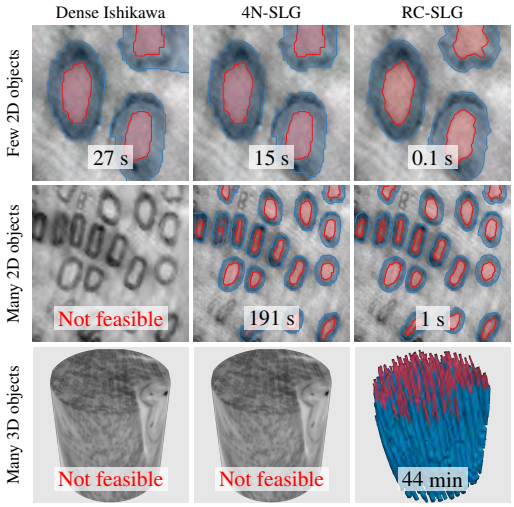


Figure 1. Time spent solving three differently sized problems using *s-t* graph cut with different graph constructions.

only important factor. When applying interactions between the objects, it is possible, and often useful, to specify a minimum margin. For a N-neighborhood regular grid structure [5], the spatial complexity of the interaction terms, and corresponding graph edges, depends polynomially on the size of the interaction margins. In practice, this means that the standard N-neighborhood graph structure is only useful for segmenting a very limited number of objects, with small interaction margins.

An alternative to the N-neighborhood structure is the ordered multi-column graph [19]. This approach is very useful for surface detection, but may also be used for object segmentation, when combined with resampling [13]. The ordered multi-column structure makes it possible to impose a certain geometry to the solution. Which geometry is imposed, depends on how the data is resampled.

While [13] describes how to handle containment interactions for ordered multi-column graphs, object exclusion is not described. Also, they assume contained objects are sampled identically, which means that objects cannot be sampled at different resolutions or have imposed different geometries. Unlike the approach used by [5], the spatial complexity of the ordered multi-column structure by [13] does not depend on the size of the interaction margins. As a result, appropriate margins can be chosen freely, without worrying about the size of the layered graph.

To overcome the complexity issue of layered Ishikawa graph structure, we introduce the concept of a Sparse Layered Graph (SLG), along with a general method for adding object interactions to layered graphs. Using this method, we construct a N-neighborhood SLG (N-SLG), which is significantly smaller than the corresponding dense Ishiskawa graph. Then, to further reduce the size of the graph, we construct an ordered multi-column SLG (C-SLG), based on the method by Li *et al*. [13]. Like the approach by [13], we need some prior knowledge in the form of approximate size and position of the objects. We propose three algorithms for incorporating interaction constraints in C-SLGs, with very few terms. Experimentally, we show that SLGs can be used to reduce segmentation time and accurately solve segmentation problems with hundreds of objects (see Fig. 1). Such tasks cannot be solved with the traditional dense Ishikawa layered graphs, used by [5, 8], due to the size of the graph.

We compare the segmentation accuracy, time and graph size of different configurations of an N-SLG, C-SLG, and the method by Li *et al*. [13], on an instance segmentation task. Our experiments show the advantages of using SLGs, and in particular C-SLG, over the traditional layered graph. They also show that ordered multi-column graphs can provide accurate segmentations, even with extremely simple models. We then demonstrate the scalability of C-SLGs by segmenting a large volume with several hundred interacting objects using a single graph cut.

Our method uses the QPBO algorithm. This means that we cannot guarantee completeness, but only partial optimality [16]. Thus, we may not be able to label all nodes if the model contains non-submodular terms. Many unlabelled nodes will result in a poor segmentation, so it is critical for the accuracy of our method that unlabelled nodes are rare. To investigate the frequency of unlabelled nodes, we segment a large set of images using SLGs. The results show that accurate segmentation is possible with a simple model, even on a varied data set. Furthermore, unlabelled nodes are rare and have little impact on the segmentation.

Along with this paper, we release an open-source Python package for constructing and solving SLGs (see Section 3).

2. Multi-object segmentation

We consider an image segmentation problem, with several objects, which may be interacting. We use the term *object*, *label* and *layer* interchangeably, depending on the context – whether we refer to the content of an image, outcome of a segmentation, or construction of a graph.

A common way to solve image segmentation problems is by minimizing an energy function of the form

$$E(\mathbf{x}) = \sum_{p \in \mathcal{V}} \theta_p(x_p) + \sum_{p,q \in \mathcal{V}} \theta_{pq}(x_p, x_q)\,. \qquad (1)$$

For images, the node-set $\mathcal{V}$ usually corresponds to image pixels, where the segmentation can be obtained as a pixel labeling with labels $x_p \in \{0, 1\}$. Unary energy terms, θ_p, usually encode a data term, while pairwise energies, θ_{pq},

encode interactions between pixels, as well as interactions between labels.

If the energy function E is submodular, meaning that all pairwise terms, θ_{pq}, between nodes p and q satisfy

$$\theta_{pq}(0,0) + \theta_{pq}(1,1) \leq \theta_{pq}(0,1) + \theta_{pq}(1,0) \qquad (2)$$

then it is possible to use the *s-t* graph cut to find the global energy minimum in polynomial time [12].

In many segmentation models, pairwise interactions are symmetric ($\theta_{pq}(0,1) = \theta_{pq}(1,0)$), finite ($\theta_{pq}(0,1) < \infty$), and submodular. Such interaction will encourage smoothness and will be balanced by the unary terms. On the other hand, asymmetric and infinite terms are useful to impose a certain geometry or object interaction.

2.1. Sparse layered graphs

In the dense Ishikawa layered structure, used by [5, 8, 13], identical graph layers are created for each object, as shown in Fig. 2b. This often results in a large number of irrelevant nodes, as only a fraction of the nodes of a given layer is usually inside or near the object. In an SLG, we remove irrelevant nodes from the layers to reduce the size of the graph. This creates what we call sparse layers, which are layers where not all pixels are represented by nodes. To determine which nodes are relevant for each layer, we use information, which is often available or can be computed in some way, such as approximate position and size of objects.

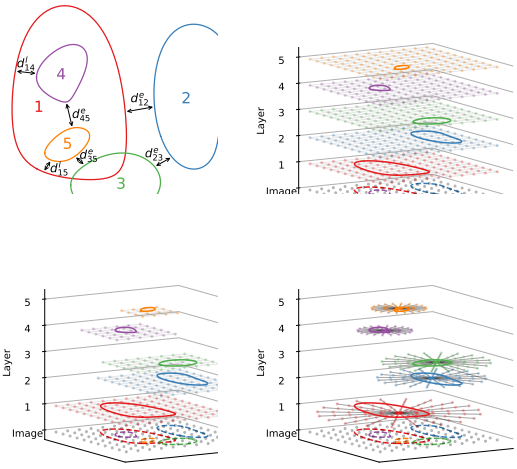
We can create a simple N-SLG, similar to the one in Fig. 2c, by first constructing a dense Ishikawa graph. Then, we crop each layer to remove nodes that are known to be outside the layer object. This way, all nodes still correspond to a single pixel, but not all pixels are represented by nodes. This approach preserves the pixel neighborhood structure between nodes in different layers used by previous methods [5, 8].

A common way to reduce the number of nodes in graph segmentation problems is to downsample data before creating the graph. However, this approach will also reduce the resolution of the segmentation. If we are segmenting interacting objects of varying sizes, it could be favorable to downsample large objects, while keeping small objects at a higher resolution. We could choose to only downsample the data for layers with large objects, but this breaks the inter-layer neighborhood structure.

It turns out that resampling can be used, not just for varying layer resolution, but also to enforce shape priors [13]. However, we need a way of adding object interactions between differently sampled layers.

2.2. Geometric interactions

We focus on two important geometric interactions:



(c) 4N-SLG (d) RC-SLG

Figure 2. Five interacting objects, with different minimum containment margins, d^l, and exclusion margins, d^e, are shown in (a). Interactions are defined independently between pairs of objects. With the Ishikawa technique, a layer is created for each object as shown in (b). The gray dots at the bottom indicate nodes that would correspond to pixels in the image. The colored dots are the layer nodes. In a 4N-SLG, shown in (c), we keep the neighborhood graph structure, but sample only a subset of the original pixels. Alternatively, we can sample radially and create a column graph (RC-SLG), as shown in (d).

- **Containment.** One object must be inside another object, with the possibility of specifying a minimum margin between the objects (d^l_{IJ} in Fig. 2a).
- **Exclusion.** Two objects cannot overlap at any point, with the possibility of specifying a minimum distance (d^e_{IJ} in Fig. 2a).

We propose a general way of applying containment and exclusion terms between non-identical graph layers, such as the ones shown in Fig. 2d. All we require, is that we can calculate a distance between nodes in interacting layers.

In the following, we will consider objects $I, J \subset \mathbb{R}^2$ and a set of graph nodes $\mathcal{V} = \mathcal{V}_I \cup \mathcal{V}_J$, where $\mathcal{V}_I$ and $\mathcal{V}_J$ are graph layers for objects I and J. We will write i when we refer to nodes from $\mathcal{V}_I$, and similarly j for $\mathcal{V}_J$. The spatial position of a node is given by a mapping $p : \mathcal{V} \to \mathbb{R}^2$, such that $p(i)$ denotes the position of node i.

Containment, *e.g.* object I contains object J, is simple to enforce by adding an energy term for all pairs of nodes $i \in \mathcal{V}_I$ and $j \in \mathcal{V}_J$

$$\theta_{ij}(0,1) = \infty \ , \ \|p(i) - p(j)\| \leq d^l_{IJ} \ . \qquad (3)$$

Here, d^l_{IJ} is the minimum margin between the outer object, I, and inner object, J. The energy term $\theta_{ij}(0,1) = \infty$ is

submodular and can therefore be translated to a single edge in a graph. Thus, we can solve problems with containment constraints using a standard maxflow solver, such as [1].

Exclusion, *e.g.* objects I and J are exclusive, can be enforced by adding energy terms

$$\theta_{ij}(1,1) = \infty \ , \ \|p(i) - p(j)\| \leq d^e_{IJ}, \tag{4}$$

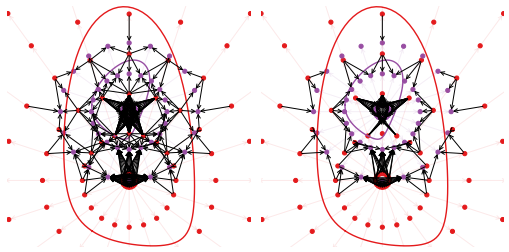
for all pairs of nodes $i \in \mathcal{V}_I$ and $j \in \mathcal{V}_J$, where d^e_{IJ} is the minimum margin between I and J. Because $\theta_{ij}(1,1) = \infty$ is a non-submodular energy term, it cannot be expressed as a single edge in a graph. To overcome this, we use the QPBO algorithm.

Since Eq. (3) and (4) can be applied as long as we can calculate a distance between nodes in interacting layers, we can now apply object interactions between objects sampled in any way. It is possible to sample different object using entirely different sampling schemes and have different graph structures in different layers. However, doing so will of course impact the segmentation results and may introduce a bias. Another important point is that the method is not limited to images in 2D. Interactions can be applied between nodes sampled in any dimension, although increased dimensionality will usually also increase the number of interaction terms significantly. To overcome this issue, we will now look closer at using SLGs with one particular intra-layer structure.

2.3. Ordered multi-column graphs

Li *et al.* [13] describe how to use an ordered multi-column graph to segment multiple interacting surfaces with a star-shaped prior. When an approximate position of an object is known, it has several advantages over a N-neighborhood structure. For instance, the surface smoothness parameter, Δ, makes it very robust, even with noisy data. Also, the number of containment terms remains almost constant for any minimum margin, δ^l. This overcomes a major problem of Eq. (3), namely that the number of terms depends polynomially on the margin size. Furthermore, the column graph allows for the specification of a maximum margin, δ^u, which can be very helpful for many segmentation problems. Such a margin cannot be specified using a N-neighborhood structure. The δ parameters, used by [13], specifically refers to the neighborhood distance on identically sampled layers. Our distance measure, d, used in Eq. (3) and (4), is an arbitrary distance measure. For simplicity, we use Euclidean distance in this paper.

C-SLGs can be seen as a generalization of the resampling-based method used by [13] for object segmentation. Because their method relies on the neighborhood distance, δ, for containment interactions, all layers must be sampled at the same positions. If instead, we use Eq. (3) to add containment terms, based on the Euclidean distance between the sample locations, layers no longer have to be sampled identically. Because of the difference in size and center position of objects, sampling them differently provides a much better basis for our segmentation.



(a) Many redundant terms (b) Few redundant terms

Figure 3. Layered containment terms enforcing a minimum margin between object 1 and 4 in Fig. 2a. (a) No removal of redundant terms. (b) Most redundant terms removed.

Exclusion interactions are not used by [13]. As they use the BK [1] implementation to cut the graph, non-submodular terms cannot be used in their model. Also, δ cannot be used for exclusion margins on radially sampled ordered multi-column graphs, as non-overlapping objects would always be sampled differently.

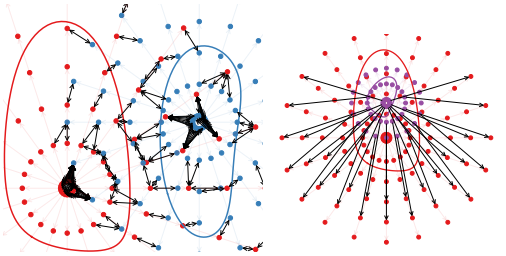
Inspired by the approach of [13], we propose two algorithms for reducing the number of interaction terms in C-SLGs. We also describe an algorithm for enforcing a maximum containment margin in C-SLGs.

To reduce the number of terms, without changing the solution, we rely on the fact that the ordered multi-column graph has infinite cost terms inside each column. As the interaction terms are also infinite cost, many of the terms added by Eq. (3) and (4) are redundant. By not adding the redundant terms to the graph, we can reduce the size of the graph significantly. Because the nodes are ordered, calculating which interaction terms are required and which are redundant, can be done quickly, before constructing the actual graph.

It should be noted that if the sampling resolution is low, meaning that the nodes are far apart, compared to the specified minimum margin, d, margins may not be enforced properly. This is a result of Eq. (3) and (4) only adding terms between nodes within the given margin. It is possible to extend both algorithm 1 and 2 to accommodate for this issue, but for now we will focus on reducing the number of terms and assume that the sampling resolution is sufficiently high compared to the margins.

2.3.1 Algorithm 1: Reducing containment terms

Fig. 3a shows the containment interaction terms for minimum margin, d^l_{14}, created using Eq. (3). Fig. 3b shows the same constraint, but with fewer terms. Our algorithm



(a) Exclusion interaction terms (b) Maximum margin containment interaction terms

Figure 4. (a) Layered exclusion terms with minimum margin between object 1 and 2 in Fig. 2a. (b) Layered containment edges for maximum margin between object 1 and 4 in Fig. 2a

for adding minimum margin interaction with few redundant terms is as follows:

1. For each *inner object* node (purple in Fig. 3), find all *outer object* nodes (red in Fig. 3) within a distance, d^l. Add all these pairs of nodes to the set of candidates, C. This is the approach from Eq. (3).
2. Remove pairs from C, so that only one pair remains for each *outer object* node, *inner object* column combination. The pair kept should be the pair with the innermost *inner object* node.
3. Remove pairs from C, so that only one pair remains for each *inner object* node, *outer object* column combination. The pair kept should be the pair with the outermost *outer object* node.
4. For each pair in C, add a pairwise term $\theta_{oi}(0,1) = \infty$ to E, where o is the *outer object* node and i is the *inner object* node.

As we will show experimentally, this algorithm can reduce the number of interaction terms by orders of magnitude. Theoretically, if N_I is the number of nodes in the inner object layer, N_O is the number of nodes in the outer object layer, and N_{IC} and N_{OC} are the number of columns in the inner and outer layers respectively, the worst-case number of containment terms is reduced from $N_I \cdot N_O$ to approx. $N_{IC} \cdot N_O + N_{OC} \cdot N_I$. In the case of radial resampling, N_{IC} and N_{OC} are the number of sample angles.

2.3.2 Algorithm 2: Reducing exclusion terms

As with containment, when we create a C-SLG, we can reduce the number of exclusion interaction terms compared to the general approach from Eq. (4). The algorithm is as follows:

1. For each *object 1* node (red in Fig. 4a), find all nodes in *object 2* (blue in Fig. 4a) within a distance, d^e. Add all these pairs of nodes to the set candidates, C. This is the general approach from Eq. (4).
2. Remove pairs from C, so that only one pair remains for each *object 1* node, *object 2* column combination. The pair kept should be the pair with the innermost *object 2* node.
3. Remove pairs from C, so that only one pair remains for each *object 2* node, *object 1* column combination. The pair kept should be the pair with the innermost *object 1* node.
4. For each pair in C, add a pairwise term $\theta_{o_1 o_2}(1,1) = \infty$ to E, where o_1 is the *object 1* node and o_2 is the *object 2* node.

The result of adding exclusion between object 1 and 2 in Fig. 2a can be seen in Fig. 4a.

2.3.3 Algorithm 3: Maximum containment margin

Because maximum containment margins cannot be enforced on N-neighborhood structured graphs, we have no general method for adding this type of interaction. Nevertheless, as shown by [13], it is possible to add this type of interaction when using an ordered multi-column graph with identical layers. However, for non-identical layers, there is no simple way of determining which columns and nodes in the two objects should interact.

We propose an algorithm for adding maximum containment interactions to C-SLGs with non-identical layers. It is designed to add intuitive maximum margin constraints using very few terms. The algorithm is as follows:

1. Calculate the node position gradient for both the *outer object* (red in Fig. 4b) and *inner object* (purple in Fig. 4b) along the columns. This indicates the direction of the column in the sample space. For columns where node positions form a straight line, such as radially sampled columns, the gradient is the same for all nodes in a column.
2. Move the *inner object* nodes in the direction of their gradient with the distance d^u.
3. For each node in the *inner object*, find the four nearest nodes in the *outer object* and add these pairs to the set of candidates, C.
4. For each pair in C, calculate the original distance between the two nodes, as it was before the *inner object* nodes were moved. Remove any pairs from C, where this distance is less than d^u.
5. For each pair in C, calculate the angle between the two nodes using the gradient from before. If the angle between the gradient vectors is more than 90 degrees, remove the pair from C.
6. Remove pairs from C, so that only one pair remains for each *outer object* node, *inner object* column combination. The pair kept should be the pair with the outermost *inner object node*.
7. Remove pairs from C so that only one pair remains for

each *inner object* node. The pair kept should be the pair with the smallest angle between the node gradients.

8. For each pair in C, add a pairwise term $\theta_{io}(0,1) = \infty$ to E, where i is the *inner object* node and o is the *outer object* node.

The result of using this algorithm is shown in Fig. 4b. We see that the algorithm effectively applies the interaction terms between nodes in columns pointing in the same direction, which is what we want.

2.4. Solving the graph

To handle non-submodular terms we use QPBO, because it is a general algorithm for solving problems of the form shown in Eq. (1). In general, unlabelled nodes may occur when using the QPBO algorithm to solve problems with non-submodular energy terms, such as exclusion. However, when using QPBO with SLGs, our experience shows that the number of unlabelled nodes is negligible.

3. Experiments

We have tested our method on two different datasets. A high-resolution 3D image of nerves, collected using μCT, and the nuclei image set BBBC038v1, available from the Broad Bioimage Benchmark Collection [14]. The primary goal of the experiments is to show the scalability of SLGs for both 2D and 3D multi-object segmentation. Secondly, we want to highlight the benefits of using C-SLGs, compared to neighborhood-based graph structures.

All experiments were run on an Azure H8m virtual machine running an Intel Xeon E5-2667 v3 CPU with 8 virtual processors and 112 GB memory.

Data, code and Jupyter notebooks for all experiments can be found at DOI 10.11583/DTU.12016941. A Python package for building and cutting sparse layered graphs has also been published to GitHub[1].

3.1. Nerve segmentation

The μCT volume is $2048 \times 2048 \times 2048$ voxels and contains several hundred nerves. Each nerve consists of a dark outer ring (myelin) and a bright core (axon). Because the axon and background have the same intensity, and because the intensities vary a lot between the nerves, accurately segmenting all nerves using the same parameters is difficult.

Before we start our experiments, center lines have been created for 216 of the nerves. Also, a single slice has been taken out of the volume and cropped to 512×512 pixels. The slice contains 17 nerves and has been segmented manually to obtain a ground truth segmentation shown in Fig. 5a.

[1]https://github.com/Skielex/slgbuilder

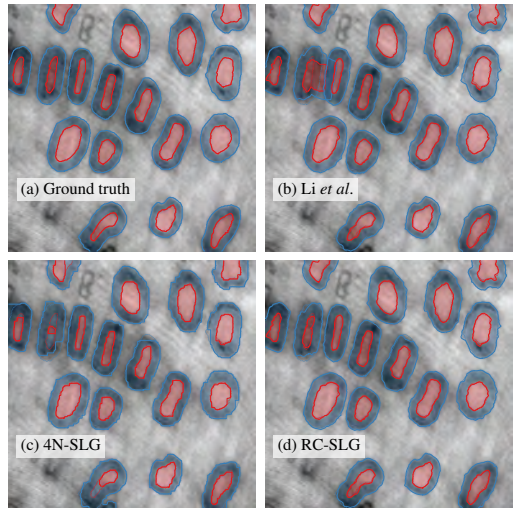


Figure 5. The most accurate nerve slice segmentation for each method and ground truth. The corresponding accuracy for each method is shown in Table 1.

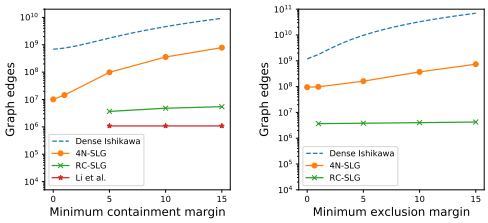
	4N-SLG		RC-SLG		Li *et al.*	
Nodes (mil.)	2.46		0.58		0.28	
	Min	Max	Min	Max	Min	Max
Edges (mil.)	6.8	1425	3.6	6.1	1.1	1.1
Time (s)	1.95	545	0.48	2.70	0.19	0.74
F1	0.91		**0.94**		0.92	
Precision	**0.95**		0.93		0.90	
Recall	0.90		**0.96**		0.95	

Table 1. Results for nerve slice segmentation using a 250 different configurations. The number of configurations was 25, 180 and 45 for the 4N-SLG, RC-SLG, and Li *et al.*, respectively. The parameters varied were the three interaction margins and the surface smoothness. For each method, the number of nodes is the same for all configurations, while the number of edges and solve time change. The accuracy is calculated as the mean score of all masks for a given configuration.

3.1.1 Single volume slice

We use the 512×512 image to compare the accuracy and graph size of the original method by Li *et al.* to a 4-neighborhood SLG (4N-SLG) and a radially resampled column SLG (RC-SLG). For each method, we evaluate several different configurations for margin sizes and surface smoothness.

Fig. 5 shows the most accurate segmentation for each method. As shown in Table 1, the RC-SLG provides the most accurate segmentation. The 4N-SLG struggles when the contrast between the myelin and the axon are low, while the method by [13] does not support exclusion and thus in-



(a) (b)

Figure 6. The two plots show the increase in the number of graph edges as the minimum interaction margins are increased for the 2D nerve segmentation problem shown in Fig. 5. The numbers for the Dense Ishikawa graph are theoretical, while the numbers for the other three methods are based on our experiments. In (a) we see how the graph size increase for the different methods as the minimum containment margin is increased. Other parameters are kept constant and as similar as possible. Li *et al*. does not have exclusion terms, which is is the main reason it has fewer edges than the RC-SLG. In (b) we similarly increase the exclusion margin. Here Li *et al*. is omitted since it does not have exclusion.

correctly overlap some segments. We also see that the number of edges and solve time vary significantly for the 4N-SLG, depending on the configuration. Fig. 6 shows how the number of edges changes depending on the margins. It is clear that the N-neighborhood-based methods do not scale well, as we increase the interaction margin. In fact, the Ishikawa graph quickly grows so big we cannot create the graph due to lack of memory. The two ordered multi-column-based methods do not have this problem, allowing us to set appropriate margins with little to no cost.

3.1.2 Full volume

To show the scalability of C-SLGs we segment all 216 nerves (432 objects) in the 2048-cubed volume with a single graph cut. For this, we used the same radial approach as for the slice, just in 3D and with a lower resolution. Along the annotated centerline, we sample points radially on planes orthogonal to the centerline. In total, we sample 182 million different positions in the volume, with which we construct the RC-SLG. It takes 44 minutes to solve the problem using the QPBO algorithm. The complete graph contains a total of 363 million nodes and 2.1 billion edges. The result contains no unlabelled nodes, which means we found the globally optimal solution to the problem. Solving this problem with a dense Ishikawa structured graph is not possible due to hardware limitations.

3.2. Nuclei segmentation

The purpose of this experiment is to compare the 4N- and RC-SLG on a large number of different images with

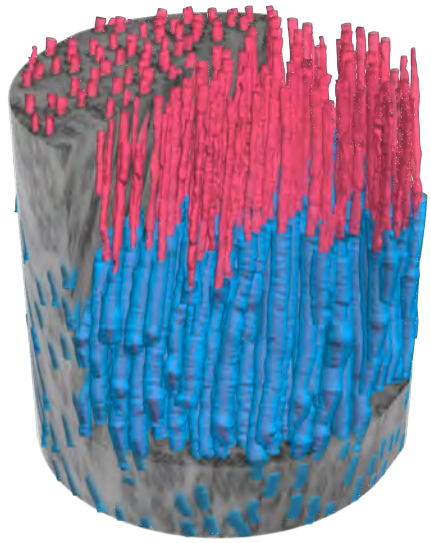


Figure 7. Nerve volume segmentation of the myelin and axon of 216 nerves created using a RC-SLG.

ground truth segmentation masks. We compare both complexity and accuracy and investigate the frequency of unlabelled nodes.

The BBBC038v1 `strage1_train` image dataset contains 670 images with a total of 29,461 segmented nuclei. The images were acquired using different imaging modalities and vary in size. The type of cells and their size vary between images and the number of nuclei per image ranges from a few to several hundred.

As a part of the experiment, we use the exact same configuration for all images. This allows us to test how sensitive/robust the methods are. Ideally, it should not be necessary to configure parameters for each individual image. For this reason, we also use the same simple gradient-based data terms for all images. One of the segmentation results is shown in Fig. 8.

In Table 2 we see that the RC-SLG significantly outperforms the 4N-SLG, both in terms of accuracy and speed. The low recall and high precision score of the 4N-SLG, indicate that it tends to underestimate the size of the nuclei. The RC-SLG does not have this issue and accurately segments nuclei of all sizes, even though they vary from a few to over a hundred pixels in diameter. Furthermore, segmentations by the RC-SLG have fewer unlabelled nodes than those by the 4N-SLG, although they hardly impact accuracy in either case. In fact, for RC-SLG, 99.8% of the masks and 97% of the images are completely labeled.

Fig. 9a further highlights the scalability of SLGs, and

	4N-SLG		RC-SLG	
Per image	Mean	Max	Mean	Max
Nodes (mil.)	2.25	19.2	0.71	6.08
Edges (mil.)	14.6	442	3.12	60.3
Time (s)	2.55	69.5	1.02	26.1
Per mask	Mean	Max	Mean	Max
Unlabelled	3.2	428	0.48	876
Per mask	Mean	Std.	Mean	Std.
F1	0.48	0.32	**0.85**	0.12
Precision	**0.97**	0.09	0.85	0.13
Recall	0.40	0.34	**0.89**	0.16

Table 2. Results for nuclei segmentation on 670 image with 29,461 segmented nuclei. The RC-SLGs are generally smaller than the 4N-SLGs, which also makes them faster to solve. In terms of accuracy, the RC-SLGs perform significantly better than the 4N-SLGs. In terms of unlabelled nodes, the RC-SLGs also perform best, leaving only 0.006% of nodes unlabelled, compared to 0.04% by the 4N-SLGs.

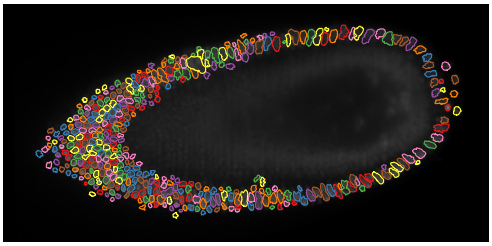
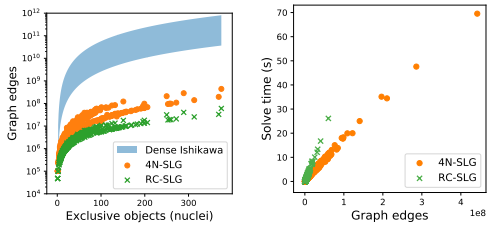


Figure 8. Largest nuclei image segmented using the RC-SLG. The image is 1272×603 pixels and has 375 segmented nuclei. The RC-SLG had approx. six million nodes and 60 million edges. A dense Ishikawa graph would have had over one billion nodes and need over 100 billion edges to enforce exclusion between all nuclei. With an exclusion margin of five, this number increases to over 10 trillion edges.

in particular C-SLGs. Fig. 9b is interesting as it shows a linear correlation between the number of graph edges and the solve time for both methods in our experiment. This means that as long as we can keep the number of edges low, we should be able to find a solution fast.

4. Discussion and conclusion

A limitation of our method is that it requires some prior knowledge about the number of objects, and where they are. However, most graph cut-based segmentation methods require this kind of prior knowledge. Another challenge is that segmentations may be incomplete, as we rely on the QPBO algorithm for solving non-submodular problems. Nevertheless, our experiments show unlabelled nodes are rare, and thus not a problem for the accuracy of the segmentation. One reason for this could be that we only



(a) (b)

Figure 9. Two plots based on our results from segmenting 670 images of nuclei. In (a), we see how the number of objects (nuclei) in the images affect the number of edges. We assume that the reason the points for the 4N-SLG and RC-SLG are not on a smooth line is that the number of interaction terms also depends on the relative positions and sizes of the objects. The numbers for the dense Ishikawa graph are theoretical. The area indicates the number of edges for the smallest and largest image in the image set. Plot (b) shows the correlation between the time it took to solve the graph cut and the number of graph edges for each image. For both methods, the Pearson correlation coefficient is over 0.99, which indicate a linear correlation between the number of edges and solve time.

use non-submodular terms for exclusion. It also appears that the RC-SLG is better for avoiding unlabelled nodes than the 4N-SLG. This is interesting, as using QPBO with N-neighborhood-based geometric priors [9] has previously been shown to result in many unlabelled nodes [8]. To label unlabelled nodes, extensions to QPBO, such as QPBO-I and QPBO-P have been proposed [16]. Although we do not use these extensions in our experiments, they could be used to further reduce the small number of unlabelled nodes in our segmentations.

Overall, our experiments show that SLGs, and in particular C-SLGs, can be used to segment very large images (2D and 3D) accurately, even with simple gradient-based data terms. These tasks are unsolvable using traditional dense graph structures. Although we have focused on images, SLGs, as well as Eq. (3) and Eq. (4) are general and can be used for 4D or point cloud data as well.

It is clear that the C-SLGs, created using our three algorithms, provide a particularly effective way of solving large segmentation tasks. In this paper, we enforced a star-like prior for the C-SLGs by using radial resampling. However, we believe there is a large potential in using C-SLGs with sampling schemes based on other priors [7, 9, 17], or by sampling based on surfaces in 3D [3, 15].

Acknowledgments This work is supported by FORCE Technology and The Center for Quantification of Imaging Data from MAX IV (QIM).

References

[1] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(9):1124–1137, Sept 2004. 1, 4

[2] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, Nov 2001. 1

[3] V. A. Dahl, A. B. Dahl, and R. Larsen. Surface detection using round cut. In *2014 2nd International Conference on 3D Vision*, volume 2, pages 82–89, Dec 2014. 8

[4] A. Delong and Y. Boykov. A scalable graph-cut algorithm for n-d grids. In *2008 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2008. 1

[5] A. Delong and Y. Boykov. Globally optimal segmentation of multi-region objects. In *2009 IEEE 12th International Conference on Computer Vision*, pages 285–292, Sept 2009. 1, 2, 3

[6] A. V. Goldberg, S. Hed, H. Kaplan, P. Kohli, R. E. Tarjan, and R. F. Werneck. Faster and more dynamic maximum flow by incremental breadth-first search. In N. Bansal and I. Finocchi, editors, *Algorithms - ESA 2015*, pages 619–630, Berlin, Heidelberg, 2015. Springer Berlin Heidelberg. 1

[7] V. Gulshan, C. Rother, A. Criminisi, A. Blake, and A. Zisserman. Geodesic star convexity for interactive image segmentation. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3129–3136, June 2010. 8

[8] H. Isack, O. Veksler, I. Oguz, M. Sonka, and Y. Boykov. Efficient optimization for hierarchically-structured interacting segments (HINTS). In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4981–4989, July 2017. 1, 2, 3, 8

[9] H. Isack, O. Veksler, M. Sonka, and Y. Boykov. Hedgehog shape priors for multi-object segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2434–2442, June 2016. 8

[10] H. Ishikawa. Exact optimization for markov random fields with convex priors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 25(10):1333–1336, Oct 2003. 1

[11] V. Kolmogorov and C. Rother. Minimizing nonsubmodular functions with graph cuts – a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(7):1274–1279, July 2007. 1

[12] V. Kolmogorov and R. Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2):147–159, Feb 2004. 3

[13] K. Li, X. Wu, D. Z. Chen, and M. Sonka. Optimal surface segmentation in volumetric images – a graph-theoretic approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(1):119–134, Jan 2006. 2, 3, 4, 5, 6

[14] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature Methods*, 9(7):637–637, 2012. 6

[15] I. . Oguz and M. Sonka. Logismos-b: Layered optimal graph image segmentation of multiple objects and surfaces for the brain. *IEEE Transactions on Medical Imaging*, 33(6):1220–1235, June 2014. 8

[16] C. Rother, V. Kolmogorov, V. Lempitsky, and M. Szummer. Optimizing binary MRFs via extended roof duality. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8, June 2007. 1, 2, 8

[17] O. Veksler. Star shape prior for graph-cut image segmentation. In D. Forsyth, P. Torr, and A. Zisserman, editors, *Computer Vision – ECCV 2008*, pages 454–467, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. 8

[18] V. Vineet and P. J. Narayanan. Cuda cuts: Fast graph cuts on the gpu. In *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–8, June 2008. 1

[19] X. Wu and D. Z. Chen. Optimal net surface problems with applications. In P. Widmayer, S. Eidenbenz, F. Triguero, R. Morales, R. Conejo, and M. Hennessy, editors, *Automata, Languages and Programming*, pages 1029–1042, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. 2

3.5 Paper B: Comparing Serial and Parallel Min-Cut/Max-Flow Algorithms for Computer Vision

While SLGs allow maxflow/mincut algorithms to solve much larger maxflow/mincut problems by providing a way to remove irrelevant nodes and terms, we still rely only on serial maxflow algorithms to cut the graph. This is less than ideal in a world where high-performance hardware relies heavily on parallelization to speed up computation.

In the second contribution, we compare state-of-the-art serial and parallel maxflow/mincut algorithms on computer vision tasks, such as image segmentation. Our work includes our own implementation of a parallel algorithm originally proposed by Jiangyu Liu and Jian Sun 2010 as well as our own optimized versions of several state-of-the-art algorithms including the BK algorithm [Boykov and Kolmogorov 2004] and EIBFS [Goldberg, Hed, Kaplan, Kohli, et al. 2015].

The paper included below is still in preparation. Once complete, the paper will be submitted to IEEE Transactions on Image Processing.

Comparing Serial and Parallel Min-Cut/Max-Flow Algorithms for Computer Vision

Patrick M. Jensen, Niels Jeppesen, Anders B. Dahl, and Vedrana A. Dahl
Department of Applied Mathematics and Computer Science
Technical University of Denmark, Kgs. Lyngby, Denmark
{patmjen,niejep,abda,vand}@dtu.dk

***Abstract*—Minimum cut / maximum flow (min-cut/max-flow) algorithms are used to solve a variety of problems in computer vision and thus significant effort has gone into developing fast min-cut/max-flow algorithms. However, most algorithmic development has targeted serial algorithms, which do not take advantage of modern multi-core processors. Furthermore, parallelization approaches often focus exclusively on the Boykov-Kolmogorov (BK) algorithm and/or are restricted to structured grids. In this paper, we review the most popular min-cut/max-flow algorithms for unstructured graphs in computer vision and the strategies proposed for parallelization. We perform a comprehensive evaluation of both serial and parallel algorithms on a number of graph cut problems w.r.t. runtime performance and memory use. To illustrate the importance of how the algorithms are implemented, we evaluate different implementations of the same algorithms. Our results show that the Hochbaum pseudoflow (HPF) algorithm is the fastest serial algorithm for computer vision problems, followed by the Excesses Incremental Breadth First Search (EIBFS) algorithm. The fastest parallel algorithm is the adaptive bottom-up merging approach by Liu and Sun. The BK algorithm is the most memory efficient serial algorithm, closely followed by EIBFS, while the adaptive bottom-up merging and the dual decomposition algorithms are the most memory efficient of the parallel algorithms. Additionally, we show significant variations in performance for different implementations of the BK and EIBFS algorithms, which highlight the impact of various implementation and optimization details. Finally, we note that good implementations of existing parallel min-cut/max-flow algorithms can scale well on multi-core systems, allowing them to solve large problems in only a fraction of the time used by the fastest serial algorithms. However, new and better implementations are needed for existing parallel algorithms to reach their full potential. Implementations of all algorithms, as well as bindings for Python and MATLAB, are available at: (link to come, see supplementary).**

I. Introduction

Min-cut/max-flow algorithms are ubiquitous in computer vision (CV) since a large variety of CV problems can be formulated as min-cut/max-flow problems. Example applications include image segmentation [8, 11, 30, 31, 35, 47], stereo matching [9, 40], surface reconstruction [43] and fitting [15, 36, 42, 44, 54, 56], and texture restoration [49]. Min-cut/max-flow algorithms have also found use in conjunction with deep learning methods – for example, to quickly generate training labels [37] or to improve CNN segmentations [25].

Formally, the application of min-cut/max-flow in computer vision involves solving energy minimization problems with an energy function, $\mathcal{E}$, of the form

$$\mathcal{E}(\mathbf{x}) = \sum_{i \in V} \mathcal{E}_i(x_i) + \sum_{(i,j) \in V \times V} \mathcal{E}_{ij}(x_i, x_j), \tag{1}$$

where V is a set of binary variables x_i, $\mathcal{E}_i$ is a unary term associated with variable i and $\mathcal{E}_{ij}$ is a binary term associated with the pair of variables i, j. In a typical application, such as binary segmentation with Markov random fields (MRFs) [8], V would represent pixels in an image and x_i the assignment of pixel i. However, variables can also describe more abstract things such as candidate positions for mesh vertices [44, 55].

For energy functions which are *submodular*, meaning that all binary terms satisfy the condition

$$\mathcal{E}_{ij}(0,0) + \mathcal{E}_{ij}(1,1) \leq \mathcal{E}_{ij}(0,1) + \mathcal{E}_{ij}(1,0), \tag{2}$$

the minimization can be solved directly as a min-cut/max-flow problem [18, 41]. If the energy function is not submodular, one can either use a submodular approximation or an approach based on quadratic pseudo-Boolean optimization (QPBO) as described in [6, 26, 39, 49].

Due to the wide applicability of min-cut/max-flow in CV, many algorithms have been developed to efficiently solve min-cut/max-flow problems [8, 23, 52]. To increase performance, several algorithms also utilize that CV problems will often use a special graph structure. For example, the resulting graph will often have a grid structure, which has been exploited to significantly reduce memory use and run-time [32, 33, 48, 53]. Authors have also focused on dynamic problems [38, 23, 57], where a series of min-cut/max-flow problems are solved in succession and the individual graphs only differ slightly. By reusing computations for one graph, the subsequent graphs can be processed much faster than the initial one. Finally, some authors [14, 50, 51, 58, 57] have explored methods to solve min-cut/max-flow problems in a distributed fashion, where the computations may be split across several computational nodes, which is suited for graphs that are too large to fit in physical memory.

In this paper, we focus on generic min-cut/max-flow algorithms, which do *not* make assumptions about the graph structure (*e.g.*, requiring a grid structure). Furthermore, we consider only static problems, where a solution is calculated once, without access to a previous solution (as opposed to dynamic problems). Finally, we do not consider whether the algorithm works well in a distributed setting, but focus only on

the case where the complete graph can be loaded into physical memory.

The goal is that our experimental results can help researchers understand the strengths and weaknesses of the current state-of-the-art min-cut/max-flow algorithms and help practitioners when choosing a min-cut/max-flow algorithm to use for a given problem.

A. Related work

In this section, we give an overview of current efforts to survey and compare serial and parallel min-cut/max-flow algorithms.

a) Serial algorithms: Several papers [10, 16, 23, 52] provide comparisons of different serial min-cut/max-flow algorithms on a variety of standard benchmark problems. However, many of these benchmark problems no longer reflect the scale and/or structure of modern vision problems solved using min-cut/max-flow, as they mostly consist of small problems and/or grid graphs. Furthermore, the papers only evaluate a subset of algorithms currently available or focus exclusively on the runtime for the min-cut computation. As shown by Verma and Batra [52], it is important for practical use to include the initialization time, as some algorithms (or their implementations) can spend as much time on initialization as on the min-cut computation. Finally, the papers mostly compare reference implementations of the different algorithms. However, as implementation details can significantly impact performance [52], it is difficult to separate the performance of the algorithm from that of the implementation.

b) Parallel algorithms: To our knowledge, a systematic comparison between parallel algorithms for min-cut/max-flow has not been conducted. Typically, papers only compare with a serial algorithm [45, 51, 58] or a single parallel algorithm [5]. The most comprehensive comparison so far was done by Shekhovtsov and Hlaváč [50]. However, due to the lack of a publicly available implementation, no paper compares with the approach by Liu and Sun [45], even though it is suspected to be the fastest [50, 51]. In addition, the papers use the same set of standard benchmark problems as the serial algorithms, which, as previously mentioned, are mostly quite small and thus neither ideal for parallelization, nor representative for the scale of modern CV problems.

B. Contributions

We evaluate generic state-of-the-art serial and parallel min-cut/max-flow algorithms on a number of CV problems with varying graph structures. We compare the algorithms on standard benchmark problems, as well as several new problems, which better reflect the scale of modern vision problems – especially in the realm of 3D image segmentation.

For the serial algorithms, we evaluate the reference implementations of the Hochbaum pseudoflow (HPF) algorithm [27, 28] and the preflow push-relabel (PPR) [22] algorithms. Furthermore, to reduce the influence of the implementation details, we evaluate different versions (including our own) of both the Excesses Incremental Breadth First Search (EIBFS) [23] algorithm and the Boykov-Kolmogorov (BK) [8] algorithm.

For the parallel algorithms, we provide the first comprehensive comparison of all major approaches. This includes our own implementation of the bottom-up merging algorithm by Liu and Sun [45], our own version of the reference implementation of the dual decomposition algorithm by Strandmark and Kahl [51], the reference implementation of the dual the region discharge algorithm by Shekhovtsov and Hlaváč [50], and an implementation of the parallel preflow push-relabel algorithm by Baunstark et al. [5]. In our comparison, we evaluate not just the runtime but also the memory use of the algorithms, which has not previously gotten much attention in the literature, even though memory use is often a limiting factor when working with large problems.

II. Min-cut/max-flow algorithms

Serial min-cut/max-flow algorithms can roughly be divided into three families: augmenting paths, preflow push-relabel, and pseodoflow algorithms. In this section, we provide an overview of how algorithms from each category work. However, before this, we first establish our notation and define the min-cut/max-flow problem.

We define a directed graph $G = (V, E)$ via its set of *nodes*, V, and its set of directed *arcs*, E. We let n and m refer to the number of nodes and arcs, respectively. Each arc $(i, j) \in E$ is assigned a non-negative *capacity* c_{ij}. For min-cut/max-flow problems, we define two special *terminal nodes*, s and t, which are referred to as the *source* and *sink*, respectively. Arcs to and from these nodes are known as *terminal arcs*.

A *feasible flow* in the graph G assigns a non-negative number (a flow), f_{ij}, to each arc $(i, j) \in E$ which satisfies *capacity constraints*, $f_{ij} \leq c_{ij}$, (*i.e.* the flow along an arc cannot exceed its capacity) and *conservation constraints*, $\sum_{(i,j)\in E} f_{ij} = \sum_{(j,k)\in E} f_{jk}$ for all nodes $j \in V \setminus \{s, t\}$ (*i.e.* the flow going into a node must equal the flow coming out). A given feasible flow also induces a *residual graph* where the set of *residual arcs*, R, is given by

$$R = \{(i, j) \mid (i, j) \in E, f_{ij} < c_{ij} \text{ or } (j, i) \in E, f_{ji} > 0\}. \tag{3}$$

Each of the residual arcs has a *residual capacity* given by $c'_{ij} = c_{ij} - f_{ij}$ if $(i, j) \in E$ or $c'_{ij} = f_{ji}$ if $(j, i) \in E$. The *maximum flow* problem refers to finding a feasible flow which maximizes the total flow from the source to the sink.

Finally, an s-t *cut* refers to partition of the nodes into two disjoint sets S and T such that $s \in S$ and $t \in T$. The sets S and T are referred to as the source and sink set, respectively. The *minimum cut* problem, which is the dual of the maximum flow problem, refers to an s-t cut which minimizes the sum of capacities for the arcs going between S and T.

A. Augmenting paths

The augmenting paths (AP) family is the oldest of the three families and was introduced with the Ford-Fulkerson algorithm [17]. AP algorithms always maintain a feasible flow and work by pushing flow along so-called *augmenting paths*, which are paths from s to t in the residual graph. Pushing flow refers to adding a flow, f, to the flow value, f_{ij}, for each arc, (i, j),

along the path. Thus, to maintain a feasible flow, the maximum flow that can be pushed is given by the minimum residual capacity along the path. The algorithms terminate when no more augmenting paths can be found.

The primary difference between various AP algorithms lies in how the augmenting paths are found. For computer vision applications, the most popular algorithm is the Boykov-Kolmogorov (BK) algorithm [8], which works by building search trees from both the source and sink nodes to find augmenting paths and uses a heuristic that favors shorter augmenting paths. The BK algorithm has great performance in practice, but the theoretical runtime bound is worse than other algorithms.

In terms of performance, the BK algorithm has been surpassed by the Incremetal Breadth First Search (IBFS) algorithm by Goldberg et al. [24]. The main difference between the two algorithms is that IBFS maintains the source and sink search trees as breadth-first search trees, which results in both a better theoretical runtime and better empirical performance.

B. Preflow push-relabel

The preflow push-relabel (PPR) algorithms were introduced by Goldberg and Tarjan [22]. These algorithms do not maintain a feasible flow but instead work with a so-called *preflow*. A preflow satisfies capacity constraints but allows nodes to have more incoming than outgoing flow (thereby violating conservation constraints). The difference between the incoming and outgoing flow for a node is denoted as its *excess*, $e_i \geq 0$. Moreover, PPR algorithms also maintain a labelling d_i for every node. For $d_i < n$ it is a lower bound on the distance from node i to t. If $d_i \geq n$ the sink cannot be reached from this node and $n - d_i$ then gives a lower bound on the distance s. Labels for s and t remain fixed at n and 0, respectively.

PPR algorithms work by repeatedly applying one of two actions [12, 22]: *push* and *relabel*. A push operation selects a node with positive excess and pushes flow to a neighbor node with a label of lower value. If no neighbor has a lower label, one can use a relabel operation to increase the label of a node by one. The algorithms terminate when no nodes with positive excess have label $d_i < n$, which means that no more flow can be pushed to the sink. Note that, at this point, only the minimum s-t cut can be extracted. To extract the maximum flow, a second phase of the algorithms must be run. However, this is generally a small part of the runtime [52] and for vision applications we are typically only interested in the minimum cut, so we will not discuss this part further.

The difference between various PPR algorithms lies in the order in which push and relabel operations are performed. Early variants used simple heuristics, such as always pushing from the node with the highest label or using a first-in-first-out queue to keep track of nodes with positive excess. More recent versions [3, 20, 21] use sophisticated heuristics and a mix of local and global operations to obtain significant performance improvements over early PPR algorithms.

C. Pseudoflow

The most recent category of min-cut/max-flow algorithms is the pseudoflow family, which was introduced with the Hochbaum pseudoflow (HPF) algorithm [27, 28]. These algorithms use a so-called *pseudoflow* and do, like the PPR algorithms, not maintain a feasible flow. A pseudoflow satisfies capacity constraints but not the conservation constraints, as it has no constraints on difference between incoming and outgoing flow. As with preflow, we refer to the difference between incoming and outgoing flow for a node as its excess, e_i. A positive excess is referred to as a *surplus* and a negative excess as a *deficit*. The difference between pseudoflow and preflow is that preflow only allows non-negative excesses.

Pseudoflow algorithms work as a hybrid between AP and PPR algorithms. They maintain a forest of trees, where each node with a surplus or deficit is the root of a tree (and all roots must have a surplus or deficit). Let $\mathcal{S}$ and $\mathcal{T}$ denote the forests of trees rooted in s and t, respectively. In each iteration, the algorithms grow these trees by adding nodes with zero excess until an arc, a, which connects a tree from $\mathcal{S}$ to a tree from $\mathcal{T}$. The path from s to t going through a now forms an augmenting path and flow is pushed along it. In contrast to AP algorithms, the only restrictions on how much flow to push are the arc capacities, since pushing flow is allowed to create new deficits or surpluses. If it is not possible to grow a tree – either by adding a free node or finding a connection to a tree in the other forest – the algorithms terminate. As with PPR algorithms, only the minimum cut can be extracted at this point and additional processing is needed to access the flow.

There are two main algorithms in this family: HPF and Excesses Incremental Breadth First Search (EIBFS) [23]. The main differences are the order in which they scan through nodes when looking for an arc connecting $\mathcal{S}$ and $\mathcal{T}$, and how they push flow along augmenting paths. Both have sophisticated heuristics for these choices which makes use of many of the same ideas developed for PPR algorithms – including a distance labelling scheme to select which nodes from each forest to merge.

D. Implementation details

As stressed by [52], the implementation details can significantly affect the measured performance of a given min-cut/max-flow algorithm. In this section we will highlight the trends of modern implementations as well as how they differ.

a) Data structures: The implementations considered in this paper all use a variant of the adjacency list structure [13] to represent the underlying graph. However, the `Arc` and `Node` data structures used for these lists vary between implementations as demonstrated by the differences in the size of the data structures listed in Table I. For all of the implementations we consider the memory footprint of the graph correlates linearly with the size of the `Arc` and `Node` data structures used. Thus, the size of these structures significantly impact the memory footprint of the graph, which in many cases also influence performance due to the way the CPU fetches and caches data from memory. For these reasons it is generally beneficial to keep the data structures small. Note that some compilers do not pack data structures densely by default, which may significantly increase the size of the `Arc` and `Node` data structures.

b) Indices vs. pointers: One way to reduce the size of the `Arc` and `Node` data structures on 64-bit system architectures is to use indices rather than pointers to reference nodes and arcs. As long as the indices can be stored using unsigned 32-bit integers, we can cut the size of the arc and node references in half by using unsigned 32-bit integers instead of pointers (which are 64-bit). This approach can significantly reduce the size of the `Arc` and `Node` data structures, as the majority of the structures consist of references to other arcs and nodes. The downside to this approach is that extra computations are required every time we need to fetch an arc or a node.

c) Arc packing: The order in which the arcs are stored may impact the performance significantly. Again, this is related to CPU cache optimization, where we want to ensure that data is stored in the same order that it is accessed. As the algorithms mostly access data in a specific order (*e.g.* by iterating over all outgoing arcs from a node), it is beneficial to store the arcs in that same order. We refer to this as *arc packaging*. However, as arcs may be added to the graph in any order, packing the arcs usually incurs extra overhead from maintaining the correct ordering or reordering all arcs as an extra step before computing the min-cut/max-flow.

d) Arc merging: In practice, it is not uncommon that multiple arcs between between the same pair of nodes are added to the graph. Merging these arcs into a single arc with a capacity equal to the sum of capacities of the merged arcs may reduce the graph size significantly.

e) Node packing: Similarly to arc packing, *node packing* to match the way they are likely to be accessed by the algorithm may improve performance.

III. Parallel min-cut/max-flow

Like serial algorithms, parallel algorithms for min-cut/max-flow problems can be split into families based on shared characteristics. A key characteristic is whether the algorithms parallelize over individual graph nodes (node-based parallelism) or split the graph into sub-graph, which are then traversed in parallel (block-based parallelism). Other important algorithmic traits include whether the algorithm is distributed, which we are not concerned with in this paper, and the guarantees in terms of convergence, optimality, and completeness provided by the algorithm.

We should note that since many (but not all) min-cut/max-flow problems in CV are defined on grid graphs, several papers [32, 33, 48, 53] have exploited this structure to create very efficient parallel implementations. However, these algorithms are not generic, as they only work for specific graph structures, and thus, are not covered in this paper.

The category of node-based parallel algorithms is generally dominated by parallel versions of PPR algorithms. In the block-based category, we investigate the three main approaches: adaptive bottom-up merging, dual decomposition, and region discharge. In the following sections, we give an overview of each approach and briefly discuss its merits and limitations.

A. Parallel preflow push-relabel

PPR algorithms have been the target of most parallelization efforts [2, 4, 5, 14, 19, 29, 53], since both push and relabel are local operations, which makes them well suited for parallelization. Because the operations are local, the algorithms generally parallelize over each node – performing pushes and relabels concurrently. To avoid data races during these operations, PPR algorithms either use locking [2] or atomic operations [29]. As new excesses are created, the corresponding nodes are added to a queue, from which threads can poll them. In [5], a slightly different approach is applied, where pushes are performed in parallel, but excesses and labels are updated in a separate later step, rather than immediately after the push.

Since parallel PPR algorithms parallelize over every node, they can achieve good speed-ups and scale well to modern multi-core processors [5], or even GPUs [53]. However, synchronization overhead often means that many threads are needed to achieve good performance compared to an efficient serial algorithm.

B. Adaptive bottom-up merging

The adaptive bottom-up merging approach introduced by Liu and Sun [45] uses block-based parallelism and has two phases, which are summarized in Fig. 1. In phase one, the graph is partitioned into a number of disjoint sets (blocks) and arcs between blocks have their capacities set to 0 – effectively removing them from the graph. For each pair of blocks connected by arcs, we store a list of the connecting arcs (with capacities now set to 0) along with their original capacities. Disregarding s and t, the nodes in each block now belong to disjoint sub-graphs and we can compute the min-cut/max-flow solution for each sub-graph in parallel. Similar to the original implementation [45], our implementation uses the BK algorithm.

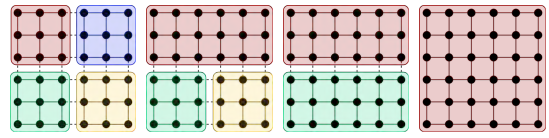


Fig. 1: Illustration of the adaptive bottom-up merging approach for parallel min-cut/max-flow. Terminal nodes and arcs are not shown. Note that the underlying graph does *not* have to be a grid graph. Phase one: (a) The graph is split into blocks and the min-cut/max-flow is computed for each block in parallel. Phase two: (b) The topmost blocks are locked, merged, and the min-cut/max-flow recomputed. (c) As the topmost block is locked, the next thread works on the bottom-most blocks (in parallel). (d) Last two blocks are merged and min-cut/max-flow recomputed to achieve the globally optimal solution.

In phase two, we merge the blocks to get the complete globally optimal min-cut/max-flow. To merge two blocks, we restore the arc capacities for the connecting arcs and then recompute the min-cut/max-flow for the combined graph. This step makes use of the fact that the BK algorithm can reuse its internal search trees [38] to efficiently recompute the min-cut/max-flow when small changes are made to the residual graph for a min-cut/max-flow solution.

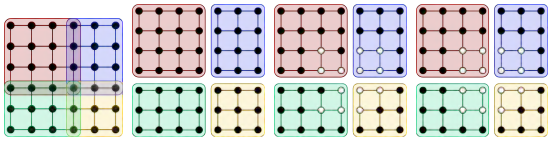
For merges in phase two to be performed in parallel, the method marks blocks being merged as locked. The computational threads then scan the list of block pairs, which were originally connected by arcs, until they find a pair where both blocks are unlocked. The thread then locks both blocks, performs the merge, and unlocks the new combined block. To avoid two threads trying to lock the same block, there is a global lock which ensures that only one thread scans the list of block pairs at a time.

As the degree of parallelism decreases towards the end of phase two – since there are few blocks left to merge – it is beneficial to performance that computationally expensive merges are performed early in phase two. To estimate the cost of merging two blocks, [45] uses a heuristic based on the potential for new augmenting paths to be formed by merging two blocks. This heuristic determines the order in which the blocks are merged.

By using block-based, rather than node-based parallelism, adaptive bottom-up merging avoids much of the synchronization overhead that parallel PPR algorithms suffer from. However, its performance depends on the majority of the work being performed in phase one and in the beginning of phase two, where the degree of parallelism is high. The theoretical speed-up from using more computational threads – determined using Ahmdahl's Law [1] – depends on how much of the computational workload is done in parallel.

C. Dual decomposition

The dual decomposition (DD) approach was introduced by Strandmark and Kahl [51] and later refined by Yu et al. [58, 57]. Their algorithm works as follows: First, the nodes of the graph are divided into a set of overlapping blocks (see Fig. 2a). Then, the graph is split into disjoint blocks, where the nodes in the overlapping regions are duplicated in each block (see Fig. 2b). It is important that the blocks overlap such that if node i is connected to nodes j in block b_j and k in block b_k, then i is also both blocks b_j and b_k.



(a) (b) (c) (d)

Fig. 2: Illustration of the dual decomposition approach. Terminal nodes and arcs are not shown. Note that the underlying graph does *not* have to be a grid graph. (a) Graph nodes are divided into a set of overlapping blocks. (b) The graph is split into disjoint sub-graphs and nodes in overlapping regions are duplicated into each blocks. (c) The min-cut/max-flow for each block is computed in parallel, and the source/sink capacities are adjusted for disagreeing duplicated nodes. (d) The min-cut/max-flow is recomputed and capacities are adjusted until all duplicated nodes agree.

Once the graph has been partitioned into overlapping blocks, the algorithm proceeds iteratively. First, the min-cut/max-flow for each disjoint block is computed in parallel using the BK algorithm. Next, each duplicated node is checked if all duplicates of that node are in the same s-t partitioned set, S or T. In that case we say that the node duplicates agree on their assignment. If all duplicated nodes agree on their assignment, the computed solution is the globally optimal one and the algorithm terminates. If not, the terminal arc capacities for the disagreeing duplicated nodes are updated according to a supergradient[1] ascent scheme and the min-cut/max-flow is recomputed. This process of updating terminal capacities and recomputing the min-cut/max-flow is repeated until all duplicated nodes agree on their assignment.

A limitation of the original dual decomposition approach is that convergence is not guaranteed. Furthermore, [58] and [50] have demonstrated that the risk of nonconvergence increases as the graph is split into more blocks. To overcome this, Yu et al. [58] introduced a new version with a simple strategy that guarantees convergence: If the duplicated nodes in two blocks do not belong to the same set, S or T, after a fixed number of iterations, the blocks are merged and the algorithm continues. This trivially guarantees convergence since, in the worst case, all blocks will merged, at which point the global solution will be computed serially. However, performance often significantly drops when merging is needed for the algorithm to converge, as merging only happens after a fixed number of iterations and all blocks may (in the worst case) have to be merged for convergence.

D. Region discharge

The region discharge approach was introduced by Shekhovtsov and Hlaváč [50], who generalized earlier work by Delong and Boykov [14]. The method first partitions the graph into a set of blocks (called regions in [50]). Each block R also has an associated *boundary* defined as the set of nodes

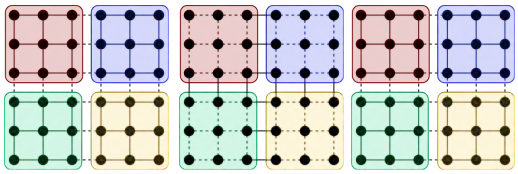
$$B^R = \{v \in V \mid (u, v) \in E, u \in R, v \neq s, t\}. \tag{4}$$

Capacities for arcs going from a boundary node to a block node are set to zero. This means that flow can be pushed *out* of the region into the boundary, but not vice versa. Furthermore, each node is allowed to have an excess.

The method then makes of the *region discharge* operation, which aims to push as much excess flow to the sink or the boundary nodes as possible (the source, s, is assumed to have infinite excess). In [50], this is done either with a PPR algorithm or the BK algorithm. When using the BK algorithm, excesses are modelled as arcs from the source, and the possibility to create excess in a node is modelled by adding an infinite capacity arc from the node to the sink.

The discharge operation is performed on each block in parallel. Afterwards, flow along boundary arcs is synchronized between neighboring blocks, which may create additional excesses in some blocks, as flow can now move from the boundary into the block. The process is repeated until no new excesses are created, at which point the algorithm terminates. It is proved in [50] that this process will terminate in a finite number of iterations.

[1] Analogous to subgradients for convex functions, *e.g.*, see [7].



(a) (b) (c)

Fig. 3: Illustration of the region discharge approach. Terminal nodes and arcs are not shown. Note that the underlying graph does *not* have to be a grid graph. (a) Graph nodes are divided into a set of blocks. Then the region discharge operation is run on each block which pushes flow to the sink or boundary nodes. (b) Flow is synchronized between boundary nodes. (c) Region discharge is run again. The process repeats until no flow crosses the block boundaries.

The guarantee of convergence, without having to merge blocks, is beneficial, as it means the algorithm can maintain a high degree of parallelism while computing the min-cut/max-flow solution. However, because flow must be synchronized between blocks, the practical performance of the method still depends on well-chosen blocks and may be limited by synchronization overhead. For details on the heuristics used in the algorithm, which are also important for its practical performance, see [50].

IV. Performance comparison

We now compare the performance of the algorithms discussed in the previous sections. For all experiments, the code was compiled with the GCC C++ compiler version 9.2.0 with `-O3` optimizations on a 64-bit Linux-based operating system. Experiments were run on a dual socket NUMA[2] system with two Intel Xeon Gold 6226R processors with 16 cores each and HTT[3] disabled, for a total of 32 parallel CPU threads. The system has enough RAM to keep all data in memory for all our experiments. The OS is linux with kernel release 3.10. For the parallel algorithm benchmarks, we ensure that cores are allocated on the same socket and memory is allocated on local RAM where possible.

Run-times were measured as the minimum time over three runs and no other process (apart from the OS) were running during the benchmarks.We split our timing into two distinct phases: build time and solve time. Build time refers to the construction of the graph and any additional data structures used by an algorithm. If the algorithm performs arc packing or similar steps, this is counted in the build time. To ensure the build time is a fair representation of the time used by an algorithm, we pre-compute a list of nodes and arcs and load these lists fully into memory before starting the timing. Solve time refers to the time required to compute the min-cut/max-flow. For algorithms, such as PPR, that only compute a minimum cut we do not include the time to extract the full maximum flow solution. The reason for this is that for most CV applications the minimum cut is of principal interest. Furthermore, converting to a maximum flow solution usually only adds a small extra overhead [52].

[2]Non-Uniform Memory Access
[3]Hyper-Threading Technology

A. Datasets

We test the algorithms on the commonly used benchmark dataset published by the University of Waterloo [46]. This dataset includes a variety computer vision problems, such as stereo matching, image segmentation, multi-view reconstruction, and surface fitting. Furthermore, our experiments include the super resolution, texture restoration, and deconvolution problems from [52]. Finally, we include problems from two recent papers [34, 35] which perform multi-object image segmentation via surface fitting. All benchmark data is available at (DOI link to come).

B. Tested implementations

BK [8] We test the reference implementation (**`BK`**) of the Boykov-Kolmogorov algorithm from http://pub.ist.ac.at/~vnk/software.html. Furthermore, we test our own implementation of BK (**`MBK`**), which contains several optimizations. Most notably, our version uses indices instead of pointers to reduce the memory footprint of the `Node` and `Arc` data structures. Finally, we test a second version (**`MBK-R`**), which reorders arcs such that all out-going arcs for a node are adjacent in memory. This increases cache efficiency but uses more memory and requires an extra initialization step.

EIBFS [23] We test a slightly modified version [30] (**`EIBFS`**) of the excesses incremental breadth first search algorithm originally implemented by [23] available from https://github.com/sydbarrett/AlphaPathMoves. This version uses slightly bigger data structures to support non-integer arc capacities and larger graphs, compared to the implementation tested in [23]. Although these changes may decrease performance slightly, we think it is reasonable to use the modified version, as several of the other algorithms have similar sacrifices in terms of performance. Additionally, we test our own modified version of `EIBFS` (**`EIBFS-I`**), which replaces pointers with indices for a reduced memory footprint. Finally, as both `EIBFS` and `EIBFS-I` performs arc reordering during initialization, we also test a version without arc reordering (**`EIBFS-I-NR`**) to better compare with other algorithms.

HPF [27] We test the reference implementation of Hochbaum pseudoflow (HPF) from https://riot.ieor.berkeley.edu/Applications/Pseudoflow/maxflow.html. This implementation has four different configurations that we test:

1) Highest label with FIFO buckets (H-FIFO).
2) Highest label with LIFO buckets (H-LIFO).
3) Lowest label with FIFO buckets (L-FIFO).
4) Lowest label with LIFO buckets (L-LIFO).

HI-PR [12] We test the implementation of the preflow push-relabel algorithm from https://cmp.felk.cvut.cz/~shekhovt/d_maxflow/index.html[4].

[4]Orignally from http://www.avglab.com/andrew/soft.html, but the link is now dead.

P-ARD [50] We test the implementation of parallel augmenting paths region discharge (P-ARD) from https://cmp.felk.cvut.cz/~shekhovt/d_maxflow/index.html. P-ARD is an example of the region discharge approach.

Liu-Sun [45] Since no public reference implementation is available, we test our own implementation of the adaptive bottom-up merging approach based on the paper by Liu and Sun [45].

P-PPR [5] We test the implementation of parallel push-relabel from https://github.com/niklasb/pbbs-maxflow.

Strandmark-Kahl [51] We test our own implementation of the Strandmark-Kahl dual decomposition algorithm based on the implementation at (link). The original implementation could only handle grid graphs with rectangular blocks. Our implementation can handle arbitrary graphs and arbitrary blocks at the cost of some additional overhead during graph construction. Note that our implementation does not implement the merging strategy proposed by [58] and therefore is not guaranteed to converge. We only include results for cases where the algorithm does converge.

Table I lists the tested implementations along with their type and memory footprint. Their memory footprint can be calculated based on the number of nodes and arcs in the graph and will be discussed further in Section V.

TABLE I: Summary of the tested implementations including their memory footprint. Shows the bytes required per node and arc assuming 32-bit indices where applicable. For arcs, the cost of a forward and backward arc is reported, since some implementations store these as a single arcs and some as two separate arcs. These numbers depend on, but are *not* the same as, the `Node` and `Arc` structure sizes.

Serial algorithms	Algorithm type	Node	Arc
HI-PPR[a] [12]	Preflow push-relabel	40 B	40 B
HPF[b] [27]	Pseudoflow	96 B	25 B
EIBFS [23]	Pseudoflow		
↳`EIBFS`[c]	Pseudoflow	72 B	72 B
↳`EIBFS-I`*	Pseudoflow	29 B	50 B
↳`EIBFS-I-NR`*	Pseudoflow	29 B	24 B
BK [8]	Augmenting path		
↳`BK`[d]	Augmenting path	44 B	64 B
↳`MBK`*	Augmenting path	23 B	24 B
↳`MBK-R`*	Augmenting path	23 B	48 B
Parallel algorithms			
P-PPR[e] [5]	Parallel preflow push-relabel	48 B	68 B
Liu-Sun* [45]	Adaptive bottom-up merging†	25 B	24 B
Strandmark-Kahl* [51]	Dual decomposition†	29 B	24 B
P-ARD[a] [50]	Region discharge†	40 B	32 B

†Uses BK (augmenting path).
*Implemented or updated by us (link will come).
[a]https://cmp.felk.cvut.cz/~shekhovt/d_maxflow/index.html
[b]https://riot.ieor.berkeley.edu/Applications/Pseudoflow/maxflow.html
[c]https://github.com/sydbarrett/AlphaPathMoves
[d]http://pub.ist.ac.at/~vnk/software.html
[e]https://github.com/niklasb/pbbs-maxflow

C. Serial algorithms

The primary experimental results for the serial algorithms are listed in Table II. The table shows the results for the fastest variant of each algorithm. For BK and EIBFS, where several different implementations were tested, only the fastest variant is shown for each row. The results from Table II are summarized in Fig. 4, which show the solve time and total time for each algorithm on each dataset relative to the fastest algorithm on the dataset. Thus, a relative performance score of 0.5 means that the algorithm was half as fast as the fastest algorithm for a given dataset. This gives a clear indication of how well the different algorithms perform relative to each other across all datasets.

From Fig. 4a, we see that EIBFS and HPF outperform the two other algorithms on the majority of the datasets in regard to solve time. Both algorithms have their upper quartiles equal to 1, meaning that they had the fastest solve time for at least 25% of the datasets. As indicated by the median, EIBFS had the fastest solve time for almost half of the datasets, while HI-PR was about five times slower than the fastest algorithm for over half of the datasets. The lower quartiles and whiskers are good indicators of the worst-case scenarios for each algorithm. We see that all four algorithms have cases where they are over four times slower than the fastest algorithm in terms of solve time, and that HPF has slightly better worst case performance than EIBFS.

While EIBFS generally has the fastest solve times, Fig. 4b shows that HPF has the fastest total time (build time + solve time) for the majority of the datasets, as indicated by a median relative performance equal to 1. Furthermore, the lower quartile and whisker indicate that HPF performs consistently well across all datasets, never dropping below 0.5 in relative performance compared to the fastest algorithm and staying above 0.7 for three out of four datasets. BK maintains similar relative performance as for solve time, being around half as fast as the fastest algorithm for the majority of the datasets. Meanwhile HI-PR falls even further behind compared to the results for solve time, with an upper quartile below 0.2 in relative performance, which means that it is at least five times slower than the fastest algorithm for three out of four datasets.

The different variants of each algorithm are compared in Fig. 5, which shows the relative performance of each implementation compared to a chosen "reference" implementation. For the BK algorithm, the `BK` implementation is used for reference, for the EIBFS algorithm, the `EIBFS` implementation is used as a reference, and for HPF the highest label FIFO configuration is used as reference, since it is the one recommended by the authors. As we are now measuring relative to a specific implementation, rather than the fastest one as we did in Fig. 4, it is possible to get a relative performance score of more than one.

For the BK algorithms, both `MBK` and `MBK-R` outperform `BK` for the vast majority of the datasets. `MBK` is consistently faster than `BK`, while `MBK-R` outperforms `MBK` for most of the datasets. Although `MBK-R` significantly outperforms the two other variants, its relative speed-up compared to `BK` is much less consistent than that of `MBK`. This clearly reflects the effect of arc packing (reordering the arcs), in that it typically decrease solve at the cost of increased build time.

The EIBFS variants show similar results to BK, with our index-based version (`EIBFS-I`) outperforming the reference implementation consistently showing over 20% improved

TABLE II: Performance comparison of serial algorithms based on both their solve and total (build + solve) times. Only the fastest variant of each algorithm (measured in total time) is included. The same variant was used for all datasets. The fastest solve time for each dataset has been underlined and the fastest total time has been marked with **bold face**.

			MBK-R [8]		EIBFS-I [23]		HPF-H-L [27]		HI-PR [12]	
Dataset	Nodes	Arcs	Solve	Total	Solve	Total	Solve	Total	Solve	Total
BL06-camel-lrg	18 M	93 M	107.53 s	111.42 s	28.55 s	31.54 s	24.44 s	**28.82 s**	291.71 s	337.91 s
BL06-gargoyle-lrg	17 M	86 M	238.08 s	241.65 s	33.76 s	36.57 s	26.51 s	**30.61 s**	208.27 s	251.10 s
LB07-bunny-lrg	49 M	300 M	8.71 s	22.95 s	6.38 s	**15.25 s**	21.87 s	32.13 s	610.24 s	820.64 s
NT32_tomo3_.raw_10	22 M	154 M	52.86 s	63.15 s	50.82 s	60.01 s	36.46 s	**44.14 s**	645.33 s	741.98 s
NT32_tomo3_.raw_100	183 M	1 B	778.39 s	860.71 s	553.50 s	627.08 s	520.26 s	**583.76 s**	9732.34 s	10618.81 s
NT32_tomo3_.raw_3	7 M	49 M	15.42 s	18.69 s	24.22 s	27.19 s	15.87 s	**18.33 s**	176.11 s	200.29 s
NT32_tomo3_.raw_30	67 M	462 M	145.23 s	**176.37 s**	194.79 s	221.90 s	179.82 s	202.73 s	2939.04 s	3260.63 s
NT32_tomo3_.raw_5	11 M	75 M	23.92 s	**28.84 s**	42.13 s	46.54 s	30.62 s	34.37 s	286.58 s	326.86 s
adhead.n26c10	12 M	327 M	35.97 s	62.76 s	18.94 s	30.26 s	22.48 s	**27.23 s**	174.19 s	370.61 s
adhead.n26c100	12 M	327 M	65.81 s	92.57 s	22.60 s	33.93 s	24.29 s	**29.03 s**	225.38 s	424.67 s
adhead.n6c10	12 M	75 M	10.62 s	14.76 s	8.31 s	**10.91 s**	10.27 s	12.00 s	45.35 s	87.82 s
adhead.n6c100	12 M	75 M	23.88 s	28.03 s	13.23 s	**15.85 s**	14.13 s	15.87 s	59.65 s	102.84 s
babyface.n26c10	5 M	131 M	42.07 s	52.65 s	23.58 s	**28.17 s**	42.87 s	45.08 s	143.68 s	188.10 s
babyface.n26c100	5 M	131 M	82.29 s	92.87 s	30.13 s	**34.74 s**	54.47 s	56.71 s	183.60 s	228.09 s
babyface.n6c10	5 M	30 M	4.28 s	5.94 s	4.36 s	**5.41 s**	10.11 s	10.86 s	46.30 s	58.77 s
babyface.n6c100	5 M	30 M	7.78 s	9.44 s	5.56 s	**6.61 s**	11.56 s	12.24 s	57.28 s	69.66 s
bone.n26c10	7 M	202 M	6.82 s	23.42 s	8.05 s	15.15 s	3.89 s	**6.80 s**	70.42 s	175.85 s
bone.n26c100	7 M	202 M	9.01 s	25.62 s	9.18 s	16.28 s	4.24 s	**7.16 s**	68.39 s	173.75 s
bone.n6c10	7 M	46 M	2.87 s	5.44 s	2.28 s	3.89 s	1.81 s	**2.86 s**	16.61 s	40.11 s
bone.n6c100	7 M	46 M	4.09 s	6.65 s	2.74 s	4.35 s	2.30 s	**3.36 s**	23.66 s	46.71 s
bone_subx.n26c10	3 M	101 M	4.95 s	13.02 s	4.35 s	7.87 s	1.95 s	**3.42 s**	20.15 s	71.88 s
bone_subx.n26c100	3 M	101 M	7.70 s	15.78 s	4.74 s	8.23 s	2.14 s	**3.61 s**	25.51 s	75.15 s
bone_subx.n6c10	3 M	23 M	2.31 s	3.57 s	1.98 s	2.74 s	1.07 s	**1.59 s**	10.48 s	21.72 s
bone_subx.n6c100	3 M	23 M	4.10 s	5.36 s	2.38 s	3.11 s	1.28 s	**1.81 s**	10.34 s	21.49 s
cells.sd2	3 M	30 M	10.25 s	12.71 s	11.32 s	12.68 s	3.49 s	**5.10 s**	24.34 s	39.14 s
foam.subset.r120.h210	8 M	113 M	3.12 s	12.93 s	1.91 s	**7.63 s**	9.25 s	14.20 s	9.24 s	77.57 s
foam.subset.r160.h210	15 M	205 M	6.09 s	23.98 s	3.58 s	**14.10 s**	17.14 s	26.18 s	23.58 s	162.54 s
foam.subset.r60.h210	1 M	24 M	622 ms	2.70 s	422 ms	**1.58 s**	1.98 s	3.01 s	2.02 s	13.37 s
graph3x3	2 K	47 K	9 ms	11 ms	3 ms	3 ms	1 ms	**1 ms**	1 ms	5 ms
graph5x5	2 K	139 K	62 ms	67 ms	6 ms	9 ms	3 ms	**4 ms**	2 ms	15 ms
liver.n26c10	4 M	108 M	6.47 s	15.06 s	8.52 s	12.22 s	4.53 s	**5.31 s**	58.78 s	114.95 s
liver.n26c100	4 M	108 M	11.78 s	20.41 s	10.49 s	14.20 s	5.72 s	**6.50 s**	71.88 s	131.00 s
liver.n6c10	4 M	25 M	4.88 s	6.20 s	4.46 s	5.21 s	4.40 s	**4.93 s**	25.63 s	37.85 s
liver.n6c100	4 M	25 M	10.08 s	11.40 s	5.82 s	6.57 s	5.70 s	**6.24 s**	30.49 s	42.71 s
simcells.sd2	955 K	8 M	1.19 s	1.92 s	574 ms	**1.00 s**	892 ms	1.35 s	5.44 s	9.38 s
simcells.sd3	3 M	35 M	12.69 s	16.38 s	2.94 s	**4.96 s**	3.84 s	5.65 s	22.56 s	38.99 s
super_res-E1	10 K	62 K	2 ms	4 ms	1 ms	**2 ms**	2 ms	3 ms	1 ms	7 ms
super_res-E2	10 K	103 K	5 ms	7 ms	2 ms	3 ms	2 ms	**3 ms**	2 ms	12 ms
super_res-Paper1	10 K	62 K	2 ms	4 ms	1 ms	**2 ms**	2 ms	3 ms	1 ms	7 ms
superres_graph	43 K	742 K	17 ms	55 ms	10 ms	26 ms	7 ms	**12 ms**	19 ms	162 ms
texture-Cremer	44 K	783 K	1.54 s	1.58 s	353 ms	374 ms	168 ms	190 ms	42 ms	**189 ms**
texture-OLD-D103	43 K	742 K	605 ms	645 ms	194 ms	210 ms	73 ms	**92 ms**	41 ms	194 ms
texture-Paper1	43 K	742 K	650 ms	688 ms	186 ms	205 ms	76 ms	**95 ms**	36 ms	171 ms
texture-Temp	14 K	239 K	219 ms	229 ms	30 ms	34 ms	9 ms	**15 ms**	6 ms	32 ms

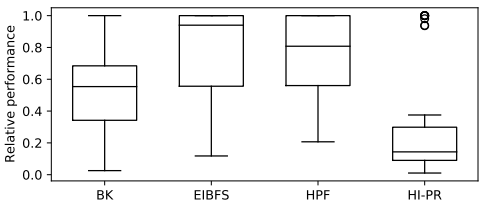
performance for the majority of the datasets and only a few instances with slightly lower performance than EIBFS. Meanwhile, EIBFS-I-NR performs worse than EIBFS on almost all datasets, which further supports the notion that arc packing is important for the performance of the implementation.

For the HPF algorithm, the H-LIFO configuration (HPF-H-L) consistently performs the best, while the L-FIFO and L-LIFO configurations mostly perform worse than the reference H-FIFO configuration. Although there are a few cases where L-LIFO performs much better than the others, it generally performs much worse than both H-LIFO and H-FIFO.

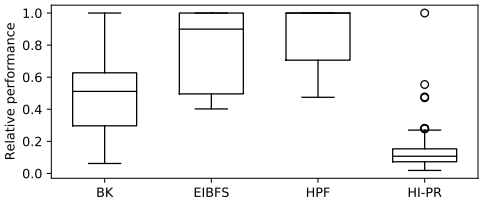
D. Parallel algorithms

Our benchmark results for the parallel algorithms are shown in Table III, which compares the build and solve time for each algorithm on each dataset. We do not include results for many of the small datasets, as the overhead of parallization typically outweighs any performance benefits. The table includes the number of CPU threads used by each of the algorithms for the listed build and solve times. Furthermore, it includes the solve time of the best serial algorithm for each dataset for comparison. Although we have included build time in the table, we focus on the solve time, since we are interested in how well the algorithms are able to distribute work as more threads are added. Additionally, a lack of optimization leads to very long build times for some of the parallel implementations, especially P-PRR, which leads to an unfair comparison. For anyone wanting to use the algorithms in practice, the build time should of course be taken into account.

From Table III, we see that our implementation of the Liu-Sun algorithm performs the best for most of the larger datasets, while the serial HPF algorithm performs better on the smaller datasets. The superiority of the Liu-Sun algorithm among the parallel algorithms is illustrated in Fig. 6, which shows that



(a) Relative solve times.



(b) Relative total times.

Fig. 4: Relative performance for the serial algorithms. For each dataset, the solve and total times for each algorithm was compared to the fastest algorithm for that dataset and a relative time computed. This shows how often an algorithm was fastest and, if it was not fastest, how much slower than the fastest it was.

Liu-Sun is faster than the fastest serial algorithm on half of the datasets and slower on the other half. However, when it is faster, it is often more than twice as fast, while rarely being less than half as fast as the fastest serial algorithm. The other parallel algorithms all have median values well below one, indicating that they are significantly slower than the serial algorithms for the majority of the datasets.

While the parallel algorithms do not outperform the fastest serial algorithm on many of the datasets, it is important to examine which datasets are better suited for the parallel algorithms. Figure 7 shows the relative speed-up for each algorithm on each dataset compared to the size of each problem, represented by the number of nodes. The results show that P-ARD and especially Liu-Sun perform better relative to the serial algorithms as the the size of the problem increases, with Liu-Sun outperforming the fastest serial algorithm several times for the datasets with more than 20 million nodes.

V. Discussion

A. Serial algorithms

Measured on solve time, EIBFS performed the best of the serial algorithms, which aligns with the results presented in [23]. However, if we look at the total time, the HPF algorithm performs the best on the majority of the datasets, which does not align with the results from [23]. We see two possible explanations for this:

1) [23] use the H-FIFO configuration for HPF, which according to our results, is consistently outperformed by H-LIFO. This significantly improves the HPF results in our comparison.
2) For the most of the datasets benchmarked in [23], they use 32-bit pointers, which significantly reduces the size of the `Node` and `Arc` data structures. This generally results in better performance (as demonstrated in our experiments).

Something we cannot explain, is why the total timings from our experiments are not significantly lower than those from [23], given the superior hardware used in our experiments. On the contrary, when using EIBFS, we actually take longer than them to build the graph and solve the problem, even for the `adhead.n6c100` dataset where they too use 64-bit pointers. Even though their EIBFS implementation has a slightly smaller memory footprint than `EIBFS-I` and avoids the overhead from index-based referencing, we do see how this could account for the difference in observed performance. To investigate this further, we would have to also benchmark the exact EIBFS implementation used in [23]. However, as previously mentioned in Section IV-B, this implementation cannot process some of the larger benchmark datasets we have used. In the interest of consistency, we have used the same implementation for all tests.

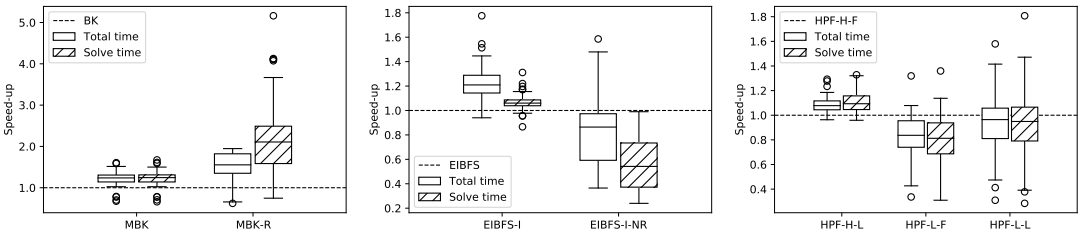
The memory footprint of an algorithm dictates the maximum size of the problems that are practical to solve on a given system. As shown in Table I, `MBK`, and the parallel implementations based on it, has the smallest memory footprint closely followed by `EIBFS-I-NR`. This allows `MBK` to handle datasets over twice the size of some of the other algorithms. However, as demonstrated by the implementations using arc packing, it may be worth using some extra memory to increase performance, given that the problems still fit in system memory. Meanwhile, reducing the size of the `Node` and `Arc` data structures will both decrease the total memory footprint and increase performance. This is demonstrated by the `EIBFS-I`, which outperforms `EIBFS` due to the reduced size of the data structures.

B. Parallel algorithms

The superior performance of the Liu-Sun algorithm, compared to the other parallel algorithms, aligns with previous results [45] and expectations [50, 51] that this approach should perform well on multi-core system. While the algorithm cannot match the serial algorithms on small datasets, it is clear that it scales well on larger datasets, where our results show it being more than five times faster than the fastest serial algorithm.

As expected due to its node-based parallelism, P-PPR scales best with a large number of parallel threads. As shown in Table III, it achieves its best results with 32 threads for nearly all datasets. This is very different to the Strandmark-Kahl algorithm, which never uses more than eight threads for its best results. This is likely due to convergence issues when splitting the graph into many blocks [58]. P-ARD is able to utilize more threads than Strandmark-Kahl, but not nearly as many as Liu-Sun, which could explain why Liu-Sun performs better. In both cases, it is clear that larger datasets allow the algorithms to scale to utilize more threads, which is an important factor if we were to use the algorithms on very large datasets with several billion arcs.

For practical use, only the Liu-Sun and P-ARD implementations appear relevant as is. And of the two, Liu-Sun is clearly



(a) BK (b) EIBFS (c) HPF

Fig. 5: Performance comparison of serial algorithm variants. The solve time and total time is compared against the times for the chosen reference algorithm for each dataset.

TABLE III: Performance of parallel algorithms based on build and solve times. The parallel algorithms were ran with 1, 2, 4, 6, 8, 12, 16, 24, 32, 40, 48, 56, and 64 threads. Only the best time is shown along with the thread count for that run. For comparison, the solve time for the fastest serial algorithm is also included. All times are in seconds. The fastest solve time for each dataset has been marked with **bold face**.

			P-ARD [50]			Liu-Sun [45]			P-PPR [5]			Strandmark-Kahl [51]			Best serial	
Dataset	Nodes	Arcs	Build	Best solve		Build	Best solve		Build	Best solve		Build	Best solve		Alg.	Solve
LB07-bunny-lrg	49 M	300 M	51.78	3.64	8T	6.47	**2.10**	16T	260.38	37.73	32T	9.01	4.51	4T	EIBFS	6.38
NT32_tomo3_.raw_10	22 M	154 M	26.91	28.45	8T	5.12	**19.02**	16T	141.09	27.67	32T	25.42	83.26	2T	HPF	36.46
NT32_tomo3_.raw_100	183 M	1 B	219.76	191.91	16T	40.35	**95.16**	32T	-	-	-	186.61	645.96	4T	HPF	498.94
NT32_tomo3_.raw_3	7 M	49 M	6.34	22.81	1T	1.64	9.94	6T	42.88	**7.79**	32T	7.44	26.26	1T	BK	15.42
NT32_tomo3_.raw_30	67 M	462 M	78.17	85.26	6T	15.28	**47.05**	32T	430.09	98.17	32T	73.04	301.38	2T	BK	145.23
NT32_tomo3_.raw_5	11 M	75 M	12.96	**12.13**	4T	2.45	12.74	12T	66.34	14.23	32T	12.41	36.80	4T	BK	23.92
adhead.n26c10	12 M	327 M	56.63	29.86	4T	7.17	**18.42**	8T	268.67	25.56	12T	27.86	25.55	4T	EIBFS	18.94
adhead.n26c100	12 M	327 M	53.39	37.27	8T	7.22	22.67	8T	262.53	**15.33**	32T	28.77	21.17	6T	EIBFS	22.60
adhead.n6c10	12 M	75 M	13.36	5.14	4T	1.68	**5.08**	4T	65.58	7.35	32T	2.46	6.00	2T	EIBFS	8.31
adhead.n6c100	12 M	75 M	13.07	8.73	4T	1.68	11.23	8T	65.84	9.13	32T	2.71	**7.57**	4T	EIBFS	13.23
babyface.n26c10	5 M	131 M	21.15	38.53	2T	2.98	47.06	4T	102.92	**11.74**	32T	10.58	28.80	4T	EIBFS	23.58
babyface.n26c100	5 M	131 M	22.22	73.55	2T	3.04	79.04	32T	103.81	**15.63**	32T	9.95	42.74	4T	EIBFS	30.13
babyface.n6c10	5 M	30 M	5.03	4.12	2T	0.68	**2.59**	24T	24.96	5.39	24T	0.85	3.88	1T	BK	4.28
babyface.n6c100	5 M	30 M	5.37	7.69	2T	0.68	5.61	4T	24.93	6.93	24T	1.10	**5.33**	4T	EIBFS	5.56
bone.n26c10	7 M	202 M	34.31	8.57	8T	4.54	3.65	16T	159.32	6.62	32T	16.10	10.82	2T	HPF	**3.11**
bone.n26c100	7 M	202 M	34.22	10.84	8T	4.52	4.02	32T	158.84	7.65	32T	16.99	5.26	8T	HPF	**3.48**
bone.n6c10	7 M	46 M	7.94	1.41	8T	1.04	**0.78**	32T	39.02	4.24	32T	1.46	1.84	2T	HPF	1.59
bone.n6c100	7 M	46 M	8.48	1.92	8T	1.04	**1.13**	16T	39.15	5.06	32T	1.47	1.89	2T	HPF	2.04
bone_subx.n26c10	3 M	101 M	18.66	7.14	1T	2.36	3.13	16T	78.56	3.90	32T	7.91	3.82	4T	HPF	**1.95**
bone_subx.n26c100	3 M	101 M	17.29	9.36	1T	2.37	4.09	16T	78.74	4.65	32T	8.27	4.77	8T	HPF	**2.14**
bone_subx.n6c10	3 M	23 M	4.33	2.27	32T	0.52	1.28	16T	19.45	2.97	16T	0.83	1.45	4T	HPF	**1.07**
bone_subx.n6c100	3 M	23 M	4.35	2.90	24T	0.53	2.83	8T	19.62	3.29	16T	0.98	2.16	8T	HPF	**1.28**
liver.n26c10	4 M	108 M	13.12	9.17	1T	2.42	11.01	32T	86.56	9.33	32T	1.02	**3.81**	1T	HPF	4.53
liver.n26c100	4 M	108 M	17.11	15.67	1T	2.44	17.88	6T	85.71	10.48	32T	1.00	7.00	1T	HPF	**5.72**
liver.n6c10	4 M	25 M	4.35	5.51	1T	0.56	4.01	6T	21.21	6.06	16T	0.64	**3.79**	1T	HPF	4.40
liver.n6c100	4 M	25 M	4.41	11.11	1T	0.56	8.31	8T	22.14	6.73	16T	0.64	7.91	1T	HPF	**5.70**

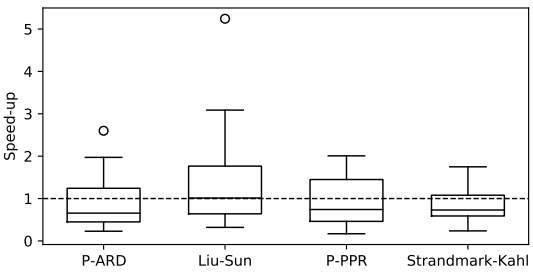


Fig. 6: Speed-up of the parallel algorithms relative to the best serial solve time for each dataset.

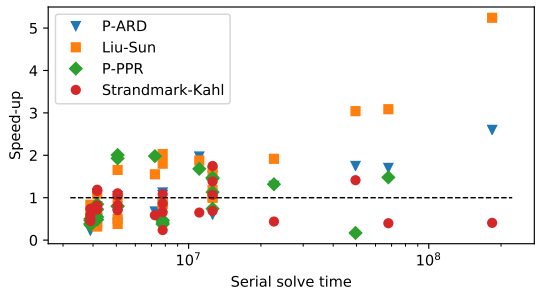


Fig. 7: Plot of the speed-up of the parallel algorithms relative to the best serial solve time for each dataset versus the number of nodes in the datasets.

the superior one, especially when build time is taken into consideration. While the node-based parallelism of P-PPR seems like a good idea in theory, the synchronization overhead results in very poor overall performance. Thus, even with 32 threads, it often cannot match the performance of the serial algorithms. On the other hand block-based parallelism appears to scale well, as long as the problems are large and we have a reasonable way of splitting them into blocks.

The Liu-Sun implementation tested in our experiments relies on the `MBK` implementation, which, as we show, is not as fast as the HPF and EIBFS implementations. As adaptive bottom-up merging can in theory be used with any min-cut/max-flow algorithm, one might consider replacing BK with HPF or EIBFS for improved performance. However, it is important to remember that for the bottom-up merging to be fast, it requires an algorithm which is fast when changing the graph and recomputing the min-cut/max-flow solution. Although EIBFS allows this, our preliminary tests showed that the overhead of changing the graph and updating the solution is significantly higher for EIBFS than BK, which means that parallel EIBFS based on bottom-up merging is often considerably slower than serial EIBFS, even for large datasets. To our knowledge, the HPF implementation tested in the paper does not support dynamically updating the min-cut/max-flow solution. However, if HPF or EIBFS can be implemented such that they can match the performance of BK for dynamic graph problems, they would definitely be suitable for adaptive bottom-up merging.

VI. Conclusions and perspectives

We performed an extensive comparison of the state-of-the-art generic serial and parallel min-cut/max-flow algorithms, some of which have seen extensive use for optimization problems in the computer vision community.

For the serial min-cut/max-flow algorithms, we have tested a total of eight different implementations across four of the fastest and most popular algorithms: PPR, BK, EIBFS, and HPF. These four algorithms include representatives for the three families of min-cut/max-flow algorithms: augmenting paths, push-relabel and pseudoflow.

Our results clearly show that the two pseudoflow algorithms, EIBFS and HPF, are significantly faster than the other algorithms, which leads us to conclude that these two algorithms should be the first choice for anyone looking for a fast min-cut/max-flow algorithm for static computer vision problems. For a comparison with dynamic problems, we refer to [23].

Based on our results, we recommend the HPF algorithm with the H-LIFO configuration, as it performs the best for the majority of the problems when looking at the total time spent building the graph and finding the min-cut/max-flow solution. However, the EIBFS algorithm (`EIBFS-I` implementations) is a very close contender and can easily replace HPF with little impact on performance. The BK algorithm (`MBK` implementation) falls significantly behind the two pseudoflow algorithms, but still provides good performance for most of the problems tested in this paper. Meanwhile, the PPR algorithm represented by HI-PPR is much slower than the three other algorithms and should only be used if there is a specific reason to, or for very small problems where performance and memory usage are irrelevant.

Concerning memory usage, the `MBK` implementation of the BK algorithm and `EIBFS-I-NR` implementation of the EIBFS algorithm are both good options, as they use significantly less memory than the standard EIBFS implementations and HPF.

We tested four different parallel algorithms for min-cut/max-flow problems: parallel PPR (P-PPR), adaptive bottom-up merging (Liu-Sun), dual decomposition (Strandmark-Kahl), and region discharge (P-ARD). Of these, we found adaptive bottom-up merging, as proposed by Liu and Sun [45], to be the best parallel approach, as our implementation of it significantly outperformed the other parallel algorithms overall. Although the other algorithms each has the fastest solve time for at least one of the benchmark problems, their practical use is affected by their slow build times. However, even if the build times were improved through new optimized implementations, Liu-Sun still appears to provide the best combination of parallel scaling and thread utilization. This allows it to keep up with serial `MBK` for smaller problems, while being several times faster than the fastest serial algorithm on large problems.

P-ARD is not able to scale as well with the number of threads as Liu-Sun, while Strandmark-Kahl is only able to utilize up to eight threads. P-PPR on the other hand scales well will the number of threads, however, it still performs worse than Liu-Sun in most cases, even when using several times more parallel threads. Of the four tested parallel implementations, P-ARD is the only one apart from Liu-Sun that appears to be able to consistently outperform the serial algorithm on large problems.

We think there are significant performance improvements to be gained from combining the optimization approaches used by the different implementations. For serial algorithms, faster and better packing of arcs and nodes could help improve the cache efficient and reduce the build time. For the parallel algorithm, it may be possible to use adaptive bottom-up merging with one of the fast pseudoflow algorithms to improve performance. Another option is to create a hybrid Liu-Sun implementation that uses the faster pseudoflow algorithms for the initial min-cut/max-flow computations but switches to BK once the merging starts. Additionally, the parallel implementations should take advantage of parallelism when building the graph.

In conclusion, it is clear that there is a lot of performance to be gained from good implementations of parallel min-cut/max-flow algorithms such as adaptive bottom-up merging when dealing with large optimization problems. Meanwhile, serial algorithms remain superior for smaller size problems due to the overhead associated with parallelism. For the serial min-cut/max-flow algorithms, implementation details, such as data structure sizes and packing also play an important role for the performance and may determine which algorithm ends up with the best performance in practice.

References

[1] Gene M Amdahl. “Validity of the single processor approach to achieving large scale computing capabilities”.

In: *Proceedings of the April 18-20, 1967, spring joint computer conference*. 1967, pp. 483–485.

[2] Richard Anderson and Joao C. Setubal. "A parallel implementation of the push-relabel algorithm for the maximum flow problem". In: *Journal of Parallel and Distributed Computing (JPDC)* 29.1 (1995), pp. 17–26.

[3] Chetan Arora et al. "An efficient graph cut algorithm for computer vision problems". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2010, pp. 552–565.

[4] David A Bader and Vipin Sachdeva. *A cache-aware parallel implementation of the push-relabel network flow algorithm and experimental evaluation of the gap relabeling heuristic*. Tech. rep. Georgia Institute of Technology, 2006.

[5] Niklas Baumstark, Guy Blelloch, and Julian Shun. "Efficient implementation of a synchronous parallel push-relabel algorithm". In: *Proceedings of the European Symposium on Algorithms (ESA)*. 2015, pp. 106–117.

[6] Endre Boros, Peter L Hammer, and Xiaorong Sun. *Network flows and minimization of quadratic pseudo-Boolean functions*. Tech. rep. Technical Report RRR 17-1991, RUTCOR, 1991.

[7] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[8] Yuri Boykov and Vladimir Kolmogorov. "An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 26.9 (2004), pp. 1124–1137.

[9] Yuri Boykov, Olga Veksler, and Ramin Zabih. "Markov random fields with efficient approximations". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1998, pp. 648–655.

[10] Bala G Chandran and Dorit S Hochbaum. "A computational study of the pseudoflow and push-relabel algorithms for the maximum flow problem". In: *Operations Research* 57.2 (2009), pp. 358–376.

[11] Xinjian Chen and Lingjiao Pan. "A survey of graph cuts/graph search based medical image segmentation". In: *IEEE Reviews in Biomedical Engineering (RBME)* 11 (2018), pp. 112–124.

[12] Boris V Cherkassky and Andrew V Goldberg. "On implementing the push—relabel method for the maximum flow problem". In: *Algorithmica* 19.4 (1997), pp. 390–410.

[13] Thomas H Cormen et al. *Introduction to algorithms*. MIT press, 2009.

[14] Andrew Delong and Yuri Boykov. "A scalable graph-cut algorithm for ND grids". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2008, pp. 1–8.

[15] Jan Egger et al. "Nugget-cut: a segmentation scheme for spherically-and elliptically-shaped 3D objects". In: *Joint Pattern Recognition Symposium (JPRS)*. 2010, pp. 373–382.

[16] B. Fishbain, Dorit S. Hochbaum, and Stefan Mueller. "A competitive study of the pseudoflow algorithm for the minimum s–t cut problem in vision applications". In: *Journal of Real-Time Image Processing (JRTIP)* 11.3 (2016), pp. 589–609.

[17] Lester Randolph Ford Jr and Delbert Ray Fulkerson. *Flows in networks*. Princeton university press, 1962.

[18] Daniel Freedman and Petros Drineas. "Energy minimization via graph cuts: Settling what is possible". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2005, pp. 939–946.

[19] Andrew V Goldberg. "Processor-efficient implementation of a maximum flow algorithm". In: *Information Processing Letters* 38.4 (1991), pp. 179–185.

[20] Andrew V Goldberg. "The partial augment–relabel algorithm for the maximum flow problem". In: *Proceedings of the European Symposium on Algorithms (ESA)*. 2008, pp. 466–477.

[21] Andrew V Goldberg. "Two-level push-relabel algorithm for the maximum flow problem". In: *International Conference on Algorithmic Applications in Management*. 2009, pp. 212–225.

[22] Andrew V Goldberg and Robert E Tarjan. "A new approach to the maximum-flow problem". In: *Journal of the ACM (JACM)* 35.4 (1988), pp. 921–940.

[23] Andrew V Goldberg et al. "Faster and More Dynamic Maximum Flow by Incremental Breadth-First Search". In: *Proceedings of the European Symposium on Algorithms (ESA)*. 2015, pp. 619–630.

[24] Andrew V Goldberg et al. "Maximum Flows by Incremental Breadth-First Search". In: *Proceedings of the European Symposium on Algorithms (ESA)*. 2011, pp. 457–468.

[25] Zhihui Guo et al. "Deep LOGISMOS: deep learning graph-based 3D segmentation of pancreatic tumors on CT scans". In: *Proceedings of the IEEE International Symposium on Biomedical Imaging (ISBI)*. 2018, pp. 1230–1233.

[26] Peter L Hammer, Pierre Hansen, and Bruno Simeone. "Roof duality, complementation and persistency in quadratic 0–1 optimization". In: *Mathematical Programming* 28.2 (1984), pp. 121–155.

[27] Dorit S. Hochbaum. "The pseudoflow algorithm: A new algorithm for the maximum-flow problem". In: *Operations Research* 56.4 (2008), pp. 992–1009.

[28] Dorit S. Hochbaum and James B. Orlin. "Simplifications and speedups of the pseudoflow algorithm". In: *Networks* 61.1 (2013), pp. 40–57.

[29] Bo Hong and Zhengyu He. "An asynchronous multithreaded algorithm for the maximum network flow problem with nonblocking global relabeling heuristic". In: *IEEE Transactions on Parallel and Distributed Systems (TPDS)* 22.6 (2010), pp. 1025–1033.

[30] Hossam Isack et al. "Efficient optimization for hierarchically-structured interacting segments (HINTS)". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 1445–1453.

[31] Hiroshi Ishikawa. "Exact optimization for Markov random fields with convex priors". In: *IEEE Transactions*

on Pattern Analysis and Machine Intelligence (PAMI) 25.10 (2003), pp. 1333–1336.

[32] Ondřej Jamriška and Daniel Sỳkora. *GridCut. Version 1.3*. https://gridcut.com. Accessed 2020-06-12. 2015.

[33] Ondřej Jamriška, Daniel Sỳkora, and Alexander Hornung. "Cache-efficient Graph Cuts on Structured Grids". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2012, pp. 3673–3680.

[34] Patrick M. Jensen, Anders B. Dahl, and Vedrana A. Dahl. "Multi-Object Graph-Based Segmentation With Non-Overlapping Surfaces". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2020, pp. 976–977.

[35] Niels Jeppesen et al. "Sparse Layered Graphs for Multi-Object Segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020, pp. 12777–12785.

[36] S Kashyap, H Zhang, and M Sonka. "Accurate Fully Automated 4D Segmentation of Osteoarthritic Knee MRI". In: *Osteoarthritis and Cartilage* 25 (2017), S227–S228.

[37] Anna Khoreva et al. "Simple does it: Weakly supervised instance and semantic segmentation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017, pp. 876–885.

[38] Pushmeet Kohli and Philip H.S. Torr. "Dynamic graph cuts for efficient inference in Markov random fields". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 29.12 (2007), pp. 2079–2088.

[39] Vladimir Kolmogorov and Carsten Rother. "Minimizing nonsubmodular functions with graph cuts-a review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 29.7 (2007), pp. 1274–1279.

[40] Vladimir Kolmogorov and Ramin Zabih. "Computing visual correspondence with occlusions using graph cuts". In: *Proceedings of the International Conference on Computer Vision (ICCV)*. Vol. 2. 2001, pp. 508–515.

[41] Vladimir Kolmogorov and Ramin Zabin. "What energy functions can be minimized via graph cuts?" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 26.2 (2004), pp. 147–159.

[42] Kyungmoo Lee et al. "Multiresolution LOGISMOS graph search for automated choroidal layer segmentation of 3D macular OCT scans". In: *Medical Imaging 2020: Image Processing*. Vol. 11313. International Society for Optics and Photonics. 2020, 113130B.

[43] Victor Lempitsky and Yuri Boykov. "Global optimization for shape fitting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2007, pp. 1–8.

[44] Kang Li et al. "Optimal surface segmentation in volumetric images-a graph-theoretic approach". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)* 28.1 (2005), pp. 119–134.

[45] Jiangyu Liu and Jian Sun. "Parallel Graph-cuts by Adaptive Bottom-up Merging". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 2181–2188.

[46] *Max-flow problem instances in vision*. https://vision.cs.uwaterloo.ca/data/maxflow. Accessed 2021-02-05.

[47] Bo Peng, Lei Zhang, and David Zhang. "A survey of graph theoretical approaches to image segmentation". In: *Pattern Recognition* 46.3 (2013), pp. 1020–1038.

[48] Yi Peng et al. "JF-Cut: A parallel graph cut approach for large-scale image and video". In: *IEEE Transactions on Image Processing* 24.2 (2015), pp. 655–666.

[49] Carsten Rother et al. "Optimizing binary MRFs via extended roof duality". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2007, pp. 1–8.

[50] Alexander Shekhovtsov and Václav Hlaváč. "A distributed mincut/maxflow algorithm combining path augmentation and push-relabel". In: *International Journal of Computer Vision (IJCV)* 104.3 (2013), pp. 315–342.

[51] Petter Strandmark and Fredrik Kahl. "Parallel and Distributed Graph Cuts by Dual Decomposition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2010, pp. 2085–2092.

[52] Tanmay Verma and Dhruv Batra. "MaxFlow Revisited: An Empirical Comparison of Maxflow Algorithms for Dense Vision Problems". In: *Proceedings of the British Machine Vision Conference (BMVC)*. 2012, pp. 1–12.

[53] Vibhav Vineet and P J Narayanan. "CUDA cuts: Fast graph cuts on the GPU". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2008, pp. 1–8.

[54] Yao Wang and Reinhard Beichel. "Graph-based segmentation of lymph nodes in CT data". In: *Proceedings of the International Symposium on Visual Computing (ISVC)*. 2010, pp. 312–321.

[55] Xiaodong Wu and Danny Z Chen. "Optimal net surface problems with applications". In: *Proceedings of the International Colloquium on Automata, Languages, and Programming (ICALP)*. 2002, pp. 1029–1042.

[56] Yin Yin et al. "LOGISMOS—layered optimal graph image segmentation of multiple objects and surfaces: cartilage segmentation in the knee joint". In: *IEEE Transactions on Medical Imaging (T-MI)* 29.12 (2010), pp. 2023–2037.

[57] Miao Yu, Shuhan Shen, and Zhanyi Hu. "Dynamic Graph Cuts in Parallel". In: *IEEE Transactions on Image Processing* 26.8 (2017).

[58] Miao Yu, Shuhan Shen, and Zhanyi Hu. "Dynamic Parallel and Distributed Graph Cuts". In: *IEEE Transactions on Image Processing* 25.12 (2015), pp. 5511–5525.

3.6 Paper C: Faster Multi-Object Segmentation using Parallel Quadratic Pseudo-Boolean Optimization

In the third contribution, we present our own parallel version QPBO algorithm. Our parallel QPBO algorithm combines the QPBO optimization techniques by Kolmogorov and Rother 2007 with the bottom-up merging strategy by Jiangyu Liu and Jian Sun 2010 discussed in Paper B. Using our parallel QPBO algorithm with SLGs, we can solve the large segmentation task from Paper A in less than two minutes, which is about ten times faster than using serial QPBO.

The included paper is the preprint submitted to the International Conference on Computer Vision (ICCV) 2021. It is currently awaiting peer review.

Faster Multi-Object Segmentation using Parallel Quadratic Pseudo-Boolean Optimization

Niels Jeppesen, Patrick M. Jensen, Anders N. Christensen, Anders B. Dahl, and Vedrana A. Dahl
Department of Applied Mathematics and Computer Science
Technical University of Denmark, Kgs. Lyngby, Denmark
{niejep,patmjen,anym,abda,vand}@dtu.dk

Abstract

We introduce a parallel version of the Quadratic Pseudo-Boolean Optimization (QPBO) algorithm for solving binary optimization tasks, such as image segmentation. The original QPBO implementation by Kolmogorov and Rother relies on the Boykov-Kolmogorov (BK) maxflow/mincut algorithm and performs well for many image analysis tasks. However, the serial nature of their QPBO algorithm results in poor utilization of modern hardware. By redesigning the QPBO algorithm to work with parallel maxflow/mincut algorithms, we significantly reduce solve time of large optimization tasks. We compare our parallel QPBO implementation to other state-of-the-art solvers and benchmark on two large segmentation tasks and a substantial set of small segmentation tasks. The results show that our parallel QPBO algorithm is up to 17 times faster than the fastest serial QPBO algorithm on the large tasks and over three times faster for the majority of the small tasks. Although we focus on image segmentation, our algorithm is generic and can be used for any QPBO problem. Our implementation and experimental results are available at: (link will come).

1. Introduction

Computational parallelism is essential to the performance and thereby the usefulness of many image segmentation algorithms. The best example is perhaps deep learning, which owes much of its success to highly efficient parallel implementations of the matrix operations used during both training and inference. However, not all algorithms used in computer vision rely on easily parallelizable matrix operations.

Graph cut algorithms are popular for solving binary optimization problems in image analysis, due to their speed and guarantee of optimality. Thus, they provide efficient solutions to a variety of computer vision problems – on their own [8, 9, 11, 16, 23, 26, 32, 34], or in combination with other methods [18, 29, 30]. While some popular graph cut algorithms have been parallelized [11, 35, 39, 41], other algorithms have remained serial, which severely limits their ability to utilize modern hardware. An example is the Quadratic Pseudo-Boolean Optimization (QPBO) algorithm [7, 19, 32, 37], which allows nonsubmodular energy terms, making it particularly useful for instance segmentation. Instance segmentation without training data is common in microscopy and material science, where manually labeling large volumetric datasets can be highly impractical. Often, the input needed for segmentation with QPBO can be obtained much easier.

In this paper, we introduce the first parallel QPBO (P-QPBO) algorithm. Our goal is to provide an efficient and scalable algorithm, which can take advantage of modern multi-core processors. With our P-QPBO algorithm, results can be obtained over an order of magnitude faster than with previous serial methods, and the scale of the tasks can be increased significantly. It enables us to segment a volume with hundreds of interacting 3D objects in minutes based on limited user input and no training data. Although we only demonstrate the advantage of P-QPBO for image segmentation, P-QPBO can be used for any QPBO problem.

This work focuses on our parallel algorithm and its time and memory efficiency on image segmentation tasks, and thus the formulation of suitable energy functions for specific computer vision tasks is outside the scope of this paper.

1.1. Related work

The QPBO algorithm [7, 19], as implemented by Kolmogorov and Rother [32, 37], relies on the serial Boykov-Kolmogorov maxflow/mincut (BK Maxflow) algorithm [9] for solving optimization tasks. Generally, maxflow/mincut algorithms can be separated into three groups [40]: push-relabel algorithms [3, 10, 14, 15], augmenting path algorithms [9, 17], and pseudoflow algorithms [16, 20, 21], which are a hybrid of the two previous categories. BK

Maxflow is an augmenting path algorithm. It is the most popular maxflow/mincut algorithm in computer vision, due to its performance and ability to handle dynamic maxflow/mincut scenarios, by reusing computations from previous solutions when changes are made to the graph [31]. In the last decade, pseudoflow algorithms like Excesses Incremental Breadth-First Search (EIBFS) [16] have outperformed BK Maxflow in most static cases, as well as some dynamic cases. However, the overhead associated with graph changes for dynamic problems is still higher for EIBFS than for BK Maxflow, giving BK an advantage for highly dynamic tasks.

Push-relabel algorithms have traditionally been the target of most parallelization efforts [2, 5, 11, 13, 22, 41], as operations mainly act locally, making them well-suited for parallel execution. However, synchronization overhead means that many threads are needed to achieve good performance [6, 38]. More recent works [35, 38, 39, 42, 43] have focused on parallelizing augmenting path algorithms. Here, a graph is partitioned into multiple sub-graphs and a serial algorithm is applied to each sub-graph in parallel. Information is then propagated between sub-graphs, or they are merged. This process is repeated until a global solution is found. Parallel pseudoflow algorithms have not been attempted yet.

Finally, as grid-based graphs are relatively common in computer vision, algorithms specialized for this structure, such as Grid-Cut [24, 25] and CUDA cuts [41], have also been developed. Assuming a grid structure allows for optimizations that significantly improve performance. However, these algorithms are only usable on a limited (but certainly important) subset of binary optimization problems. In this paper, we are concerned with a general-purpose parallel QPBO algorithm and will therefore not discuss the grid-based algorithms further.

1.2. Contribution

We introduce a fast parallel algorithm for solving QPBO problems. It is based on the efficient two-stage approach of the QPBO algorithm as presented in [32] and the bottom-up merging approach from [35]. Our algorithm is fully compatible with the original QPBO algorithm and we prove that it is guaranteed to find equivalent solutions.

We show that our parallel algorithm reduces the solve time significantly on a large multi-object 3D segmentation task compared to current state-of-the-art approaches. We also benchmark our algorithm for segmentation using a large set of 2D images and show significant performance improvements, even for smaller segmentation tasks.

Our C++ implementation, along with a wrapper for Python and benchmark code, is available at (link will come).

2. QPBO

We briefly summarize the original QPBO algorithm here. Both the QPBO algorithm and general-purpose maxflow/mincut algorithms, can be used to minimize energy functions of the form

$$E(\mathbf{x}) = \sum_{p\in\mathcal{V}} \theta_p(x_p) + \sum_{p,q\in\mathcal{V}} \theta_{pq}(x_p, x_q)\,. \qquad (1)$$

Here, $\mathcal{V}$ is a set of nodes, $x_p \in \{0,1\}$ are the node labels, θ_p are unary energy terms, and θ_{pq} are pairwise energy terms. If E is submodular, meaning that all pairwise energies satisfy

$$\theta_{pq}(0,0) + \theta_{pq}(1,1) \leq \theta_{pq}(0,1) + \theta_{pq}(1,0)\,, \qquad (2)$$

maxflow/mincut-based algorithms (including QPBO) are guaranteed to find the global optimal solution to the minimization problem [9, 32]. However, non-submodular energy terms cannot be represented as edges in the graph [12, 33]. To overcome this limitation, the QPBO algorithm uses a primal-dual graph approach, in which every node is represented by two graph nodes: a primal node $p \in \mathcal{V}$ and a dual node $\overline{p} \in \overline{\mathcal{V}}$. Every energy term is then represented by two graph edges (see Table 1). This allows the non-submodular terms to be represented as graph edges between the primal and the dual graph. Computing the maximum flow/minimum cut of this primal-dual graph, corresponds to minimizing the energy function [32].

Table 1: Conversion from energy terms to graph edges [32]. Here, $p, q \in \mathcal{V}$ represent nodes from the primal graph, $\overline{p}, \overline{q} \in \overline{\mathcal{V}}$ are the corresponding nodes of the dual graph, and s, t represent, respectively, the source and sink nodes.

Energy term	Corresponding edges	Edge capacity
$\theta_p(0)$	$(p \to t), (s \to \overline{p})$	$\frac{1}{2}\theta_p(0)$
$\theta_p(1)$	$(s \to p), (\overline{p} \to t)$	$\frac{1}{2}\theta_p(1)$
$\theta_{pq}(0,1)$	$(p \to q), (\overline{q} \to \overline{p})$	$\frac{1}{2}\theta_{pq}(0,1)$
$\theta_{pq}(1,0)$	$(q \to p), (\overline{p} \to \overline{q})$	$\frac{1}{2}\theta_{pq}(1,0)$
$\theta_{pq}(0,0)$	$(p \to \overline{q}), (q \to \overline{p})$	$\frac{1}{2}\theta_{pq}(0,0)$
$\theta_{pq}(1,1)$	$(\overline{q} \to p), (\overline{p} \to q)$	$\frac{1}{2}\theta_{pq}(1,1)$

However, the support for non-submodular terms means that the guarantee of finding the global optimal solution is replaced by one of finding a partial optimal solution [32]. This means that we may get a solution with unlabeled nodes, which may lead to bad segmentation results [23]. However, when non-submodular terms are used only for exclusion, [26] has shown that unlabeled nodes are rare and hardly affect the resulting segmentation.

The original QPBO algorithm uses a two-stage approach to reduce the solve time [32]. In the first stage, only the primal graph is created and the maxflow solution is computed without adding edges corresponding to non-submodular terms. In the second stage, the dual graph is created by copying the residual primal graph and reversing the edges. Afterwards, the edges for the non-submodular terms are added and the solution is updated. This two-stage approach reduces the solve time significantly, but it relies on maxflow solvers that can handle dynamic graphs efficiently, such as the BK algorithm [31].

3. Parallel QPBO

We now describe our new Parallel QPBO (P-QPBO) algorithm, which combines the two-stage QPBO approach described in Section 2 with the bottom-up merging parallelization approach by Liu and Sun [35]. Judging from previous work [35, 38, 39, 42, 43], bottom-up merging provides good performance on non-distributed multi-core systems. Like Liu and Sun's algorithm, ours has two phases. In Phase I, the QPBO problem is split into disjoint subproblems, which are solved independently in parallel. In Phase II, these partial solutions are merged and re-solved, also in parallel, to get the complete solution.

In contrast to Liu and Sun's algorithm, which strictly works as a maxflow/mincut solver, our algorithm also considers each sub-problem as a QPBO problem. Specifically, each sub-problem is kept in Stage 1 (*i.e.* we only consider the primal graph) as long as it contains only submodular terms. Thus, in the case of few non-submodular terms, most sub-problems will remain in Stage 1 for most of Phase I and II, which delays the introduction of the dual graph. This significantly reduces the solve time. We now provide a detailed description of the two phases. Here, we also describe the specific conditions, which will trigger a conversion to Stage 2 for a sub-problem. A visual summary is shown in Figure 1.

Phase I: Partitioning of the QPBO problem is done by splitting the underlying primal-dual graph $\mathcal{G}$. We split the primal node set $\mathcal{V}$ into N disjoint sets $\mathcal{V}_1, \mathcal{V}_2, ..., \mathcal{V}_N$. This gives a partition of the graph nodes into blocks $\mathcal{W}_1, \mathcal{W}_2, ..., \mathcal{W}_N$ where $\mathcal{W}_i = \{p, \overline{p} \mid p \in \mathcal{V}_i\}$. Then, for each pair of blocks $\mathcal{W}_i, \mathcal{W}_j$ connected by one or more edges, we identify inter-block edges and store these in separate lists. From now on, we refer to these edge lists as *block interfaces*. After building the block interfaces we remove the corresponding edges from $\mathcal{G}$.

We now have a series of sub-graphs $\mathcal{G}_i = \langle \{\mathcal{W}_i, s, t\}, \mathcal{E}_i \rangle$ where $\mathcal{E}_i = \{(p \rightarrow q) \in \mathcal{E} \mid p, q \in \mathcal{W}_i\}$ and $\mathcal{E}$ is the set of all edges in $\mathcal{G}$. Because these sub-graphs are disconnected, except through the source and sink nodes, we can compute their individual maxflow solutions in parallel (see Figure 1a). For each sub-graph, we adapt the two-stage approach from the serial QPBO algorithm. First, we only consider the primal graph without adding edges from non-submodular terms. Then, if (and only if) a sub-graph contains non-submodular terms, we transition the sub-graph to Stage 2. During this transition, the dual graph is constructed by copying the primal graph, the remaining non-submodular edges are added, and the maxflow solution is updated (see Figure 1b). When all sub-graphs have been solved, we move to Phase II.

Phase II: In this phase we merge the sub-graphs to recreate the original complete graph, $\mathcal{G}$. Merging two sub-graphs is done by re-adding the inter-block edges, which were removed at the beginning of Phase I. If all sub-graphs and inter-block edges correspond to submodular terms, we only add primal edges and keep both sub-graphs in Stage 1 (see Figure 1c). If some of the inter-block edges correspond to non-submodular terms, both sub-graphs are transformed to Stage 2 (see Figure 1d) before the inter-block edges are added (see Figure 1e). Furthermore, if the two sub-graphs are in different stages, the sub-graph in Stage 1 is transformed to Stage 2 before inter-block edges are re-added and the sub-graphs are merged (see Figure 1f). After merging, the solution of the combined graph is updated.

To further reduce the solve time, we want merges to happen in parallel, for which we use the strategy from [35]. As updating the maxflow solution is a serial process, only one thread can work on a sub-graph at a time. For synchronization, each sub-graph can be locked (meaning it is being worked on) or unlocked (meaning it is free for merging).

To decide which sub-graphs to merge, each thread scans through the list of block interfaces created in Phase I, until it finds one that connects two unlocked sub-graphs. The thread then locks the sub-graphs and merges them. Then, it re-computes the maxflow solution for the merged sub-graph and the sub-graph is unlocked. Note that after sub-graphs have been merged, there may be several block interfaces connecting the previously merged sub-graphs. Therefore, when a thread finds a pair of sub-graphs to merge, it continues to scan the list of block interfaces to find all block interfaces connecting the pair. The block interfaces are then removed from the global list and the merge proceeds. We use a global synchronization object to ensure that only one thread can scan the list of block interfaces at a time.

At the end of Phase II, the number of remaining merges will be less than the number of running threads (unless only one thread is used). Therefore, if a thread scans the whole list of block interfaces without encountering a pair of unlocked sub-graphs, it terminates. As a result, the degree of parallelism is gradually reduced near the end of Phase II. However, for most problems, the time required for the last merge will be small compared to the total solve time. In total, the number of merges performed will be one less than the number of blocks. The process for each thread is sum-

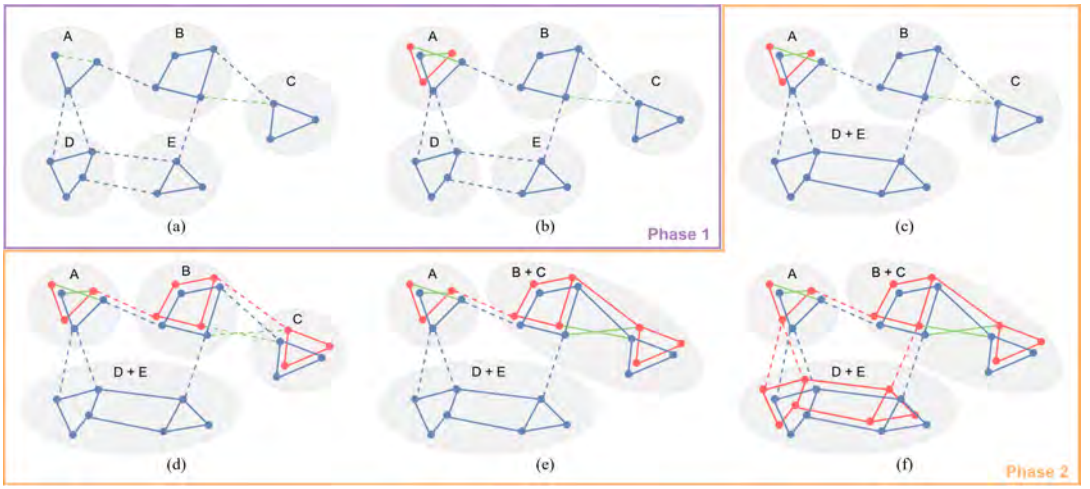


Figure 1: Illustration of merging strategy. Blue dots and lines represent nodes and edges in the primal graph, while red dots and lines are nodes and edges in the dual graph. Green lines represent edges between the primal and dual graphs, corresponding to non-submodular terms. Dashed lines represent edges between sub-graphs, which are re-added when the sub-graphs are merged. The exception is green dashed lines between two primal (blue) nodes. These represent non-submodular energy terms, which will be translated to edges, once the dual graphs are added. Source/sink nodes and edges are not shown. (a) The graph is split into sub-graphs and the maxflow solution for each primal sub-graph is computed in parallel. (b) Sub-graph A contains internal non-submodular terms, so it is transformed to Stage 2 and the solution is updated. (c) Sub-graph D and E are merged. Inter-block edges are re-added and the sub-graph solution is updated. As all intra- and inter-block terms are submodular the sub-graph remains in Stage 1. (d) A term between B and C is non-submodular, so the sub-graphs are transformed to Stage 2 to prepare for merge. (e) Sub-graphs B and C are merged. (f) Sub-graph D + E is transferred to Stage 2 to allow merges with the remaining sub-graphs. All sub-graphs are now in Stage 2, and merging can proceed as normal bottom-up merging.

marized in Algorithm 1.

3.1. Correctness

Our P-QPBO algorithm will always give a solution equivalent to that of the serial QPBO algorithm.

Energy: The energy of the solution is given by the unique value of the minimum cut for the primal-dual graph. Since the final graph is identical for the serial and parallel algorithms, and they both compute a minimum cut, the solutions must have the same energy.

Labeling: There may be several min-cuts which have the same cost/energy but label a different number of nodes [32]. However, given a residual graph, the algorithm from [4, 32] will choose the min-cut that labels the maximum number of nodes. It can be shown (proof in supplementary material) that since both QPBO and P-QPBO compute a min-cut of the same graph, they must label the same nodes after running this extra algorithm. Since this extra step is an insignificant part of the overall runtime, we do not include it in our runtime experiments.

3.2. Efficient graph partitioning and merging

The partitioning of the graph nodes into blocks is important for the performance of the P-QPBO algorithm. Ideally, we want as much work as possible to be done in Phase I (computing the partial solutions) and as little as possible to be done in Phase II (merging sub-graphs and updating solutions). A good way to achieve this is to separate the nodes into blocks that are densely packed (many intra-block terms) and sparsely related (few inter-block terms). This speeds up the merging by reducing the number of changes made to the graph. However, the ideal partitioning very much depends on the energy function.

For image segmentation, we can use the spatial position of the nodes/pixels when partitioning them into blocks. Cutting the image into evenly sized rectangular blocks, as done by [35, 39] should result in many intra-block terms, compared to inter-block terms, as long as the blocks are not very small. When solving instance segmentation tasks using Sparse Layered Graphs (SLG) [26], an intuitive way to partition the nodes is to create a block per label/object. This

Algorithm 1: Phase II of the parallel QPBO algorithm for each thread.

while true **do**
  Lock synchronization object
  Let $S = \emptyset$
  foreach block interface s **do**
    Let $\mathcal{G}_i$ and $\mathcal{G}_j$ be sub-graphs connected by s
    if both $\mathcal{G}_i$ and $\mathcal{G}_j$ are unlocked **then**
      $S = \{s_i \mid s_i \text{ connects } \mathcal{G}_i \text{ and } \mathcal{G}_j\}$
      break
  Remove entries of S from list of block interfaces
  if S is empty **then**
    Unlock synchronization object
    return
  Lock sub-graphs $\mathcal{G}_i$ and $\mathcal{G}_j$ connected by S
  Unlock synchronization object
  / Ensure sub-graphs are in stage 2 if needed. */*
  if S contains non-submodular terms **or** $\mathcal{G}_i$ in stage 2 **or** $\mathcal{G}_j$ in stage 2 **then**
    if $\mathcal{G}_i$ in stage 1 **then** Transform $\mathcal{G}_i$ to stage 2
    if $\mathcal{G}_j$ in stage 1 **then** Transform $\mathcal{G}_j$ to stage 2
  Unite blocks $\mathcal{G}_i$ and $\mathcal{G}_j$ to block $\mathcal{G}_{ij}$
  / Re-insert boundary edges */*
  foreach boundary edge e in S **do**
    Reinsert e in graph
    if $\mathcal{G}_{ij}$ in stage 2 **then**
      Reinsert dual edge of e in graph
    if nodes of a have different labels **then**
      Mark nodes of reinserted edges as active
  Update maxflow for subgraph $\mathcal{G}_{ij}$
  / Make $\mathcal{G}_{ij}$ available for merges */*
  Lock synchronization object
  Unlock block $\mathcal{G}_{ij}$
  Unlock synchronization object

works well when the interaction between the objects is low compared to the size of the objects (which is usually the case), and we have at least as many objects as the number of parallel threads available on the system. We use this natural way of partitioning the nodes for all our experiments, as most of our images contain many objects.

For determining the merging order, P-QPBO uses the same approach as Liu and Sun [35]. After Phase I, we loop over each block interface and count the number of potential new augmenting paths, when merging the sub-graphs containing the blocks. This serves as a heuristic for how much work must be done when merging the sub-graphs. The list of block interfaces is then sorted in descending order based on the number of potential new augmenting paths, in the hope that threads will perform the most expensive merges first. The goal is to do as much work as possible early in Phase II, while the degree of parallelism is high.

4. Benchmark results

To test the scalability of our P-QPBO algorithm, we compare it with two serial QPBO implementations. The first is a slightly optimized implementation of the original QPBO algorithm by Kolmogorov – we call it K-QPBO. The reason we are using a slightly modified version is that the original implementation has overflow issues for large graphs. The second serial implementation is our own re-implementation of K-QPBO, which contains numerous improvements in data structures and optimizations of the code that improves performance. We call this implementation Modern QPBO (M-QPBO). M-QPBO is included to provide a more fair comparison between a serial and parallel implementation since M-QPBO contains the same performance optimization as P-QPBO. When referring to results for our parallel implementation, we use the notation P-QPBO(t), where t is the number of parallel threads used by P-QPBO.

We test the QPBO implementations on the two datasets used in [26], and use the exact energy functions shared in [27]. Our notebooks (based on [27]), used to formulate the energy functions and to benchmark the QPBO algorithms, are included in our supplementary material (and will be shared online). However, as our focus in this paper is purely on the computational performance, the energy formulations are not included in the paper.

The first dataset used for our experiments is a high-resolution μCT 3D image of nerves [28] shown to the left in Figure 2. This is a large segmentation task with many non-overlapping objects. It allows us to test the scalability of the parallel QPBO implementation across many CPU threads. The second dataset is the BBBC038v1 nuclei image set from the Broad Bioimage Benchmark Collection [36]. An image from the dataset along with the instance segmentation results is shown to the right in Figure 2. Using these images, we test the performance of the QPBO implementations on a variety of small and medium-sized segmentation tasks. Unlike general maxflow problems, where a number of commonly used benchmark datasets exist [16], there are no commonly used benchmark datasets specifically for QPBO.

We use two Intel Xeon Gold 6226R (16 cores / 16 threads) CPUs in dual socket configuration for all our benchmarks. With this, we test how our implementation scales on a modern architecture with up to 32 threads executing in parallel.

4.1. Large segmentation tasks

The goal of this experiment is to compare the solve times of the K-QPBO and M-QPBO to those of P-QPBO at various parallel thread counts on large segmentation tasks. Although solve times vary between system architectures, this experiment shows the benefit of using P-QPBO, depending

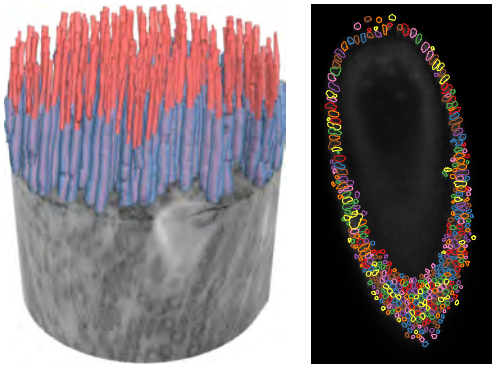


Figure 2: Left: Result of N1 nerve segmentation task. Nodes are split into blocks such that all nodes associated with either the inner (red) or outer (blue) surface of a nerve are in the same block (two blocks per nerve). Right: Example of nuclei segmentation on an image from the Broad Bioimage Benchmark. Nodes are split into blocks such that all nodes associated with a cell are in the same block (one block per cell).

on the number of CPU cores available.

In the experiment, we segment the myelin and axon of 216 nerves in a $2048 \times 2048 \times 2048$ volume at two different radial sampling resolutions, using the SLG method of [26]. The first resolution (N1) is the one used by [26], while the second resolution (N2) is higher, resulting in a graph more than twice the size of N1 (see Table 2). In both cases, the output is a 3D multi-label segmentation with a total of 432 interacting objects (two per nerve). For P-QPBO we use one block per object, see Figure 2, left.

As shown in Table 2, our M-QPBO implementation reduces the solve time for the N1 task by 24% compared to the K-QPBO implementation, while P-QPBO(1) outperforms M-QPBO slightly for this task, with a 34% reduction compared to K-QPBO, using only a single thread. Using two threads, P-QPBO(2) provides a 62% reduction in solve time, compared to K-QPBO, and a 50% reduction compared to M-QPBO. The best result is achieved using 40 threads, in which case P-QPBO is over 10 times faster than K-QPBO. Figure 3a show the relative speed-up when using P-QPBO compared to K-QPBO. We see that the performance increases up to, and even beyond, the number of CPU cores (32) in our test system.

For the larger N2 task, we observe even larger performance improvements and better scaling of P-QPBO than for N1 (see Figure 3b). From the solve times in Table 2, we see that the bottom-up merging strategy, even without parallelism, provides a reduction in solve time of 48% for P-QPBO(1) compared to K-QPBO. Meanwhile,

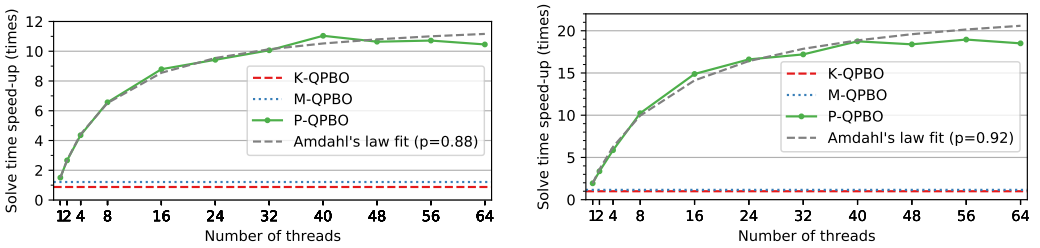
Table 2: Graph details for the nerve segmentation tasks. Nodes and edges refer to the size of the full primal-dual graph. The memory footprint is the total memory footprint of the graph and relevant bookkeeping, which depends on the number of nodes and edges. P-QPBO and M-QPBO use 32-bit indices for N1, but 64-bit edge indices for N2, because it has more than 2^{31} edges. K-QPBO always uses 64-bit pointers for indexing. The solve times are shown for each of the three algorithms, with a number of different thread configurations for P-QPBO. Each solve time is the minimum of ten runs for N1 and three runs for N2.

	N1	N2
Nodes	363,748,800	818,434,800
Edges	2,124,073,474	4,864,255,488
Memory footprint		
K-QPBO [32]	134.2 GB	306.8 GB
M-QPBO	60.1 GB	182.7 GB
P-QPBO	70.0 GB	224.9 GB
Fastest solve time		
K-QPBO [32]	844 s	4,534 s
M-QPBO	638 s	3,897 s
P-QPBO (1)	561 s	2,338 s
P-QPBO (16)	96 s	305 s
P-QPBO (32)	83 s	264 s
P-QPBO (40)	**76 s**	242 s
P-QPBO (56)	79 s	**239 s**

M-QPBO only provides a 14% reduction over K-QPBO. In other words, P-QPBO clearly improves its relative performance as the task grows, while M-QPBO actually performs slightly worse for N2 than for N1, when looking at the relative improvement over K-QPBO. One explanation for the reduced relative performance improvement of M-QPBO on N2 could be the change to 64-bit indices for edges, which results in a larger relative memory footprint, as shown in Table 2.

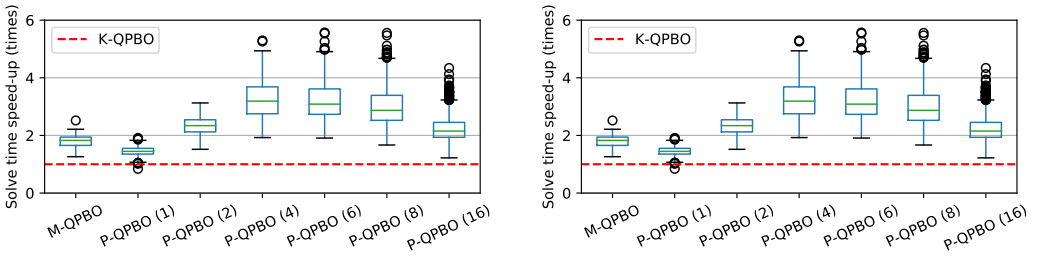
Both Figures 3a and 3b show that the speed-up increases significantly less past 16 threads. This is expected, as we are testing on a dual socket system, which means we are likely to experience some degree of computational overhead when using both CPUs, especially for cache and memory intensive tasks such as computing the maximum flow. Yet, despite the overhead, the combined 32 CPU cores allow P-QPBO to scale past 32 threads for both N1 and N2, with P-QPBO(40) significantly outperforming P-QPBO(32) in both cases. This is perhaps a result of some threads idling while waiting for the synchronization lock to be released.

Another reason for the way P-QPBO scales with the number of threads is the reduction in the degree of paral-



(a) N1 (b) N2

Figure 3: Plots showing the relative speed-up in the solve time when using P-QPBO compared to K-QPBO and M-QPBO. The relative speed-up is calculated using the minimum of ten runs for N1 and three runs for N2. K-QPBO and M-QPBO are represented as horizontal lines, as they always use a single thread. For P-QPBO, we show results with the number of parallel threads ranging from one to 64 threads. We also show a fit of Amdahl's law [1] and the parallel fraction, p. Keep in mind that two 16 core CPUs were used, which means we would expect the speed-up to stagnate or even decrease when using more than parallel 32 threads. For these tasks, the stagnation appears to start at 40 threads on our test system.



(a) All images (b) Images with 16 or more nuclei

Figure 4: Box plots showing the relative speed-up for each image in the Broad Bioimage Benchmark `stage1_train` dataset, for M-QPBO and P-QPBO compared to K-QPBO. Following Tukey's definition, the green line in the box is the median, the box marks the two quartiles and the whiskers show the minimum and maximum values, excluding outliers. Outliers are defined as values more than 1.5 times the interquartile range from the nearest quartile and are shown as rings. In (a) the results are shown for all 670 images, while (b) only includes the 502 images with 16 or more nuclei. The relative speed-up is calculated using the fastest solve time for each method, with each method having been run ten times.

lelism at the end of Phase II. According to Amdahl's law [1], this puts a theoretical maximum to the speed-up, which in our case will depend on the energies and blocking strategy. For the nerve segmentation tasks we estimate a parallel fraction of 0.88 and 0.92 (including overhead) for N1 and N2, respectively, meaning that most of the work is done in parallel.

We expect that most of the performance improvement of M-QPBO over K-QPBO is due to the smaller memory footprint of the graphs shown in Table 2. We achieve this reduction by using more compact data structures for nodes and edges. Instead of 64-bit pointers, we use 32-bit indices where possible. Furthermore, we store forward and backward edges adjacent in memory to avoid storing pointers between these. P-QPBO and M-QPBO use the same fundamental data structures for nodes and edges. The increased memory footprint of the P-QPBO graph is a result of extra bookkeeping needed for the bottom-up merging. Reducing the memory footprint of the graph structures is important for two reasons. Firstly, we increase performance due to improved CPU cache and memory efficiency. Secondly, it allows us to solve larger tasks, without running out of memory.

It is important to remember that the scaling depends both on the optimization problem and the system architecture. Generally, we would expect M-QPBO to outperform P-QPBO(1) on smaller tasks, due to the overhead of merging the sub-graphs. However, for large tasks, using bottom-up merging, even without parallel computations, actually turns out to be faster. This behavior was previously noted by [16, 35] and is probably due to a combination of shorter augmenting paths and better cache efficiency.

For the N1 task, [26] reported a solve time of 44 minutes for K-QPBO, which is much higher than the 14 minutes we found in our experiments. We suspect that the main reason for the big difference is that their system only had 112 GB memory, while the graph has a footprint of at least 134 GB. This could have caused memory swapping, which would likely impact performance negatively.

Comparison with other solvers: As an ablation study, we compare P-QPBO with other state-of-the-art maxflow/mincut algorithms. To test the effect of the two-stage QPBO strategy, we compare with our own implementation of the parallel maxflow/mincut algorithm by Liu and Sun [35]. Furthermore, we compare with the serial EIBFS algorithm [16], as it is currently the fastest serial maxflow/mincut solver, and we compare with our own parallel version of EIBFS (P-EIBFS) based on bottom-up merging. Finally, we include a best case estimate for a QPBO implementation using EIBFS as its maxflow/mincut solver (EIBFS-QPBO).

We compare the algorithms on the N1 nerve segmentation task. The maxflow/mincut algorithms are evaluated by first converting the QPBO problem to the primal-dual graph and then running the algorithm on this graph. We do not include this conversion time in the benchmark. Results are shown in Table 3. We see that our algorithm significantly outperforms the other methods.

Table 3: Results of ablation experiment on the N1 nerve segmentation task. The maxflow/mincut solvers were run on the full primal-dual graph. We do not include the time used to convert the QPBO problem to a primal-dual graph.

	Mem. footprint	Fastest solve time
M-QPBO	60.1 GB	638 s
P-QPBO(40)	70.0 GB	**76 s**
Liu-Sun(32) [35]	70.0 GB	191 s
EIBFS [16]	175.1 GB	922 s
P-EIBFS(32)	175.8 GB	1435 s
EIBFS-QPBO*	175.1 GB	461 s

*Best case estimate (half of EIBFS solve time).

4.2. Smaller segmentation tasks

We use the Broad Bioimage Benchmark, previously used in [26], to compare the performance of P-QPBO, M-QPBO, and K-QPBO on a large set of 2D (non-grid) segmentation problems of varying sizes. Figure 5 shows the distributions of graph nodes and edges for the images. With a median of 437,400 nodes and 1,598,060 edges, we consider most of these segmentation tasks relatively small. A few of the tasks are significantly larger, with the largest having just over six million nodes and 60 million edges. To examine the overall performance of M-QPBO and P-QPBO for these small to medium-sized tasks, we compute the relative speed-up when using our implementations compared to K-QPBO. An example segmentation is shown in Figure 2, right. Nodes associated with each cell nuclei are in the same block.

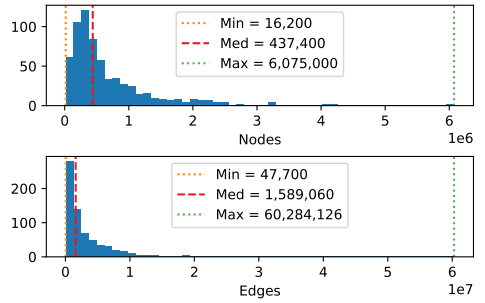


Figure 5: Histograms of the distribution of nodes and edges for the 2D image segmentation tasks.

Figure 4 shows the relative speed-up for M-QPBO and P-QPBO for each image in the data set (Figure 4a) and for each image with 16 or more nuclei (Figure 4b). Both M-QPBO and P-QPBO show a significant improvement compared to K-QPBO. When we include all images, there are cases where the relative performance drops. In these few cases, the tasks are very small (few nodes and terms), such that the overhead of P-QPBO outweighs the benefits. If we look only at the 502 images with 16 or more nuclei (259,200 nodes or more), M-QPBO and P-QPBO significantly outperform K-QPBO for all images, except when using P-QPBO with a single thread. For these smaller tasks, the overhead of merging blocks is not outweighed by the shorter augmenting paths. Thus, when using only a single thread, the best performance is achieved without bottom-up merging.

For the images with 16 or more nuclei (Figure 4b), M-QPBO gives a median speed-up of 1.8x, with a maximum of 2.5x. P-QPBO(4) achieves the best overall performance, with a median speed-up of 3.2x and a maximum of 5.3x. While P-QPBO(6) and P-QPBO(8) show the best performance in a few of the largest tasks, the overall performance decreases slightly when compared to using four threads, due to the majority of the tasks being relatively small.

5. Conclusion

Our P-QPBO algorithm is the first parallel QPBO algorithm. It scales much better than the serial K-QPBO algorithm on modern multi-core hardware, by partitioning the task into sub-tasks and solving them in parallel. It uses a bottom-up merging strategy to combine the solutions, also in parallel. This allows P-QPBO to solve optimization

tasks, such as image segmentation, significantly faster than other current algorithms.

Our experiments show that P-QPBO solves large multi-object segmentation tasks over 17 times faster than K-QPBO, with lower memory usage. It does so while remaining fully compatible with K-QPBO, making no constraining assumptions about the graph structure. Even for smaller tasks, with just a few hundred thousand nodes, P-QPBO is 2-5 times faster than K-QPBO, using only four threads. This indicates that P-QPBO will significantly outperform K-QPBO, even on consumer hardware.

The scalability of P-QPBO, when combined with modern hardware, makes P-QPBO suitable for solving much larger optimization tasks than previously possible. Furthermore, because it is a parallel algorithm, we expect the relative performance of P-QPBO to keep increasing in the future. Finally, P-QPBO is a general algorithm, which is suitable for many binary optimization tasks, not just image segmentation. Thus, we are confident that P-QPBO can be used not just for faster image segmentation, but also for a wide range of other tasks, both in computer vision and other fields.

References

[1] Gene M Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18-20, 1967, spring joint computer conference*, pages 483–485, 1967. 7

[2] Richard Anderson and Joao C. Setubal. A parallel implementation of the push-relabel algorithm for the maximum flow problem. *Journal of Parallel and Distributed Computing (JPDC)*, 29(1):17–26, 1995. 2

[3] Chetan Arora, Subhashis Banerjee, Prem Kalra, and SN Maheshwari. An efficient graph cut algorithm for computer vision problems. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 552–565, 2010. 1

[4] Bengt Aspvall, Michael F Plass, and Robert Endre Tarjan. A linear-time algorithm for testing the truth of certain quantified boolean formulas. *Information Processing Letters*, 8(3):121–123, 1979. 4

[5] David A Bader and Vipin Sachdeva. A cache-aware parallel implementation of the push-relabel network flow algorithm and experimental evaluation of the gap relabeling heuristic. Technical report, Georgia Institute of Technology, 2006. 2

[6] Niklas Baumstark, Guy Blelloch, and Julian Shun. Efficient implementation of a synchronous parallel push-relabel algorithm. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 106–117, 2015. 2

[7] Endre Boros, Peter L Hammer, and Xiaorong Sun. Network flows and minimization of quadratic pseudo-boolean functions. Technical report, Technical Report RRR 17-1991, RUTCOR, 1991. 1

[8] Yuri Boykov and Gareth Funka-Lea. Graph Cuts and Efficient N-D Image Segmentation. *International Journal of Computer Vision*, 70(2):109–131, nov 2006. 1

[9] Yuri Boykov and Vladimir Kolmogorov. An Experimental Comparison of Min-Cut/Max-Flow Algorithms for Energy Minimization in Vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(9):1124–1137, 2004. 1, 2

[10] Boris V Cherkassky and Andrew V Goldberg. On implementing the push—relabel method for the maximum flow problem. *Algorithmica*, 19(4):390–410, 1997. 1

[11] Andrew Delong and Yuri Boykov. A scalable graph-cut algorithm for nd grids. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2008. 1, 2

[12] Daniel Freedman and Petros Drineas. Energy minimization via graph cuts: Settling what is possible. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 939–946, 2005. 2

[13] Andrew V Goldberg. Processor-efficient implementation of a maximum flow algorithm. *Information Processing Letters*, 38(4):179–185, 1991. 2

[14] Andrew V Goldberg. The partial augment–relabel algorithm for the maximum flow problem. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 466–477, 2008. 1

[15] Andrew V Goldberg. Two-level push-relabel algorithm for the maximum flow problem. In *International Conference on Algorithmic Applications in Management*, pages 212–225, 2009. 1

[16] Andrew V Goldberg, Sagi Hed, Haim Kaplan, Pushmeet Kohli, Robert E Tarjan, and Renato F Werneck. Faster and More Dynamic Maximum Flow by Incremental Breadth-First Search. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 619–630, 2015. 1, 2, 5, 7, 8

[17] Andrew V Goldberg, Sagi Hed, Haim Kaplan, Robert E Tarjan, and Renato F Werneck. Maximum Flows by Incremental Breadth-First Search. In *Proceedings of the European Symposium on Algorithms (ESA)*, pages 457–468, 2011. 1

[18] Zhihui Guo, Ling Zhang, Le Lu, Mohammadhadi Bagheri, Ronald M Summers, Milan Sonka, and Jianhua Yao. Deep LOGISMOS: deep learning graph-based 3D segmentation of pancreatic tumors on CT scans. pages 1230–1233, 2018. 1

[19] Peter L Hammer, Pierre Hansen, and Bruno Simeone. Roof duality, complementation and persistency in quadratic 0–1 optimization. *Mathematical Programming*, 28(2):121–155, 1984. 1

[20] Dorit S. Hochbaum. The pseudoflow algorithm: A new algorithm for the maximum-flow problem. *Operations Research*, 56(4):992–1009, 2008. 1

[21] Dorit S. Hochbaum and James B. Orlin. Simplifications and speedups of the pseudoflow algorithm. *Networks*, 61(1):40–57, 2013. 1

[22] Bo Hong and Zhengyu He. An asynchronous multithreaded algorithm for the maximum network flow problem with non-blocking global relabeling heuristic. *IEEE Transactions on Parallel and Distributed Systems (TPDS)*, 22(6):1025–1033, 2010. 2

[23] Hossam Isack, Olga Veksler, Ipek Oguz, Milan Sonka, and Yuri Boykov. Efficient optimization for hierarchically-structured interacting segments (HINTS). In *Proceedings*

of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1445–1453, 2017. 1, 2

[24] Ondřej Jamriška and Daniel Sỳkora. GridCut. Version 1.3. https://gridcut.com, 2015. Accessed 2020-06-12. 2

[25] Ondřej Jamriška, Daniel Sỳkora, and Alexander Hornung. Cache-efficient Graph Cuts on Structured Grids. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3673–3680, 2012. 2

[26] Niels Jeppesen, Anders N Christensen, Vedrana A Dahl, and Anders B Dahl. Sparse layered graphs for multi-object segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12777–12785, 2020. 1, 2, 4, 5, 6, 8

[27] Niels Jeppesen, Anders Nymark Christensen, Vedrana Andersen Dahl, and Anders Bjorholm Dahl. Sparse Layered Graphs for Multi-Object Segmentation (notebooks). 6 2020. 5

[28] Niels Jeppesen, Anders Nymark Christensen, Vedrana Andersen Dahl, Anders Bjorholm Dahl, Hans Martin Kjer, Martin Bech, and Lars Dahlin. Sparse Layered Graphs for Multi-Object Segmentation (data). 11 2020. 5

[29] Anna Khoreva, Rodrigo Benenson, Jan Hosang, Matthias Hein, and Bernt Schiele. Simple does it: Weakly supervised instance and semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 876–885, 2017. 1

[30] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: from edges to instances with multicut. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5008–5017, 2017. 1

[31] Pushmeet Kohli and Philip H.S. Torr. Dynamic graph cuts for efficient inference in Markov random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(12):2079–2088, 2007. 2, 3

[32] Vladimir Kolmogorov and Carsten Rother. Minimizing nonsubmodular functions with graph cuts-a review. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 29(7):1274–1279, 2007. 1, 2, 3, 4, 6

[33] Vladimir Kolmogorov and Ramin Zabin. What energy functions can be minimized via graph cuts? *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 26(2):147–159, 2004. 2

[34] Kang Li, Xiaodong Wu, Danny Z Chen, and Milan Sonka. Optimal surface segmentation in volumetric images-a graph-theoretic approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 28(1):119–134, 2005. 1

[35] Jiangyu Liu and Jian Sun. Parallel Graph-cuts by Adaptive Bottom-up Merging. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2181–2188, 2010. 1, 2, 3, 4, 5, 7, 8

[36] V. Ljosa, K. L. Sokolnicki, and A. E. Carpenter. Annotated high-throughput microscopy image sets for validation. *Nature Methods*, 9(7):637–637, 2012. 5

[37] Carsten Rother, Vladimir Kolmogorov, Victor Lempitsky, and Martin Szummer. Optimizing binary MRFs via extended roof duality. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007. 1

[38] Alexander Shekhovtsov and Václav Hlaváč. A distributed mincut/maxflow algorithm combining path augmentation and push-relabel. *International Journal of Computer Vision (IJCV)*, 104(3):315–342, 2013. 2, 3

[39] Petter Strandmark and Fredrik Kahl. Parallel and Distributed Graph Cuts by Dual Decomposition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2085–2092, 2010. 1, 2, 3, 4

[40] Tanmay Verma and Dhruv Batra. MaxFlow Revisited: An Empirical Comparison of Maxflow Algorithms for Dense Vision Problems. In *Proceedings of the British Machine Vision Conference (BMVC)*, pages 1–12, 2012. 1

[41] Vibhav Vineet and P J Narayanan. CUDA cuts: Fast graph cuts on the GPU. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1–8, 2008. 1, 2

[42] Miao Yu, Shuhan Shen, and Zhanyi Hu. Dynamic Parallel and Distributed Graph Cuts. *IEEE Transactions on Image Processing*, 25(12):5511–5525, 2015. 2, 3

[43] Miao Yu, Shuhan Shen, and Zhanyi Hu. Dynamic Graph Cuts in Parallel. *IEEE Transactions on Image Processing*, 26(8), 2017. 2, 3

3.7 Contribution summary

The three research contributions presented above represent a significant step forward in the application of maxflow/mincut algorithms for large optimization problems, in particular large multi-object segmentation problems. The SLG structure algorithms, formulated in Paper A, significantly expand the range of segmentation problems that can be solved using binary optimization. This is critical for the applicability of graph cut-based methods on high-resolution 3D and 4D datasets, which appear increasingly frequent due to technological advancements in scanning equipment. By reducing the graph size by orders of magnitude, many multi-object segmentation tasks that were previously impractical are now solvable using graph cut methods.

In practice, Paper A only presents a glimpse of the possibilities made possible by the SLG structure. The radial resampling scheme and gradient-based energies used in the experiments work well for many tasks. However, the flexibility provided by the layered structure is not really explored, as the paper focuses on the scalability of the method. Furthermore, while the object/label layers in the experiments are all created the same way, this is not a requirement for the SLG layers. In fact, layers can be constructed using completely different energy functions/graph structures and still interact. Similarly, while the experiments only apply simple multi-exclusion and containment interactions, the Ishikawa, and thus SLG structure, allows any logically sound combination of valid geometric interactions to be expressed. Another way to put this is that there are no explicit limitations on which interactions can be used in combination, as long as they are logically sound. An example of something that is not logically sound would be to add both containment and exclusion interactions between two labels at the same time.

While Paper A shows how the size of multi-label problems can be reduced by orders of magnitude, Paper B and Paper C focus on how to improve the performance of graph cut algorithms using parallelism. Parallel maxflow/mincut and QPBO algorithms are essential for proper utilization of modern hardware. We show that it is possible to implement parallel maxflow and QPBO algorithms, which scale well for larger segmentation tasks, with speed-ups of more than an order of magnitude compared to their serial counterparts. Because our implementations will be made publicly available and are compatible with previous standard implementations, everyone currently using these algorithms will be able to benefit from our contributions.

3.8 Blade segmentation

At this point, I understand if the relationship between my work on maxflow/mincut algorithms and the quality control of wind turbine blades covered in Chapter 2 is unclear. As previously explained, due to the confidentiality of the data (and, to some extent, the methods), I cannot present results from my work on segmentation of blade structures in this thesis. However, I will explain the general concepts and

challenges of segmenting ultrasound data from blades, and why graph cut methods are well suited for this task.

My work on graph cut-based segmentation methods was motivated by the need for a robust method for detecting/segmenting surfaces in large ultrasound volumes. The blade structure often consists of a series of layers (e.g., pultruded carbon stacks and/or fiberglass), where the approximate thickness, order, and number of material surfaces are known. As previously covered, the peaks in the ultrasound signals are located at the material interfaces/surfaces, but the data is noisy and often incomplete. Based on these factors, I decided to use the graph cut-based surface detection method by Kang Li et al. 2006 for segmenting major parts of the spar cap structure. The method is well-suited for this due to the ability to enforce both smoothness and interaction constraints for the surfaces. However, as the size of the structures grew, and new equipment dramatically increased the resolution and thereby the size of the datasets, it was clear there was a need for more efficient graph structures and maxflow/mincut algorithms.

3.8.1 Ultrasound data from blades

While the B-scan images shown in Section 2.4 consist of a few hundred A-scans, the volumetric ultrasound datasets used for modern inspection of a single spar cap may consist of millions of A-scans. Depending on the sampling frequency, a single volume may take up anywhere from 4 GB to 70 GB when stored as 16-bit floating points. As blades continue to increase in size, it is very likely that we will be seeing single ultrasound volumes with over $20,000 \times 500 \times 3,000$ voxels, and a voxel resolution around $5\,\text{mm} \times 1\,\text{mm} \times 0.02\,\text{mm}$, before the end of 2021.

Figure 3.15a illustrates a cross section of a carbon reinforced spar cap and web bonding area of a wind turbine blade, similar to what was shown in Figure 2.5, except the web is shaped and attached differently. The primary features, shown in Figure 3.15a, are the outer blade/shell surface, the carbon pultrusion stacks, the filler material (e.g., balsa wood or foam), the inner shell surface, the glue, and the web. The mechanical properties of the materials used for the different structures vary significantly and determine where and how much of the ultrasound is reflected from the different material interfaces.

As shown in Figure 2.8, the probe is placed on outer surface and moved across the blade collecting A-scans at a certain interval. Typically, more than one probe is used to reduce the scan time and modern solutions may consist of more than a hundred piezoelectric crystals in a so-called phased-array configuration. This allows the scanner to cover the entire width of the space cap, and allows it to focus the sound waves by using multiple crystals for transmitting and receiving at the same time. Because these large scanners can cover the entire width of the spar cap with a resolution down to 1 mm, the scanner can move along the blade in the spanwise direction (from root to tip or vice versa) without having to move across the spar cap

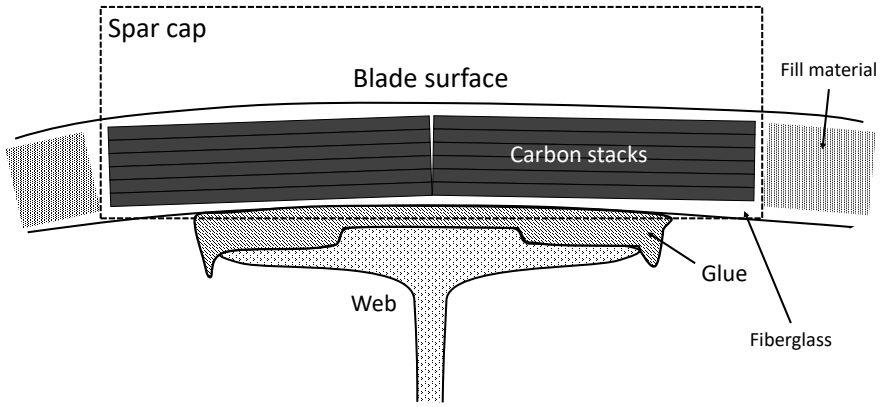
in the chordwise direction (vertical axis in Figure 3.15). This allows the spar cap to be scanned in less than an hour, even for a large blade. Naturally, there is also the option to use smaller scanners, which follow a zigzag or meander pattern to cover the full width of the spar cap. However, this dramatically increases the time it takes to scan the blade. I will not cover the scanning equipment and process any further in this thesis. The primary thing to note is that modern equipment allows very large high-resolution datasets to be captured quickly. In fact, much faster than the time it currently takes to evaluate the data.

3.8.2 Interpreting ultrasound images of the spar cap

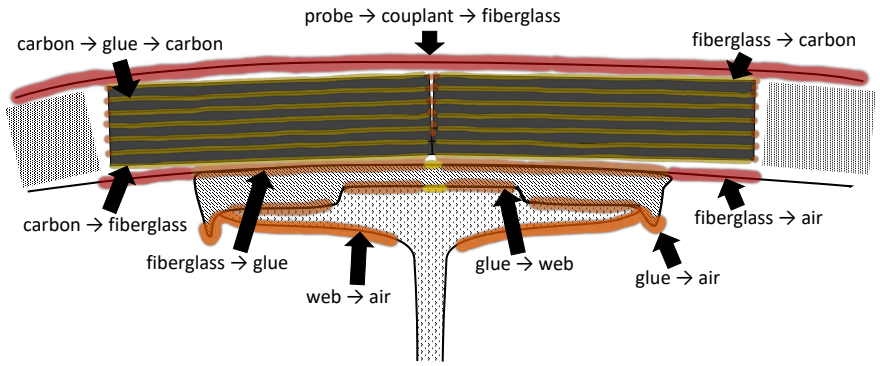
As mentioned a few times at this point, the interpretation of ultrasound data is generally more difficult than attenuation-based techniques, such as X-ray CT. For blades, the interpretation is made even harder by the extensive use of different composite materials, which are notoriously difficult to image with ultrasound due to the anisotropic nature and inconsistent microscopic structure of the materials. However, with the proper equipment, it is possible to capture reflections from most of the material interfaces in the spar cap and web bond as shown in Figure 3.15b. Here, the red color represents the high amplitude reflections, orange marks weaker reflections, and yellow indicates very weak interface reflections. The reflections in Figure 3.15b represent an ideal scenario. The orange, and in particular yellow signals, are often broken into pieces, or even completely invisible in much of the data. This is primarily due to the way the sound propagates through the structure, from the surface down into the structure, and then, ideally, back to the surface again.

The primary surface echo should always be clearly visible, as the signal is strong when it passes from the probe into the couplant (usually water) and then, almost immediately, into the spar cap. These two interfaces reflect some of the signal, which corresponds to the top red line in Figure 3.15b. However, as soon as the signal enters the fiberglass laminate, the signal gets noisy, as the mixture of glass fibers and resin cause many small reflects. On either side of the spar cap, the signal passes into the filler material, usually made of balsa wood or foam. These materials completely absorb the signal, which means we have no reflections from underneath them. Inside the spar cap, the signal passes from the fiberglass into the pultruded carbon stacks, through each of the slabs, and out into another thin layer of fiberglass. Each of these interfaces reflects a small portion of the signal, marked with yellow in Figure 3.15b. These are minor reflections, which are often difficult to see.

Afterwards, the signal reaches the inner surface of the blade shell. Near the edge of the spar cap, the signal hits the fiberglass-to-air interface, reflecting most of the remaining energy. This interface (two bottom red lines in Figure 3.15b) should be clearly visible. If this is not the case, it is most likely because a defect inside the shell is shadowing what is beneath it, or due to issues with the equipment. By “shadowing”, I mean that something is either absorbing, reflecting, or refracting the signal in such a way that there is little or no signal beyond this point. Another place shadowing



(a) Spar cap cross section



(b) Reflecting interfaces

Figure 3.15 – Cross section of a carbon reinforced spar cab and web bond in a wind turbine blade. The most important structural components are shown in (a), while the different interface echoes visible in ultrasound images are shown in (b).

occurs is where the carbon stacks come together in the center of the spar cap. Here, a combination of small air pockets and material interfaces almost parallel to the signal often results in a very weak signal in the center of the bond area.

The fiberglass-to-glue interface is not as clear as the fiberglass-to-air interface, as the difference in sound velocity is much less for the fiberglass and glue than for fiberglass and air. If a strong signal is observed at the fiberglass-to-glue interface, it is often a course of major concern, as it usually indicates that the glue is missing, as shown in Figure 3.16b.

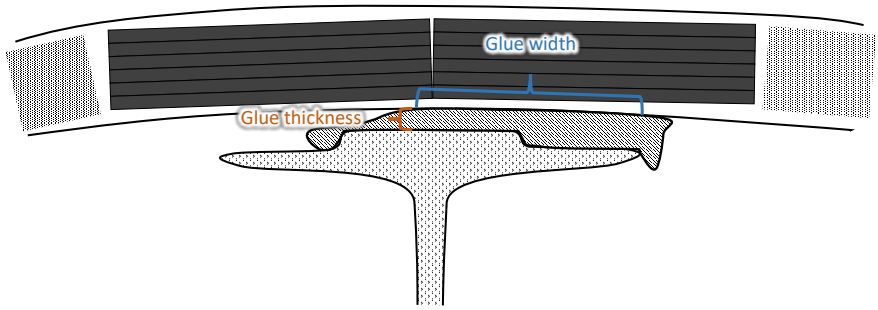
When the signal reaches the web, it has lost a lot of its energy. To compensate for the energy lost as the signal travels further from the probe, it is common to amplify the signal artificially as a function of time. This is known as time-varied gain (TVG). Applying TVG can often ease the human interpretation, as reflections at different depths are more even in terms of amplitude. However, it does not improve the signal-to-noise ratio.

Because the signal has to travel through the entire thickness of the shell before reaching the web, the reflections from the web structure are often weak and inconsistent. This is the case for both the web-to-air and glue-to-web interfaces. However, the web-to-air interface usually appears stronger than the glue-to-web interface due to the large difference in sound velocity between the web material or air. Although both web interfaces should reflect a significant part of the signal energy, the orientation of the interfaces, along with shadowing from imperfections in the structure above the web, means that web interfaces are often not well defined in the ultrasound images.

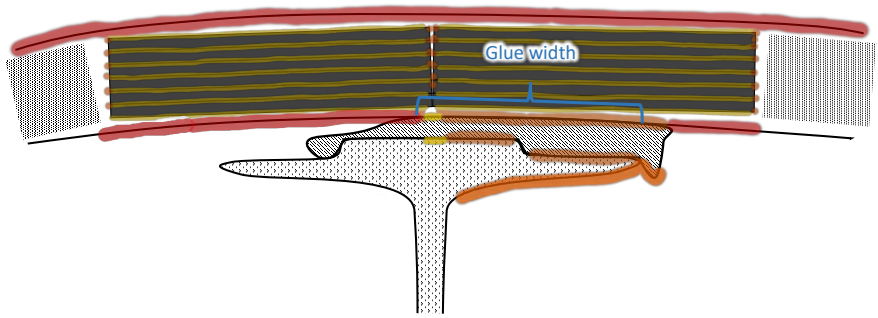
For automated evaluation of the blades, accurately detecting and characterizing the material interfaces in the spar cap and web bonding region is essential. A good example of this is when measuring the glue width and thickness as shown in Figure 3.16. Keep in mind that the structure sketched in the figure is not visible in the ultrasound data. We only have the interface reflects, which are often very difficult to separate from the noise. However, if we are able to detect and characterize the glue and web interfaces, we can measure the glue width and thickness, which are important to the blade's structural integrity.

3.8.3 Automated surface detection

Human evaluation of ultrasound data from blades relies heavily on prior knowledge and contextual information to find and evaluate the structural interfaces in the signal. An image analysis or machine learning method for automated evaluation would also benefit from being able to utilize prior knowledge about the structure and context. Surface detection based on the approach by Kang Li et al. 2006 allows both these things through interaction and smoothness constraints. However, without the scalability and improvements in performance offered by SLGs and parallel maxflow/mincut algorithms, the use of Li's surface detection is limited to smaller datasets.



(a) Spar cap cross section



(b) Reflecting interfaces

Figure 3.16 – Cross section of a carbon reinforced spar cap and web bond in a wind turbine blade, where glue is partially missing from the bond area. (a) shows how glue thickness and width can be measured, while (b) shows the effect of the missing glue on the reflected signal. The lack of glue causes strong reflections from the entire left inner surface, while no reflections are visible from the left side of the web due to shadowing.

3.8.3.1 Dynamic maxflow algorithms

A topic that was only briefly mentioned in the contributions is the dynamic solving features of some of the maxflow algorithms, in particular BK Maxflow. It allows cut graphs to be modified and re-cut, where the runtime of computing the new optimal solution depends on how many changes were made to the graph and how much they change the solution. In principle, this allows real-time user interaction, where the user can correct minor errors in the labeling and recompute the optimal solution fast enough for the interaction to feel responsive to the user input. I think this user interaction is critical to the long-term success of graph cut-based segmentation and surface detection, at least for use in commercial software. From my experience, one of the biggest issues of applying automated methods in production is that they are expected to be right every time. It is very much a question of all or nothing, as they are often thought of as a replacement of human labor, rather than a supplement. I believe one of the key issues with many automated solutions is that it is very difficult for users to correct small errors made by the automated solution. In such cases, users are often required to do everything from scratch through a type of manual backup procedure, which is completely decoupled from the automated system.

While completely automated evaluation of ultrasound data from blades will likely happen sometime in the future, for now, I think assisted evaluation is actually a much more practical solution. Dynamic graph cuts, along with active learning algorithms, could very well be a part of the solution. An assisted solution that does 90% of the work 100% of the time is much more reliable than an automated one that does 100% of the work but only works 90% of the time – because – how do you know when it did not work? In other words, it is probably better to design an interactive system that does most of the work than a fully automated non-interactive system that sometimes fails. At least while failures are relatively common, and in particular if failures may have large consequences.

3.9 Summary

In this chapter, I have described and discussed some of the most important techniques, methods, and algorithms related to graph cut-based segmentation and surface detection. These form the foundation for my contributions, which allow graph cuts to effectively solve much larger optimization tasks than previously possible. Sparse layered graphs bring flexibility to the Ishikawa layered technique, making it possible to segment hundreds or even thousands of interacting objects. Meanwhile, our comparison of serial and parallel maxflow implementations shows that there is still plenty of performance to be gained through optimization and parallelization of existing algorithms. We further demonstrate this with our own parallel QPBO implementation that outperforms the serial algorithm by an order of magnitude on large segmentation tasks.

Our work on faster and more scalable graph cut methods is based on the need for 3D image segmentation and surface detection in a world where data resolution, and thus size, is ever-growing. One such example is ultrasound data from wind turbine blades, where the need for assisted and automated evaluation is growing, as both blade production and blade sizes increase. The work presented in this chapter does not solve this task on its own, but it provides some useful tools for building solutions, which can be used to bring down the cost and improve the accuracy of QC on blades.

CHAPTER 4

Structure Tensor Analysis

This chapter includes two papers about characterization and quantification of fiber orientations in composite materials. Both contributions focus on unidirectional fiber-reinforced composite materials used for structural components, such as the spar cap in wind turbine blades. Using *structure tensor* analysis, we estimate the fiber orientations in µCT scans of composite samples and calculate a number of different metrics used to characterize and quantify the orientation and alignment of the fibers.

Before presenting the two papers, I give a brief introduction to our structure tensor formulation. This introduction is similar to, but more detailed than, the descriptions given in the two papers. Afterwards, I discuss some of the challenges of working with orientations in 3D and applications of our approach in material science research and QC.

4.1 Structure tensor

In 3D image analysis, a structure tensor is a 3-by-3 matrix, which summarizes the orientation of imaged structures in a small neighborhood around a point in space. The structure tensor $\mathbf{S}$ is computed from the gradients of the 3D image and thus describes the change in image intensity around the point. For orientation analysis, we exploit the following property of the structure tensor: If $\mathbf{u}$ is a (column) unit vector, then $\mathbf{u}^\intercal\mathbf{S}\mathbf{u}$ is the *sum of the squared change* in the intensities for a window displaced slightly in the direction $\mathbf{u}$. Since the change in image intensity is smallest when moving parallel to structures in the image, finding the dominant direction of structures corresponds to finding the $\mathbf{u}$ that minimizes $\mathbf{u}^\intercal\mathbf{S}\mathbf{u}$, i.e., the direction of least change.

To understand why this is the case, we must examine the definition of the structure tensor for discrete 3D images. For a volumetric (3D) image V, the structure tensor can be written as

$$\mathbf{S}(\mathbf{p}) = \sum_{\mathbf{r}\in N(\mathbf{p})} \left(\nabla V(\mathbf{r})\right)\left(\nabla V(\mathbf{r})\right)^\intercal \ , \tag{4.1}$$

where N is a fixed neighborhood around a voxel, $\mathbf{p}$ is a point, and $\mathbf{r}$ is a window determined by the neighborhood of a point.

Meanwhile, the displacement of the window $\mathbf{r}$ by the unit vector $\mathbf{u}$ can be written as $V(\mathbf{r}+\mathbf{u})$. Using first-order Taylor expansion, we can write the displacement of the window as

$$V(\mathbf{r}+\mathbf{u}) \approx V(\mathbf{r}) + \mathbf{u}^\intercal \nabla V(\mathbf{r})\,. \tag{4.2}$$

Thus, the sum of the squared change of intensities from the displacement is

$$\begin{aligned}\sum_{\mathbf{r}\in N(\mathbf{p})} \left(V(\mathbf{r}+\mathbf{u}) - V(\mathbf{r})\right)^2 &= \sum_{\mathbf{r}\in N(\mathbf{p})} \left(\mathbf{u}^\intercal \nabla V(\mathbf{r})\right)^2 \\ &= \mathbf{u}^\intercal \left(\sum_{\mathbf{r}\in N(\mathbf{p})} \left(\nabla V(\mathbf{r})\right) \left(\nabla V(\mathbf{r})\right)^\intercal \right) \mathbf{u} \\ &= \mathbf{u}^\intercal \mathbf{S} \mathbf{u}\,.\end{aligned} \tag{4.3}$$

This shows that $\mathbf{u}^\intercal\mathbf{S}\mathbf{u}$ is the sum of the squared change in the intensities of V, when displaced slightly in the direction of $\mathbf{u}$. Furthermore, because $(\nabla V)(\nabla V)^\intercal$ is symmetric at every point, $\mathbf{S}$ is symmetric. Lastly, because we know that $\mathbf{u}^\intercal\mathbf{S}\mathbf{u} \geq 0$, $\mathbf{S}$ must be positive semidefinite.

Due to the properties described above, $\mathbf{S}$ has three non-negative eigenvalues $\lambda_1 \leq \lambda_2 \leq \lambda_3$ and mutually orthogonal eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$, which can be calculated analytically. The eigenvector $\mathbf{v}_1$ corresponding to the smallest eigenvalue, λ_1, points in the direction of least change, i.e., the dominant structural direction for fiber-like structures.

The relationship between the three eigenvalues can be interpreted as indicators of the structure "type" at the given position. The case where $\lambda_1 \ll \lambda_2 \approx \lambda_3$ is often interpreted as a linear/fiber-like structure and the metric $c_l = \frac{\lambda_2 - \lambda_1}{\lambda_3}$ is the linear anisotropy [Westin et al. 2002; Nelson, Smith, and Mienczakowski 2018]. While the contributions in this thesis use only $\mathbf{v}_1$, the other eigenvectors and eigenvalues may certainly be useful. For instance, we could use c_l for weighing the importance of the eigenvectors and for tasks such as image segmentation.

For calculating the structure tensor $\mathbf{S}$, we rely on Gaussian kernels for computing gradients and integrating those gradients, similar to Weickert 1998. This leads to the formulation

$$\mathbf{S} = K_\rho * \left(\nabla V_\sigma \left(\nabla V_\sigma\right)^\intercal\right)\,, \tag{4.4}$$

where the parameter σ is the standard deviation of the Gaussian derivative kernel used for computing the gradient ∇V_σ, and ρ is the standard derivation of the Gaussian kernel K_ρ used for integration by convolution.

Our approach is different from the commonly used finite difference, as well as the 5-point numerical differentiation proposed by Straumit, Lomov, and Wevers 2015.

While computing the gradient using the finite difference would be faster, the approach is sensitive to noise and image resolution. One way to reduce the noise is to smooth the data using a Gaussian kernel before computing the gradient, however, this is equivalent to, and less efficient than, computing the gradient using the Gaussian derivative. For this reason, we use the Gaussian derivative kernel, which depends on the noise scale σ, to compute the gradients. For integrating (averaging) the gradients in a certain neighborhood, the Gaussian kernel also works well, as it is separable, which means the integration can be done fast, even for large neighborhoods. The neighborhood size depends on the kernel size determined by the integration scale ρ. While the use of Gaussian kernels requires two parameters to be chosen, it has the advantage that it can accommodate different resolutions and noise levels, all while remaining fast as a result of being separable.

4.2 Paper D: Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor

In the fourth paper of this thesis, we use structure tensor analysis to characterize fiber orientations in a fiberglass material used for wind turbine blades. We show that the method can be used to estimate fiber orientation distributions based on a laboratory CT scan of a sample. These distributions can be used to quantify important material properties and segment fiber bundles in the stitched non-crimp fabric. We share the entire pipeline, which includes our scalable GPU implementation for calculating the structure tensor, performing the eigendecomposition, and computing derived metrics, such as angles from the expected fiber orientation. The work was presented at the 41st Risø International Symposium on Materials Science and published in the IOP Conference Series: Materials Science and Engineering. Links to code, notebooks and data can be found in the paper.

The paper included below is the Open Access version published by IOP Publishing Ltd, which can be found at DOI: 10.1088/1757-899x/942/1/012037.

IOP Conference Series: Materials Science and Engineering

PAPER • **OPEN ACCESS**

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

To cite this article: N Jeppesen *et al* 2020 *IOP Conf. Ser.: Mater. Sci. Eng.* **942** 012037

View the article online for updates and enhancements.

This content was downloaded from IP address 194.62.169.40 on 25/02/2021 at 19:06

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

Content from this work may be used under the terms of the Creative Commons Attribution 3.0 licence. Any further distribution of this work must maintain attribution to the author(s) and the title of the work, journal citation and DOI.
Published under licence by IOP Publishing Ltd

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 $[\mathrm{b}_{\mathrm{biax}}/\mathrm{biax}, \mathrm{b}_{\mathrm{UD}}/\mathrm{UD}, \mathrm{b}_{\mathrm{UD}}/\mathrm{UD}]_\mathrm{s}$ where the "b" indicates the location of the backing layer, which is orientated in the transverse

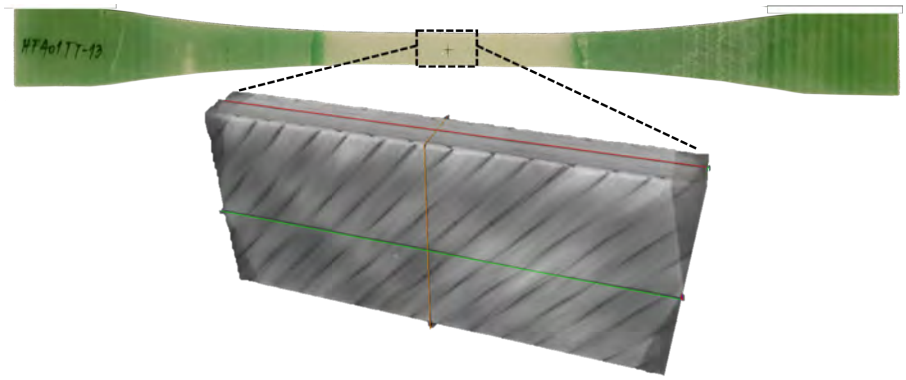


Figure 1. The test-sample and the scanned part of the gauge-section with the dimension $32 \times 15 \times 4.5\,\mathrm{mm}^3$.

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 mm×16.5 mm×16.5 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	$3\times16.5\,\mathrm{mm}^3$	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].

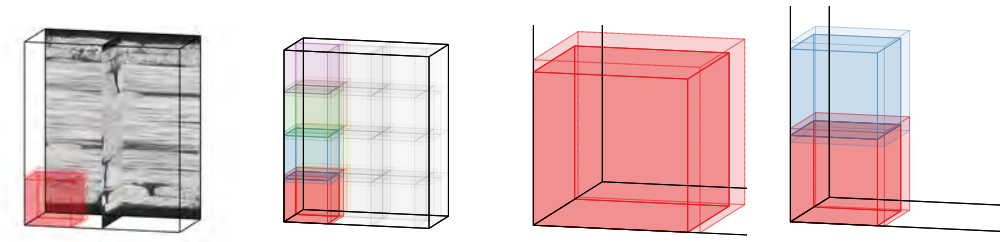
Step 0: Pre-process data. Depending on available tools and experience it may be useful to pre-process data. This includes cropping and rotating data, as well as converting it to a suitable

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

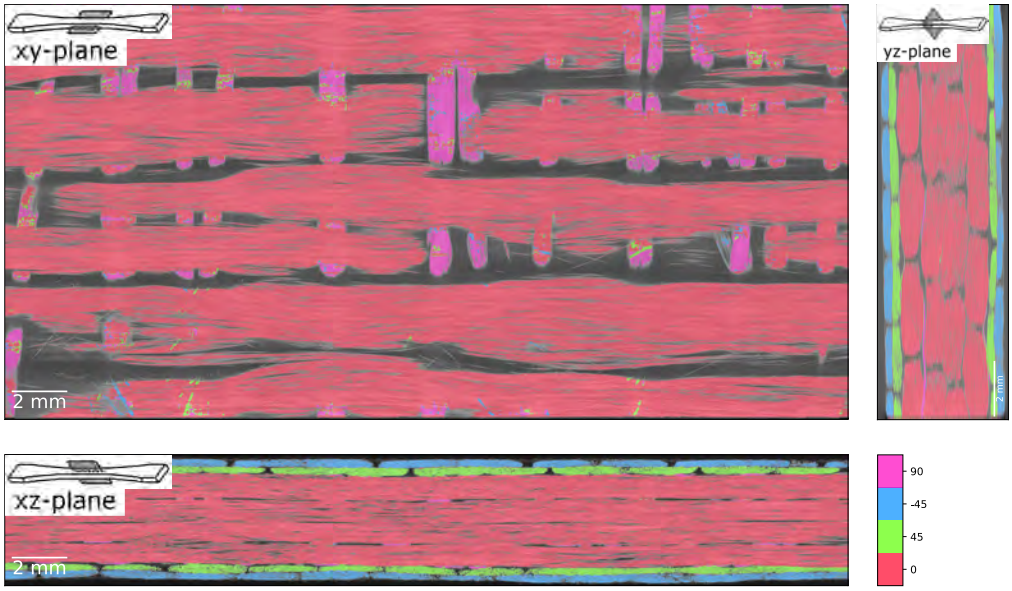


Figure 3. Segmentation based on fiber orientations of the non-crimp fabric shown in the center of the sample in the xz and yz-plane and in the xy-plane in a plane with backing bundles. The four different fiber classes are represented by four different colors, while the background is gray. The segmentation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

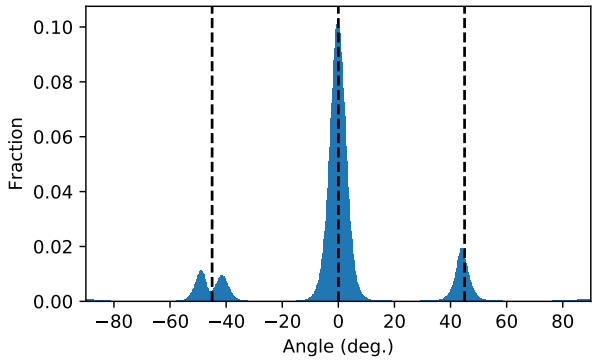


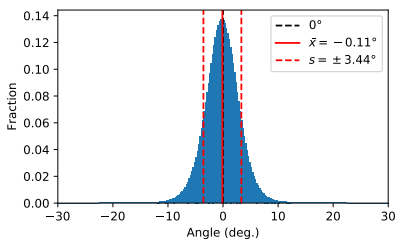
Figure 4. Distribution of the fiber orientations projected onto the xy-plane.

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.

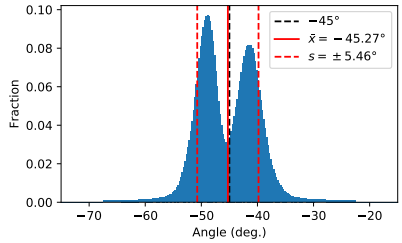
To quantitatively evaluate the fabric and bundles (fiber rowings) used for building the fabric, we can plot the in-plane fiber orientation distributions. Figure 4 shows the distribution orientations of the fibers projected onto the xy-plane for all fibers in the analyzed sample. In figure 5 the same 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.

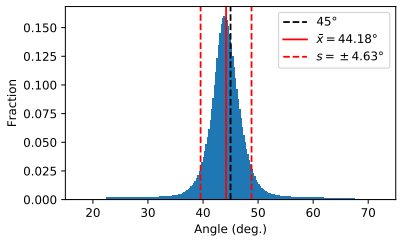
Figure 7 shows the out-of-plane fiber orientations for the xy-plane. Here, all classes should have the same 0° orientation, which also appears to be the case, with all means being within one degree of the expected orientation. Again the UD (0°) fibers appear to align almost perfectly, while the biax-ply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



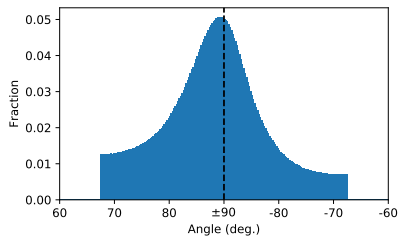
(a) Distribution of 0°



(b) Distribution of −45°

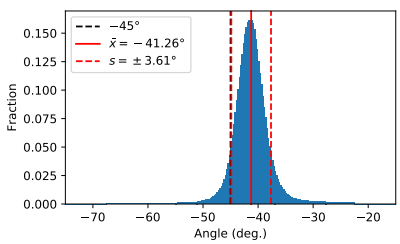


(c) Distribution of +45°

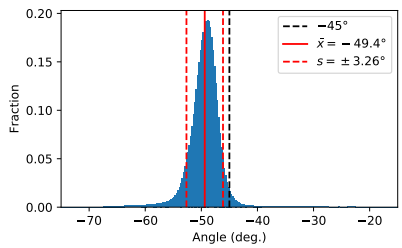


(d) Distribution of 90°

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.

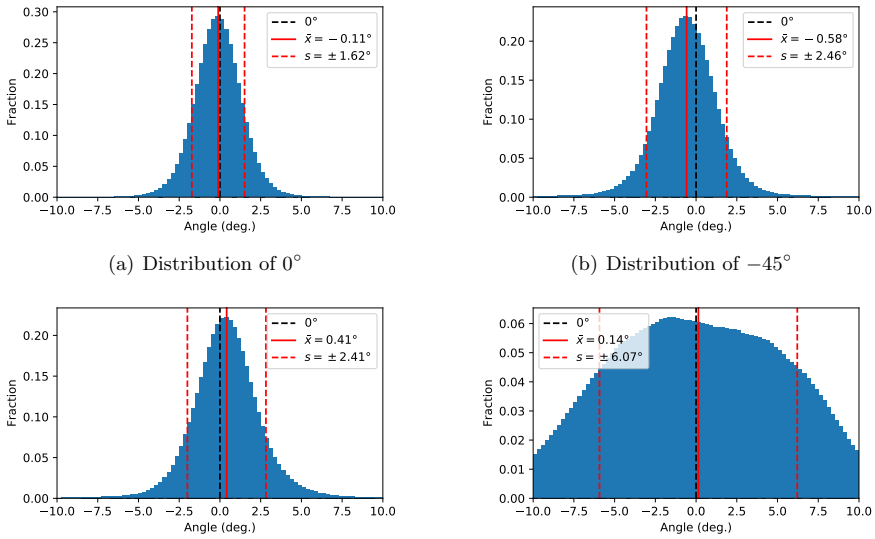


(a) Distribution of the upper ply



(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)

4.3 Paper E: Quantifying effects of manufacturing methods on fiber orientation in unidirectional composites using structure tensor analysis

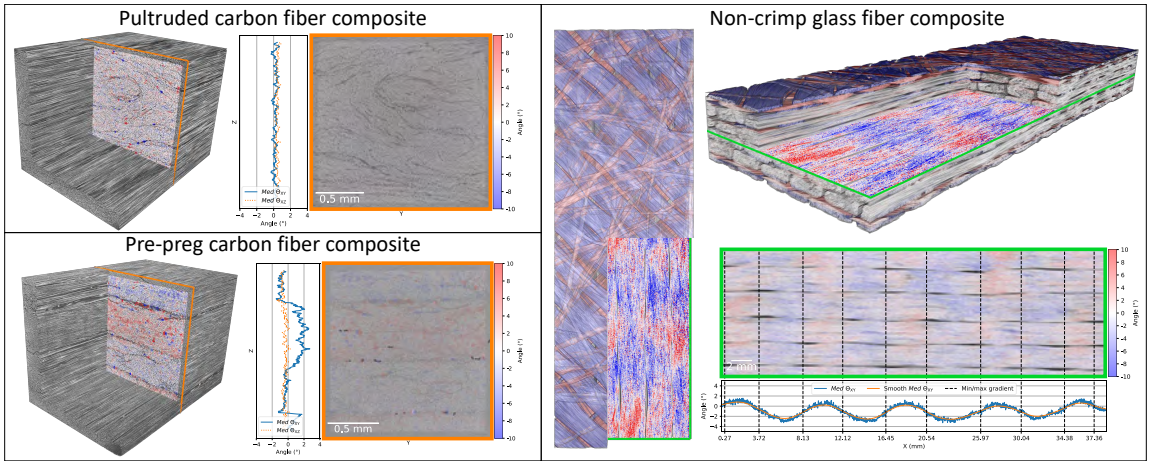
The fifth paper in my thesis also concerns the use of structure tensor analysis for quantification of fiber orientation. However, where Paper D focuses on the pipeline, describing how to obtain distributions and apply them for segmentation on non-crimp fabric, Paper E investigates and compares fiber orientations in three different composite materials used in wind turbine blades. Here, we focus on local variations, which influence material quality and depend on production parameters. We show that these local variations in the fiber orientations can be quantified using our pipeline, which allows manufacturers and customers to compare the quality of the material samples quantitatively. Again, a key feature of our pipeline is that the analysis can be performed in only a few minutes using a modern GPU. All code, notebooks and data can be found through the link found in the paper (once it has been published).

The version included below is a preprint of the paper, which includes the graphical abstract. The paper has been submitted to Composites Part A: Applied Science and Manufacturing and is currently awaiting peer review.

Graphical Abstract

Quantifying effects of manufacturing methods on fiber orientation in unidirectional composites using structure tensor analysis

N. Jeppesen,L. P. Mikkelsen,A. B. Dahl,A. N. Nymark,V. A. Dahl



Highlights

Quantifying effects of manufacturing methods on fiber orientation in unidirectional composites using structure tensor analysis

N. Jeppesen,L. P. Mikkelsen,A. B. Dahl,A. N. Nymark,V. A. Dahl

- Fiber reinforced composites
- X-ray computed tomography (CT)
- Structure tensor analysis
- Fiber orientations

Quantifying effects of manufacturing methods on fiber orientation in unidirectional composites using structure tensor analysis

N. Jeppesen[a], L. P. Mikkelsen[b,*], A. B. Dahl[a], A. N. Nymark[a] and V. A. Dahl[a]

[a]*Department of Applied Mathematics and Computer Science, Technical University of Denmark, Richard Petersens Plads, Building 324, 2800 Kgs. Lyngby, Denmark*

[b]*Department of Wind Energy, Technical University of Denmark, DTU Risø Campus, 4000 Roskilde, Denmark*

ARTICLE INFO

Keywords:
Structure tensor analysis A. Carbon fibers A. Glass fibers C. Statistics D. Non-destructive testing

ABSTRACT

The three most common fiber reinforced composite materials in wind turbine blades are pultruded carbon fiber, pre-preg carbon fiber and non-crimp glass fiber reinforced composites. Important properties for these materials, such as stiffness, compression strength and fatigue resistance are sensitive to the fiber alignment. In this paper, we use a combination of thresholding, structure tensor decomposition and nearest neighbor classification to characterize the fiber orientations in CT scans of each of the three materials. The results show that the unidirectional fibers in the pultruded sample are aligned the most, while the non-crimp fabric fibers are the least aligned. Through local quantitative analysis we show that misalignment of the individual pre-preg layers contribute to the overall fiber misalignment in the material. Similarly, for the non-crimp composite, we show that both the stitching of the unidirectional bundles and the backing bundles affect the fiber alignment in unidirectional bundles. Quantifying the misalignment caused by these effects allow manufacturers to tune production parameters, such as stitching thread tension, to minimize the misalignment of the fibers. Notebooks, code, and data for all our experiments are available online.

1. Introduction

Fiber reinforced composites are widely used due to their high stiffness and excellent strength-to-weight ratio. Some of the largest composite structures currently made are wind turbine blades, with lengths of over 100 meters. For these structures to withstand the enormous forces applied to them continuously for several decades, understanding the properties of structural composite materials is critical. This paper investigates the micro-structure of three fiber composites, commonly used in the load-carrying laminates of wind turbine blades: pultruded profiles, pre-impregnated composites (pre-preg) and vacuum infused non-crimp fabric (NCF) composites. We show how the alignment of the unidirectional (UD) fibers can be quantified using a simple and fast approach, based on structure tensor analysis and X-ray computed microtomography (μCT). This alignment is important for the material stiffness, compression strength and fatigue resistance – all key material design parameters for the load-carrying laminates in wind turbine blades.

The quality of the composite materials is of high importance to the manufacturers of large structures such as wind turbine blades. Both destructive testing (DT) and non-destructive testing (NDT) methods are widely used by material and blade manufacturers to verify material properties. DT is typically used for material certification for wind turbine blades. However, determining the compression strength and fatigue resistance with destructive testing for UD composites often result in invalid failure modes (failure in the gripping area) of the test coupons leading to conservative strength and resistance values. This makes it difficult to use strength values determined experimentally for comparing the quality of high-quality UD composites. NDT on blade composites is often done using X-ray tomography or ultrasonic testing. To image the material microstructures in 3D, μCT is the preferred method in most cases. It allows material samples to be imaged in 3D at high resolution, such that individual fibers are visible. As axial stiffness, compression strength and fatigue resistance are highly sensitive to the fiber alignment, quantification of the fiber microstructures is useful for estimating these material properties. One of the more simplified quantification measures, which can we can calculate, is the fiber orientation efficiency factor proposed by [1].

*Corresponding author

niejep@dtu.dk (N. Jeppesen); lapm@dtu.dk (L.P. Mikkelsen); abda@dtu.dk (A.B. Dahl); anym@dtu.dk (A.N. Nymark); vand@dtu.dk (V.A. Dahl)

ORCID(s): 0000-0001-7844-9180 (N. Jeppesen)

It is well established that the compression strength of composite materials depends on fiber alignment [2, 3], and the link between fiber orientations and the compression strength of composite materials has been studied extensively, both in 2D and 3D [4]. While the initial work used generic distributed fiber orientations with a local maximum fiber-orientation angle, later work looks into the effect of realistic fiber orientation distributions on the compression strength [5]. By comparison, our method estimates the actual orientations in a given composite component.

The three fiber materials examined in this paper are all UD fiber composites, meaning that most of the fibers are oriented in the same direction. These materials are used in the spar cap (load carrying part of the blade). The goal of the UD composites is to maximize the material stiffness, compression, and fatigue strength in the fiber direction (the axial/spanwise direction of the blade). However, misalignment of the fibers in the materials will reduce the stiffness and strength in the given direction. Thus, knowledge about how process parameters affect the UD fiber orientations is highly valuable to the manufacturers and can be used for optimizing the manufacturing process. One way to get this knowledge is through quantitative analysis of fiber orientations and micro-structure, both locally and globally, in the materials.

1.1. Related work

A number of different methods can used to characterize the orientations of fibers in composites based on μCT images. There are two fundamentally different approaches for characterizing fiber orientations, which we will refer to as *instance*-based methods and *semantic* methods. With instance-based methods, fibers are segmented individually, after which aggregated information like fiber diameters and orientations can be calculated. Commercial software, such as Avizo, GeoDict, and VGStudio Max, can be used for this [6]. However, these generally require very high-resolution data, are computationally expensive and time-consuming, and struggle to segment densely packed fibers [6]. Other methods for tracking fibers in 3D include template matching [7] and dictionary-based methods [8, 9], which have been shown to work well, even for densely packed fibers at lower resolutions. However, the instance-based methods are slow for large datasets and often designed for tracking UD fibers only, requiring tracked fibers to be oriented in approximately the same direction.

Semantic methods do not detect and track individual fibers, but only distinguishes between different classes of fibers (e.g., UD and backing fibers) and background (e.g., matrix). A popular semantic method is structure tensor-based orientation characterization [6, 10, 11, 12, 13], as it is a fast and reliable approach for determining fiber orientation distributions. It can be used to determine fiber orientations across large datasets in only a few minutes, using high-performance GPU implementations [13]. Furthermore, it works well, even for lower resolution data, where the instance methods are unable to detect and track individual fibers [6].

1.2. Contribution

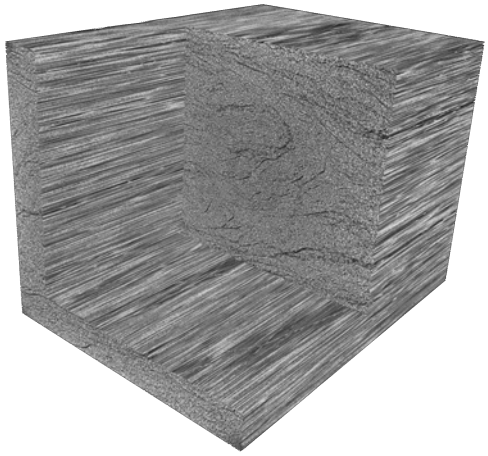
Our primary contribution is showing that it is possible to visualize and quantify local misalignment in UD fiber materials, which are directly related to specific production parameters, using the structure tensor analysis and simple intensity-based thresholding. This allows manufacturers of these materials to improve the quality of their products. Not only can they use our approach to detect defects in the materials, they are also able to adjust process parameters based on local measurements of misalignment and to compare the quality of different material batches.

Using the structure tensor approach, we estimate the global orientation distributions for one sample of each of the three UD fiber-reinforced materials. For the pultruded and pre-preg materials, we further examine the individual cross-sectional slices of the sample to determine the effects of each of the manufacturing processes on the orientations. In particular, we visualize and quantify the effect of the pre-preg layers on the fiber orientations and compare the results to those of the pultruded sample. For the NCF, we segment the different bundles based on their orientation and estimate the orientation distributions for the different bundle types. Lastly, we show the effects of the stitching and backing fibers on the alignment of the UD fibers in the fabric. We show that these effects can be quantified – information that could be used by manufacturers to adjust the stitching to improve UD fiber alignments.

We share our complete pipeline, including data and all experimental results online (see Section 4). The results are shared as Jupyter notebooks, making it easy for others to reproduce our results or test our pipeline with their own data.

2. Materials

In this section we introduce the three UD fiber composites examined in this paper.



(a) Pultruded carbon composite (`HyFiSynCF2-01 B2`)

(b) Pre-preg carbon composite (`DY06 B2`)

(c) Non-crimp fabric reinforced composite (`Stitch-0-1-2-3-4`)

Figure 1: 3D renderings of the three CT scans.

2.1. Pultruded carbon fiber composite

The pultruded carbon fiber reinforced composite was supplied in the form of a rectangular profile with a cross-section of $105 \times 1.9\,\mathrm{mm}^2$. The characteristic fiber diameter was reported in [14] to be $D_f = (7.0 \pm 0.3)\,\mu\mathrm{m}$ and the fiber volume fraction was reported in [15] to be $V_f = 0.67 \pm 0.01$. Note that the carbon fibers are often not cylindrically shaped [18], and the characteristic fiber diameter specified above is defined as the diameter of a circle, with the same area as the cross-section area of the carbon fibers, measured using a string vibration method.

The pultruded carbon fiber profile was cut into a 2 mm wide sample before scanning. The cut sample was scanned with the settings given in Table 1 and the result is shown in Fig. 1a. The 3D reconstruction was done in the Reconstructor Scout-and-Scan software using the default settings for the Zeiss Xradia 520 scanner. During the reconstruction, the scan was aligned in the YZ-plane with respect to the outer surfaces in the Y-direction of the pultrution. For information

Material case	**Pultrusion**	**Pre-preg**	**Non-crimp fabric**
Sample ID	HyFiSynCF2-01	DY06	Stitch-0-1-2-3-4
Reference	[14, 15]	[16]	[17]
	Material		
Fibers	Carbon	Carbon	Glass
Fiber diameter [µm]	7.0±0.3	7.4±0.3	17±1
Matrix	Vinyl ester	Epoxy	Epoxy
Fiber volume fraction	0.67	0.61	0.57
Sample size (y;z) [mm]	(2.0; 1.9)	(2.0; 2.0)	(15; 4.3)
	Scanning		
Voxel size [µm]	2.87	2.87	16.16
FoV: (x; y; z) [mm]	(2.84; 2.91; 2.88)	(2.84; 2.91; 2.88)	(45.0; 15.4; 4.82)
No. of slices (x; y; z)	(992; 1013; 1003)	(992; 1013; 1003)	(2787; 950; 298)
No. of single stitches	1	1	5
Accelerating Voltage [kV]	30	30	50
Power [W]	2	2	4
Filter	Air	Air	LE3
Optical magnification	4x	4x	0.4x
Detector to sample distance [mm]	10.5	10.5	100
Source to sample distance [mm]	7.79	7.79	31
Exposure time [s]	3	3	6
No. of projections	4501	4501	5501
Rotation	360	360	360
Binning	2	2	2
Total scanning time [h]	6	6	69
File type	txm	txm	nii
File-size [GB]	1.99	1.99	1.54
	Analysis		
Analyzed FoV (x; y; z) [mm]	(2.45; 1.89; 1.83)	(2.45; 1.73; 1.67)	(38.6; 12.8; 4.57)
No. of analyzed slices (x; y; z)	(853; 658; 637)	(853; 603; 582)	(2387; 790; 283)
Threshold value (Otsu)	25181	14463	40956
V_f (Otsu)	0.653	0.547	0.893
σ	1.275	1.275	0.525
ρ	2.55	2.55	1.05

Table 1
Material properties, scan parameters and analysis parameters for the pultruded, pre-preg and NCF samples, respectively.

on the mechanical performance of the fibers and the composite material see [14] and [15], respectively.

2.2. Pre-preg carbon fiber composite

The pre-preg carbon fiber composite was manufactured using six pre-preg plies, resulting in a carbon fiber reinforced epoxy laminate with a thickness of 3.3 mm. The fiber volume fraction was estimated to $V_f = 0.61$ in [16]. Before scanning, the carbon fiber laminate was cut into a stick in the fiber-direction with the cross-section area $2.0 \times 2.0\,\mathrm{mm}^2$, i.e., cut in both the width and thickness direction. The sample was scanned using the settings shown in Table 1 and the result is shown in Fig. 1b. Settings are identical to those used for the pultruded profile. Again, 3D reconstruction was done using Reconstructor Scout-and-Scan with default settings, and the reconstruction was aligned in the YZ-plane with respect to the cut surfaces. For more details on the mechanical performance of the pre-preg, see [16].

2.3. Non-crimp fabric glass fiber composite

The non-crimp glass fiber-based composite was manufactured using vacuum infusion, based on a [biax/UD_4/biax] layup of non-crimp glass-fibre fabrics. The resulting composite has a thickness of approximately 4.3 mm and was cut into typical tension-tension fatigue test samples with a width in the scanned gauge-section of 15 mm. The sample was scanned using the settings shown in Table 1 and the result, consisting of five stitched scans, is shown in Fig. 1c. The four UD fabrics used in the non-crimp laminate consist of $1152\,\mathrm{g/m}^2$ (2400 tex) fibers in the 0° direction, each stitched to a backing consisting of $100\,\mathrm{g/m}^2$ ±45° and $19\,\mathrm{g/m}^2$ 90° (both 200 tex). The two biax fabrics consist of $528\,\mathrm{g/m}^2$

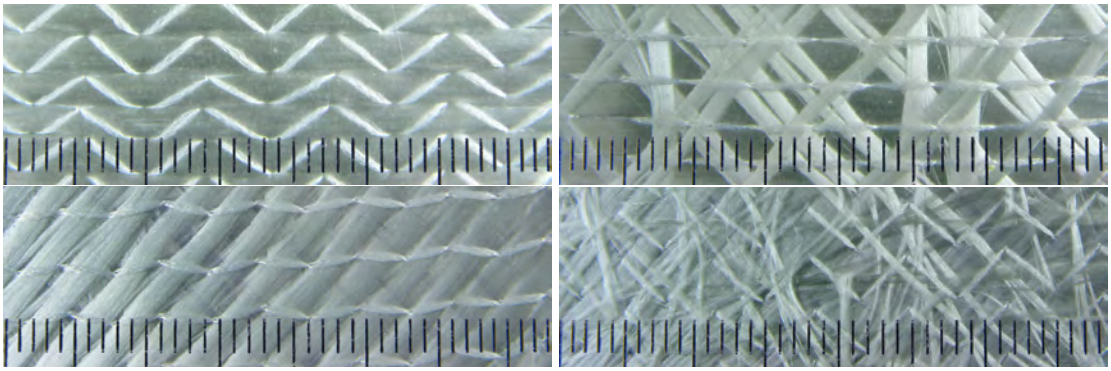


Figure 2: Images of the UD + backing (upper) and biax + backing (lower) fabric seen from the two side. The area shown in the pictures correspond approximately with the area visible in scan `Stitch-0-1-2-3-4`.

fibers at ±45°, in addition to a backing of 19 g/m^2 90° and 100 g/m^2 randomly oriented chopped strand mat (CSM) glass fibers. The information is summarized in the first row in Table 2. For more details on the material system see [17], which includes a tension-tension fatigue damage study performed on this specific material system.

We separate the fiber orientations into four different classes based on their angle, θ, from the UD orientation. The classes are: $C = \{0, -45, 45, 90\}$. We distribute the randomly oriented CSM fibers evenly across the four classes. The fibers in class 0 (UD) have an average diameter of 17 µm, while the fibers in the other classes have an average diameter of 16 µm. The expected volume fractions, $V_c \mid c \in C$ (with distributed CSM), for each fiber bundle class in the material (also reported in Table 2) are:

$$V_0 = 0.7160\,, \quad V_{-45} = V_{45} = 0.1294 \quad \text{and} \quad V_{90} = 0.0252\,. \tag{1}$$

Using these volume fractions and the expected orientations of the bundles, we calculate the expected fibre orientation efficiency factor [1] as

$$\eta_o = \sum_{c \in C} V_c \cos^4(\theta_c) = 0.7807\,. \tag{2}$$

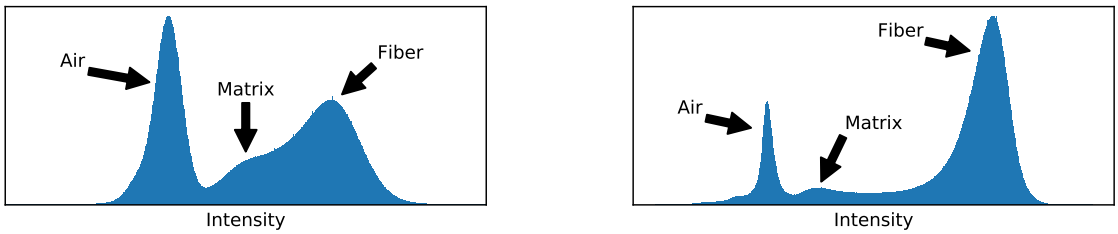
Our analysis focuses on the UD bundles, visible in the upper left picture in Fig. 2, as these are most important for the material stiffness in the axial direction. In this picture, we clearly see the stitching holding the UD bundle onto the backing bundles (shown in the upper right picture). Although the stitching is clearly visible here, the polymeric threads used for stitching are difficult to see in the CT scans, as their density is similar to that of the matrix material. The two sides of the biax-fabric are shown in the lower pictures in Fig. 2.

3. Method

Our analysis of the materials is done through an estimation of the local fiber orientation at every fiber voxel position in each scanning volume. Once the fiber orientations have been estimated for at each voxel position, we can examine both global orientation distribution, as well as local variations in the fiber orientations. To estimate the local fiber orientations across the volumes, we use the approach by [13], which uses a combination of intensity-based thresholding, structure tensor decomposition, and nearest-neighbor classification (for the NCF sample). We use the classification of the bundles to separate the UD fiber bundles from backing and biax bundles.

3.1. Separating fibers from matrix

The first step of the method is to separate the fibers from the background (i.e., matrix and air). Since the fibers are denser than the matrix and air, voxels containing fibers have higher intensities than matrix and air voxels in the reconstructed CT volume. The contrast, and thus our ability to separate fiber voxels from the background, is highly dependent on the scan resolution, scan time, and difference in density between the fiber and matrix materials. The density of the carbon fibers is close to the matrix density, which results in a low contrast between the fibers and matrix



(a) Pultruded carbon fiber

(b) Non-crimp glass fiber

Figure 3: Volume intensity distribution for the pultruded carbon scan (a) and non-crimp scan (b).

in the pultruded carbon and pre-preg materials. For the NCF, the contrast is higher due to the larger difference in density between the glass fibers and the matrix material.

To examine the intensity distribution, we can plot a histogram of the intensities of the voxels, as shown in Fig. 3. The histogram can be used to manually pick a threshold value, which is used to classify voxels as either fiber or background voxels. However, when the contrast between the matrix and fibers are low, as is the case for our carbon fiber scans (see Fig. 3a), choosing a good value becomes difficult. Furthermore, the voxel intensities, and thereby the appropriate threshold value, will vary from scan to scan, depending on scanner settings, samples being scanned, etc. This means a new value should be chosen for every scan, but at the same time, the value should be chosen in a consistent manner to reduce variation in the results. In cases where V_f is already known, we could choose the threshold based on this.

One method for automatically choosing an appropriate threshold value, without accurate prior knowledge about V_f, is Otsu's method [19]. The method works by maximizing the intra-class intensity variance and works well if the intensities can be assumed to have a bimodal distribution, preferably with a valley between the two peaks. However, this is clearly not the case for our scanning data (see Fig. 3), which contains three different distributions (fiber, matrix and air). Luckily, as the air is almost exclusively outside the sample, we can easily create a smaller subvolume, strictly inside the sample, which contains little to no air. This leaves us with only matrix and fiber distributions, which we can separate using Otsu's method to obtain a threshold value. We can use this threshold value to determine whether a voxel is fiber or background. The accuracy of this naive approach depends on the data quality, i.e. contrast, noise and resolution.

3.2. Calculating the structure tensor

The structure tensor method is an effective way of determining orientations of imaged structures in both 2D (images) and 3D (volumes). In particular, it has been shown to work well for fiber-like structures in 3D [6, 10, 11, 12, 13]. In this paper, we use the same structure tensor implementation as [13]. For completeness, a short description of the structure tensor method follows.

In 3D, a structure tensor is a 3-by-3 matrix that captures the local orientation around a point in space. For each point in the volume, V, we can compute the structure tensor, $\mathbf{S}$, as

$$\mathbf{S} = \sum \nabla V \, (\nabla V)^T \,, \tag{3}$$

where $\nabla V = \begin{bmatrix} V_x & V_y & V_z \end{bmatrix}^T$ is the gradient of V and the summation/integration is limited to a certain neighborhood around the point. There are different approaches to calculating gradients and integrating the neighborhood information. Here, we use a Gaussian kernel window for integration and a Gaussian derivative kernel for computing the gradients, which has several nice properties [20]. This leads to the formulation

$$\mathbf{S} = K_\rho * \left(\nabla V_\sigma \, (\nabla V_\sigma)^T \right) \,, \tag{4}$$

where the parameter σ the standard deviation of the Gaussian derivative kernel used for computing the gradient ∇V_σ and the parameter ρ is the standard derivation of the Gaussian kernel K_ρ used for integration. The parameters σ and ρ are also known as the *noise* scale and *integration* scale, respectively. The noise scale σ determines the scale at which the gradient is calculated and is useful for suppressing high-frequency noise. Higher values of σ will result in smoother

gradients by suppressing features that are small relative to σ. The integration scale ρ should be chosen relative to the size of the structures that we are calculating the orientations of. It should be large enough to include the context around a point required to see the structure of interest. For fibers, this means that ρ should depend on the fiber diameter, given that the resolution is greater than the fiber diameter. When the voxels are similar in size or larger than the fiber diameter, choosing ρ based on fiber diameter no longer makes sense, as the individual fibers are not clearly visible in the data. It that case it is less clear how to chose ρ. However, as shown by [6], ST-based estimation of orientation distributions is still viable.

3.3. Estimating local fiber orientations

For fiber-like structures in a CT volume, the orientation of a structure corresponds to the direction of least change in intensity. This corresponds to minimizing the expression $\mathbf{u}^T\mathbf{S}\,\mathbf{u}$ in each point, where $\mathbf{u}$ is a unit vector. In other words, the direction of $\mathbf{u}$, for which $\mathbf{u}^T\mathbf{S}\,\mathbf{u}$ gives the smallest value for each point, is the direction of the least change for that point. The solution can be found analytically using eigendecomposition of $\mathbf{S}$. Because $\mathbf{S}$ is symmetric and positive semi-definite, the eigendecomposition results in three positive eigenvalues and their corresponding mutually orthogonal eigenvectors. The predominant orientation (direction of least change) is that of the eigenvector, $\mathbf{v}$, corresponding to the smallest eigenvalue. Thus, for fiber-like structures, $\mathbf{v}$ is the local orientation of the fiber-like structure at a certain point in space. Note that fiber-like structures are characterized by the smallest eigenvalue being much smaller than the other two eigenvalues. However, we currently do not use the eigenvalues in our analysis. Also, the eigenvector captures only orientation, not direction. Depending on the implementation of the eigensolver, an eigenvector may be computed as either $\mathbf{v}$ or $-\mathbf{v}$. This has to be taken into account in the subsequent analysis.

Using the eigendecomposition, we can estimate the structure orientations in every single point where $\mathbf{S}$ was calculated. However, we are actually only interested in determining the orientation of points that are inside fibers. Unlike $\mathbf{S}$, which depends on the neighborhood, the eigendecomposition is calculated independently for each point. As a result, we can choose not to perform the decomposition for points that are not of interest to us. We only decompose $\mathbf{S}$ at points that are classified as being inside fibers based on our segmentation. This means that $\mathbf{v}$ should generally correspond to local fiber orientations, as $\mathbf{v}$ is only calculated at fiber voxel positions. These local orientations can be used to describe the global fiber orientation distribution, and investigate local distributions in the sample.

3.4. Fiber classification

For the pultruded and pre-preg carbon samples, all the fibers are UD, so we assume all fibers belong to the same distribution. Meanwhile, the NCF is composed of a number of differently oriented fiber bundles. To determine which fiber voxels belong to each of the different bundle orientations (classes), we can use the estimated fiber orientations. This paper uses a naive, yet effective way of determining which bundle type each fiber voxel belongs to. Classification of image pixels/voxels, as done here, is known as semantic segmentation. The segmentation is done by having the user specify the orientation of each fiber class using a class unit vector. Each fiber voxel is then assigned to the class where the angle between the class vector and the local fiber orientation at the voxel is the least. This strategy corresponds to using a K-nearest neighbor classifier with $K = 1$.

3.5. Calculating θ, η_o, in-plane and out-of-plane orientations

Once the local fiber orientations have been estimated, we can calculate a number of different metrics, which can help us interpret the orientations. Since all three materials studied in this paper are UD fiber materials, the first metric is the angle between the expected UD orientation (X-axis) and the local fiber orientation $\mathbf{v} = [v_x, v_y, v_z]$. We denote this value $\theta(\mathbf{v}) = |\arccos(v_x)|$. As previously noted, $\mathbf{v}$ captures orientation, not direction. Thus, we take the absolute value when calculating θ. Besides providing information about the alignment of the UD fibers, θ can also be used to calculate the fiber orientation efficiency factor [1]

$$\eta_o = \sum_{i=1}^{N} a_i \cos^4 \theta_i \,, \tag{5}$$

where θ_i is the angle with respect to the loading direction of the i-th fraction a_i. In our case N is simply the number estimated angle values and $a_i = 1/N$.

While θ is useful for determining the deviation from the expected fiber orientation, it does not fully explain the actual orientation of the fibers. Describing and visualizing 3D orientation distributions in not trivial. Thus, projecting

the orientations onto a 2D plane can be useful. For fibers oriented along the X-axis, projections onto the XY- and XZ-planes are useful. After projecting, we compute the angle between the projection and the X-axis. The angles between the X-axis and the projections of $\mathbf{v}$ onto the XY- and XZ-planes can be calculated as

$$\Theta_{\mathrm{XY}}(\mathbf{v}) = \arctan \frac{v_y}{v_x} \quad \text{and} \quad \Theta_{\mathrm{XZ}}(\mathbf{v}) = \arctan \frac{v_z}{v_x} \,, \tag{6}$$

respectively. Due to the lay-up of the different fiber bundles in the NCF, we want to calculate the signed out-of-XY-plane angle

$$s_{\mathrm{X}}\theta_{\mathrm{XY}}(\mathbf{v}) = \operatorname{sgn}(v_x) \arcsin(v_z) \,, \tag{7}$$

rather then calculating $\Theta_{\mathrm{XZ}}(\mathbf{v})$. Here, sgn is the sign function, which is used to flip all vectors with a negative x-component. We need this to consistently interpret the sign of the z-component, used to calculate the out-of-plane angle. However, it also means that vector orthogonal to the X-axis ($v_x = 0$) will be interpreted as having an out-of-plane angle $s_{\mathrm{X}}\theta_{\mathrm{XY}}(\mathbf{v}) = 0°$, even if $v_z \neq 0$. But since $v_x = 0$ is very unlikely, this has little to no effect on the results. This approach makes sense for investigating the out-of-plane waviness of all the fiber bundles that are angled $\pm 45°$ or less to the X-axis. If we wanted to investigate the signed out-of-plane waviness of the 90° backing bundles, we should use $s_{\mathrm{Y}}\theta_{\mathrm{XY}} = \operatorname{sgn}(v_y) \arcsin(v_z)$ instead. Note that both these signed approaches only work when fibers are relatively straight, which is the case for the three materials investigated here. An alternative is to calculate the absolute out-of-XY-plane angle

$$\theta_{\mathrm{XY}} = \arcsin \left| v_z \right| \,. \tag{8}$$

Here, we do not have to choose a positive direction, but this also means that we no longer distinguish between positive and negative z-components. In other words, without choosing a positive fiber direction, we cannot distinguish between fibers moving up and down through the XY-plane.

4. Results

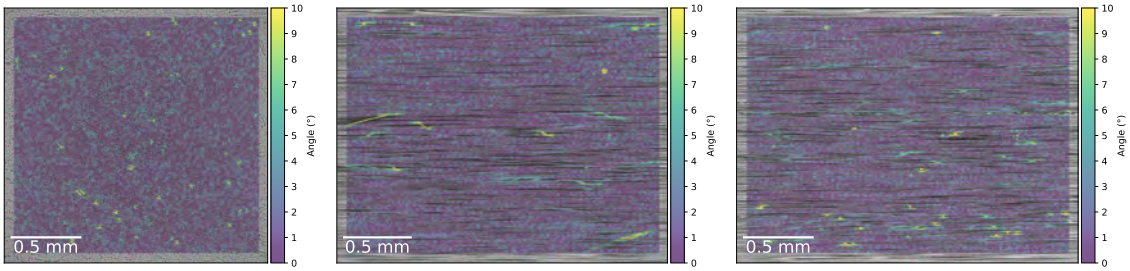
Using the method described in the previous section, we examine the CT scans of the three fiber materials from Table 1. For each scan, we show a number of histograms of the fiber orientation distributions describing the degree of misorientation in the fiber samples compared to the expected orientations of the fiber material. For the NCF scan, we also segment the fibers into different classes, depending on their expected orientations, and show the in-class orientation distribution for the 0° UD class. Lastly, we plot local orientations in key areas of the scans, highlighting the misalignment of fibers due to the manufacturing processes. All results are available in, and can be reproduced by, the Jupyter notebooks and data published online [21]. For structure tensor calculations we rely on our `structure-tensor` package for Python [22].

4.1. Pultruded carbon fiber composite

The pultruded carbon sample in scan `HyFiSynCF2-01 B2` has the simplest structure of the three samples examined in this paper. This is evident in Fig. 4, which shows a single slice of data along each axis overlaid with colors showing θ. Note that θ angles of 10 or more degrees are all shown using the same color to enhance the contrast for values between 0 and 10 degrees. From the cross-section of the fibers (Fig. 4a), the material appears quite homogeneous, although there are clearly some fibers that are misaligned.

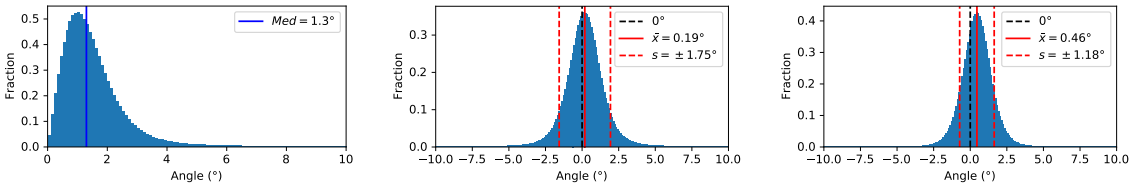
The distribution of θ for the estimated orientations, along with distributions of Θ_{XY} and Θ_{XZ}, are shown in Fig. 5. We see that the fibers of the pultruded material appear very well aligned overall, with a median θ angle of only 1.3°. The reason the distribution of θ decays close to 0° is that the θ angles are calculated as the distance between two points on a half sphere. As the the area on the sphere, and thus the bin $[0, 1)°$ is much smaller than that for $[1, 2)°$, points are less likely to fall into the $[0, 1)°$ bin. In other words, it very unlikely that two points on the half sphere are very close to each other.

Some of the estimated misalignment could be explained by misalignment of the sample in the scanner. This could also explain why the mean values of the projected orientations are not zero, but rather 0.19° and −0.46° for the Θ_{XY} and Θ_{XZ} respectively. However, it is also possible that the fibers are just slightly misoriented overall in the cut out sample. The standard deviation is 1.75° for Θ_{XY} and 1.18° for Θ_{XZ}, which corresponds with our observation from Fig. 4 that



(a) YZ-slice (b) XY-slice (c) XZ-slice

Figure 4: Slices of the pultruded composite, `HyFiSynCF2-01 B2`, along each of the three axis. The intensities of the CT data are shown as grey-scale, overlaid with colors representing the estimated local θ angle. Data is oriented so that the fibers should align with the X-axis and high values of θ (angle from X-axis) corresponds to more misalignment.



(a) Angle from X-axis (θ) (b) In XY-plane (Θ_{XY}) (c) In XZ-plane (Θ_{XZ})

Figure 5: Estimated orientation distributions of the pultruded carbon sample, `HyFiSynCF2-01 B2`. (a) is the angle between the estimated local orientation and the X-axis, (b) is the angle of the orientation projected onto the XY-plane and (c) is the angle of the orientation projected onto the XZ-plane.

the fibers are very well aligned overall. As a result, we get an orientation efficiency factor, defined in Eq. (2), very close to unity, with a value of $\eta_o = 0.9973$. The higher variation in the XY-plane orientations could result from the large width (Y-axis), compared to thickness (Z-axis) of the pultruded material.

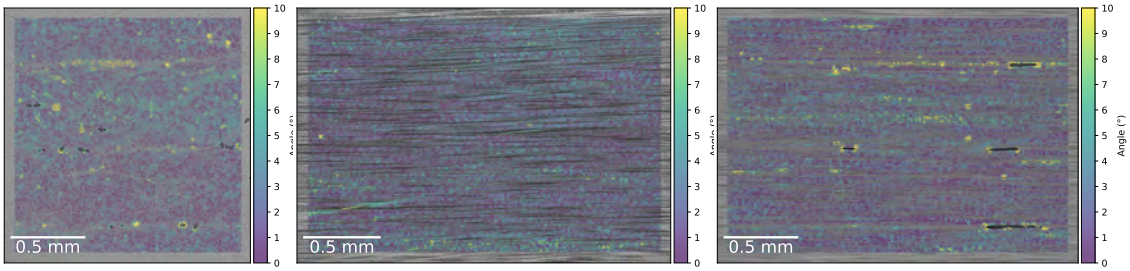
4.2. Pre-preg carbon fiber composite

The pre-preg carbon material from scan `DY06 B2` is structurally slightly more complex than the pultruded sample. The material is composed of layers of pre-impregnated layers of fibers. We see the interfaces between some of these layers in both Fig. 6a and 6c, where small pockets of air (dark spots) are visible and θ is higher in some cases. Compared to the pultruded sample, the pre-preg appears to have more variation in the fiber orientations (see Fig. 7), with a median θ of 1.9° and a standard deviation of 3.42° and 2.75° for Θ_{XY} and Θ_{XZ}, respectively. In particular, the larger variation in the XY-plane, which corresponds to the pre-preg layers, is interesting.

Further investigating Θ_{XY} locally, it is clear that the individual layers of the sample are not oriented the same. Fig. 8a shows a single cross-sectional (YZ) slice from the scan, overlaid with colors representing Θ_{XY}, as well as a plot of the median orientations along the Z-axis. From the colored image, it appears that some layers have been rotated a bit, causing fibers in those layers to be slightly misoriented in the XY-plane. This is confirmed by the plot, which takes the median of Θ_{XY} along the horizontal slice axis (Y-axis). The plot clearly shows that the fibers of some layers are oriented ~2° different from others. This pattern is consistent throughout our pre-preg sample, but not present for the pultruded sample as shown in Fig. 8b.

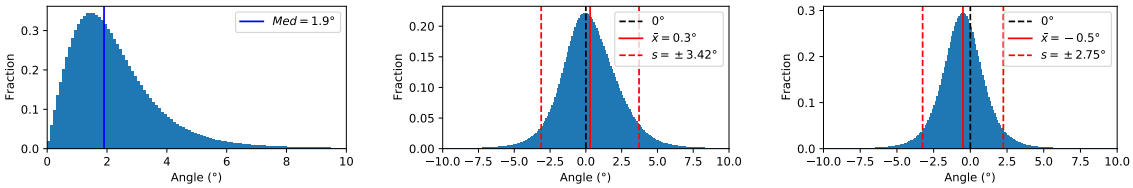
4.3. Non-crimp fabric glass fiber composite

The non-crimp glass fiber laminate has the most complex structure of the materials examined in this paper. The primary UD fibers are stitched together in bundles and supported by bundles of backing fibers as shown in Fig. 2 and 9. The expected (spec.) fiber class fractions are shown in Table 2, along our estimated fractions, based on our semantic segmentation of the fibers. The table shows agreement between the expected class fractions and the estimated ones. The biggest difference is for the 90° bundle class, where the estimated fraction is 20% below expected. This



(a) YZ-slice (b) XY-slice (c) XZ-slice

Figure 6: Slices of the pre-preg composite, DY06 B2, along each of the three axis. The intensities of the CT data are shown as grey-scale, overlaid with a colors representing the estimated local θ angle. Data is oriented so that the fibers should align with the X-axis and high values of θ (angle from X-axis) corresponds to more misalignment.



(a) Angle from X-axis (θ) (b) In XY-plane (Θ_{XY}) (c) In XZ-plane (Θ_{XZ})

Figure 7: Estimated orientation distributions of pre-preg carbon sample, DY06 B2. (a) is the angle between the estimated local orientation and the X-axis, (b) is the angle of the orientation projected onto the XY-plane and (c) is the angle of the orientation projected onto the XZ-plane.

$[_b$Biax$/_b$UD$/_b$UD$/$UD$_b/$UD$_b/$Biax$_b]$	0°	−45°	45°	90°	CSM
Area weight [g/m^2]	4×1152	4×50+2×296	4×50+2×296	6×19	2×100
V_c (spec.)	0.7083	0.1217	0.1217	0.0175	0.0307
V_c (spec. CSM dist.)	0.7160	0.1294	0.1294	0.0252	
V_c (out method)	0.7304	0.1273	0.1222	0.0200	
Difference [%]	2.01	1.62	5.56	20.63	

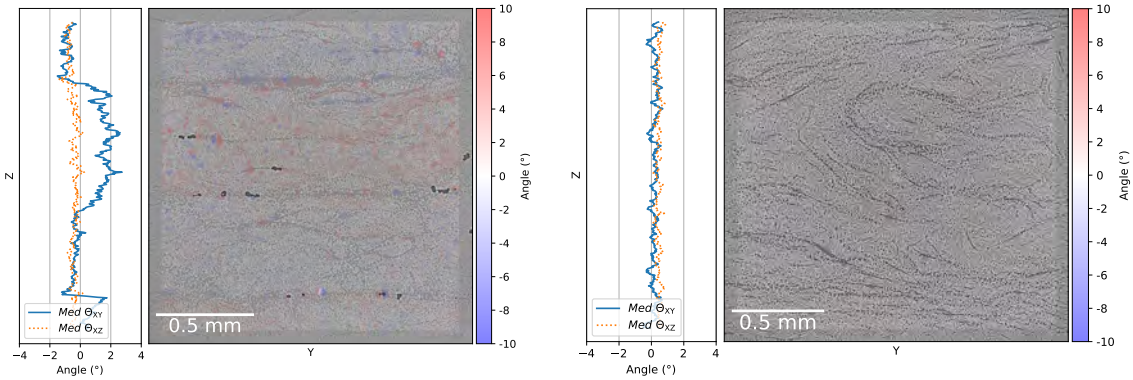
Table 2
Composition of the non-crimp fiber sample. The fraction calculated from the material specification is compared to the fraction estimated using structure tensor-based semantic segmentation.

is likely due to scarcity of the 90° bundles, which only represent about 2.5% of the total fiber volume. Based on the specification, we calculated an η_o value of 0.7807, which is slightly higher than the value of 0.7761 estimated using our method.

For the NCF, we can investigate the distributions per fiber bundle class. Fig. 10 shows the distributions of θ, Θ_{XY} and $s_X\theta_{XY}$ for all fibers, while Fig. 11 shows the distributions for the 0° UD class only. The UD fibers are generally well-aligned, with a median θ of 4.87°, and a standard deviation of 5.44° for Θ_{XY} and 4.53° for $s_X\theta_{XY}$. The mean of −0.06° for both the in- and out-of-plane angles indicate that the data is aligned correctly with the X-axis.

All three backing bundle classes also appear to be aligned correctly, based on their mean orientations (see [21]). There is a clear correlation between the volume fraction of the class and the in-class variation of the orientations. This is probably partially related to the integration scale of the structure tensor method, which causes some "bleeding" between the bundles. This impacts the less common classes the most, since they have more surface area compared to their volume.

Although the standard deviation of Θ_{XY} is quite low for the 0° class, it is significantly higher than for the carbon samples. One contributing factor is an in-plane waviness, which appears to be the result of the stitching of the UD



(a) DY06 B2

(b) HyFiSynCF2-01 B2

Figure 8: Estimated local Θ_{XY} for a single slice of the pre-preg carbon sample (a), `DY06 B2`, and pultruded sample (b), `HyFiSynCF2-01 B2`, respectively. The images show a single slice overlaid with colors representing Θ_{XY} values. The plots show the median of the Θ_{XY} along the horizontal axis (Y-axis). In (a), we see that the pre-preg layers are rotated slightly, creating a significant variation in the orientations along the Z-axis. As expected, (b) does not contain this type of variation.

bundles. Fig. 12 shows local Θ_{XY} values for a single slice inside the UD bundles and a plot of median Θ_{XY} values along the X-axis. The waviness of the bundles matches the stitching pattern of the material. The black dashed lines, corresponding to the peak gradient of the smoothed median, mark the estimated location of the stitching. The mean interval between these lines along the X-axis is 4.381 mm with a standard deviation of 0.453 mm. This aligns with the stitching interval of approximately 4.1 mm measurable in the upper left image in Fig. 2. The pattern is persistent across all four UD layers of the NCF (see [21]). The degree of in-plane waviness is likely dependant on the tension of the stitching thread.

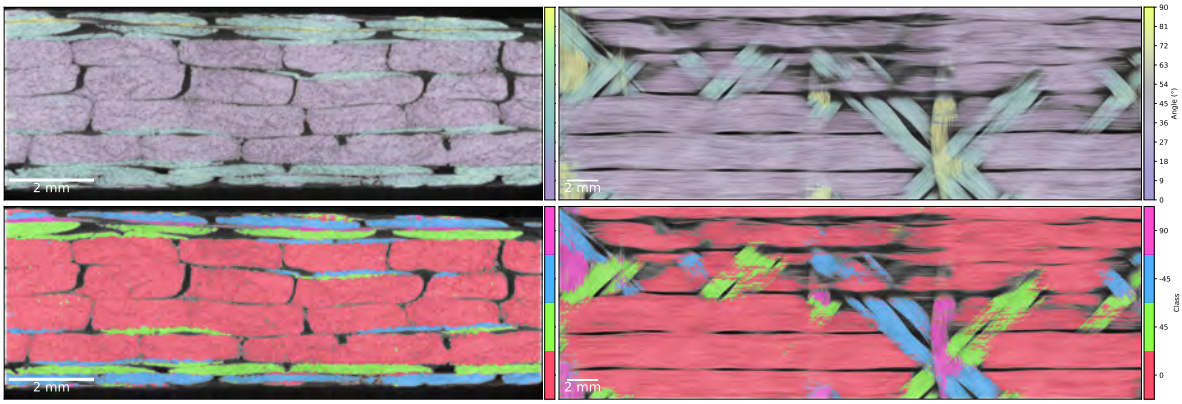
Another periodic pattern visible in the UD bundles is found in the out-of-XY-plane angles, $s_X\theta_{XY}$. The pattern, shown in Fig. 13, appears to be a result of the adjacent $\pm 45°$ backing fiber bundles exerting pressure on the UD bundles, which creates a slight out-of-plane waviness. By rotating the slice 45° and calculating the median $s_X\theta_{XY}$ values for the slice along both axes, we clearly see the periodic pattern (see Fig. 13a). We can then find the local extrema using the smoothed median values along both axes, which are likely located near the edge of the backing bundles. This is confirmed by drawing the lines on top of a nearby slice, where the backing bundles are visible (Fig. 13b). Along both axes, the dashed lines align well with the visible 45° bundles. The degree of this out-of-plane waviness will likely depend on the structure of the backing bundles and the pressure on the fabrics during resin infusion.

5. Discussion

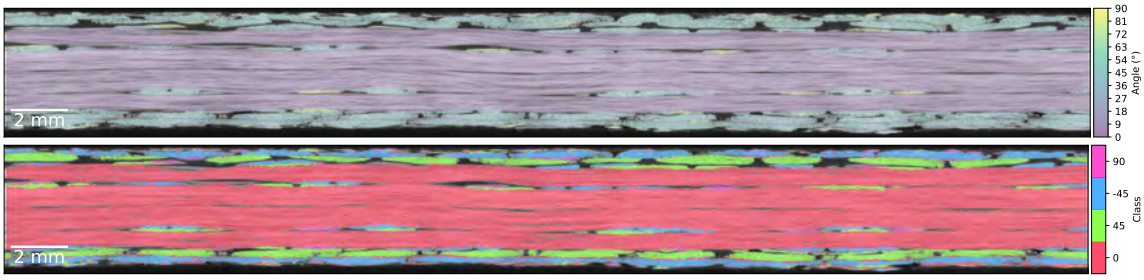
Our investigation of the three composite material systems using structure tensor analysis demonstrates that we can measure small differences in fiber orientation that are important for material characteristics. This has also previously been shown, and it has likewise been shown that the structure tensor measures orientations that are similar to what can be measured using other methods [6, 23].

Our analysis approach, however, takes the quantification further by integrating the measured orientations in the directions of change, such that deviations become very clear. This is the case for the pre-preg sample, where we see deviations in the layers that have been laid up, but most pronounced in the stitched NCF composite sample. The stitching gives deviations relative to the main UD fiber direction that are largest between the stitches, which can be seen from the color overlay in Fig. 12, but even more when aggregating these, as illustrated in the plot showing clear periodic patterns. This implies that the structure tensor analysis enables quantifying orientation differences that allow for optimizing the production of the fiber fabric and the composite.

The structure tensor takes two parameters, σ and ρ, that were adjusted according to the size of the structures that we measured, i.e. the fiber diameters. We did not do an extensive study of how these parameters should be set, because we experienced little change in the quantification when changing these parameters. Another parameter is the intensity threshold, but that too did not influence the result very much. This makes the basis for our structure tensor analysis

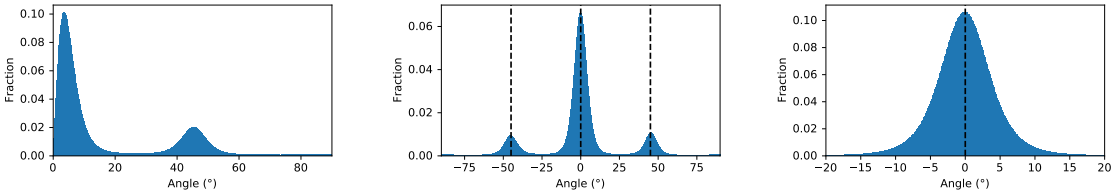


(a) YZ-slice (b) XY-slice



(c) XZ-slice

Figure 9: Slices of NCF composite, `Stitch-0-1-2-3-4`, along each of the three axis. The intensities of the CT data are shown as grey-scale, overlaid with a colors representing the estimated local θ angle (top) and semantic class (bottom). Data is oriented so that the UD fibers should align with the X-axis and high values of θ (angle from X-axis) corresponds to more misalignment.



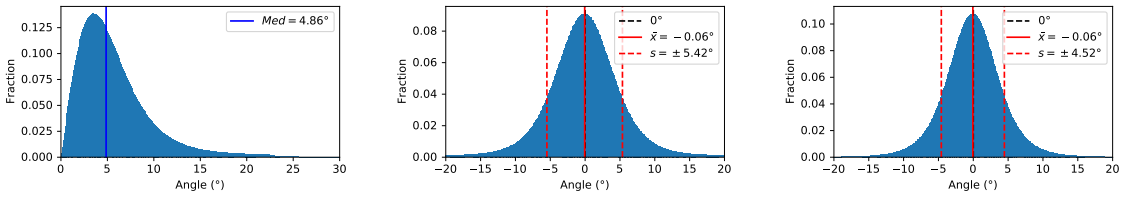
(a) Angle from X-axis (θ) (b) In XY-plane (Θ_{XY}) (c) Out of XY-plane ($s_X\theta_{XY}$)

Figure 10: Estimated orientation distributions of non-crimp sample, `Stitch-0-1-2-3-4`. (a) is the angle between the estimated local orientation and the X-axis, (b) is the angle of the orientation projected onto the XY-plane and (c) is the signed out-of-XY-plane angle.

easy, and we trust the measures that come out.

An important contribution of our work is the open-source GPU-based implementation of structure tensor and eigendecomposition [22], which makes computation exceptionally fast without a limit on how large a volume that can be processed [21]. The ability to process a volume in few minutes, that on a lab-scanner takes many hours or days to record, gives interesting perspectives. Instead of lab µCT we could use synchrotron imaging, and here the analysis could be done in a similar pace as the scanning, give the possibility for large-scale studies.

We find it interesting that our pultruded sample shows better fiber alignment than the pre-preg sample, with a lower standard deviation for both Θ_{XY} and Θ_{XZ}. As previously mentioned, the `DY06` sample is 2 mm^2 cut from a 3.3 mm



(a) Angle from X-axis (θ) (b) In XY-plane (Θ_{XY}) (c) Out of XY-plane ($s_X\theta_{XY}$)

Figure 11: Estimated orientation distributions of non-crimp sample, `Stitch-0-1-2-3-4`, for the 0° (UD) class (red in Fig. 9). (a) is the angle between the estimated local orientation and the X-axis, (b) is the angle of the orientation projected onto the XY-plane and (c) is the signed out-of-XY-plane angle.

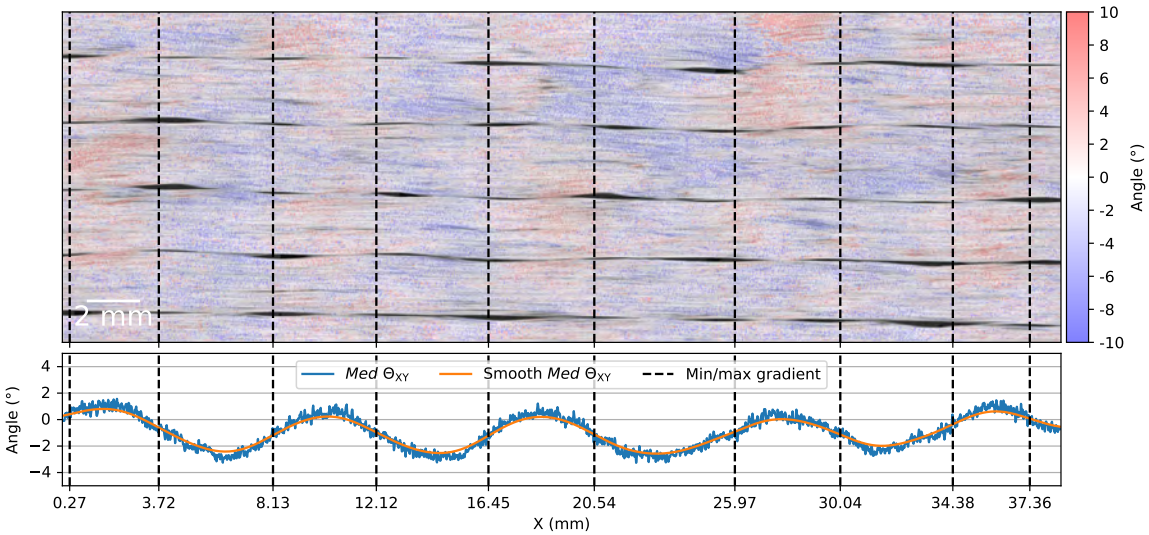


Figure 12: Single XY-slice inside one of the four UD layers overlaid with estimated local Θ_{XY} values, and a plot of the median Θ_{XY} value along the vertical axis. We see a clear periodic pattern in the Θ_{XY} values along the X-axis. Using the gradient smoothed median value, local extrema have been found and marked with dashed black lines. These are the estimated positions of the stitches.

thick 6-ply laminate. The cut contains four plies, with the first and last being cut in half. As shown in Fig. 8a, the large standard deviation for Θ_{XY} (3.42°) partially explained by inconsistent alignment of the plies. However, this does not explain the larger deviation for Θ_{XZ}. Further examination of the orientation distributions of the two central plies of the pre-preg sample (see Fig. 14) show that the standard deviation for Θ_{XY} and Θ_{XZ}, within each of the plies, is still larger than that of our pultruded sample (Fig. 5). The difference in the ply orientation between ply 2 and 3 is also clearly visible in Fig. 14a and 14b. We should note that the pre-preg and pultruded samples are not manufactured using the same type of carbon fiber bundles. It should also be noted that Θ_{XY} and Θ_{XZ} are not normally distributed. However, we still think the standard deviation is a convenient metric for describing the variation in the UD fiber distributions.

6. Conclusion

We show that it is possible to estimate and quantify fiber orientations, both globally and locally, from X-ray CT images using structure tensor-based analysis. In particular, we show that it is possible to detect and quantify different effects of the manufacturing process and material structure in three different fiber composites used in wind turbine blades.

For all three materials, we show how to estimate the fiber orientations, calculate the in-plane and absolute mis-

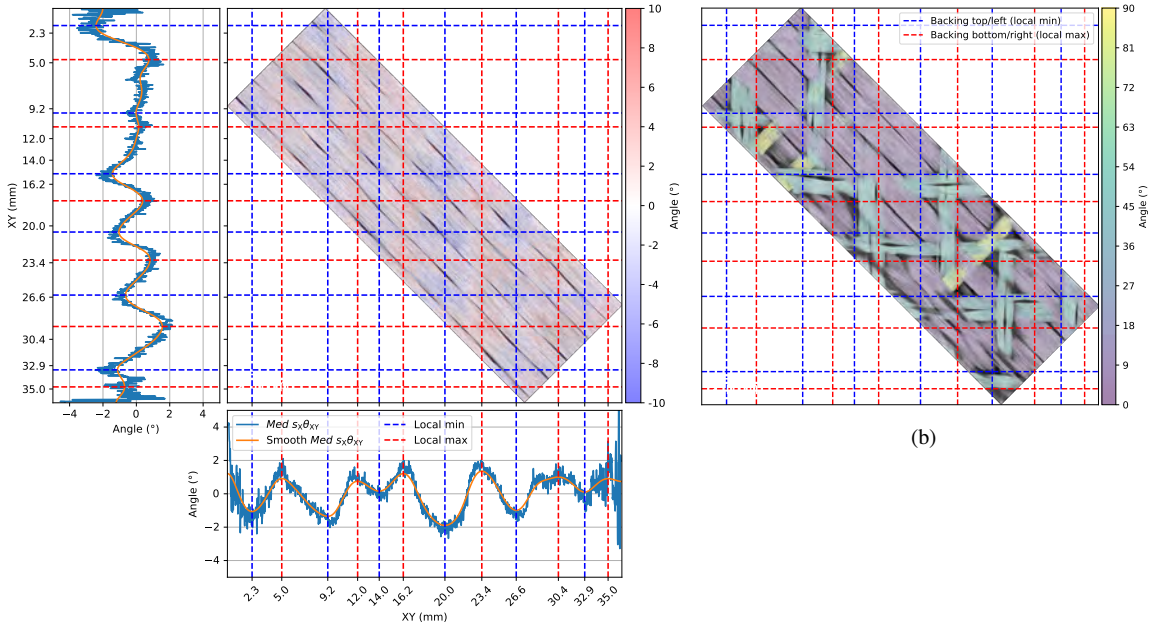


Figure 13: Out of plane waviness in UD bundles due to ±45° backing bundles. (a) shows a single slice inside one of the four UD bundle layers rotated 45°. The slice is overlaid with colors representing the estimated local $s_X\theta_{XY}$ values. The plots left and below the slice show the median of $s_X\theta_{XY}$ along the vertical and horizontal axes, respectively. The blue and red lines are the local extrema on the smoothed median. (b) shows the extrema from (a) on top of a nearby slice (0.34 mm further along the Z-axis), where we see the ±45° backing bundles. We see that the local extrema align well with the edges of the bundles. The coloring is based on the estimated local fiber angle to the X-axis, (θ).

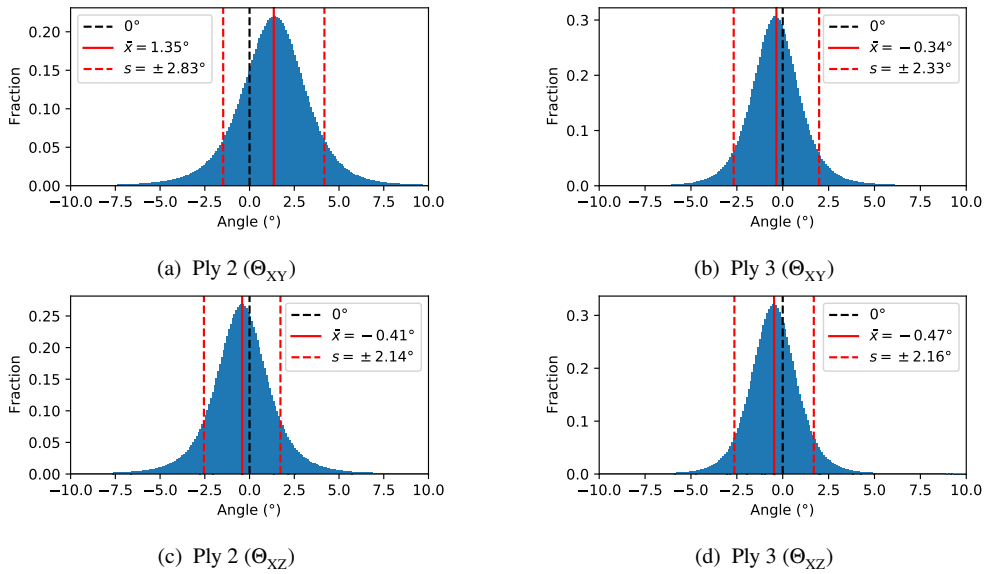
alignment from the specified UD direction. For the pre-preg sample, we show how misalignment of the plies, which may reduce the overall material quality, can be found and quantified. We also see that even when excluding the effects of misalignment of the plies, our pultruded sample still shows better fiber alignment than the pre-preg sample. To determine if pultruded carbon composites are generally more aligned than pre-preg carbon composites, more sample will have to be examined. Our method is well suited for such a study, as the computations only take a few minutes on modern GPUs. Thus, collecting and scanning samples are by far the most time-consuming part of this task.

For the NCF composite, we estimate orientations and separate the bundles into classes, as previously done in [13]. Our contribution is showing the effects of stitching and backing bundles on the UD fiber orientations. We quantify these effects locally in the images and show that the periodic waviness of the UD bundles correlate with the stitching and backing bundles. The quantification of these effects allow manufacturers to choose production parameters, which minimize the waviness of the UD fibers, thereby improving important material properties. For instance the tension of the stitching thread can be tuned to minimize in-plane waviness of the UD bundles. Of course, we examine only a few of many potential effects, which are related to production parameters.

All in all, our publicly available code and notebook, built entirely on open source libraries, allow both manufacturers of composite structures (e.g., wind turbine blades), manufacturers of composites, and researchers, to perform quantitative quality control of UD composite materials. While this paper focuses on UD fiber materials used in blades, most of our approach is applicable to CT images of many different types of fiber materials.

Acknowledgments

N Jeppesen's work is supported by FORCE Technology and RELIABLADE. The X-ray scan was captured using a Zeiss Xradia Versa 520 scanner at DTU Center for Advanced Structural and Material Testing (CASMaT). The work is



(a) Ply 2 (Θ_{XY}) (b) Ply 3 (Θ_{XY})

(c) Ply 2 (Θ_{XZ}) (d) Ply 3 (Θ_{XZ})

Figure 14: Estimated in-plane angle distributions for the pre-preg carbon sample, `DY06 B2`, second and third ply from the top, respectively.

also supported by The Center for Quantification of Imaging Data from MAX IV (QIM) funded by The Capital Region of Denmark.

CRediT authorship contribution statement

N. Jeppesen: Methodology, Software, Writing - original draft, Writing - review & editing. **L. P. Mikkelsen:** Conceptualization, Data Curation, Writing - original draft, Writing - review & editing. **A. B. Dahl:** Supervision, Writing - review & editing. **A. N. Nymark:** Writing - review & editing. **V. A. Dahl:** Methodology, Writing - review & editing.

References

[1] H. Krenchel, Fibre reinforcement. Theoretical and practical investigations of the elasticity and strength of fibre-reinforced materials, Akademisk Forlag, 1964.

[2] B. Budiansky, Micromechanics, Comput Struct 16 (1983) 3–12. https://doi.org/10.1016/0045-7949(83)90141-4.

[3] B. Budiansky, N. Fleck, Compressive failure of fibre composites, J Mech Phys Solids 41 (1993) 183–211. https://doi.org/10.1016/0022-5096(93)90068-Q.

[4] T. J. Vogler, S. Y. Hsu, S. Kyriakides, Composite failure under combined compression and shear, Int J Solids Struct 37 (2000) 1765–1791.

[5] D. Liu, N. Fleck, M. Sutcliffe, Compressive strength of fibre composites with random fibre waviness, J Mech Phys Solids 52 (2004) 1481–1505. https://doi.org/10.1016/j.jmps.2004.01.005.

[6] R. Karamov, L. M. Martulli, M. Kerschbaum, I. Sergeichev, Y. Swolfs, S. V. Lomov, Micro-CT based structure tensor analysis of fibre orientation in random fibre composites versus high-fidelity fibre identification methods, Compos. Struct. 235 (2020) 111818. https://doi.org/10.1016/j.compstruct.2019.111818.

[7] P. J. Creveling, W. W. Whitacre, M. W. Czabaj, A fiber-segmentation algorithm for composites imaged using X-ray microtomography: Development and validation, Compos. Part A Appl. Sci. Manuf. 126 (2019) 105606. https://doi.org/10.1016/j.compositesa.2019.105606.

[8] M. J. Emerson, K. M. Jespersen, A. B. Dahl, K. Conradsen, L. P. Mikkelsen, Individual fibre segmentation from 3D X-ray computed tomography for characterising the fibre orientation in unidirectional composite materials, Compos. Part A Appl. Sci. Manuf. 97 (2017) 83–92. https://doi.org/10.1016/j.compositesa.2016.12.028.

[9] V. A. Dahl, M. J. Emerson, C. H. Trinderup, A. B. Dahl, Content-based propagation of user markings for interactive segmentation of patterned images, in: Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops, 2020, pp. 994–995. https://doi.org/10.1109/CVPRW50498.2020.00505.

[10] I. Straumit, S. V. Lomov, M. Wevers, 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 (2015) 150–158. https://doi.org/10.1016/j.compositesa.2014.11.016.

[11] I. Straumit, C. Hahn, E. Winterstein, B. Plank, S. V. Lomov, M. Wevers, Computation of permeability of a non-crimp carbon textile reinforcement based on X-ray computed tomography images, Compos. Part A Appl. Sci. Manuf. 81 (2016) 289–295. https://doi.org/10.1016/j.compositesa.2015.11.025.

[12] N. Q. Nguyen, M. Mehdikhani, I. Straumit, L. Gorbatikh, L. Lessard, S. V. Lomov, 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 (2018) 14–23. https://doi.org/10.1016/j.compositesa.2017.10.018.

[13] N. Jeppesen, V. A. Dahl, A. N. Christensen, A. B. Dahl, L. P. Mikkelsen, Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor, IOP Conf. Ser. Mater. Sci. Eng. 942 (2020) 012037. https://doi.org/10.1088/1757-899X/942/1/012037.

[14] R. Kumar, L. P. Mikkelsen, H. Lilholt, B. Madsen, Understanding the mechanical response of glass and carbon fibres: stress-strain analysis and modulus determination, IOP Conf. Ser. Mater. Sci. Eng. 942 (2020) 012033. https://doi.org/10.1088/1757-899X/942/1/012033.

[15] R. Kumar, L. P. Mikkelsen, H. Lilholt, B. Madsen, Investigation of accurate determination of strain to failure of unidirectional carbon fibre composites, Under preparation (2021).

[16] L. P. Mikkelsen, S. H. Madsen, J. I. Bech, R. Baldini, Alternative geometries for test coupons measuring tensile properties of thicker unidirectional carbon fiber reinforced epoxy, Under preparation (2021).

[17] K. M. Jespersen, L. P. Mikkelsen, Three dimensional fatigue damage evolution in non-crimp glass fibre fabric based composites used for wind turbine blades, Compos. Sci. Technol. 153 (2017) 261–272. https://doi.org/10.1016/j.compscitech.2017.10.004.

[18] J. J. Huether, W. V. Liebig, Investigations of the carbon fibre cross-sectional areas and their non-circularities by means of laser diffraction, IOP Conf. Ser. Mater. Sci. Eng. 942 (2020). https://doi.org/10.1088/1757-899X/942/1/012034.

[19] N. Otsu, A Threshold Selection Method from Gray-Level Histograms, IEEE Trans. Syst. Man Cybern. Syst. 9 (1979) 62–66. https://doi.org/10.1109/TSMC.1979.4310076.

[20] J. Weickert, Anisotropic Diffusion in Image Processing, Teubner Stuttgart, 1998.

[21] N. Jeppesen, L. P. Mikkelsen, V. A. Dahl, A. N. Christensen, A. B. Dahl, Quantifying effects of manufacturing methods on fiber orientation in unidirectional composites using structure tensor analysis [data set] (2021). https://doi.org/10.5281/zenodo.4446499.

[22] N. Jeppesen, V. A. Dahl, structure-tensor v0.1.2 (2021). https://doi.org/10.5281/zenodo.4573913.

[23] F. Salling, N. Jeppesen, S. M.R., J. Hattel, L. P. Mikkelsen, Individual Fibre Inclination Segmentation from X-ray Computed Tomography using Principal Component Analysis, Under preparation (2021).

4.4 Contribution summary

The two papers show that structure tensor analysis is very effective for extracting fiber orientations from CT scans of fiber-reinforced composites, and that this information is detailed enough to detect and quantify local variations in the orientations. Because the information can be retrieved in a matter of minutes, the barrier to access for researchers and manufacturers is reduced significantly.

As mentioned in both contributions, one of the key advantages of using the structure tensor method for estimating fiber orientation, compared to methods based on individual fiber tracking, is the speed and scalability of the structure tensor method. By utilizing vector operations on the CPU, or in particular the GPU, we can compute the structure tensor and the eigendecomposition fast. Both the separability of the Gaussian filter and the analytically calculated eigendecomposition play a significant role in the performance of our method.

As shown in Paper D, the structure tensor is computed locally, which means we can split large problems into blocks and compute the structure tensor for each block individually. This allows us to deal with volumes of any size and to distribute the task across any number of CPUs and/or GPUs. The proposition of being able to work on very large volumes without having to wait days for the results is very interesting, as it allows researchers and analysts to work on much larger, and thus more representative samples.

The results in Paper E show that we still have much to learn about how the manufacturing process effects the fiber microstructures. They also show that structure tensor analysis is an excellent tool for extracting the information about fiber orientations needed to characterize and quantify these variations.

4.5 Discussion

A major challenge with the type of work presented in Paper D and Paper E is validating the results, specifically the estimated orientations. These depend on data quality, background segmentation, and structure tensor implementation details (e.g., how gradients are calculated and which integration window is used). In both contributions, we perform qualitative validation by coloring volume slices based on the estimated orientations. This shows that the results align well with the fibers visible in the data. However, performing a quantitative validation of the results is more difficult, as it requires accurate knowledge about the fiber orientations, which is what we are trying to obtain in the first place. One way to test the validity of the structure tensor results is to compare them to results obtained using a completely different method. This was done by Karamov et al. 2020, who compared their structure tensor results to results obtained with several other methods. Similarly, we have compared our structure tensor results to results obtained using a different method based on individual fiber tracking (see Appendix B). In both cases, the fiber orientations obtained with

structure tensor analysis align well with results from other methods. However, I still think there is a need for more quantitative validation of the accuracy of structure tensor-based orientation analysis of fibers.

One way to quantify the accuracy of the structure tensor method would be to create a simulated CT volume, where all fiber orientations are known. This way, the accuracy of the estimated fiber orientations could be studied in detail. Furthermore, this approach would be well-suited for studying the effects of varying resolution, noise, and choice of structure tensor parameters σ and ρ, on the accuracy. However, for the results to generalize to real data, the simulated data needs to be realistic.

Our two contributions use a relatively simple approach for identifying which voxels belong to the fibres. This is done intentionally to keep the pipeline simple and keep focus on the estimation and characterization of the fiber orientations. Thus, we use simple intensity-based thresholding to separate fiber and background voxels. This approach is sufficiently accurate for our purpose, as we always aggregate at least a few hundred samples at a time. However, we could improve the background segmentation accuracy by using more of the information acquired from the eigendecomposition, such as the eigenvalues and derived metrics such as linear anisotropy, c_l. These values could also be used to weigh the contribution of the individual eigenvectors when calculating the orientation distributions, rather than relying on binary segmentation of the background.

Although the interpretation of the eigenvectors is simple for fiber-like structures (the eigenvector corresponding to the smallest eigenvalue, $\mathbf{v}$, encodes the fiber orientation), it is important to remember that we want to estimate orientations, not directions, and that the eigenvectors $\mathbf{v}$ and $-\mathbf{v}$ both encode the same orientation. This is known as antipodal symmetry and is important to keep in mind whenever we want to say something about an orientation based on a vector.

As long as the fibers are expected to be oriented in the same direction, it is reasonable to describe the orientation distribution by choosing the pole to be at the expected orientation and flipping vectors, so all vectors are pointing towards the pole. We use this approach in both Paper D and Paper E to describe and compare the different unidirectional fiber composites. However, it is important to remember that although we think it is reasonable to use the mean and standard deviation here, the angles calculated from the projected orientations are *not* normally distributed.

4.6 Summary

In this chapter, I have introduced our structure tensor method for 3D images. The method uses separable Gaussian filters to calculate the structure tensor and an analytical solution to the eigendecomposition. This makes the method both fast, robust to noise, and resolution invariant.

Then, I presented two papers where we apply structure tensor analysis to three different types of unidirectional glass and carbon fiber-reinforced composites used in wind turbine blades. We show that our method can be used to estimate fiber orientation, which can be used to characterize and segment the fibers. The estimated distributions also allow quantitative analysis and comparison of materials. Furthermore, we show that subtle local variations in the fiber orientations can be detected and quantified, allowing for parameter optimization of production parameters to improve material quality.

Finally, we have discussed validation of the accuracy of the estimated orientations, utilization of the eigenvalues, and the importance of keeping the antipodal symmetry in mind when dealing with orientations.

CHAPTER 5

Conclusion

As of March 2021, 191 out of the 197 Parties to the COP21 convention in Paris have ratified the Paris Agreement[1], committing them to combat climate change and adapt to its effects. While countries are scrambling to take the actions required to reduce greenhouse gas emissions, it is clear that renewable sources of electrical energy play a key role in these efforts.

Wind energy is the second largest source of renewable energy, after hydropower.[2] Thanks to technological development fueled by investments from consumers, industry, and governments, wind energy has not just seen a large expansion in capacity, but has also become economically competitive compared to traditional fossil-fueled energy sources. As the demand for renewable energy remains high and the price of wind energy is decreasing, it is hard to imagine that the demand for wind energy will slow down anytime soon.

The cost of electricity from wind turbines depends largely on the nameplate capacity, especially for off-shore sites. The nameplate capacity is limited by the swept area of the wind turbine, which increases significantly as the rotor diameter increases. However, increasing the rotor diameter is not easy and is therefore both the limiting and driving factor behind the increasing size of wind turbines. Due to the advantages of larger capacities, I expect wind turbines to keep increasing in size as long as our engineering skills allow. With the increasing size, structures and materials are pushed closer to their limits, and QC becomes increasingly important to reduce the risk of costly structural failures.

5.1 Better quality control

Artificial intelligence has the potential to provide more consistent, much faster, and even more accurate QC than human experts. However, for highly complex tasks, such as inspection of wind turbine blades using ultrasound, creating reliable automated solutions based on AI is challenging. Even with state-of-the-art deep learning techniques, such tasks generally require large high-quality labeled datasets and years of development. Furthermore, creating labeled datasets, particularly for 3D image data, can be very costly. While such an investment might be worthwhile for generic

[1] https://unfccc.int/process/the-paris-agreement/status-of-ratification, March 13, 2021
[2] See Appendix A.

problems like natural image and language processing, this is often not the case for less generic problems, such as blade inspection. Therefore, we need solutions that do not require large investments into the creation of labeled datasets for training machine learning algorithms.

The work that has been presented in this thesis takes a non-learning approach to the extraction of important material and structural properties from 3D image data. This allows us to rely on prior knowledge, which we can model, rather than large labeled datasets, which are impractical to obtain. The non-learning approach is suitable for many QC scenarios, where we have a clear expectation of what the data should look like and a high degree of control over the data capturing process.

However, learning and non-learning methods are not mutually exclusive. In fact, a feasible way of obtaining large amounts of labeled data for training machine learning models could be to use non-learning methods, such as the ones presented in this thesis. Refining such datasets generally requires human intervention to correct bad labels. The interactive approach discussed in Section 3.8.3.1, would allow users to easily fix bad labels, while still benefiting from the assistance of the AI model. This could also be combined with active learning techniques, which allow users to refine the trained models interactively.

I think the key to better QC on blades using AI is to provide users with a good initial AI model, which can extract relevant information from the data with little to no user interaction. However, the user must be able to easily interact with the model to correct any errors made by the AI. This is important not only to allow automated reporting of the evaluation results, but also to ensure that the results are correct, and thus suitable for training machine learning methods, which can further improve the AI.

5.2 Contributions

The goal of the work, which has been presented in this thesis, is to reduce the cost and improve the consistency of QC of wind turbine blade structures and materials.

5.2.1 Faster graph cut-based segmentation

This thesis includes three papers related to graph cut-based image segmentation, which is suitable for analysis of blade structures based on ultrasound images.

- In Paper A, we proposed the sparse layered graph structure, which can be used to reduce the size of graphs in multi-label segmentation tasks by orders of magnitude compared to previous methods. This enables the use of graph cut-based for multi-label segmentation of large 3D volumes, which was not previously feasible.

- In Paper B, we benchmarked state-of-the-art serial and parallel maxflow/mincut algorithms on a set of computer vision tasks, including our own implementations and optimized versions of known algorithms. Our results provide useful insights into which algorithms perform the best for image segmentation tasks. Furthermore, the results highlight the potential of parallel maxflow/mincut algorithms for large segmentation tasks.
- In Paper C, we proposed our own parallel version of the QPBO algorithm. This is, to my knowledge, the first parallel implementation of the QPBO algorithm. The results presented in the paper show that our parallel algorithm can speed up the segmentation of large 3D images by an order of magnitude compared to the serial QPBO algorithm.

When combined, these contributions significantly increase the size of image segmentation tasks, for which graph cut methods are useful. Furthermore, as these tasks can be solved faster, it opens the door for real-time user interaction, which is important for the correction and refinement of the segmentation results. Some of the methods presented here are already being used in AI models for blade evaluation by FORCE Technology. All code, notebooks, and data used for the experiments presented in the three papers are (or will be) shared online as open-source, allowing others to benefit from our fast implementations.

5.2.2 Structure tensor analysis of fiber orientations

The two last papers included in this thesis are concerned with estimating and quantifying fiber orientations in fiber-reinforced composites used in blades.

- In Paper D, we presented a structure tensor-based pipeline for fast estimation of fiber orientations in fiber-reinforced composites. The usefulness of our approach was demonstrated on a unidirectional stitched fiberglass composite commonly used in blades. The results show that we can estimate the different orientation distributions in the glass fiber fabric and segment the different bundles based on the estimated orientations.
- In Paper E, we used our fast structure tensor pipeline to examine local variations in fiber orientations in three unidirectional fiber-reinforced composites commonly used in blades. We showed that our method can be used to quantify local changes in orientation, which depend on manufacturing process parameters. Besides quantifying misalignment effects, our method is fast and scalable, making it suited for process parameter optimization, as many large volumes can be analyzed in a short time.

Structure tensor analysis appears to provide a robust, and certainly fast, way of estimating fiber orientations in composites. The two contributions included in this thesis

describe our pipeline, which we have also shared online, allowing both manufacturers and researchers to estimate and characterize fiber orientations. This provides a foundation for automated quantitative QC of fiber-reinforced materials.

5.3 Summary

The work presented in this thesis provides new methodological and algorithmic tools, which can be used for better and cheaper QC of wind turbine blades and fiber-reinforced composites.

The presented graph cut-based method provides a solution to the challenge of detecting surfaces in ultrasound images of blade structures. Surface detection is essential for the extraction of structural features in the AI-based solution for automated blade inspection developed by FORCE Technology. In this solution, the graph cut-based method plays a key role when measuring structural properties such as glue thickness, which can be measured automatically for an entire blade in less than a minute with a resolution 200 times higher than that usually recorded through manual evaluation. As such, the graph cut-based method achieves the goal of providing better and faster QC of blades.

Similarly, the contributions related to fiber characterization provide an effective tool in the form of a pipeline for automatic extraction of fiber properties that can be used directly in QC. The pipeline can be used by manufacturers of composite materials to determine the quality of their products in terms of fiber alignment in only a few minutes, once they have acquired the necessary X-ray CT scans of the materials. This allows both faster and more comprehensive QC of the composites than previously possible.

These new tools for faster and better QC are of high importance, not just to the manufacturers of wind turbines, who want to cut the cost and improve the quality of their products, but also to the international community, in the process of transitioning to renewable sources of energy.

This does, however, not mean that the contributions presented in the five papers included in this thesis are only useful for image analysis of blades and blade materials. Both the methodological contributions and the open-source software implementations are generic and represent important steps forward within the respective fields of graph cut-based computer vision, maxflow/mincut algorithms and structure tensor-based fiber characterization.

APPENDIX A

Electricity generation and greenhouse gas emissions

Year	1990	1995	2000	2005	2010	2015	2018
Coal	4,429,911	4,993,261	5,994,185	7,316,600	8,662,447	9,534,199	10,159,646
Oil	1,322,975	1,228,863	1,183,808	1,129,445	970,042	1,027,686	783,703
Natural gas	1,748,624	2,020,958	2,774,747	3,706,208	4,841,878	5,525,879	6,150,200
Biofuels	105,435	95,068	113,780	169,500	277,740	415,631	518,467
Waste	24,142	34,770	49,544	58,142	89,291	101,843	118,773
Nuclear	2,012,902	2,331,951	2,590,624	2,767,952	2,756,288	2,570,070	2,710,430
Hydro	2,191,674	2,545,918	2,695,591	3,019,509	3,535,266	3,982,151	4,325,111
Geothermal	36,426	39,895	52,171	58,284	68,094	80,562	88,956
Solar PV	91	197	800	3,732	32,038	250,076	554,382
Solar thermal	663	824	526	597	1,645	9,605	11,321
Wind	3,880	7,959	31,348	104,465	342,202	833,732	1,273,409
Tide	536	547	546	516	513	1,006	1,005
Other sources	19,939	23,864	22,049	32,983	33,704	35,741	34,662

All numbers are in GWh.
Based on data from IEA (2020) IEA Electricity Information, www.iea.org/statistics. All right reserved.

Table A.1 – Worldwide electricity generation by source.

Year	1990	1995	2000	2005	2010	2015	2018
Geothermal	36,426	39,895	52,171	58,284	68,094	80,562	88,956
Solar thermal	663	824	526	597	1,645	9,605	11,321
Hydro	2,191,674	2,545,918	2,695,591	3,019,509	3,535,266	3,982,151	4,325,111
Solar PV	91	197	800	3,732	32,038	250,076	554,382
Tide, wave, ocean	536	547	546	516	513	1,006	1,005
Wind	3,880	7,959	31,348	104,465	342,202	833,732	1,273,409

All numbers are in GWh.
Based on data from IEA (2020) IEA Renewables Information, www.iea.org/statistics. All right reserved.

Table A.2 – Worldwide renewable electricity generation by source (non-combustible).

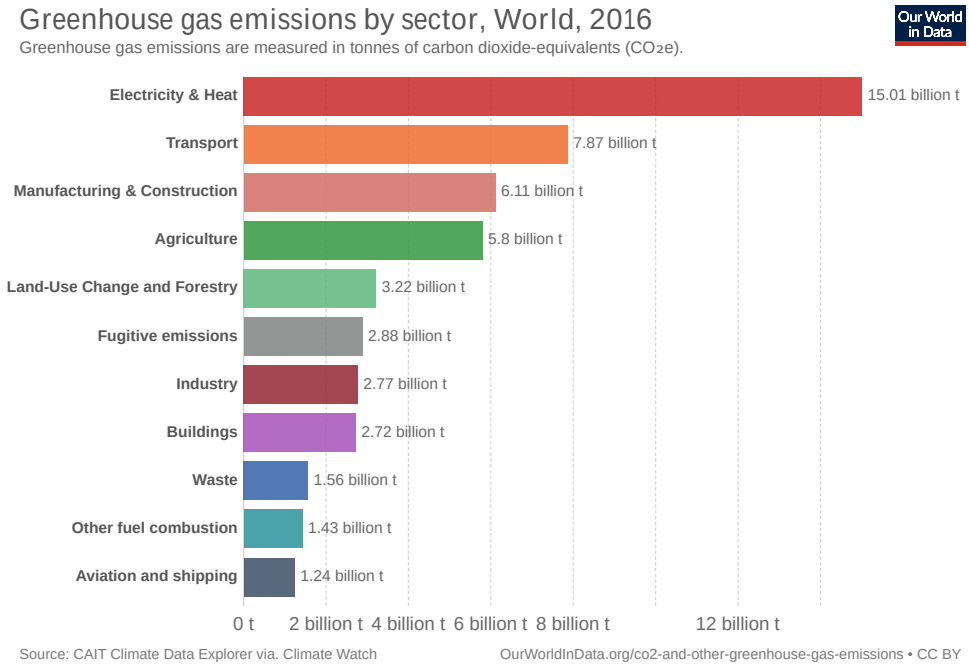


Figure A.1 – Greenhouse gas emission by sector.

APPENDIX B

Paper: Individual Fibre Inclination Segmentation from X-ray Computed Tomography using Principal Component Analysis

---

This paper has been submitted to the journal Composites Part A: Applied Science and Manufacturing.

Individual Fibre Inclination Segmentation from X-ray Computed Tomography using Principal Component Analysis

Filip Bo Salling[a1], **Niels Jeppesen**[b], **Mads Rostgaard Sonne**[a], **Jesper H. Hattel**[a] and **Lars Pilgaard Mikkelsen**[c2]

[a] Department of Mechanical Engineering, Technical University of Denmark (DTU), 2800 Kgs. Lyngby, Denmark.

[b] Department of Applied Mathematics and Computer Science, DTU, Denmark.

[c] Department of Wind Energy, DTU, Denmark.

Corresponding authors: [1]fsras@mek.dtu.dk, [2]lapm@dtu.dk

Abstract. This study presents a holistic segmentation procedure, which can be used to obtain individual fibre inclination angles from X-ray computed tomography. The segmentation approach is based on principal component analysis and was successfully applied for a unidirectional and an air textured glass fibre reinforced composite profile. The inclination results show a weighted mean fibre inclination of 2.1° and 8.0° for the unidirectional and air textured profile, respectively. For the air textured composite, fibre inclinations of up to 55° were successfully segmented. The results were verified by comparative analysis with equivalent results obtained from structure tensor analysis – showing no notable deviation. The comparable characteristics in combination with the distinct differences of the two material systems make this case study ideal for verification and validation of idealized models. It is shown how this approach can provide fast, accurate and repeatable inclination estimates with a high degree of automation.

Keywords: Glass fibre composites, Fibre undulation, Segmentation verification, Singular value decomposition, Structure tensor.

1. Introduction

The demand for lightweight materials in the form of fibre reinforced polymer (FRP) composites has consistently increased and the growth is expected to continue [1]. FRP composite materials are heavily used for structural applications where high specific mechanical properties are required, e.g. high stiffness to weight ratio. In that regard, the FRPs can easily compete with most metal alloys and are therefore heavily used in e.g. offshore-, construction-, and wind turbine industry [2-3]. This is one of the reasons why the growth rate of composites usage has exceeded that of steel and aluminum in the last decades [4].

The macroscopic properties of FRPs are inherently related to their microstructure, which is mainly described by the fibre volume fraction and the fibre architecture. This enables tailoring of mechanical properties for a given application. Hence, numerous studies are found in literature investigating the relation between composite microstructure and macroscopic properties, e.g. stiffness, compressive strength and fatigue properties.

In the study by Paluch [5], the effect of fibre undulation in unidirectional (UD) composite was investigated by destructive characterization, i.e. regularly cut sections examined by optical microscopy. It was concluded that considering a composite material as a series of mutually parallel fibres undulating in phase is a poor assumption. Non-destructive confocal laser scanning microscopy was used in [6] to investigate fibre undulation. However, this method is limited to sufficient transparent samples, i.e. composites where the matrix fluoresces strongly and with fibre volume fraction below 30 percent.

X-ray Computed Tomography (XCT) is widely used to characterize material properties of fibre reinforced polymers, as well as for materials like concrete and metals [7-9]. XCT has the advantage of non-destructive testing and a high resolution, i.e. micron level voxel size [10-11]. However, the resolution obtained by XCT cannot compete with that obtained by scanning electron microscopy (SEM), reaching sub-micron resolutions [12]. XCT was used to investigate the effect of varying off-axis angles in non-crimp fabrics [13-14].

In a recent study [15], the effect of fibre misalignment on the longitudinal compressive and tensile failure was investigated by micromechanical modelling, introducing the fibre misalignment using a

stochastic process. Similar mechanical modelling was conducted on characterized microstructures by Auenhammer et al. [16]. The main objective of this study was to develop a highly automated process to transfer XCT data into robust finite element models. The model was evaluated by its ability to predict Young's moduli of UD fibre composites loaded in tension and it was concluded that the methodology did not suffice to predict the correct trend. It was concluded that the reason for the deviations most likely was attributed to the single bundle fibre tracking segmentation process and in particular the mean fibre orientation, fibre volume fraction and bundle to volume ratio. The influence of the segmentation method was investigated in [17] and the waviness induced by backing bundles was captured by generating a boundary between the backing and UD bundles. The approach in [16-17] focusses on the bundle level rather than the fibre level, hence, the influence of the single fibre scale was not investigated. It is argued that the advantage of the methodology is automation, but it is also mentioned that the user needs experience in Avizo to set up the right parameters depending on the composite architecture and the fibre to matrix attenuation contrast.

The structure tensor (ST) segmentation method was used by Advani and Tucker to predict fibre orientations in short fibre composites [18]. The ST method was recently used to characterize a non-crimp glass fibre fabric and the Jupyter notebooks and Python code behind the segmentation process were publicly shared [19]. This ST segmentation approach uses thresholding to separate fibre and matrix material, which necessitates a proper attenuation contrast.

Emerson et al. [20] developed an interactive fibre tracking algorithm, which was used to characterize a UD composite. The algorithm was developed in Matlab, publicly shared, and statistical validated in [10]. Its applicability for a textured fibre composite was investigated in [21]. It was found that the algorithm made some erroneous fibre detections for high angle fibres but it was in general applicable for textured fibre composites.

In a study by Amrehn et al. [22], it was concluded that interactive segmentation methods are beneficial for complex structures as opposed to fully automated segmentation methods that inherently lack domain knowledge.

1.1. Objective

FRP composites are often designed with large safety margins because of incomplete knowledge about the connection between the microstructure and the macroscopic properties.

The main objective of this study is to use XCT to quantify the fibre undulation of two mutually distinct composite systems. The first one is a pultruded UD glass fibre profile and the second one is a pultruded air textured glass fibre profile. The air textured fibre reinforced composite is expected to have a complex structure, which calls for an interactive segmentation method. The segmentation will build upon the individual fibre tracking algorithm developed in [20]. The inclination of the fibres will be analysed using a novel segmentation method developed by the authors of the present work. The new segmentation approach will include a semi-automatic segmentation module based on principal component analysis (PCA). The quantified inclination results will be verified by comparison with corresponding results obtained by the ST method, as described in [19]. Finally, the prospects of novel insight into the connection between microstructure and macroscopic properties will be discussed.

2. Materials

This study is carried out on two mutually distinct FRP composite systems, namely a UD glass fibre reinforced thermoset composite (UD-fib) and an air textured glass fibre reinforced thermoset composite (air-tex). The two composites are both manufactured by the resin injection pultrusion process using solely fibre bundle rovings. Hence, the fibre architecture is inherently continuous through the profile and any observable fibre inclinations are thus not a result of misaligned layup, which sometimes is the case in pre-preg and non-crimp fabric composites [23].

The UD-fib and the air-tex composites are manufactured as a rod with a diameter of ~13mm and a large beam with a square cross-section of ~100×100mm^2, respectively. But, for the purpose of the current study, both pultruded samples were machined to obtain a rod with a diameter of ~5mm, see Figure 1. The continuous UD- and air textured glass fibre rovings used for the two composites are of the same glass fibre type with an estimated average diameter of ~23.4µm, i.e. the air textured rovings are a textured version of the UD rovings. This makes the two composite systems suitable for comparative analysis. The two composite rods together with a schematic illustration of their respective fibre reinforcement are depicted in Figure 1.

FIGURE 1

It should be noticed how Figure 1 is set up with a) UD-fib and b) air-tex. This figure setup is continued throughout the paper to ease comparative analysis and is additionally directly comparable with the corresponding a) and b) figures in the data article published by the authors [12].

3. X-ray Computed Tomography

To recap, the scope of the current study is to characterize the fibre orientation of the two mutually distinct FRP composite systems, i.e. the UD-fib and air-tex material described in the previous section. This is done using XCT followed by an advanced novel segmentation approach. In the following section the experimental XCT method is described.

3.1. Experimental Method (XCT)

The XCT experiments were carried out on a *Zeiss Xradia Versa 520* scanner with a voltage of 40kV. The scan was performed with 4× optical magnification obtaining 4501 projections at binning 2. A tomographic volume of 988×1013×999 voxels with a voxel size of 1.99µm was obtained. The tomographic data is published in the data article [12], including a detailed description of the X-ray tomography settings. The same procedure was used for the two composite systems and the resulting 3D tomograms are depicted together with an example of a tomographic cross-sectional slice in Figure 2.

FIGURE 2

The tomographic reconstruction shown in Figure 2 has a high resolution, i.e. a voxel size of 1.99µm. It also has a high attenuation contrast, which eases the separation of the constituent fibre and matrix materials by the naked eye. However, obtaining a segmented mathematical description of the individual fibres is not a trivial task, as the fibres are often bundled closely together, making separation difficult, even with high attenuation contrast data. A novel segmentation process is described in the following chapter.

4. Individual Fibre Segmentation

In the current study the segmentation process is divided into three steps: i) a slice by slice fibre center detection, ii) a fibre tracking procedure relating centers of the individual image slices to each other to form continuous fibres, and iii) a fibre inclination segmentation step, i.e. assigning an inclination angle to each individual fibre trajectory. The first two segmentation steps were conducted using the Matlab algorithm

developed by Emerson et al. in [20]. The third and final segmentation step was performed using a novel approach developed in this study. The three segmentation steps are the subject of the following 3 sections. The segmentation process was continuously evaluated for both material systems to ensure a generic segmentation process applicable for both material systems. This chapter is concluded with a summary including a table with an overview of the user input and running time for each of the three segmentation process steps (see Table 1).

4.1. Fibre center detection

The first step of the segmentation process, i.e. the slice-by-slice fibre center detection, was conducted using the algorithm developed my Emerson et al [20]. This segmentation algorithm uses a training step to set up a dictionary of corresponding image and label patches. The dictionary was set up using the following steps:

i) Per default the middle slice of the tomographic stack of slices was chosen for the training step, i.e. slice no. 500 out of the 999 slices

ii) The region of interest (RoI) was chosen to include the entire tomographic volume.

iii) An image region for the training step was chosen somewhat smaller than the RoI but still covering a major part of a tomographic slice.

iv) An optimal patch size of 11 pixels, corresponding to approximately 1 fibre diameter, was chosen based on the voxel size of 1.99µm.

v) Finally, the graphical user interface was used to setup the dictionary.

The accuracy of the fibre center detection algorithm was evaluated qualitatively by manual inspection of individual tomographic slices and the corresponding fibre centers detected. Figure 3 illustrates an example of a tomographic slice and the corresponding detected fibre centers. It should be noticed how the dictionary successfully detects the fibre centers with very few exceptions. The few erroneous double detections and missing fibre detections are considered insignificant for the overall evaluation of the microstructure and hence the scope of this study (see Figure 3).

FIGURE 3

The dictionary based fibre center detection algorithm was applied for the stack of 999 images resulting in a processing time of approximately one hour for each of the two tomographic volumes. The total 3D volume of detected fibre centers are depicted by four evenly spaced stacks of five image slices, see Figure 4.

FIGURE 4

The fibre centers shown in Figure 4 clearly illustrates a pattern of continuous fibres for both material systems. In addition, it should be noticed how the air-tex sample indicates a trend of fibres with notable inclination angles.

At the top and bottom of the tomographic stack of image slices significantly increased noise was observed. The noise arises from the well-known cone beam effect and results in missing fibre detections. However, missing fibre detections in one image has no influence on the fibre detections in the other images. It does however influence the second step of the segmentation process, i.e. the individual fibre tracking. This problem is accounted for by excluding the top and bottom 50 image slices. This will be discussed in the following section about the fibre tracking segmentation step.

4.2. Fibre tracking

When the fibre center detection has been conducted for the full stack of tomographic image slices (cf. Figures 3 and 4), the next step is to track the 2D slices of fibre centers through the depth of the volume, i.e. along the x-axis. Hence, connecting fibre centers belonging to the same fibre and thereby obtaining the individual fibre trajectories. This segmentation step is also performed using the algorithm by Emerson et al. [20]. The only user input for the fibre tracking algorithm is how many pixels a fibre center is allowed to move from one slice to the next, i.e. a 2D planar distance. Thus, this is naturally a function of the voxel size and the fibre diameter. From simple geometrical considerations an allowable fibre center movement from one slice to the next of 6 pixels will allow for a maximum inclination angle of approximately 80°. Hence, an allowable movement of 6 pixels was chosen as a conservative input (cf. Figure 2).

The fibre tracking algorithm is set up so the number of fibre centers detected in the very first image slice dictates how many fibres in total the tracking algorithm will allow. Thus, the less accurate fibre center detection at the very top or bottom image slices, results in too few fibre trajectories. Therefore, it was chosen to ignore the top and bottom 50 image slices. This solved the above-mentioned problem at the expense of a decreased tomographic volume.

The individual fibre tracking algorithm was applied for both material systems, each with a processing time of less than five minutes. The 3D volume of all fibre trajectories obtained from the fibre tracking algorithm are shown in Figure 5.

FIGURE 5

Figure 5 illustrates how the fibre tracking segmentation algorithm was successfully used to track the fibre centers for both material systems. A 3D volume representing 10 percent of the fibre trajectories is depicted in Figure 6.

FIGURE 6

Figure 6 shows that the tracking algorithm is able to track both low and high inclination fibres. However, taking a closer look at Figure 6(b) one can notice a few fibre trajectories with piecewise unlikely geometries, i.e. UD fibre kink segments along the edge of the tomographic volume. This is most likely due to the fact that the fibre tracking algorithm does not allow termination or initiation of fibre trajectories running in or out of the tomographic volume. Hence, two individual fibres might get connected by a UD fibre segment to form a single fibre connected by this "fictive" UD fibre kink. However, this error is only rarely observed and would therefore not have any significant influence on the scope of this study, i.e. characterizing the fibre undulation. Using the fibre trajectories it is now possible to evaluate and quantify the inclination of the individual fibres.

4.3. Fibre inclination estimation using PCA

The third and final step of the segmentation process is inclination estimation of the individual fibre trajectories. The segmentation algorithm, used for fibre detection and fibre tracking, does have a third module for orientation estimation which was successfully used for UD fibre composites [10,20]. However, the module assumes a straight fibre trajectory between the fibre centers at the top and bottom image slice and does not take any undulation into account. From Figure 6 it is evident that the air textured fibres in the air-tex sample exhibit significant undulation, which should be resolved. To do so, a novel inclination segmentation algorithm was developed. The idea behind the current approach is to evaluate the inclination of each individual fibre as a weighted mean of fibre sub-segments, e.g. each individual fibre is divided into a number of sub-segments by dividing the total stack of image slices into smaller stacks. For instance, the stack of 900 image slices is divided into two stacks of 450 image slices, hence, each fibre is represented by two fibre sub-segments (see Figure 7).

FIGURE 7

The straight lines representing the fibre sub-segments (cf. Figure 7) are obtained from linear regression analysis using Principal Component Analysis (PCA) [24].

The PCA analysis is a statistical interpretation of the singular value decomposition (SVD) of a data matrix $\boldsymbol{A}$. Once the matrix $\boldsymbol{A}$ is set up, the SVD is done in Matlab. The results of the SVD are the matrices $\boldsymbol{U}$, $\boldsymbol{\Sigma}$ and $\boldsymbol{V^T}$. The covariance matrix $\boldsymbol{U}$ contains the left singular vectors, $\boldsymbol{\Sigma}$ is a diagonal matrix with the singular values ordered by importance and $\boldsymbol{V^T}$ is a transposed matrix containing the right singular vectors. Hence, the SVD yields

$$\boldsymbol{A} = \boldsymbol{U\Sigma V^T} \tag{1}$$

The steps of the regression analysis are: i) Setup a matrix $\boldsymbol{X}$ with all fibre center coordinates from the fibre segment of interest. ii) Compute the mean fibre center coordinate $\overline{\boldsymbol{X}}$, i.e. the centre point of the regression line of interest. iii) Compute the matrix $\boldsymbol{A} = \boldsymbol{X} - \overline{\boldsymbol{X}}$. iv) Compute the covariance matrix $\boldsymbol{U}$ by singular value decomposition of $\boldsymbol{A}$. The first left singular vector $\boldsymbol{u_1}$ of the covariance matrix $\boldsymbol{U}$, corresponding to the highest singular value σ_1, is the first principal component and gives the direction of the best fit line. Hence, the regression line, representing the fibre sub-segment of interest, goes through the mean center point $\overline{\boldsymbol{X}}$ and has the direction $\boldsymbol{u_1}$. v) Compute the coordinates of the best fit line using the parametric vector equation of a line in three dimensions given by

$$\boldsymbol{X_{fit}} = \overline{\boldsymbol{X}} + t \cdot \boldsymbol{u_1} \tag{2}$$

where, $\boldsymbol{X_{fit}}$ is the resulting coordinates of the best fit line and t is a scaling factor

Finally, the increments Δx, Δy and Δz are readily obtained from the endpoint coordinates of $\boldsymbol{X_{fit}}$. The inclination of the linear fibre sub-segments is then computed using simple trigonometry. The inclination angle, i.e. the angle with respect to the x-axis, is calculated as

$$\theta_x = \arctan\left(\frac{\sqrt{\Delta y^2 + \Delta z^2}}{\Delta x}\right) \tag{3}$$

The individual fibre inclination is weighted by their corresponding fibre length. Thus, the fibre length of each fibre sub-segment is also calculated. The length of each fibre segment is given as

$$L_f = \sqrt{\Delta x^2 + \Delta y^2 + \Delta z^2} \tag{4}$$

The alignment of the samples prior to the X-ray scan can be evaluated by projected inclinations and these are therefore also computed. The projected inclination angles and corresponding projected ‘fibre

lengths' are calculated using the simplified 2D version of equations (3) and (4). The equations for the xy-projections are

$$\theta_{xy} = \arctan\left(\frac{\Delta y}{\Delta x}\right) \tag{5}$$

$$L_f^{xy} = \sqrt{\Delta x^2 + \Delta y^2} \tag{6}$$

and the equivalent equations for the xz-projection are (5) and (6) with z substituted for y.

4.3.1. Segmentation discretization. An example of a fibre trajectory and the corresponding representation by five linear sub-segments are illustrated in Figure 8.

FIGURE 8

Figure 8 shows that the linear segments, obtained from the regression analysis, represents the undulation of the individual fibre well, i.e. each straight sub-segment follows the local fibre trajectories, while imposing a piecewise smoothing effect to the trajectory. This is beneficial because it minimizes the effect of the noise, which arises from the uncertainty of the fibre center detections. Hence, there is a trade-off between resolving the fibre undulation while maintaining the smoothening effect, i.e. the uncertainty related to the fluctuating fibre center detections should not be resolved (cf. Figure 8(a)). Therefore, a proper discretization of the fibre trajectories must be chosen to obtain the best results. The fibre trajectories are now discretized into fibre sub-segments with a fixed height Δx corresponding to an integer number of image slices. For instance, each of the five fibre sub-segments in Figure 8 contains fibre centers of (up to) 180 image slices corresponding to a fibre sub-segment height of approximately $\Delta x = 358\mu\text{m}$.

For the sake of generalization, the segmentation mesh will be described using the normalized mesh size $\Delta x/D_f$, using an average fibre diameter of $D_f = 23.4\mu\text{m}$, confer chapter 2. Which, in the case with five fibre sub-segments illustrated in Figure 8 corresponds to a normalized segmentation mesh size of $\Delta x/D_f \approx 15$. The method was tested by varying the normalized mesh size from 77 (i.e. including all 900 tomographic image slices to represent the fibre trajectory as a single straight fibre segment, cf. Figure 7) to 1 (i.e. representing the fibre trajectories by 81 straight fibre sub-segments each obtained by 12 tomographic image slices). Figure 9 illustrates a fibre trajectory and the corresponding representation by straight fibre sub-segments obtained using the fine segmentation mesh where $\Delta x/D_f \approx 1$.

FIGURE 9 **FIGURE 10**

Figure 9 shows that a fine segmentation mesh captures some of the noise related to the uncertainty of the fibre center detections. This indicates that a mesh size of $\Delta x/D_f \approx 1$ does not provide sufficient smoothing. The goal is to obtain a good resolution of the fibre undulation while maintaining the smoothing effect from the regression analysis. A dimensionless segmentation mesh size of $\Delta x/D_f \approx 10$, which corresponds to including 113 image slices for each fibre sub-segment, provides a good trade-off for the current data. The choice of $\Delta x/D_f \approx 10$ is based on the known voxel size of 1.99μm, an expected fibre diameter of ~23.4μm as well as considerations regarding the physical limitation of the frequency of the fibre undulation. Figure 10 shows a well-defined fibre trajectory from the air-tex sample using the dimensionless segmentation mesh size of $\Delta x/D_f \approx 10$. Each linear fibre sub-segment was only accepted for PCA if it contains at least 95% of the possible fibre centers. This conservative criterion was chosen to minimize any artificial inclination effects from erroneous fibre detection and fibre tracking.

The PCA based individual fibre inclination segmentation algorithm was applied for both materials systems using a normalized segmentation mesh size of $\Delta x/D_f \approx 10$, resulting in a processing time for each of the material systems of less than 15 seconds.

4.4. Individual fibre segmentation summary

A schematic overview of the three-step segmentation method used in the present study is summarized in Table 1.

TABLE 1

The three segmentation steps consist of: i) Fibre center detection, ii) Fibre tracking and iii) Inclination estimation. The first two steps are performed using the fibre tracking algorithm [20], while a novel segmentation algorithm, using PCA, was developed for inclination estimation in this study. The user inputs for the three step segmentation modules are summarized in Table 1, together with the running time for each of the three segmentation steps (using a standard laptop). The total processing time of the complete segmentation procedure is only a little more than one hour. This is, to the knowledge of the authors, faster than similar commercial fibre tracking algorithms, which usually require several hours to perform a similar analysis.

5. Results

The PCA algorithm was used to obtain the fibre inclination distribution for both material systems. The results are presented in section 5.2, but first the choice of segmentation mesh is evaluated in the following section.

5.1. Segmentation mesh

The choice of $\Delta x/D_f = 10$ as an 'optimal' dimensionless segmentation mesh size (cf. chapter 4) was determined by varying the dimensionless segmentation mesh size from 77 (representing each fibre using a single straight line) to a value of 1 (representing each fibre by 81 sub-segment straight lines). The influence of the variation in segmentation mesh on the overall inclination estimation was evaluated by comparing the weighted mean inclination for the total volume, calculated as

$$\bar{\theta}_x = \sum_{i=1}^{N_s} \left(\theta_i \cdot \frac{L_{f,i}}{L_f^{tot}} \right) \tag{7}$$

where the summation index i is used to sum over the total number of fibre sub-segments N_s and L_f^{tot} is the total fibre length of the segmented 3D volume, given by the sum of all the fibre sub-segments

$$L_f^{tot} = \sum_{i=1}^{N_s} L_{f,i} \tag{8}$$

The weighted mean inclination as a function of the dimensionless segmentation mesh size is depicted in Figure 11.

FIGURE 11

Figure 11 shows how the refinement of the segmentation mesh from $\Delta x/D_f = 77$ to $\Delta x/D_f = 10$ results in an increase in the mean fibre inclination of approximately 15% and 20% for the UD-fib and air-tex material systems, respectively. Hence, the segmentation mesh has a significant influence on the inclination estimation for both material systems, i.e. the fibre undulation is resolved.

It should further be noticed how the mean inclination increases significantly at fine mesh sizes, in particular when $\Delta x/D_f = 1$. This increase highlights the importance of reducing the impact of the noise, by using more slices for each segment (cf. Figure 9). This is particularly evident for the UD-fib sample, where the deviations in the center detections cause a significant relative increase in the mean fibre inclination when $\Delta x/D_f$ is low. This is expected because the noise becomes significant due to the low inclination angles inherently expected from a UD fibre reinforced composite.

Hence, a value of $\Delta x/D_f = 10$ was found to provide a good tradeoff between resolving fibre undulation while maintaining the smoothing effect of the analysis, and is therefore used in the analysis henceforth.

5.2. Inclination distribution

The weighted inclination distribution obtained using a segmentation mesh size of $\Delta x/D_f = 10$ is illustrated in Figure 12.

FIGURE 12

The inclination distributions depicted in Figure 12 consist of the actual inclination distribution (top) and the projected inclination distributions Figure 12 (middle and bottom). The inclination distributions in Figure 12 (top) are right-angled distributions, which inherently arises from the nature of the fibre reinforcement, i.e. continuous fibre bundles. The main characteristics of the inclination distributions illustrated in Figure 12 are summarized in Table 2, i.e. the weighted mean and the quartiles of the inclination distribution as well as the weighted mean of the projected inclination distributions.

TABLE 2

The results illustrated and summarized in Figure 12 and Table 2 show expected magnitudes of inclination angles compared with the corresponding 3D tomograms depicted in Figure 2. From Table 2 it should be noticed how $\bar{\theta}_x \gg \bar{\theta}_{xy}$ and $\bar{\theta}_x \gg \bar{\theta}_{xz}$, which indicate little tilting of the samples in the XCT setup. Hence, the inclination estimation is qualitatively acceptable. The fibre trajectories are plotted with colors corresponding to their individual mean fibre inclination using the quartiles as interval boundaries (see Figure 13).

FIGURE 13

The fibre trajectories coloured according to their inclination angle (see Figure 13) show that the PCA-based segmentation algorithm estimates the inclination of the individual fibres well. A cross-sectional view of the coloured fibre trajectories is illustrated in Figure 14.

FIGURE 14

The cross-sectional view in Figure 14 shows how fibres with similar inclination angles tend to group in bundle like structures. A study of these trends could be of interest for future studies.

The estimated inclinations can be used to evaluate mechanical properties, e.g. estimation of longitudinal stiffness using the fibre efficiency formula proposed by Krenchel [25]. The microstructural effect on mechanical properties is the subject of ongoing studies by the authors. The preliminary results indicate that

the inclination estimations obtained in this study can be used to explain some of the deviation in longitudinal stiffness observed comparing tensile test results with classical rule of mixture estimates. Hence, this would be of great interest in future studies.

For verification, the results of the PCA segmentation method were compared to equivalent fibre inclination estimates obtained using the structure tensor method - this is the subject of chapter 6.

6. Verification

Microstructure investigations using XCT and subsequent numerical segmentation will inherently contain errors since the segmented tomogram is a discretized mathematical representation of the microstructure features. Hence, verification is necessary to evaluate the error of this mathematical segmentation method. The developed mathematical method using PCA is verified by comparison with equivalent results obtained for the same XCT scan using the structure tensor (ST) method.

6.1. The structure tensor method

The ST method, when applied on volumetric data, can be used to extract local structural information from the data, such as local orientations of fibre-like structures. The structure tensor itself is a 3-by-3 matrix, summarizing local gradient information around a point in space. One way to calculate the structure tensor is using a Gaussian derivative for computing the gradient and a Gaussian window for integration [19]. The size of the Gaussian derivative kernel is determined by the parameter σ, while the size of the Gaussian integration kernel is determined by the parameter ρ. The parameters σ and ρ are also known as the *noise scale* and *integration scale*, respectively. Formally, the structure tensor $\mathbf{S}$ can be formulated as

$$\mathbf{S} = K_\rho \cdot (\nabla V_\sigma (\nabla V_\sigma)^T) \tag{9}$$

where ∇V_σ is the gradient computed using the Gaussian derivative kernel that depends on σ and K_ρ is the Gaussian integration kernel, which depends on ρ.

Once $\mathbf{S}$ for a point has been computed, the predominant orientation can be determined through eigendecomposition of $\mathbf{S}$. The result is three positive eigenvalues $\lambda_1 \leq \lambda_2 \leq \lambda_3$ and their corresponding mutually orthogonal eigenvectors $\mathbf{v}_1$, $\mathbf{v}_2$ and $\mathbf{v}_3$. The direction of least variation is the one of $\mathbf{v}_1$ corresponding to the smallest eigenvalue λ_1.

The `structure-tensor` Python package [19] is used to compute $\mathbf{S}$ for each voxel in the tomographic volume. The parameters used are $\sigma = 1.045$ and $\rho = 4.18$, which were chosen based on the voxel resolution and fibre diameter. After this, the dominant orientation, $\mathbf{v_1}$, is calculated using the same Python package. The structure tensor $\mathbf{S}$ is computed for each voxel position. However, the goal is to estimate the fibre orientation distribution, not the volume voxel orientation distribution. To exclude non-fibre material, such as matrix and air, an intensity threshold is used to filter out non-fibre voxels. The threshold value is determined using Otsu's threshold, which gives a threshold value of 30.463 for UD-fib and 32.700 for air-tex. Only voxels with values larger than the threshold value are considered to be fibre voxels.

Having filtered out values of $\mathbf{S}$ corresponding to non-fibre voxel positions, the dominant orientation $\mathbf{v_1}$ is computed for all remaining (fibre) voxels. This orientation can be interpreted as the orientation of a small piece of a fibre with the size of a single voxel. The orientations of all the fibre voxels are the fibre orientation distribution. At this point, θ_x, θ_{xz} and θ_{xy} values for each fibre voxel can be calculated. θ_x is calculated as the absolute angle between $\mathbf{v_1}$ and the x-axis. θ_{xz} is calculated by projecting $\mathbf{v_1}$ onto the xz-plane and taking the angle between the projection and the x-axis. θ_{xy} is calculated by projecting $\mathbf{v_1}$ onto the xy-plane and taking the angle between the projection and the x-axis.

The ST method described above has a processing time of less than 10 min. for each of the tomographic volumes analyzed in this study.

6.2. Comparative verification

The results using the PCA approach (cf. Figure 14) are compared with the equivalent inclination distribution estimations obtained using the ST method, see figure 15.

FIGURE 15

The inclination distributions obtained using the ST method show a great resemblance with the results of the PCA method developed in this study (see Figure 15). No significant deviations are observed between the two methods and since the methods are inherently different the resemblance of the results serve as mutual verification of both methods.

In the current case with both high attenuation contrast and high resolution, the fibre tracking approach is considered superior to the thresholding approach used for the PCA and ST methods. However, it is be noted that the ST method has a statistical advantage while the PCA method has the advantage of a straightforward physical interpretation.

7. Conclusion

Continuous fibre reinforced composites are increasingly used for structural applications demanding a high stiffness to weight ratio. However, in-depth knowledge of the relation between the, often complex, fibre architecture and macroscopic properties is still lacking. Many recent studies have used X-ray computed tomography (XCT) to characterize the internal three-dimensional microstructure of composite materials. In this study, a novel semi-automatic segmentation procedure is presented. The segmentation algorithm uses an existing fibre tracking algorithm and a novel algorithm using principal component analysis (PCA) to obtain individual fibre inclinations from XCT data. The present approach using PCA is to the knowledge of the authors not found anywhere in literature. The focus of the current segmentation approach is on the single fibre level rather than on the bundle level and is therefore suitable for complex random fibre architectures.

The PCA segmentation algorithm was used to characterize two mutually distinct composites systems, namely a unidirectional glass fibre reinforced thermoset composite (UD-fib) and an air textured glass fibre reinforced composite (air-tex).

The novel segmentation approach resolves the fibre undulation by discretizing the tomographic volume into finite volumes along the length direction of the continuous fibres, i.e. the x-axis. A normalized dimensionless segmentation mesh size of $\Delta x/D_f = 10$ was found to be suitable for both material systems.

The segmentation algorithm was used to obtain the inclination distribution for both material systems. The results show a weighted mean inclination of 2.1° and 8.0° as well as the corresponding set of quartiles [1.4, 1.9, 2.6, 11.3]° and [3.7, 5.9, 10.0, 55,7]° for the UD-fib and air-tex sample, respectively. These results are well within the expected inclination angles for the two material systems. The inclination results for the air-tex sample clearly show how the PCA segmentation approach is capable of resolving high inclination fibre undulation, which was the major goal of the study.

The segmentation approach using PCA was verified by comparison with equivalent inclination distribution results obtained using the structure tensor (ST) method. The qualitative comparative analysis of the two methods showed no significant deviations, i.e. the methods are mutually verified.

The main advantages of the suggested PCA segmentation approach and the individual fibre inclination results obtained can be summarized as:

a) Semi-automated segmentation approach with physical based user input, allowing accurate and repeatable segmentation results.
b) The full segmentation procedure is presented, i.e. fibre detection, fibre tracking and fibre inclination estimation.
c) The novel inclination estimation approach using PCA is verified by comparison with the structure tensor method.
d) The interactive user input is chosen from physical considerations, i.e. the normalized segmentation mesh size $\Delta x/D_f$.
e) Resolving individual fibre undulation of complex fibre architectures.
f) The quantified inclination results serve as a benchmark dataset for future segmentation research because both the raw XCT data and the corresponding inclination results are available for download in [12].

8. Data availability

The raw XCT data and the corresponding inclination segmentation results produced in this study are available for download in [12].

CRediT authorship contribution statement

Filip Bo Salling: Conceptualization, Methodology, Software, Formal Analysis, Investigation, Visualization, Writing – Original Draft, Writing – Review & Editing. **Niels Jeppesen:** Software (Chp. 6), methodology (Chp. 6), Writing – Original Draft (section 6.1), Writing – Review & Editing. **Mads R. Sonne:** Supervision. **Jesper H. Hattel:** Funding acquisition, Writing - Review & Editing. **Lars P. Mikkelsen:** Conceptualization, Methodology, Supervision, Writing - Review & Editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships which have or could be perceived to have influenced the work reported in this article.

Acknowledgements

This work is part of the *Resin Injection Pultrusion (RIP)* project which has been granted by the Danish Council for Independent Research | Technology and Production Sciences (DFF/FTP), Grant no. DFF-6111-00112.

References

[1] Markets and Markets, Lakshmi Narayanan, July 2020, Composites Market by Fibre Type (Glass Fibre Composites, Carbon Fibre Composites, Natural Fibre Composites), Resin Type (Thermoset Composites, Thermoplastic Composites), manufacturing process, End-use Industri and Region – Global Forecast to 2025 https://www.marketsandmarkets.com/Market-Reports/composite-market-200051282.html, Visited 05-11-2020.

[2] Mikkelsen LP, Emerson MJ, Jespersen KM, Dahl VA, Conradsen K, Dahl AB 2016. X-ray based micromechanical finite element modeling of composites materials. *Proc. Of 29th Nordic Seminar on Computational Mechanics*.

[3] Mikkelsen LP 2020. *The fatigue damage evalution in the load carrying composites laminates of wind turbine blades.* Chp. 16 in: *Fatigue Life Prediction of Composites and Composite Structures (2nd ed.)* , Woodhead Publishing series in Compos. Sci. and Eng., p 569-603.

[4] Agarwal BD, Broutman LJ 1990. *Analysis and performance of fibre composites* 4th ed. (Hoboken/NJ, USA: Wiley).

[5] Paluch B 1996. Analysis of geometric imperfections affecting the fibers in unidirectional composites. *Jour. of Composite Materials.* **30**:454-85.

[6] Clarke AR, Archenhold G, Davidson NC 1995. A novel technique for determining the 3d spatial distribution of glass fibres in polymer composites. *Composites Science and Technology*. **55**:75-91.

[7] Baran I, Straumit I, Shishkina O, Lomov SV 2018. X-ray computed tomography characterization of manufacturing induced defects in a glass/polyester pultruded profile. *Composite Structures*. **195**:74-82.

[8] Klingaa CG, Dahmen T, Baier S, Mohanty S, Hattel JH 2020. X-ray CT and image analysis methodology for local roughness characterization in cooling channels made by metal additive

manufacturing. *Additive Manufacturing*. **32**.

[9] Ren W, Yang Z, Sharma R, Zhang C, Withers PJ 2015. Two-dimensional X-ray CT image based meso-scale fracture modelling of concrete. *Engineering Fracture Mechanics*. **133**:24-39.

[10] Emerson MJ, Dahl VA, Conradsen K, Mikkelsen LP, Dahl AB 2017. Statistical validation of individual fibre segmentation from tomograms and microscopy. *Composites Science and Technology*. **160**:208-15.

[11] Garcea SC, Wang Y, Withers PJ 2018. X-ray computed tomography of polymer composites. *Composites Science and Technology.* **156**:305-19.

[12] Salling FB, Hattel JH, Mikkelsen LP 2020. X-ray computed tomography and scanning electron microscopy datasets of unidirectional and air textured glass fibre composites. *Data in Brief*, In press

[13] Wilhelmsson D, Mikkelsen LP, Fæster S, Asp LE 2019. Influence of in-plane shear on kink-plane orientation in a unidirectional fibre composite. *Composites Part A: App. Sci. and Manufacturing*. **119**:283-90.

[14] Wilhelmsson D, Mikkelsen LP, Fæster S, Asp LE 2019. X-ray tomography data of compression tested unidirectional fibre composites with different off-axis angles. *Data in Brief*. **25**

[15] Varandas LF, Catalanotti G, Melro AR, Tavares RP, Falzon BG 2020. Micromechanical modelling of the longitudinal compressive and tensile failure of unidirectional composites: The effect of gibre misalignment introduced via a stochastic process. *Int. Journ. Of Solids and Structures*. **203**:157-76.

[16] Auenhammer RM, Mikkelsen LP, Asp LE, Blizler BJ 2020. Automated X-ray computer tomography segmentation method for finite element analysis of non-crimp fabric reinforced composites. *Composite Structures*, **256**.

[17] Auenhammer RM, Mikkelsen LP, Asp LE, Blinzler BJ 2020. X-ray tomography based numerical analysis of stress concentration in non-crimp fabric reinforced composites – assesment of segmentation methods. *IOP Conference Serie: Materials Science and Engineering*. **942** [012038].

[18] Advani SG, Tucker CL 1987. The use of tensors to describe and predict fiber orientation in short fiber composites. *Jour. of Rheology*. **31(8)**:751-84.

[19] Jeppesen N, Dahl VA, Christensen AN, Dahl AB, Mikkelsen LP 2020. Characterization of the fiber orientations in non-crimp glass fiber reinforced composites using structure tensor. *IOP Conference Serie: Materials Science and Engineering, 942, [012037]*.

[20] Emerson MJ, Jespersen KM, Dahl AB, Conradsen K, Mikkelsen LP 2017. Individual fiber segmentation from 3D x-ray computed tomography for characterising the fibre orientation in unidirectional composite materials. *Compos. part A: App. Sci. and Manuf.* **97**:83-92.

[21] Rasmussen FS, Emerson MJ, Sonne MR, Hattel JH, Mikkelsen LP and Dahl VA 2019. Fiber segmentation from 3D X-ray computed tomography of composites with continuous and textured glass fiber yarns. *Proc. of 2019 Int. Conf. on Tomography of Mater. & Struct.* (Cairns, Australia).

[22] Amrehn M, Steidl S Kortekaas R, Strumia M, Weingarten M, Kowarschik M, Maier A 2019. A semi-automated usability evaluation framework for interactive image segmentation systems. *Int. Jour. Of Biomedical Imaging.*

[23] Jeppesen N, Mikkelsen LP, Dahl AB, Christensen AN and Dahl VA 2021. Quentifying effects of manufacturing methods on fiber orientation in unidirectional composites using structure tensor analysis. *Composite Science and Technology*, (submitted).

[24] Serafin J, Olson E, Grisetti G 2016. Fast and robust 3D feature extraction from sparse point clouds. *IEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, Daejeon (South Korea), pp. 4105-12.

[25] Krenchel H 1964. *Fiber reinforcement: Theoretical and practical investigations of the elasticity and strength of fiber-reinforced materials.* Akademisk forlag, Danmarks tekniske højskole.

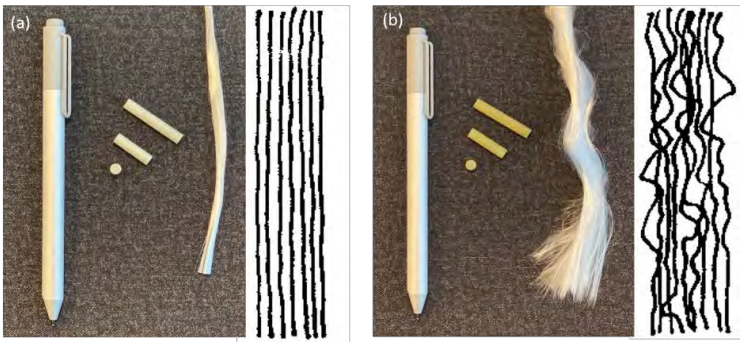


Figure 1. Pictures of composite samples and their respective fibre reinforcement:

a) UD-fib, b) air-tex.

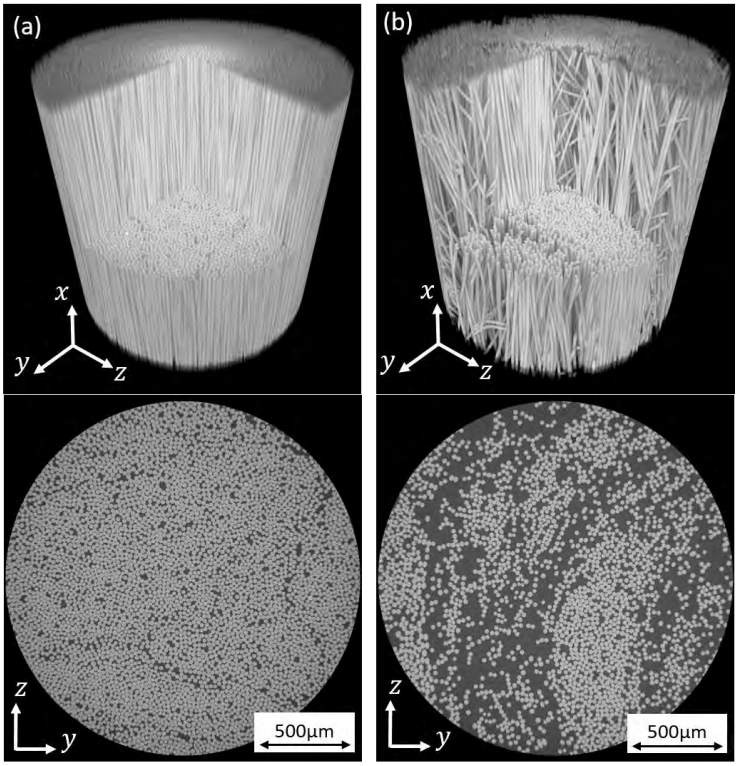


Figure 2. 3D tomograms of 988×1013×999 voxels with a voxel size of 1.99µm together with a corresponding 2D cross-sectional image slice in the yz-plane:

a) UD-fib, b) air-tex.

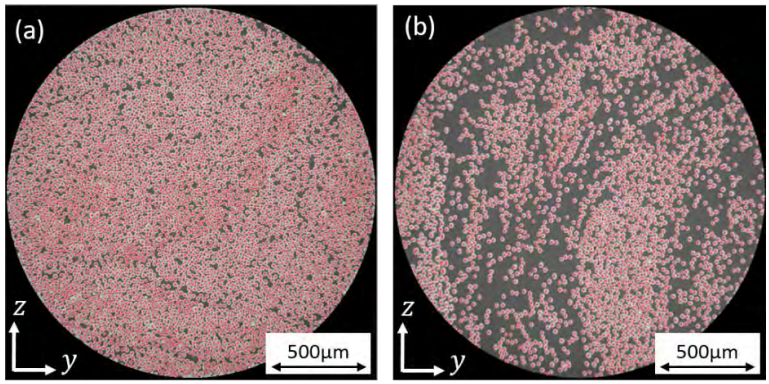


Figure 3. 2D image slices from XCT with fibre center detections:

a) UD-fib, b) air-tex.

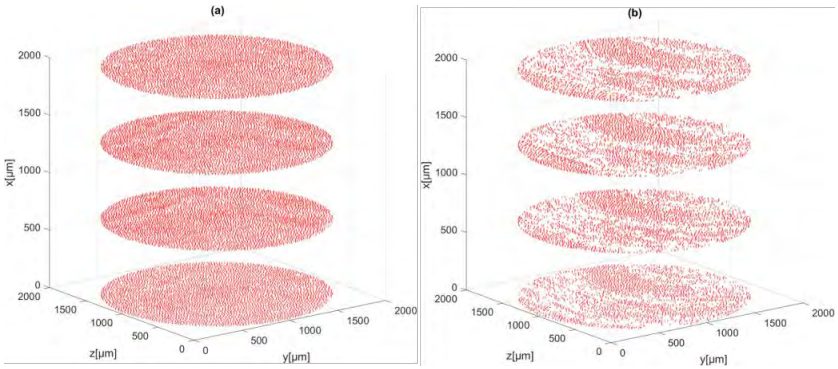


Figure 4. Four evenly spaced stacks of five image slices with detected fibre centers:

a) UD-fib, b) air-tex.

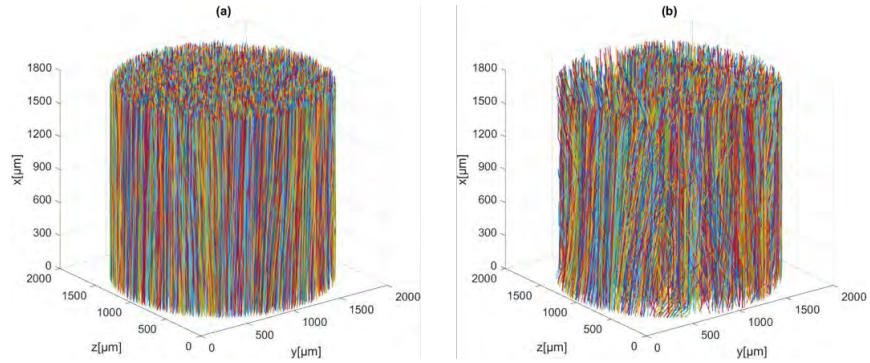


Figure 5. All the 3D fibre trajectories obtained by the fibre tracking segmentation algorithm:

a) UD-fib, b) air-tex.

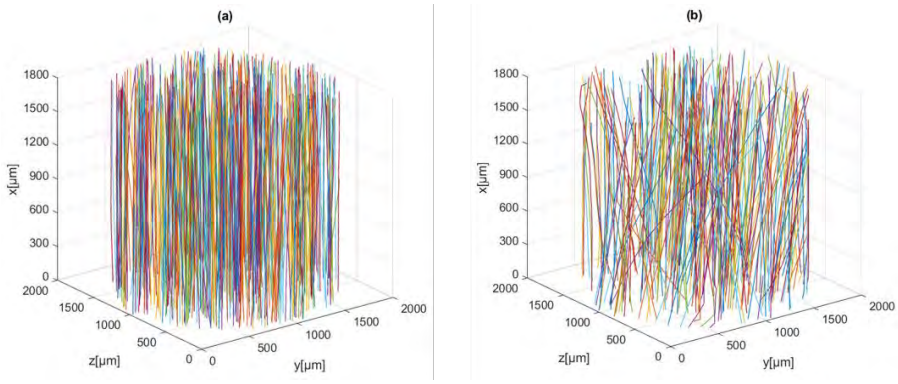


Figure 6. 10% of the 3D fibre trajectories obtained by the fibre tracking algorithm:

a) UD-fib, b) air-tex.

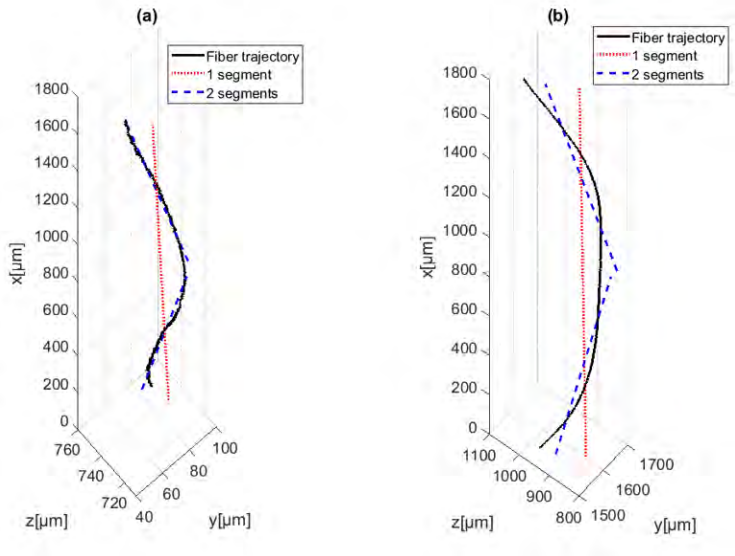


Figure 7. Example of a fibre trajectory and the corresponding linear representation by 1 or 2 segments: a) UD-fib, b) air-tex.

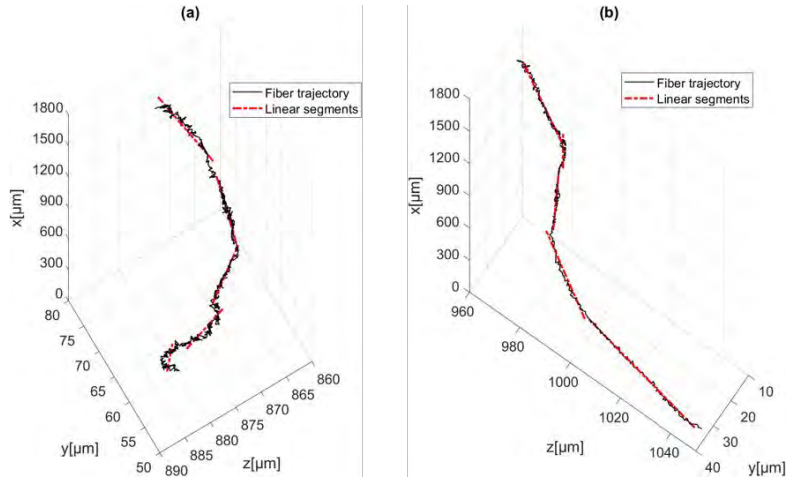


Figure 8. Example of a fibre trajectory and the corresponding linear representation by five fibre sub-segments with a dimensionless height of $\Delta x/D_f \approx 15$ (to be explained):

a) UD-fib, b) air-tex.

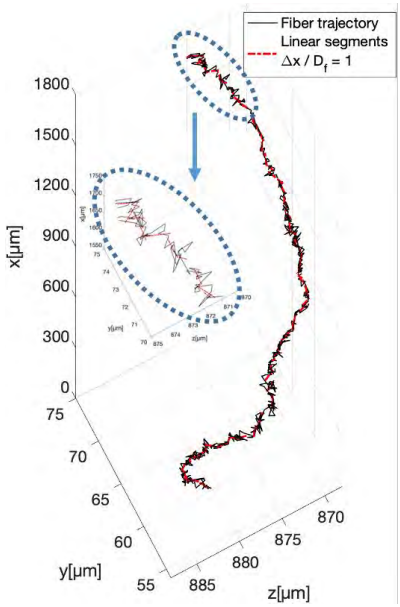


Figure 9. Fibre trajectory and corresponding linear representation using a segmentation mesh with $\Delta x/D_f \approx 1$, UD-fib.

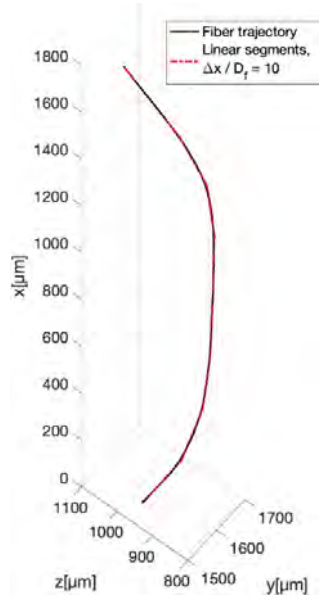


Figure 10. Fibre trajectory and corresponding linear representation using a segmentation mesh with $\Delta x/D_f \approx 10$, air-tex.

Table 1. Summary of the three individual fibre segmentation steps.

Segmentation step	**Segmentation algorithm**	**User input**	**Processing time**
i) Fibre center detections	[20]	RoI: the full 988×1013×999 voxels volume Patch size: 11 pixels Threshold: 0.5	~1h
ii) Fibre tracking	[20]	Image slices: 51-950 $\Delta C_{max}{}^{*}$ = 6 pixels	< 5min
iii) Inclination estimation	PCA	$\frac{\Delta x}{D_f} = [77:1]$ $\left\{\frac{\Delta x}{D_f}\right\}_{opt} = 10$ $\min(N_C)^{**} \geq 95\%$	< 15s

* Maximum allowed fibre center movement from one slice to the next.

** Minimum fraction of fibre centers detected to accept PCA.

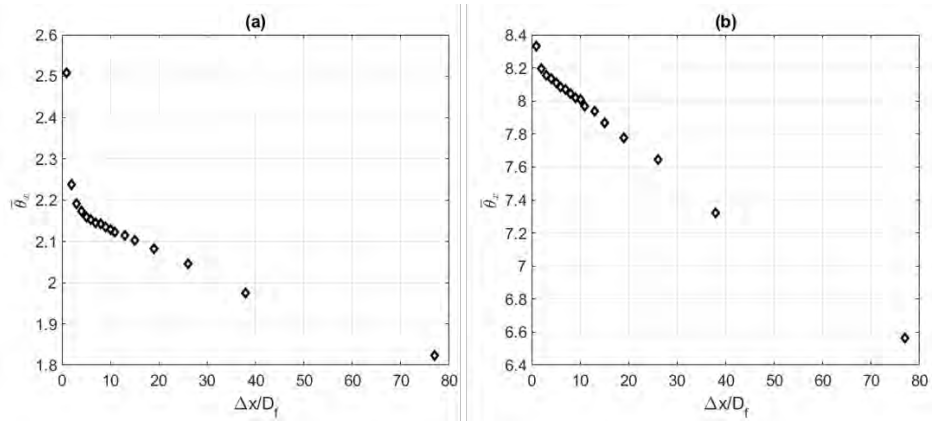


Figure 11. Weighted mean inclination as a function of dimensionless segmentation mess:

a) UD-fib, b) air-tex.

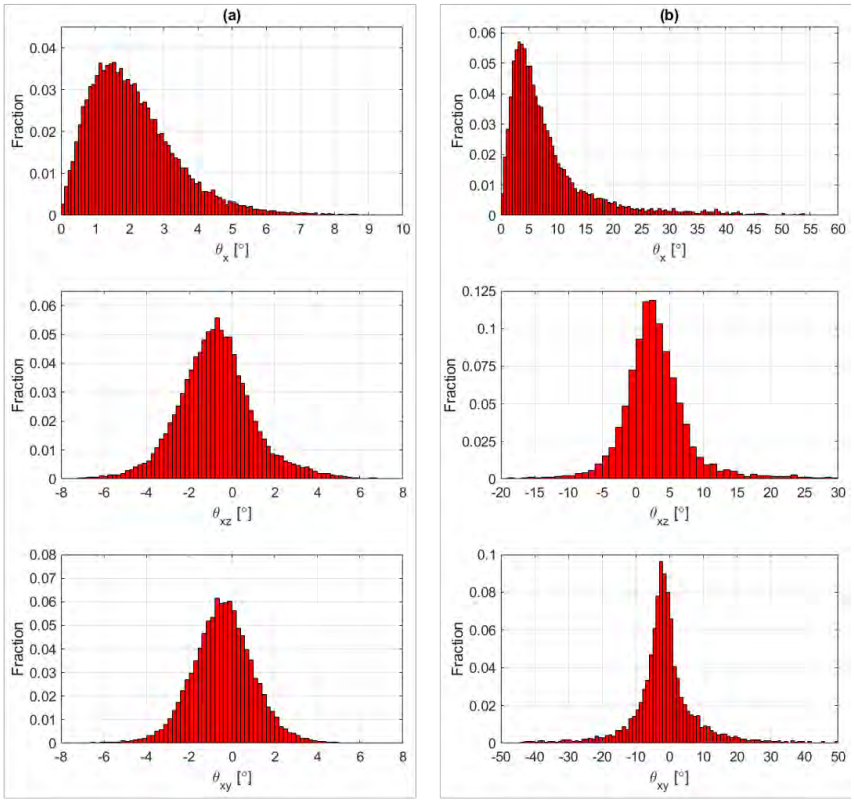


Figure 12. Inclination distribution and projected inclination distributions obtained with a segmentation mesh size of $\frac{\Delta x}{D_f} = 10$: a) UD-fib, b) air-tex.

Table 2. Weighted mean and quartiles of the inclination distribution and weighted mean projection inclinations (cf. Figure 12).

Material System	$\bar{\theta}_x$	$\bar{\theta}_{xy}$	$\bar{\theta}_{xz}$	Q_{x1}	Q_{x2}	Q_{x3}	Q_{x4}
(a) UD-fib	2.1°	-0.7°	-0.4°	1.4°	1.9°	2.6°	11.3°
(b) Air-tex	8.0°	-1.5°	2.6°	3.7°	5.9°	10.0°	55.7°

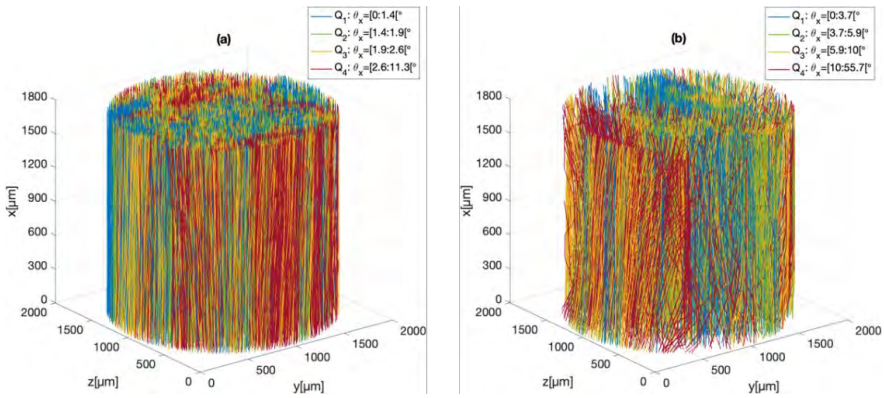


Figure 13. 3D volume of fibre trajectories colored according to their individual mean inclination using the quartiles as interval boundaries: a) UD-fib, b) air-tex.

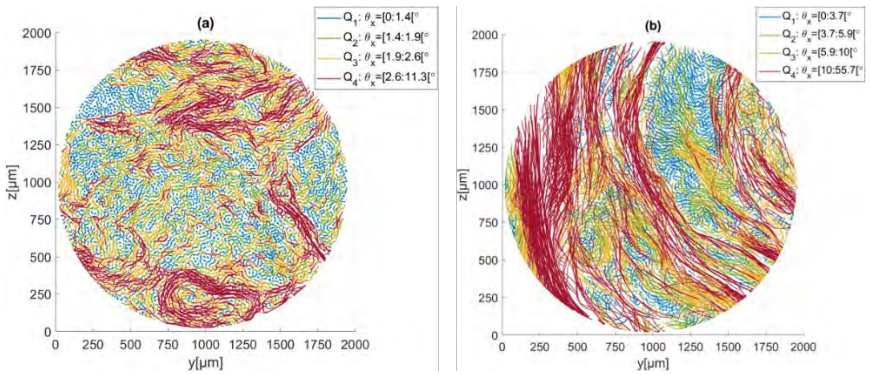


Figure 14. Cross-sectional view of fibre trajectories colored according to their individual mean inclination using the quartiles as interval boundaries: a) UD-fib, b) air-tex.

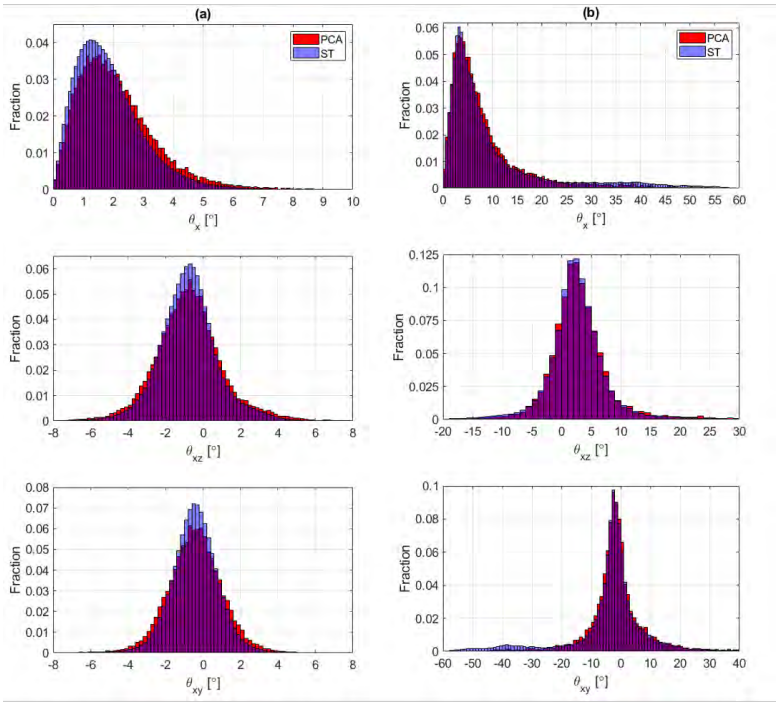


Figure 15. Comparison of the inclination distributions obtained using the PCA method (red) and the ST method (blue): a) UD-fib, b) air-tex.

Bibliography

Boros, E. and P. L. Hammer (November 2002). "Pseudo-Boolean optimization". In: *Discrete Applied Mathematics* 123.1-3, pp. 155–225. ISSN: 0166-218X. DOI: 10.1016/S0166-218X(01)00341-9.

Boykov, Y. and V. Kolmogorov (September 2004). "An experimental comparison of min-cut/max- flow algorithms for energy minimization in vision". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.9, pp. 1124–1137. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.60.

Boykov, Y., O. Veksler, and R. Zabih (2001). "Fast approximate energy minimization via graph cuts". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23.11, pp. 1222–1239. ISSN: 0162-8828. DOI: 10.1109/34.969114.

Boykov, Y. and G. Funka-Lea (November 2006). "Graph Cuts and Efficient N-D Image Segmentation". In: *International Journal of Computer Vision* 70.2, pp. 109–131. ISSN: 1573-1405. DOI: 10.1007/s11263-006-7934-5.

Delong, A. and Y. Boykov (June 2008). "A scalable graph-cut algorithm for N-D grids". In: *26th IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. June. IEEE, pp. 1–8. ISBN: 978-1-4244-2242-5. DOI: 10.1109/CVPR.2008.4587464.

Dinic, E. A. (1970). "Algorithm for solution of a problem of maximum flow in networks with power estimation". In: *Soviet Math. Dokl.* Vol. 11, pp. 1277–1280.

Fishbain, B., D. S. Hochbaum, and S. Mueller (March 2016). "A competitive study of the pseudoflow algorithm for the minimum s–t cut problem in vision applications". In: *Journal of Real-Time Image Processing* 11.3, pp. 589–609. ISSN: 1861-8219. DOI: 10.1007/s11554-013-0344-3.

Ford Jr, L. R. and D. R. Fulkerson (1962). *Flows in networks*. Princeton university press.

Goldberg, A. V., S. Hed, H. Kaplan, P. Kohli, et al. (2015). "Faster and More Dynamic Maximum Flow by Incremental Breadth-First Search". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Ed. by N. Bansal and I. Finocchi. Vol. 9294. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 619–630. ISBN: 978-3-662-48350-3. DOI: 10.1007/978-3-662-48350-3_52.

Goldberg, A. V., S. Hed, H. Kaplan, R. E. Tarjan, et al. (2011). "Maximum Flows by Incremental Breadth-First Search". In: *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes*

in Bioinformatics). Vol. 6942 LNCS, pp. 457–468. ISBN: 978-3-642-23719-5. DOI: 10.1007/978-3-642-23719-5_39.

Goldberg, A. V. and R. E. Tarjan (October 1988). "A new approach to the maximum-flow problem". In: *Journal of the ACM* 35.4, pp. 921–940. ISSN: 0004-5411. DOI: 10.1145/48014.61051.

Hammer, P. L., P. Hansen, and B. Simeone (February 1984). "Roof duality, complementation and persistency in quadratic 0–1 optimization". In: *Mathematical Programming* 28.2, pp. 121–155. ISSN: 1436-4646. DOI: 10.1007/BF02612354.

Hochbaum, D. S. (August 2008). "The Pseudoflow Algorithm: A New Algorithm for the Maximum-Flow Problem". In: *Operations Research* 56.4, pp. 992–1009. ISSN: 0030-364X. DOI: 10.1287/opre.1080.0524.

Isack, H. et al. (July 2017). "Efficient Optimization for Hierarchically-Structured Interacting Segments (HINTS)". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Vol. 2017-Janua. IEEE, pp. 4981–4989. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.529.

Ishikawa, H. (October 2003). "Exact optimization for markov random fields with convex priors". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 25.10, pp. 1333–1336. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2003.1233908.

Jamriska, O., D. Sykora, and A. Hornung (June 2012). "Cache-efficient graph cuts on structured grids". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 3673–3680. ISBN: 978-1-4673-1228-8. DOI: 10.1109/CVPR.2012.6248113.

Jiangyu Liu and Jian Sun (June 2010). "Parallel graph-cuts by adaptive bottom-up merging". In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 2181–2188. ISBN: 978-1-4244-6985-7. DOI: 10.1109/CVPR.2010.5539898.

Kang Li et al. (January 2006). "Optimal Surface Segmentation in Volumetric Images-A Graph-Theoretic Approach". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.1, pp. 119–134. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2006.19.

Karamov, R. et al. (March 2020). "Micro-CT based structure tensor analysis of fibre orientation in random fibre composites versus high-fidelity fibre identification methods". In: *Composite Structures* 235.June 2019, p. 111818. ISSN: 0263-8223. DOI: 10.1016/j.compstruct.2019.111818.

Kolmogorov, V. and R. Zabih (February 2004). "What energy functions can be minimized via graph cuts?" In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26.2, pp. 147–159. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2004.1262177.

Kolmogorov, V. and C. Rother (July 2007). "Minimizing Nonsubmodular Functions with Graph Cuts-A Review". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29.7, pp. 1274–1279. ISSN: 0162-8828. DOI: 10.1109/TPAMI.2007.1031.

Martin, R. W. et al. (November 2018). "Comparison of nondestructive testing techniques for the inspection of wind turbine blades' spar caps". In: *Wind Energy* 21.11, pp. 980–996. ISSN: 1095-4244. DOI: 10.1002/we.2208.

Nelson, L., R. Smith, and M. Mienczakowski (January 2018). "Ply-orientation measurements in composites using structure-tensor analysis of volumetric ultrasonic data". In: *Composites Part A: Applied Science and Manufacturing* 104, pp. 108–119. ISSN: 1359-835X. DOI: 10.1016/j.compositesa.2017.10.027.

Rother, C. et al. (June 2007). "Optimizing Binary MRFs via Extended Roof Duality". In: *2007 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, pp. 1–8. ISBN: 1-4244-1179-3. DOI: 10.1109/CVPR.2007.383203.

Straumit, I., S. V. Lomov, and M. Wevers (February 2015). "Quantification of the internal structure and automatic generation of voxel models of textile composites from X-ray computed tomography data". In: *Composites Part A: Applied Science and Manufacturing* 69, pp. 150–158. ISSN: 1359-835X. DOI: 10.1016/j.compositesa.2014.11.016.

Weickert, J. (1998). *Anisotropic Diffusion in Image Processing*. Teubner Stuttgart. ISBN: 3519026066.

Westin, C.-F. et al. (June 2002). "Processing and visualization for diffusion tensor MRI". In: *Medical Image Analysis* 6.2, pp. 93–108. ISSN: 1361-8415. DOI: 10.1016/S1361-8415(02)00053-1.

Wu, X. and D. Z. Chen (2002). "Optimal Net Surface Problems with Applications". In: pp. 1029–1042. ISBN: 978-3-540-45465-6. DOI: 10.1007/3-540-45465-9_88.