Труды МАИ. 2025. № 144
Trudy MAI. 2025. No. 144. (In Russ.)

ИНФОРМАТИКА, МОДЕЛИРОВАНИЕ И УПРАВЛЕНИЕ

Научная статья
УДК 004.932+519.254
URL: https://trudymai.ru/published.php?ID=186313
EDN: https://www.elibrary.ru/OUPNNE

АЛГОРИТМ ОПРЕДЕЛЕНИЯ ГРАНИЦ МОНОСЛОЕВ КОМПОЗИЦИОННОГО МАТЕРИАЛА НА ОСНОВЕ ДАННЫХ КОМПЬЮТЕРНОЙ ТОМОГРАФИИ

Андрей Владимирович Пантелеев, Николай Васильевич Турбин✉, Николай Александрович Тучков, Роман Львович Талья, Исак Азизович Ахмедов

Московский авиационный институт (национальный исследовательский университет)

Москва, Россия

✉turbinnv@mai.ru

Аннотация. В работе предложен алгоритм и описано программное обеспечение для автоматического определения границ монослоев в слоистых композитных материалах на основе последовательности томографических изображений. Актуальность задачи обусловлена необходимостью анализа микроструктуры композитов для оценки их механических свойств и долговечности с помощью расчетных моделей, поскольку границы монослоев затруднительно выявить

традиционными методами неразрушающего контроля. Разработанный алгоритм включает шесть основных этапов: извлечение данных о включениях из каждого изображения с применением фильтров и морфологических операций, выделение надежных сегментов границ с использованием фильтра Собеля, добавление информации о трещинах и обработка данных для каждого изображения индивидуально, усреднение информации всего пакета изображений, построение границ монослоев с применением геометрических и эмпирических методов и финальное уточнение границ путем дополнительного цикла построения границ по всей собранной информации о монослоях. Программное обеспечение принимает на вход последовательность томографических изображений микроструктуры композита и возвращает множество векторов, описывающих координаты границ монослоев. Результаты работы представлены в виде визуализированных границ на выходных изображениях. Предложенный подход позволяет автоматизировать процесс анализа внутренней структуры композитных материалов и использовать полученные данные в расчетных моделях.

***Ключевые слова*:** композитный материал, монослой, компьютерная томография, фильтрация, бинаризация, морфологические преобразования.

***Для цитирования*:** Пантелеев А.В., Турбин Н.В., Тучков Н.А., Талья Р.Л., Ахмедов И.А. Алгоритм определения границ монослоев композиционного материала на основе данных компьютерной томографии // Труды МАИ. 2025. № 144. URL: https://trudymai.ru/published.php?ID=186313

Original article

ALGORITHM FOR MONOLAYER BOUNDARIES IDENTIFICATION IN COMPOSITE MATERIAL FROM COMPUTER TOMOGRAPHY IMAGES

Andrei V. Panteleev, Nikolay V. Turbin✉, Nikolay A. Tuchkov, Roman L. Talya, Isak A. Akhmedov

Moscow Aviation Institute (National Research University),

Moscow, Russian Federation

✉turbinnv@mai.ru

Abstract. The paper proposes an algorithm and software for the automatic detection of monolayer boundaries in layered composite materials based on a sequence of tomographic images. The relevance of the problem stems from the need to analyze the microstructure of composites to assess their mechanical properties and durability, as well as the complexity of identifying monolayer boundaries using traditional non-destructive testing methods. The developed algorithm consists of six main stages: extraction of inclusion data using filters and morphological operations, identification of reliable boundary segments with the Sobel filter, addition of crack data and processing of the obtained data for each image individually, data averaging for the entire package, construction of monolayer boundaries using geometric and empirical methods and final border refinement through an additional cycle of border construction on the aggregated monolayer data. The software accepts a sequence of tomographic images of the composite microstructure as input and outputs a set of vectors

describing the coordinates of monolayer boundaries. The results are presented as visualized boundaries on output images. The proposed approach enables automation of the analysis of the internal structure of composite materials and usage of the extraccted data in computational models.

***Keywords*:** composite material, matrix cracking, computer tomography, filtration, binarization, morphological transformations.

***For citation*:** Panteleev A.V, Turbin N.V., Tuchkov N.A., Talya R.L., Akhmedov I.A. Algorithm for monolayer boundaries identification in composite material from computer tomography images // Trudy MAI. 2025. No. 144. (In Russ.) URL: https://trudymai.ru/published.php?ID=186313

Введение

Современные отрасли машиностроения, включая авиакосмическую, автомобильную и ветроэнергетическую промышленность, активно используют слоистые композитные материалы благодаря их уникальным механическим свойствам, таким как высокая прочность при низком весе. Микроструктура этих материалов, включающая распределение монослоев и возможные дефекты, такие как полости или трещины, существенно влияет на их эксплуатационные характеристики и долговечность [1–5]. Точное определение границ монослоев позволяет в дальнейшем привязать типовое распределение дефектов к конкретным слоям укладки композитов и спрогнозировать поведение пакета под нагрузкой с помощью расчетных моделей.

Традиционные методы неразрушающего контроля, такие как ультразвуковая диагностика или рентгенография, часто сталкиваются с ограничениями при анализе границ монослоев из-за малых размеров характерных структурных элементов в этих зонах. Компьютерная томография (КТ) предоставляет возможность получения высококачественных трехмерных изображений внутренней структуры композитов, что делает её мощным инструментом для анализа микроструктуры [6–9]. Однако обработка томографических данных для выделения границ монослоев стандартными подходами, например сегментацией, остается сложной задачей из-за наличия шумов и низкого контраста между слоями.

Ручной анализ томографических изображений является трудоемким и субъективным процессом, что подчеркивает необходимость разработки автоматизированных методов. Современные подходы к обработке изображений, включая морфологические операции, фильтрацию и алгоритмы контурного анализа, позволяют эффективно решать задачи сегментации и классификации структурных элементов [10–15]. В то же время, для повышения точности необходимо учитывать последовательность томографических срезов, что требует интеграции информации из множества изображений.

Целью данной работы является разработка алгоритма и его программной реализации для автоматического определения границ монослоев в слоистых композитах на основе данных томографических изображений. Алгоритм включает этапы суммирования информации из последовательности изображений, извлечения данных о включениях и сегментах границ, их обработки и построения окончательных границ монослоев с последующим уточнением. Реализация выполнена на языке

Python с использованием библиотек NumPy и OpenCV. Результатом является множество векторов, описывающих координаты границ монослоев, и визуализированные изображения, что способствует автоматизации анализа микроструктуры композитов и повышению объективности оценки их качества. Статья является частью более общего исследования. В [16] описан алгоритм выделения трещин с помощью анализа последовательности томографических изображений. Выделение положения монослоев позволит определить наличие и расположение трещин в каждом слое и, как следствие, оценить состояние композиционного материала на основе расчетных моделей.

1. Техническая постановка задачи

Дана последовательность изображений (пакет изображений) в градациях серого, на каждом из которых представлен срез микроструктуры композита, полученный путем томографического анализа, и информация об этом срезе. Томографические снимки (сканы) сделаны с шагом ~0.005 мм. и имеют масштаб 6.5×6.5 мм. Композит состоит из 32 монослоев, шириной от 0.1 до 0.2 мм., между которыми могут содержатся пустоты, воздух или плохо отвержденное связующее, представляющие из себя включения черного цвета в материале. Внутри монослоя могут находиться трещины, являющиеся вертикальными черными полосами, соединяющими верхнюю и нижнюю границу монослоя.

Требуется по заданной последовательности изображений определить границы монослоев.

2. Математическая постановка задачи

Дана последовательность изображений в градациях серого, которая представляется тензором $P = \left(P_{ijk}\right)^{h \times w \times n}$, $P_{ijk} \in [0,255]$, где h – высота изображения, w – ширина, n – количество изображений. Элементы тензора соответствуют яркости пикселей соответствующего изображения.

Границы монослоев представляются векторами, образованными координатами точек: $b = ((x_0, y_0) \quad (x_1, y_1) \quad \ldots \quad (x_n, y_n))^T$, n – количество точек, образующих границу монослоя.

Требуется разработать алгоритм и его программную реализацию, которые на основе матричного представления изображения микроструктуры композита при помощи методов обработки и анализа изображений определяют границы всех монослоев.

3. Алгоритм обнаружения монослоев

В алгоритме обнаружения монослоев можно выделить шесть основных шагов.

3.1. Извлечение информации о включениях в каждом изображении пакета. Включения определяются как затемнения на сканах, возникающие из-за присутствия в материале пустот, термовуалей и других подобных дефектов в структуре композита. Включения имеют примерно прямоугольную или округлую форму, вытянутую по горизонтальному направлению.

3.2. Извлечение информации о наиболее надежных сегментах границ в каждом изображении пакета.

3.3. Добавление информации о трещинах, суммирование данных и их обработка для каждого изображения пакета.

3.4. Усреднение информации всего пакета изображений.

3.5. Построение первоначальной аппроксимации границ монослоев в каждом скане.

3.6. Суммирование информации из полученных границ по пакету входных изображений. Выбирается изображение с числом слоев, наиболее близких к заданному заранее, а полученная из него информация о расположении монослоев улучшается соседними сканами.

4. Извлечение информации о включениях

Данный и следующий шаги алгоритма построения границ являются наиболее важными для решения задачи, так как от чистоты и количества данных сильно зависит качество нахождения границ монослоев. Сильно зашумленные данные могут помешать построить границы в принципе, а большое количество данных может значительно замедлить выполнение программы и ухудшить качество получаемого результата.

4.1. Бинарная морфология. Морфологические операции.

Общей частью шагов алгоритма являются морфологические операции, которые являются нелинейными преобразованиями. Помимо этого, результат морфологических операций зависит не от яркости пикселей, а от их относительного расположения. Данная особенность делает морфологические операции особенно

полезными при обработке двоичных изображений, т.е. изображений только с двумя цветами, например, черным и белым.

В бинарной морфологии двоичное изображение является подмножеством Евклидового пространства R^2 или целочисленной сетки Z^2. Для проведения операций над ним вводится понятие структурного элемента, который также является двоичным изображением. Проведение операции имеет вид применения какой-либо функции в области ненулевых значений структурного элемента, который от шага к шагу меняет положение относительно исследуемого изображения.

Основные структурные элементы (используемое пространство Z^2):

- Открытый диск радиуса r с центром в начале координат. Для $r=3$ структурный элемент имеет вид (рис. 4.1, а)

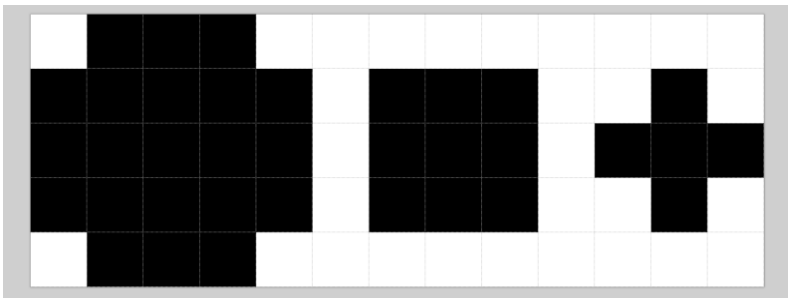
$$Disk(3)=\begin{Bmatrix}(2,0),(2,1),(1,1),(1,2),(0,2),\\(-1,2),(-1,1),(-2,1),(-2,0),\ldots,\\(1,0),(1,1),(-1,0),(0,-1),(0,0)\end{Bmatrix}. \qquad (4.1.1)$$

- Прямоугольник с размерами H и W. Например, для $H=W=3$ структурный элемент имеет вид (рис. 4.1, б)

$$Sqr(3)=Rect(3,3)=\begin{Bmatrix}(-1,1),(0,1),(1,1),(1,0),\\(1,-1),(0,-1),(-1,-1),(0,0)\end{Bmatrix}. \qquad (4.1.2)$$

- Перекрестие в пространстве Z^2 с размерами H и W. Для $H=W=3$ (рис. 4.1, в):

$$Cross(3,3)=\{(-1,0),(0,0),(1,0),(0,1),(0,-1)\}. \qquad (4.1.3)$$



а б в

Рис. 4.1. Двоичные изображения основных структурных элементов бинарной морфологии

Рассмотрим основные морфологические операции.

- Перенос изображения A на вектор t: $A_t = \{a + t \mid a \in A\}$.

Данная операция смещает все пиксели a изображения A на вектор t. Это преобразование является основной частью большинства дальнейших морфологических операций.

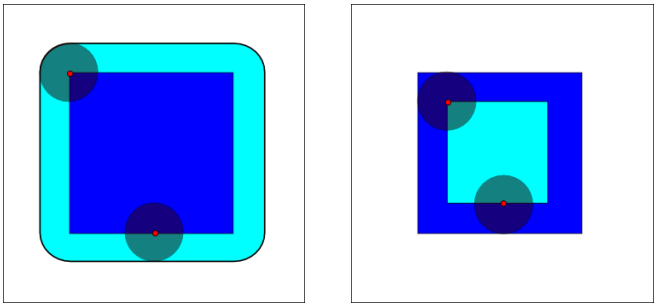
Пусть структурный элемент B применяется к двоичному изображению A.

- Наращивание (dilate): $dilate(A, B) = \bigcup_{b \in B} A_b$.

Данная операция увеличивает размеры изображения A при помощи объединения копий исходного изображения, полученных переносом A на вектор b. В свою очередь вектор b определяет координаты конкретного ненулевого пикселя структурного элемента B (рис. 4.2, а). На больших размерах такая операция помогает удалить пробелы между пикселями, объединяя разъединенные узоры в один.

- Эрозия (erode): $erode(A,B) = \{a \in A \mid B_a \subseteq A\}$.

Данная операция уменьшает размеры изображения, отсеивая все пиксели изображения A, в которых структурный элемент имеет ненулевое пересечение с областью вне изображения (рис. 4.2, б). Такая операция помогает на больших изображениях сделать толстые линии более тонкими, что особенно полезно при обработке нечеткого текста или поиске четких границ.



а б

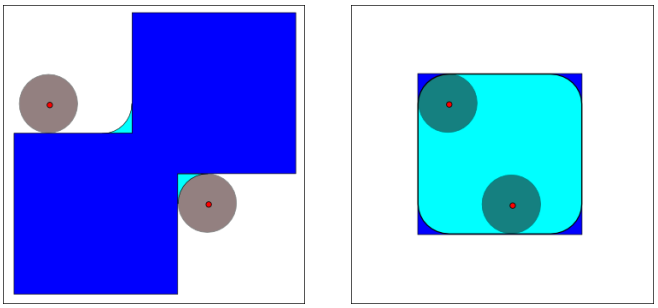
Рис. 4.2. Примеры применения операции наращивания и эрозии с дисковым структурным элементом. Внешний и внутренний квадраты – соответствующие результирующие изображения

- Замыкание (close): $close(A,B) = erode(dilate(A,B),B)$.

Данная производная операция исключает малые по сравнению со структурным элементом “дырки и щели” из изображения A и убирает углубления в его контуре (рис. 4.3, а). Однако такая операция одновременно увеличивает изображение, из-за чего имеет смысл сразу после нее применить эрозию. Изображение становится полным после замыкания, ввиду чего эрозия не испортит получившееся изображение.

- Размыкание (open): $open(A,B) = dilate(erode(A,B),B)$.

Данная производная операция создает обратный эффект по сравнению с замыканием и убирает все малые по сравнению со структурным элементом “вкрапления” и линии в изображении (рис. 4.3, б). Стоит заметить, что, как и замыкание, данная операция изменяет размер исходного изображения, а именно уменьшает и сглаживает контуры, ввиду чего иногда требуется использовать операцию наращивания сразу после применения замыкания.



а б

Рис. 4.3. Примеры применения операций замыкания и размыкания с дисковыми структурными элементами. Исходное множество имеет темно-синий цвет

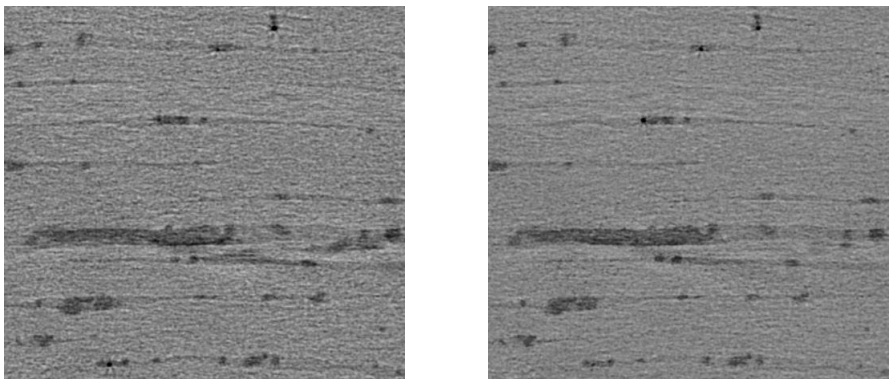
Важно заметить, что операции бинарной морфологии не обязательно коммутативны. Если использовать эрозию со структурным элементом размеров больших, чем изображение, то после применения оно станет пустым, а все дальнейшие операции его никак не изменят. За счет этого и можно осуществлять чистку изображения от шума или удаления ненужных деталей, не теряя при этом всю информацию об исходном изображении.

4.2. Алгоритм обнаружения включений.

4.2.1. Применяется размытие двусторонним фильтром (bilateral filter) для уменьшения шума на изображении (рис. 4.4, а) с сохранением четкости границ (рис. 4.4, б):

$$
\begin{gathered}
Inc_{x,y}^{(1)} = \frac{1}{W_p} \sum_{(x_0,y_0)\in\Omega(x,y)} G_{\sigma_1}(\Delta x) G_{\sigma_2}(\Delta I) I_{x_0,y_0}, \\
W_p = \sum_{(x_0,y_0)\in\Omega(x,y)} G_{\sigma_1}(\Delta x) G_{\sigma_2}(\Delta I), \\
\Delta x = \sqrt{(x_0 - x)^2 + (y_0 - y)^2}, \quad \Delta I = \left| I_{x_0,y_0} - I_{x,y} \right|, \\
G_{\sigma_i}(d) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{d^2}{2{\sigma_i}^2} \right), \; i = 1,2.
\end{gathered}
\tag{4.2.1}
$$

где $Inc_{x,y}^{(1)}$ – пиксель отфильтрованного изображения; $I_{x,y}$ – пиксель оригинального изображения; (x,y) – координаты фильтруемого пикселя; $\Omega(x,y)$ – окно, центрированное на (x,y); $G_{\sigma_1}(d)$ – ядро пространства, сглаживающее разницу в координатах; $G_{\sigma_2}(d)$ – ядро дистанции, сглаживающее разности в интенсивности цвета; σ_i – стандартные отклонения соответствующих гауссовых ядер.



а б

Рис. 4.4. Результаты применения двустороннего фильтра. Слева – фрагмент оригинального изображения

4.2.2. Применяется бинаризация. Данная операция превращает изображение (рис. 4.5, а) в двоичное, в котором фон будет представлен черными пикселями, а нужные для дальнейшей фильтрации детали – белыми (рис. 4.5, б). Так как в изображении возможны вариации яркости из-за различной освещенности образца, применяемая бинаризация называется адаптивной.

В общем случае бинаризация описывается формулой

$$Inc_{x,y}^{(2)} = \begin{cases} m, & Inc_{x,y}^{(1)} > T_{x,y}, \\ 0, & Inc_{x,y}^{(1)} \le T_{x,y}, \end{cases} \qquad (4.2.2)$$

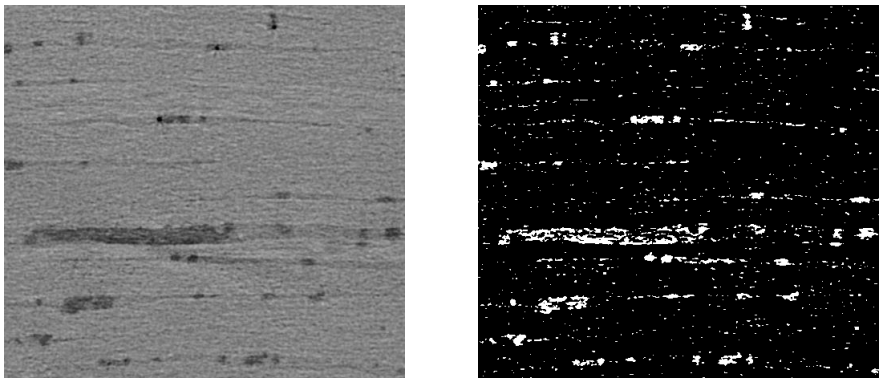
где $Inc_{x,y}^{(2)}$ – пиксель бинаризованного изображения с координатами (x, y); $Inc_{x,y}^{(1)}$ – фильтруемый пиксель исходного изображения; m – максимальное значение бинаризации; $T_{x,y}$ – порог бинаризации на пикселе с координатами (x, y).

В данной работе используется гауссово ядро для значения порога бинаризации:

$$T_{x,y} = \sum_{(x_0,y_0) \in \Omega(x,y)} Inc^{(2)}_{x,y} G_{\sigma_3}(\Delta x) - C,$$
$$\Delta x = \sqrt{(x_0 - x)^2 + (y_0 - y)^2}, \qquad (4.2.3)$$

где $\Omega(x,y)$ – окно, центрированное на пикселе с координатами (x,y); C – константа, вычитаемая из значения порога; $G_{\sigma_3}(\Delta x)$ – ядро пространства, сглаживающее разницу в координатах.

В решаемой задаче размер окна $k = 81$, вычитаемая константа $C = 25$.



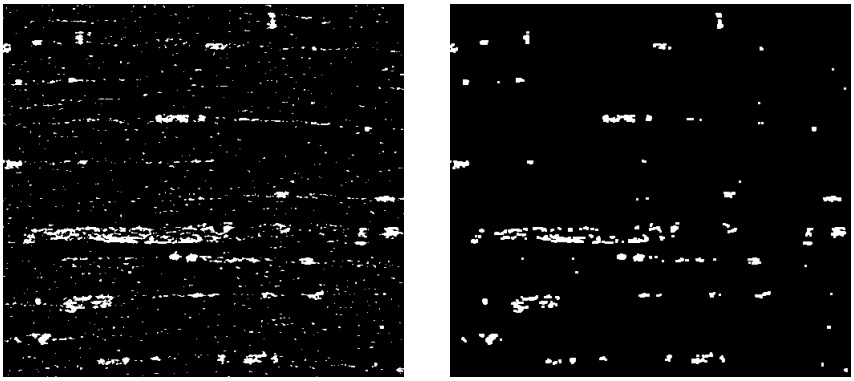
а б

Рис. 4.5. Результат применения адаптивной бинаризации

4.2.3. К текущему изображению (рис. 4.6, а) применяется морфологическая операция размыкания с квадратным структурным элементом $Sqr(3)$ размера 3. Такая обработка позволяет снизить шум от малых вкраплений (рис. 4.6, б):

$$Inc^{(3)} = open(Inc^{(2)}, Sqr(3)). \qquad (4.2.4)$$

Так как включения имеют примерно округлую форму и достаточно большие по размеру в сравнении с шумом, то более продвинутых схем уменьшения шума не требуется.

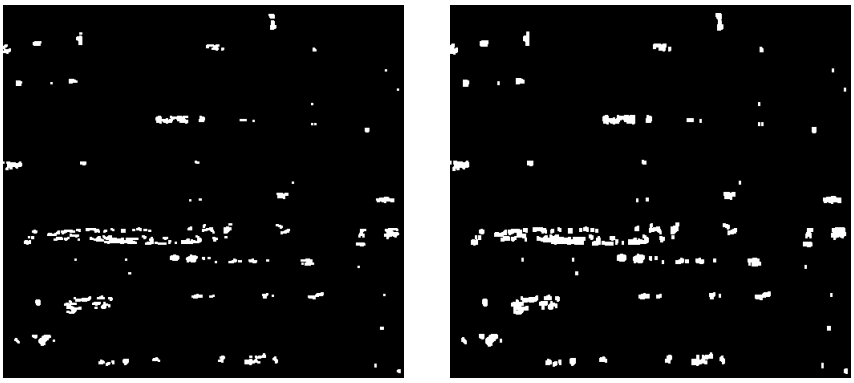


а б

Рис. 4.6. Результат применения операции размыкания

4.2.4. Применяется операция наращивания на изображение с прямоугольным структурным элементом $Rect(3,1)$ размеров 3×1. Такая морфологическая операция помогает слить некоторые вкрапления на текущем изображении (рис. 4.7, а) в одно большего размера, что снижает количество точек, которые требуется обработать далее (рис. 4.7, б):

$$Inc^{(4)} = dilate(Inc^{(3)}, Rect(3,1)). \qquad (4.2.5)$$

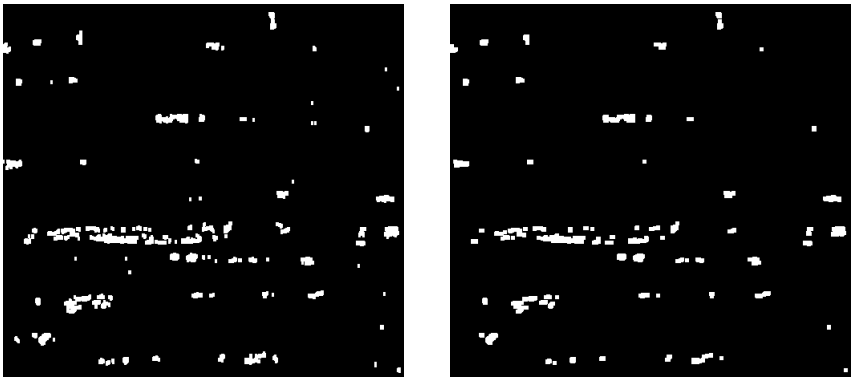


а б

Рис. 4.7. Результат применения операции наращивания

4.2.5. Последним шагом морфологических операций, применяемым к текущему изображению (рис. 4.8, а), является двукратное повторение шага 4.2.3, что еще более уменьшает количество вкраплений, полученных от шума в изображении (рис. 4.8, б):

$$Inc^{(5)} = open(open(Inc^{(4)}, Sqr(3)), Sqr(3)). \qquad (4.2.6)$$



а б

Рис. 4.8. Результат применения двукратной операции размыкания

4.2.6. На полученном изображении проводится поиск всех контуров при помощи алгоритма Сузуки [12], чтобы извлечь примерные прямоугольные области включений:

$$IncCr = \left\{ \left(x_i, y_i, w_i, h_i \right) \right\},\ i = \overline{0, N^{(IncCr)}}, \qquad (4.2.7)$$

где x_i, y_i – координаты левого верхнего угла i-й области, w_i, h_i – ее ширина и высота, $N^{(IncCr)}$ – число контуров.

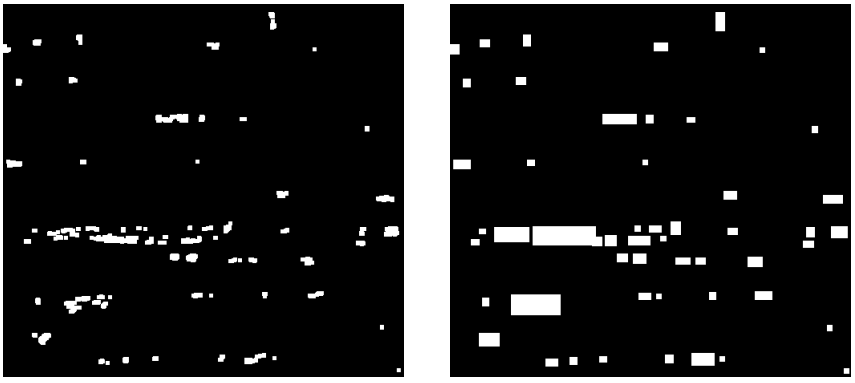
4.2.7. При получении контуров из текущего изображения (рис. 4.9,а) их ограничивающие прямоугольники наносятся на новое изображение $IncCrTmp$, к которому потом применяется наращивание с квадратным структурным элементом размера 3 (рис. 4.9, б):

$$IncCrTmp := dilate(IncCrTmp, Sqr(3)). \quad (4.2.8)$$

4.2.8. Проводится повторный поиск контуров на новом изображении. Они затем наносятся на исходное изображение, которое было предварительно очищено. К нему применяется операция эрозии с таким же структурным элементом, как и при наращивании:

$$Inc^{(6)} = erode(IncCrTmp, Sqr(3)). \quad (4.2.9)$$

4.2.9. Шаги 5.2.7 и 5.2.8 повторяются дважды, чтобы слить как можно больше точек включений и уменьшить количество информации для дальнейшего алгоритма. Наличие большого количества близлежащих точек может плохо сказаться на точности построения границ, ввиду чего требуется описанная здесь последовательность извлечений контуров.



а б

Рис. 4.9. Полученные прямоугольные области включений

4.2.10. У полученных ограничивающих прямоугольников ищется центр, который подается на вход дальнейшим шагам алгоритма. Эти точки (рис 4.10) являются примерными центрами наиболее четких и крупных включений и дефектов на изображении:

$$Incs = \left\{ \left(x_i + \frac{w_i}{2}, y_i + \frac{h_i}{2} \right) \right\},\ i = \overline{0, N^{(Incs)}}, \qquad (4.2.10)$$

где $N^{(Incs)}$ – число включений.

В результате фильтраций центры трещин должны отсутствовать, а горизонтальные вкрапления, не являющиеся включениями, стерты при применении размытий и очисток от шума.

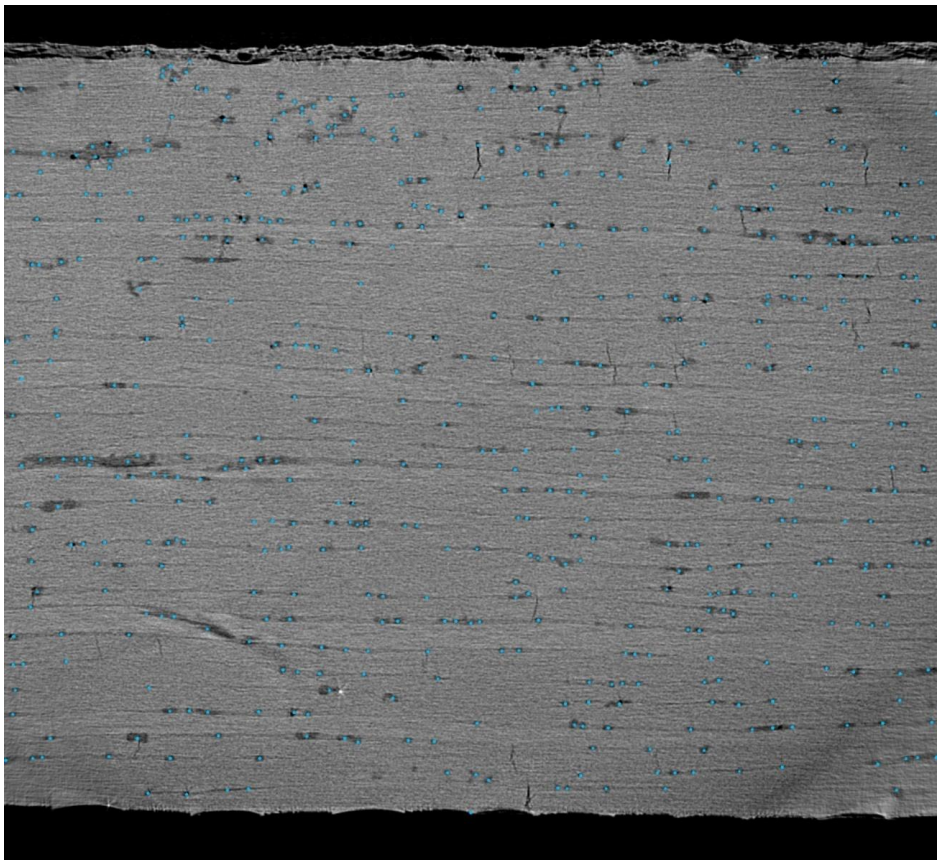


Рис. 4.10. Извлеченные центры включений (синие точки)

5. Извлечение информации о наиболее надежных сегментах границ

Суть метода, реализуемого на данном шаге, такая же, как на предыдущем.

5.1. Применяется размытие двусторонним фильтром:

$$Seg_{x,y}^{(1)} = \frac{1}{W_p} \sum_{(x_0,y_0)\in\Omega(x,y)} G_{\sigma_1}(\Delta x) G_{\sigma_2}(\Delta I) I_{x_0,y_0},$$

$$W_p = \sum_{(x_0,y_0)\in\Omega(x,y)} G_{\sigma_1}(\Delta x) G_{\sigma_2}(\Delta I),$$

$$\Delta x = \sqrt{(x_0 - x)^2 + (y_0 - y)^2}, \quad \Delta I = \left| I_{x_0,y_0} - I_{x,y} \right|, \tag{4.11}$$

$$G_{\sigma_i}(d) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{d^2}{2\sigma_i^2} \right), \; i = 1,2.$$

Результат аналогичен шагу 4.2.1 извлечения включений (рис 4.4, б)

5.2. Применяется фильтр Собеля, чтобы извлечь все горизонтальные и окологоризонтальные линии, наиболее крупные из которых характеризуют границу разделения слоев (рис. 5.1, б). Для этого используется свертка исходного изображения (рис. 5.1, а) с ядром фильтра S размеров $m \times n$:

$$Seg_{x,y}^{(2)} = S * Seg_{x,y}^{(1)} = \sum_{\Delta x=0}^{m} \sum_{\Delta y=0}^{n} S_{\Delta x,\Delta y} Seg_{x-\Delta x,y-\Delta y}^{(1)}. \tag{4.12}$$

Этот фильтр является аппроксимацией градиента изображения, измеряя разницу в интенсивности в определенном направлении. Например, можно определить градиент по направлению x при использовании ядра S_X:

$$S_X = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}. \tag{4.13}$$

В таком случае результирующее изображение будет иметь яркие вертикальные линии в местах меньшего изменения интенсивности.

Для определения горизонтальных линий используется S_Y:

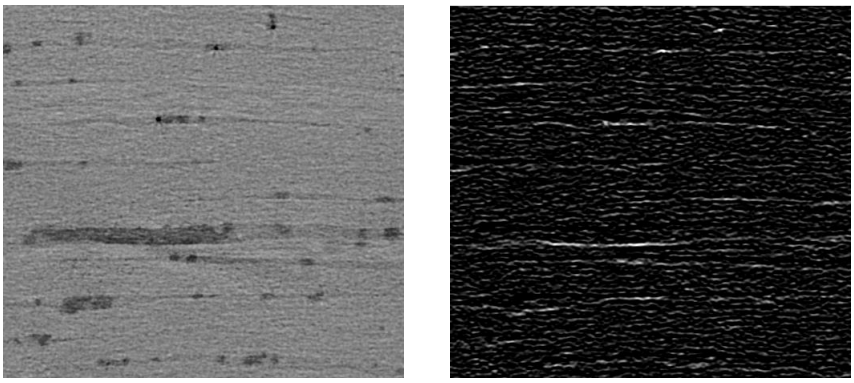
$$S_Y = \begin{pmatrix} -1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & 2 & 1 \end{pmatrix}. \qquad (4.14)$$

Если же требуется найти сам градиент, то можно скомбинировать результат применения обоих ядер S_X, S_Y в один:

$$G_X = S_X * I, \quad G_Y = S_Y * I, \quad G = \sqrt{G_X^2 + G_Y^2}. \qquad (4.15)$$

Стоит заметить, что операция извлечения корня при вычислении G выполняется поэлементно.

В случае поиска примерно горизонтальных линий границ слоев используется ядро S_Y.

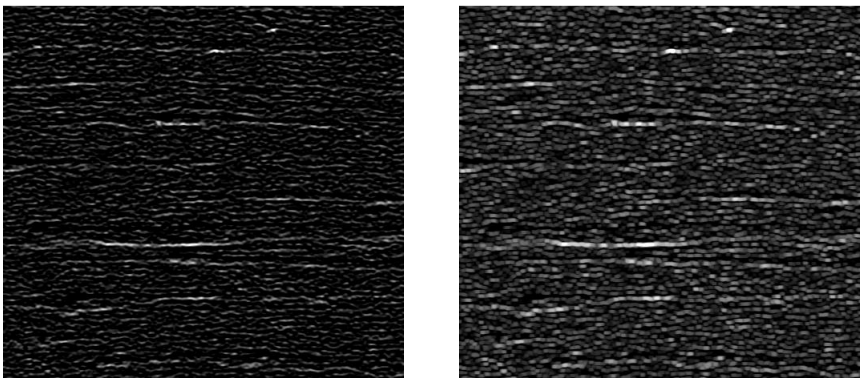


а б

Рис. 5.1. Результат применения фильтра Собеля

5.3. К отфильтрованному изображению (рис. 5.2, а) применяется операция наращивания с квадратным структурным элементом размера 3 (рис. 5.2, б):

$$Seg^{(3)} = dilate(Seg^{(2)}, Sqr(3)). \qquad (4.16)$$

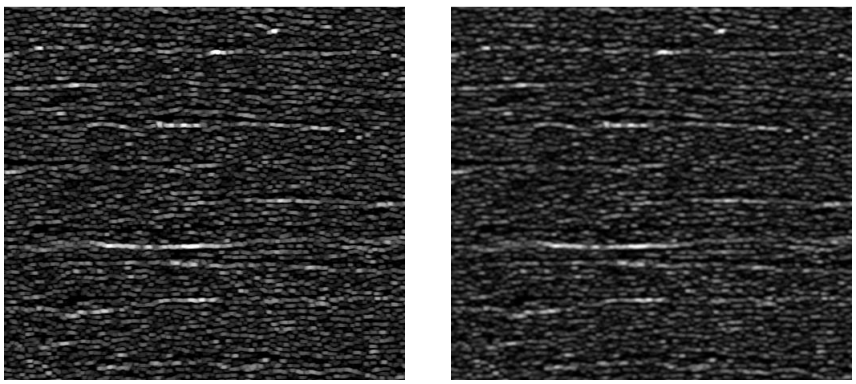


а б

Рис. 5.2. Результат применения операции наращивания

5.4. К текущему изображению (рис. 5.3, а) применяется размытие по Гауссу с размером окна 5 (рис. 5.3, б). Размытие по Гауссу производится по вычислениям, аналогичным адаптивной бинаризации в извлечении включений:

$$\begin{aligned} Seg_{x,y}^{(4)} &= \sum_{(x_0,y_0)\in\Omega(x,y)} Seg_{x,y}^{(3)} G_{\sigma_4}(\Delta x), \\ \Delta x &= \sqrt{(x_0 - x)^2 + (y_0 - y)^2}. \end{aligned} \qquad (4.17)$$



а б

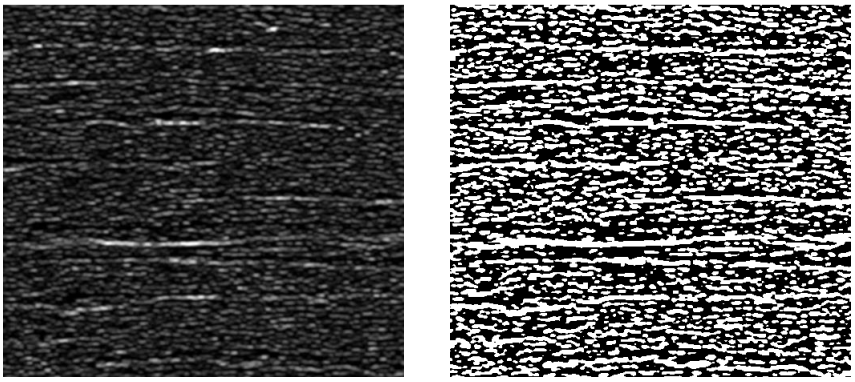
Рис. 5.3. Результат применения операции размытия по Гауссу

5.5. К текущему изображению (рис. 5.4, а) применяется адаптивная бинаризация с пределом, вычисляемым гауссовым ядром с размером окна $k = 101$ (рис. 5.4,б):

$$Seg_{x,y}^{(5)} = \begin{cases} m, & Seg_{x,y}^{(4)} > T_{x,y}, \\ 0, & Seg_{x,y}^{(4)} \le T_{x,y}, \end{cases}$$

$$T_{x,y} = \sum_{(x_0,y_0)\in\Omega(x,y)} Seg_{x,y}^{(4)} G_{\sigma_4}(\Delta x), \qquad (4.18)$$

$$\Delta x = \sqrt{(x_0 - x)^2 + (y_0 - y)^2}.$$

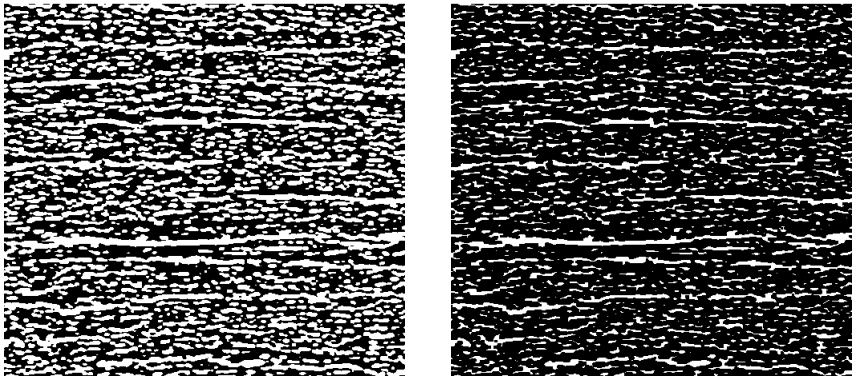


а б

Рис. 5.4. Результат применения адаптивной бинаризации

5.6. К текущему изображению (рис. 5.5, а) применяется операция эрозии с квадратным структурным элементом размера 3 (рис. 5.5, б):

$$Seg^{(6)} = dilate(Seg^{(5)}, Sqr(3)). \qquad (4.19)$$



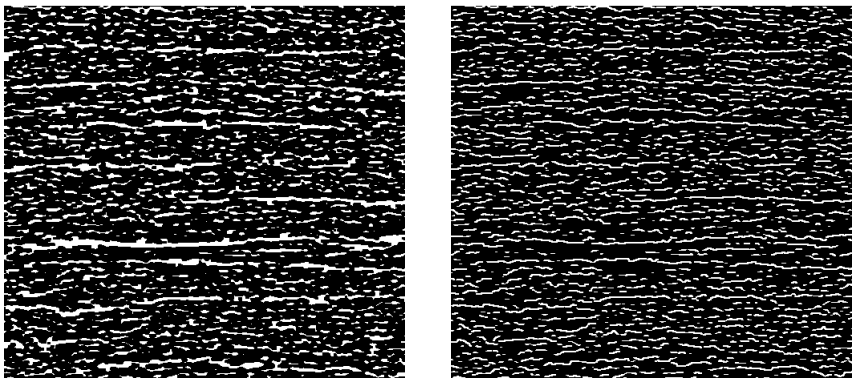
а б

Рис. 5.5. Результат применения эрозии

5.7. К текущему изображению (рис. 5.6, а) применяется фильтр Собеля (рис. 5.6, б):

$$Seg^{(7)} = S * Seg^{(6)}. \qquad (4.20)$$

Заметим, что полученные линии на шаге 5.5 могут быть достаточно большими и размазанными, а также могут присутствовать лишние линии шума. Поэтому используется пара операций 5.6, 5.7, которая уменьшает количество вкраплений от шума и уточняет положение наиболее горизонтальных линий.



а б

Рис. 5.6. Результат повторного применения фильтра Собеля

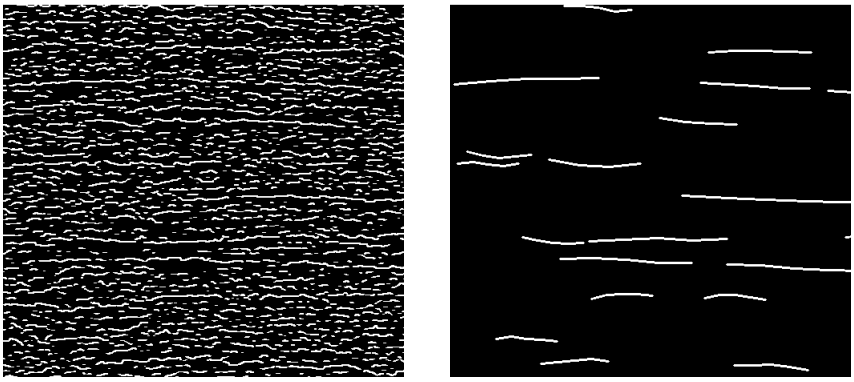
5.8. Производится поиск контуров при помощи алгоритма Сузуки [17] на текущем изображении (рис. 5.7, а).

5.9. Полученные контуры преобразуются в их средние линии. Для этого точки контура из верхней и нижней половин усредняются по y и попарно сливаются, получая примерную среднюю линию контура, которая аппроксимирует исходный сегмент границы слоя.

5.10. У средней линии отсекаются все сегменты, длина которых меньше некоторого заданного предела, а оставшиеся фрагменты соединяются в ломаную. Данная операция снижает вариацию полученных контуров при запусках алгоритма на схожих изображениях и, как следствие, увеличивает точность дальнейших шагов.

5.11. Полученные линии наносятся на отдельное изображение, к которому далее применяется наращивание с квадратным структурным элементом размера 3 (рис. 5.7, б). Эта операция сливает близлежащие контуры в один соединенный контур.

5.12. Предыдущие три шага повторяются два раза, причем предел $thresh$ длины сегмента контура и количество применений операции $iter$ наращивания различаются от итерации к итерации. В данной работе используются значения $thresh = [60, 75]$ и $iter = [1, 2]$.



а б

Рис. 5.7. Найденные сегменты границ монослоев

5.13. У полученных ломаных, интерполирующих средние линии исходных контуров, добавляются дополнительные точки. Данный шаг происходит по следующему алгоритму.

5.13.1. Задается рациональное количество сегментов s_{opt} в ломаной и "взвешенный кумулятивный угол" $A = 0$.

5.13.2. Вычисляется длина l сегмента ломаной.

5.13.3. На сегменте строятся дополнительные точки в количестве $N = \left[\frac{l}{l_{opt}} \right]$, где $l_{opt} = \frac{100}{s_{opt}}$ – рациональная длина сегмента ломаной. Точки размещаются равномерно между началом и концом сегмента.

5.13.4. "Взвешенный кумулятивный угол" суммируется со "взвешенным углом" обрабатываемого сегмента:

$$A := A + \operatorname{arctg}\left(\frac{\Delta y}{\Delta x}\right) \min\left(1, \frac{l}{l_{opt}}\right), \qquad (4.21)$$

где $\Delta x, \Delta y$ – разница в координатах конца и начала сегмента ломаной.

5.13.5. Шаги 5.13.2 – 5.13.4 реализуются для каждого сегмента каждой ломаной.

5.13.6. Вычисляется "взвешенный средний угол":

$$A_{avg} = \frac{A}{N_{seg}},$$

где N_{seg} – суммарное количество сегментов всех ломаных.

Такой “взвешенный” расчет среднего угла помогает исключить эффект сильно осциллирующих коротких линий, которые могли быть ошибочно извлечены из изображения. В данной работе рациональное количество сегментов было выбрано в количестве $s_{opt} = 4$.

После выполнения данного алгоритма полученная информация о сегментах границ содержится в множестве Seg. Извлеченные точки изображены на рис. 5.8.

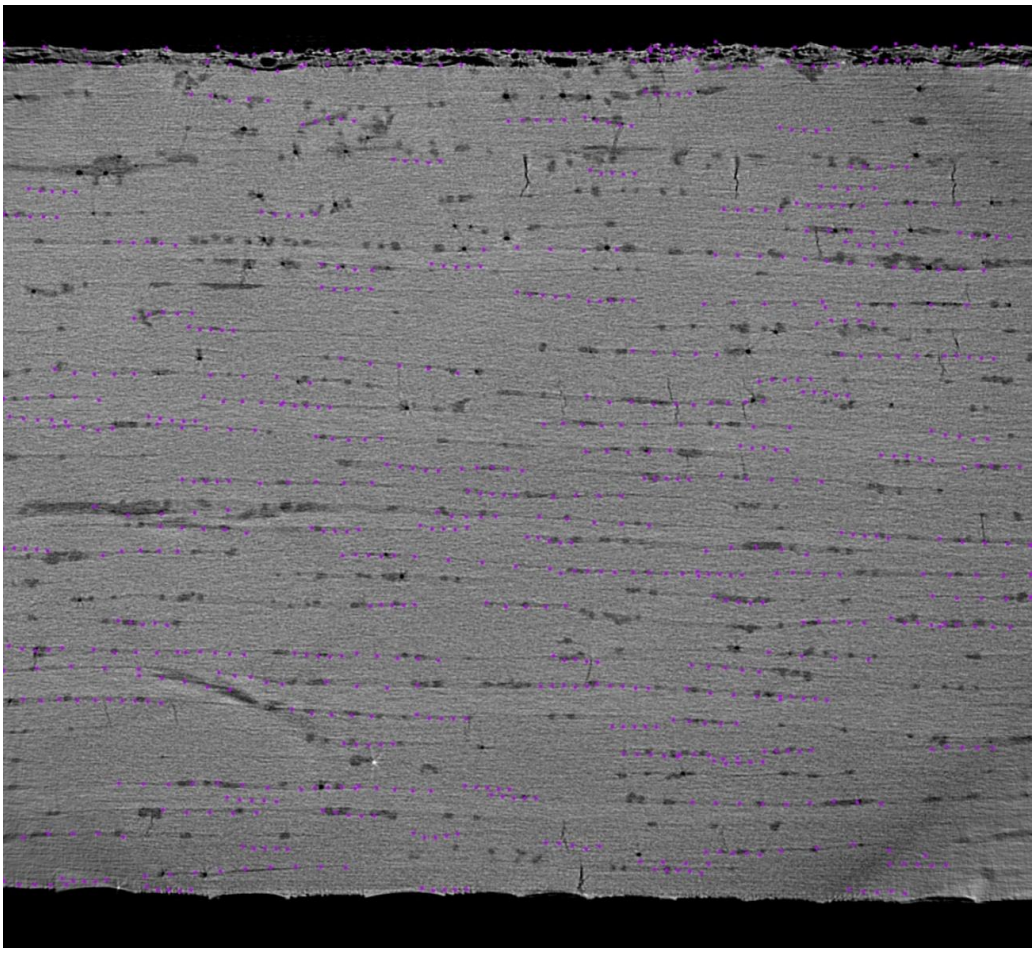


Рис. 5.8. Извлеченные точки сегментов границ монослоев

6. Добавление информации о трещинах, суммирование данных одного изображения и их обработка

6.1. Информация из алгоритмов поиска включений и сегментов границ суммируется в один массив точек. Для уменьшения информации и увеличения качества набора данных применяется слияние точек, которые расположены ближе дистанции слияния $mThresh$ (в работе используется значение $mThresh = 30$):

$$data^{(1)} = Incs \bigcup Seg. \tag{5.1}$$

6.2. К уже существующему массиву добавляется информация о результатах поиска трещин. Для этого используется алгоритм, описанный в [16].

Происходит обработка новой информации, т.е. по каждому элементу $bbox = (l, t, w, h)$ строятся две дополнительные точки:

$$\begin{aligned} &data^{(2)} = data^{(1)} \bigcup_{n=0}^{N^{(cracks)}} \{(c_u{}^{(n)}, c_b{}^{(n)})\}, \\ &c_u{}^{(n)} = \left(l^{(n)} + \frac{w^{(n)}}{2}, t^{(n)} \right), \quad c_b = \left(l^{(n)} + \frac{w^{(n)}}{2}, t^{(n)} + h^{(n)} \right), \\ &(l^{(n)}, t^{(n)}, w^{(n)}, h^{(n)}) \in Cracks, \end{aligned} \tag{5.2}$$

где $Cracks$ – множество трещин, $N^{(cracks)}$ – количество трещин.

Здесь добавлены точки верхней и нижней сторон прямоугольника. Данные точки важны для дальнейшей работы алгоритма, так как трещины проходят строго внутри одного монослоя. Это значит, что данные точки обязательно являются частью сегмента двух границ монослоя. Однако могут произойти ситуации, когда две трещины соединяются в одной точке, ввиду чего могут определиться как одна. Из-за

данной сложности в этом алгоритме для добавления точек используются лишь трещины с высокой “уверенностью” алгоритма, т.е. трещины, которые не сливаются в одной точке границы.

Суммарная извлеченная информация отражена на рис. 6.1.

6.3. Проводится еще одно слияние точек с дистанцией слияния 15, чтобы в дальнейшем улучшить устойчивость алгоритма и уменьшить количество обрабатываемой информации (рис. 6.2).

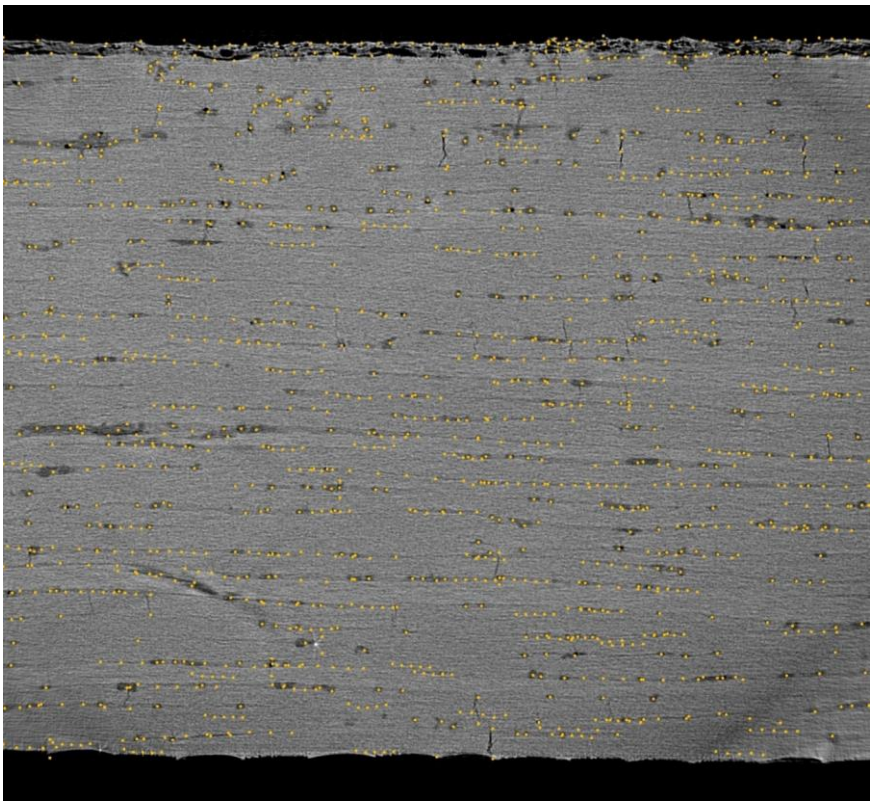


Рис. 6.1. Информация, сложенная из алгоритмов поиска включений, сегментов границ монослоев и трещин

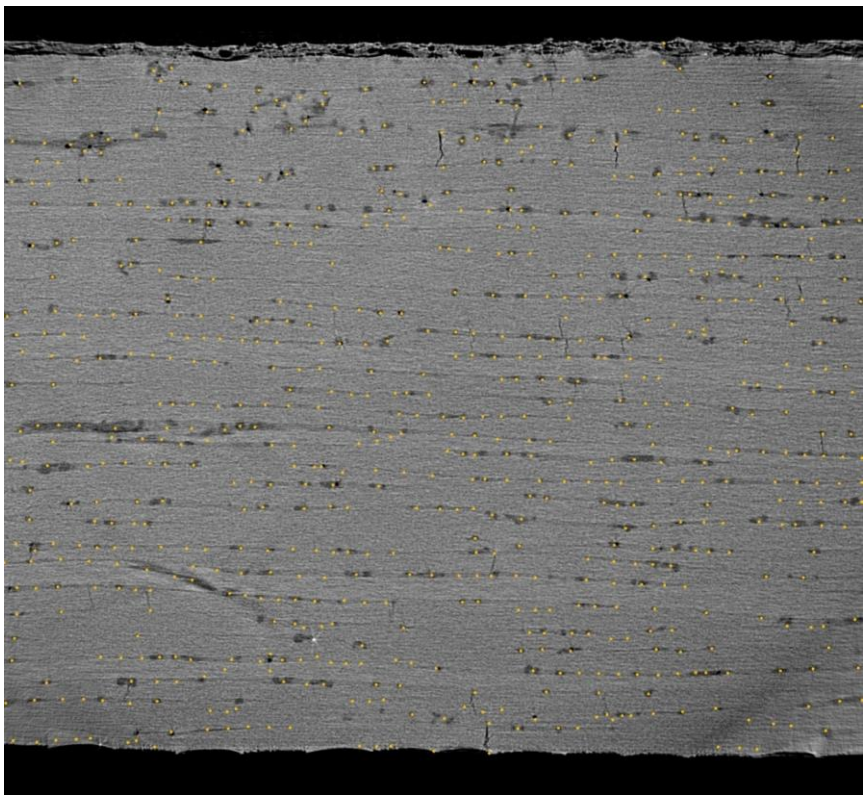


Рис. 6.2. Извлеченная информация после повторного проведения слияния точек

6.4. К каждому элементу массива точек добавляется параметр состояния точки. Данный параметр может принимать значения -1 (точка не отмечена), 1 (отмечена) и 0 (пропущена). Точка находится в состоянии “не отмечена” только до запуска алгоритма построения границ и при каждой итерации алгоритма поиска пропущенных границ. Точка принимает состояние “отмечена”, если она является частью построенной границы. Состояние “пропущена” задается точке, если она не является частью построенных границ и также не может быть элементом новой границы без пересечений с предыдущими:

$$data^{(3)} = \{(data^{(2)}_i, -1)\}_{i=\overline{0,N^{(data)}}}, \qquad (5.3)$$

где $N^{(data)}$ – количество точек в массиве информации.

7. Усреднение информации пакета изображений

Если алгоритму подается лишь одно изображение, то алгоритм может дать значительно худший результат, нежели с пакетом последовательных изображений. При обработке пакета обеспечивается некоторый уровень стабильности, чтобы при запусках на схожих наборах данных на выходе получались примерно такие же границы монослоев.

Значительная проблема при обработке пакета изображений – неполное извлечение информации из входных данных. Небольшие вариации в освещении или помехи на картинке могут значительно уменьшить количество и качество извлеченных точек и их координат соответственно. Для решения такой проблемы используется смешивание данных из соседних изображений.

Суть данного алгоритма заключается в дополнении массива точек, извлеченных из изображения, информацией из соседнего изображения. При малом расстоянии между сканами образца точки ближайших срезов будут отражать примерно одни и те же включения или фрагменты границ. Смешивать можно информацию из любого количества изображений, но лучше всего ограничиться ближайшими несколькими изображениями. В данной работе используется только одно ближайшее изображение с каждой стороны от обрабатываемого.

После дополнения массивов возможно скопление большого количества точек в одном месте, что увеличит время обработки и может послужить лишним шумом в данных. Ввиду этого используется слияние по расстоянию $mThresh$, чтобы усреднить характер таких кластеров одной центральной точкой (рис. 7.1). В данной работе используется расстояние слияния $mThresh = 10$.

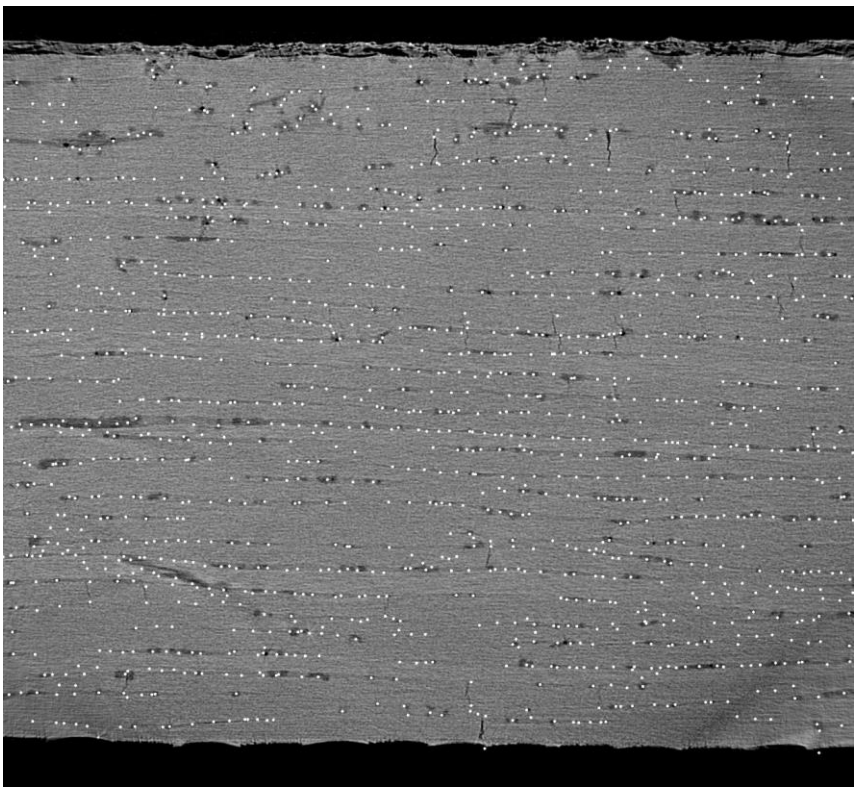


Рис. 7.1. Дополненная и усредненная информация

8. Первоначальное построение границ

Суть алгоритма заключается в выборе точек вдоль области поисковой прямой с углом, равным среднему по линиям из контуров, полученных фильтром Собеля, и в дальнейшем соединении точек наиболее прямой линией, задающей границу монослоя. Соединение точек происходит за счет слияния наиболее близких точек поисковой прямой с реальными. Если количество закрепленных таким образом точек меньше некоторого предела, то угол поисковой линии начинает варьироваться в небольшой области среднего, после чего производится еще одна попытка построить границу. В случае неудачи построения после варьирования угла происходит сдвиг всей прямой на шаг вниз по изображению и производится повторение всего алгоритма. Если же построенная граница удовлетворяет всем критериям, то производится ее сглаживание, чтобы слить большее число точек прямой с реальными

данными и уменьшить осцилляции в ломаной, характеризующей границу, а точки, оказавшиеся над границей, помечаются как пропущенные и игнорируются в построении дальнейших границ.

Перед выполнением алгоритма производится инициализация всех нужных параметров: количество монослоев m, номинальная толщина слоя w_{nom}, разрешение r по x, шаг поиска Δy, порог закрепления границы по количеству слитых точек t_s и по длине покрытия ширины изображения t_w. В работе использовались значения $m = 32,\ w_{nom} = 50,\ r = 20,\ \Delta y = 5,\ t_s = 0.75,\quad t_w = 0.5$.

Номинальная толщина слоя (в пикселях) позволяет сделать оценку максимального размаха границы монослоя. Эвристически было определено, что в среднем толщина монослоя варьируется от $0.15w_{nom}$ до $1.5w_{nom}$. Важным здесь является максимальное значение толщины, которое берется за расстояние поиска при закреплении точек за реальными данными при построении границы.

Разрешение r означает количество точек в поисковой прямой, которые будут проверяться на слияние с реальными точками. Большое количество может привести к малому числу успешно построенных границ, а слишком малое количество влечет собой плохую точность построения границ и неравномерность распределения границ по изображению.

Шаг поиска Δy – количество пикселей, на которое сдвигается поисковая линия после неудачи при построении границ с каждым допустимым углом на предыдущем значении y.

Величины t_s и t_w – проценты количества точек поисковой прямой и ширины изображения, которые являются порогами для построения границы: если у построенной ломаной количество закрепленных точек $\left|p_{snapped}\right| \geq rt_s$ или суммарная длина сегментов, образованных закрепленными точками $l_{snapped} \geq t_w W$, где W – ширина изображения, то граница считается успешно построенной.

Алгоритм работает циклично m раз, чтобы построить соответствующее количество границ в лучшем случае. Если количество построенных границ меньше m, то после выполнения данных действий производится выполнение алгоритма поиска пропусков.

8.1. Происходит выбор подходящих для построения границы точек. Для этого массив всех точек p на изображении фильтруется по близости от поисковой линии $x + \mathrm{tg}\left(A_{avg}\right) y + y_{cur} = 0$, где y_{cur} есть величина y поиска на данной итерации:

$$\frac{\left|\mathrm{tg}\left(A_{avg}\right) p_x - p_y + y_{cur}\right|}{\sqrt{\mathrm{tg}^2\left(A_{avg}\right)+1}} \leq 0.25 w_{nom}. \qquad (7.1)$$

Если расстояние от точки до прямой не более $0.25 w_{nom}$, то она используется в дальнейших шагах алгоритма построения границы.

8.2. Принятые точки передаются на вход алгоритму построения границы. Данный алгоритм для каждой точки поисковой прямой реализует следующие действия.

8.2.1. Находится расстояние поиска x_f:

$$x_f = \begin{cases} 0, & i = 1 \vee |b|, \\ 0.7b_{i+1_x} + 0.3b_{i-1_x}, & 1 < i < |b|, \end{cases} \tag{8.2}$$

где B – начальная аппроксимация искомой границы.

8.2.2. Происходит поиск всех допустимых точек, с которыми можно закрепить элемент прямой. Фильтрации точек по попаданию в прямоугольную область с шириной w_f и высотой h_f, центрированную на точке $(B_{i_x} + x_f, B_{i_y})$:

$$w_f = \begin{cases} b_{i+1} - b_i, & i = 1, \\ b_{i-1} - b_i, & i = |b|, \\ \min(b_{i+1} - b_i, b_{i-1} - b_i), & 1 < i < |b|, \end{cases} \tag{8.3}$$

$$h_f = \max\left(\frac{w_{nom}}{5}, \min\left(\frac{w_{nom}}{2}, \frac{w_f}{3}\right)\right).$$

8.2.3. Из всех точек выбирается та, расстояние от которой до центра поиска минимально, и элемент аппроксимации границы заменяется на этого кандидата, а состояние найденной точки в исходном массиве данных устанавливается на значении -1. Если ни одна точка не попала в область поиска, то координаты элемента границы остаются неизменными.

Пример шага работы алгоритма построения границы представлен на рис. 8.1.

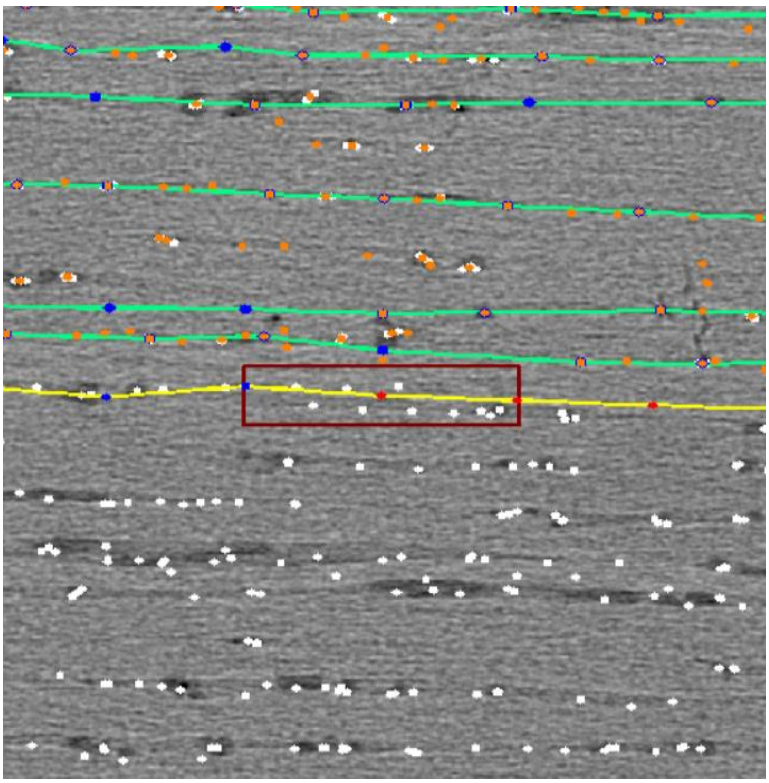


Рис. 8.1. Шаг работы алгоритма построения границ монослоя

8.3. После построения границы оценивается ее пригодность по критериям количества закрепленных точек и покрытия ширины изображения:

$$|b_s| \geq t_s, \quad L_s \geq t_w, \tag{7.2}$$

где b_s – множество всех закрепленных точек границы b; L_s – суммарная длина всех сегментов границы b, образованных закрепленными точками.

Если закрепленных точек слишком мало, то есть риск того, что граница была построена по шуму в данных, т.е. по включениям внутри слоя. Если покрытие ширины слишком мало, то есть вероятность, что граница построена недостаточно точно или пересекает монослой, из-за чего большая часть точек не нашла соответствующий аналог в массиве реальных данных. Выполнение одного из этих

критериев достаточно, чтобы граница считалась построенной верно и передалась на дальнейшую обработку.

8.4. Если граница не удовлетворяет ни одному из критериев, то происходит изменение поисковой линии и повторяется построение. Сначала варьируется угол прямой $\alpha \in \left[0.5\mathrm{tg}\left(A_{avg}\right), 1.5\mathrm{tg}\left(A_{avg}\right)\right]$. Если после вариации угла граница так и не была построена, то смещается величина y_{cur} поисковой линии на величину Δy.

8.5. Производится сглаживание границы. Для этого применяется фильтр Гаусса, но в одномерном случае для каждой координаты точек границ:

$$b_i^* = \frac{1}{W}\sum_{r=-4}^{4} G_\sigma(r)\hat{b}_{i+r}, \quad i = \overline{0,|b|}$$
$$W = \sum_{r=-4}^{4} G_\sigma(r), \qquad (7.3)$$
$$G_\sigma(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right),$$

где b – точки исходной границы, b^* – сглаженная граница, $G_\sigma(r)$ – ядро пространства, сглаживающее разницу в координатах; σ – стандартное отклонение ядра, $\hat{b}$ – бесконечно отраженный вектор точек границы b с дублированием крайних элементов (под элементами $\hat{b}$ указан их номер ind):

$$\hat{b} = \left\{ \underset{ind:}{\ldots} \underset{-(n+2)}{b_{n-1}}, \underset{-(n+1)}{b_n}, \underset{-n}{b_n}, \ldots, \underset{-1}{b_0}, \underset{0}{b_0}, \underset{1}{b_1}, \ldots, \underset{n-1}{b_{n-1}}, \underset{n}{b_n}, \underset{n+1}{b_n}, \underset{n+2}{b_{n-1}}, \ldots, \underset{2n}{b_0}, \underset{2n+1}{b_0}, \underset{2n+2}{b_1}, \ldots \right\}. \qquad (7.4)$$

Сглаживание делает полученную кривую более похожей на искомую границу, а также захватывает меньше точек, которые могли бы использоваться для построения новых линий.

8.6. Начинается поиск новой границы, если их число меньше m, иначе алгоритм заканчивает работу, построив при этом m границ, что представлено на рис. 8.2.

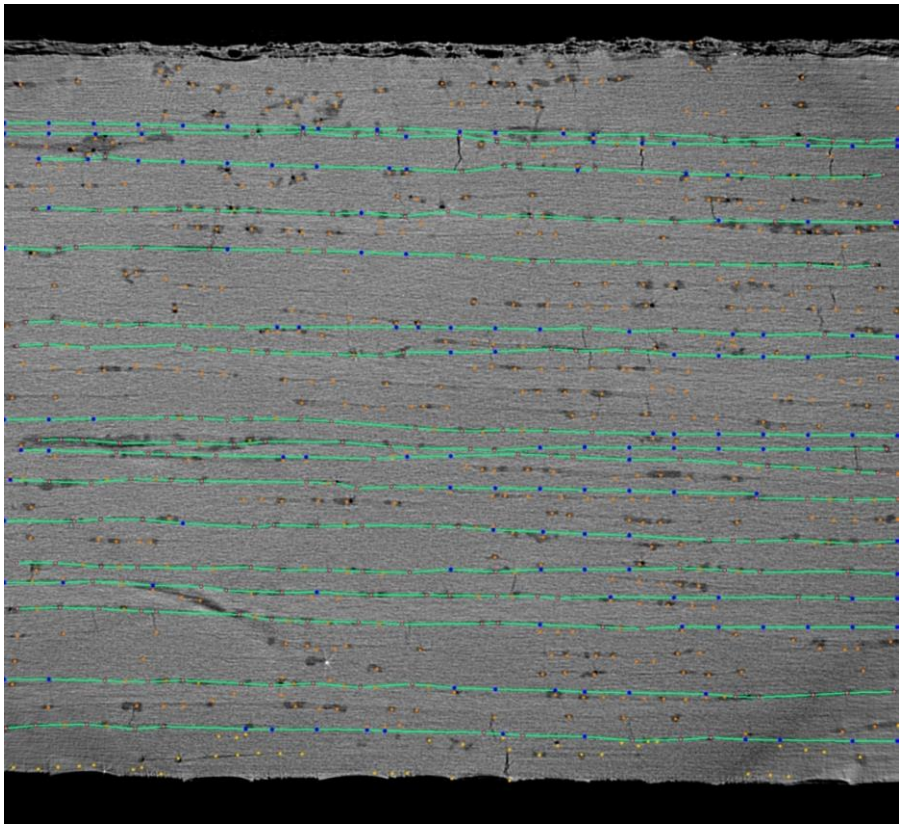


Рис. 8.2. Построенные границы монослоев после одного цикла работы алгоритма их построения

8.7. Производится удаление лишних границ в местах высокой скученности. Для этого находится среднее расстояние Δ_{avg} между границами, разделенными на один шаг. Для этого вычисляется площадь ΔS_j многоугольника, образованного границами с номерами $j-1$ и $j+1$. Если окажется, что эта площадь меньше установленного порога $S_{dense} = Wh_{dense}$, то граница $b^{(j)*}$ удаляется из множества.

В данной работе используется значение порога $h_{dense} = 0.8w_{nom}$.

8.8. Производится поиск пропущенных границ. Для этого аналогичным образом находится площадь многоугольника, образованного соседними границами. Если она больше установленного порога $S_{gap} = Wh_{gap}$, то номера этих границ записываются в множество пропусков.

В данной работе используется значение порога $h_{gap} = 1.2w_{nom}$.

Полученные данные о пропусках подаются на вход модифицированному алгоритму поиска границ. В отличие от предыдущего, данный уже имеет информацию об использованных точках и производит поиск не по всему изображению, а между линиями, где, вероятно, есть пропущенная граница. В качестве начального набора точек для построения границ используются все точки между границами с пропуском. Значения порогов принятия границы t_s и t_w выбраны равными 0.5 и 0.6 соответственно, что увеличивает надежность нахождения границы. Помимо этого, расстояние до поисковой линии, на котором точки считаются пригодными для построения границы, увеличено с $0.25w_{nom}$ до $0.35w_{nom}$. В остальном алгоритм идентичен оригинальному.

Результат работы (рис. 8.3) – дополненный набор границ, который лучше отражает реальную структуру слоев исследуемого материала.

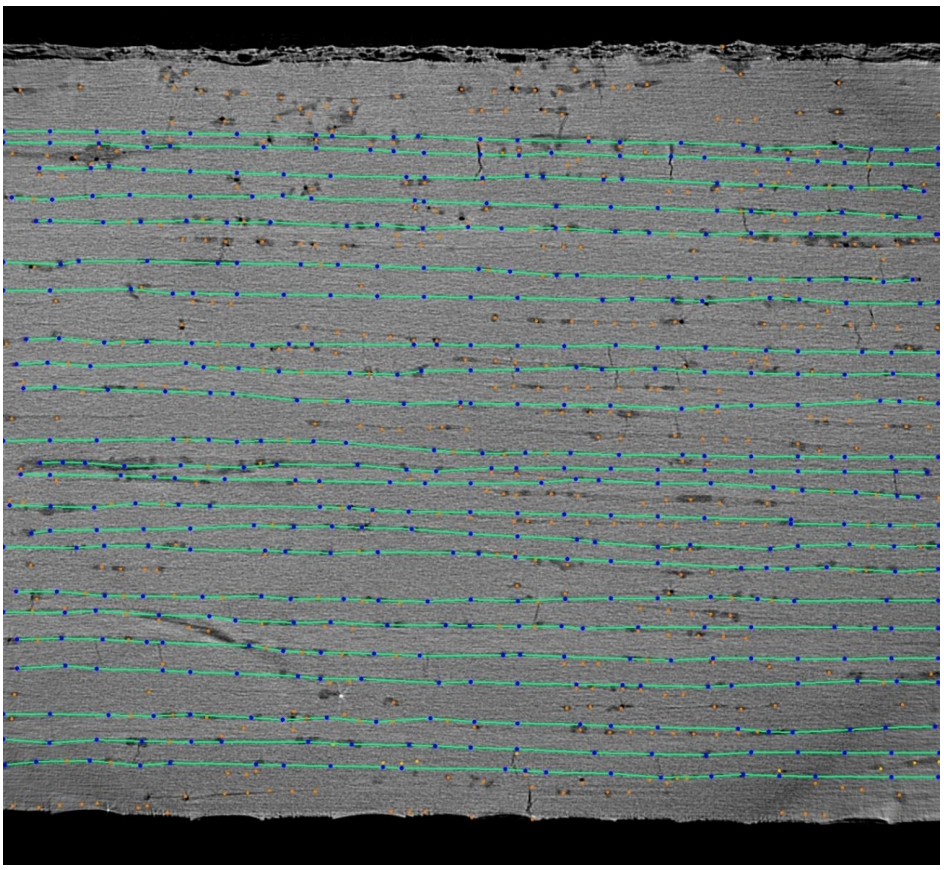


Рис. 8.3. Построенные границы монослоев после работы алгоритма выравнивания плотности границ

9. Алгоритм уточнения границ

Алгоритм построения границ позволяет получить наиболее полную картину границ монослоев для подаваемого на вход изображения. Однако обычно прогонки алгоритма на одном изображении, даже с комбинацией информации из ближайших изображений, недостаточно. Чаще всего полученное количество границ сильно меньше, чем установлено заранее. Ввиду этого требуется дополнительный алгоритм усреднения информации о границах. Этот алгоритм реализуется поиском наиболее полной начальной аппроксимации и повторением алгоритма выравнивания плотности границ, ориентируясь на точки каждой границы из каждого набора слоев пакета изображений, кроме усредняемого.

9.1. Для каждого изображения вычисляется количество найденных слоев. Среди всех выбирается усредняемый набор границ $\bar{B}$, у которого количество найденных слоев ближе всего к заданному заранее номинальному числу слоев m.

9.2. Формируется глобальный массив точек границ P_b. В него включаются точки каждой границы из каждого набора, кроме $\bar{B}$. Точки сливаются по расстоянию, чтобы уменьшить нагрузку на алгоритм и уменьшить шум в данных.

В данной работе было выбрано значение расстояния слияния $mThresh = 25$.

9.3. Далее запускается алгоритм выравнивания плотности границ с теми же параметрами h_{dense}, h_{gap}, что и при первичной обработке.

Финальный набор границ изображен на рис. 9.1.

10. Описание программной реализации

Алгоритм поиска монослоев был исполнен на языке Python 3.12 в среде программирования PyCharm.

Основные используемые библиотеки:

- NumPy 2.0.2 [17]
- OpenCV 4.10 [18]
- Shapely 2.0.7 [19]
- SciPy 1.15.2 [20]

Общие принципы создания соответствуют идеям из [21].

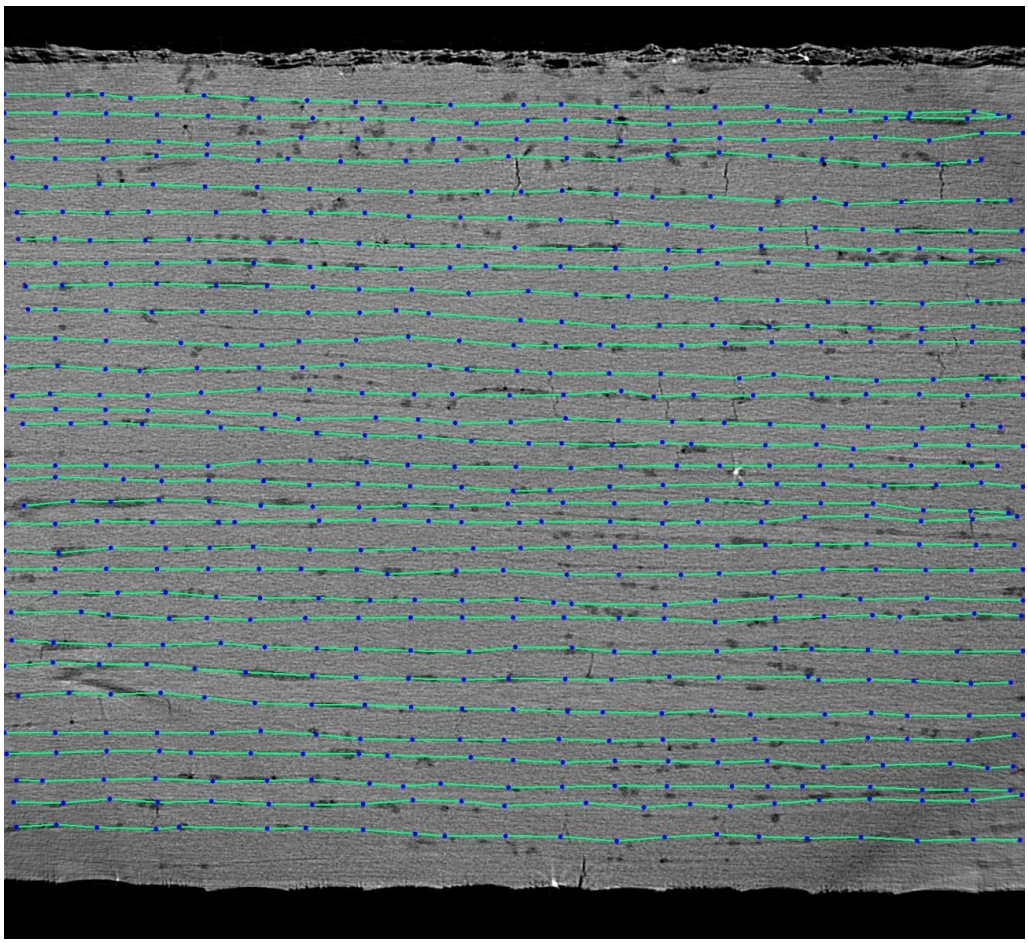


Рис. 9.1. Итоговые построенные границы монослоев после их уточнения

На вход алгоритму подается либо отдельное изображение, которое требуется обработать, либо массив сканов. В результате работы алгоритма программа дает информацию о границах монослоев в форме списка массивов точек, представляемых параметрами координат и их закрепленность.

Основные шаги работы программы:

10.1. *getMonolayers*

Основная функция программы. Принимает на вход все гиперпараметры, нужные для работы дальнейших блоков, и выполняет вызовы дальнейших модулей программы и дополнительные действия для сложения информации из пакета входных изображений.

Входные параметры:

- imgs – входной пакет изображений;
- upCrop, downCrop – обрезка изображений сверху и снизу соответственно;
- targetImgInd – целевое изображение, для которого стоит построить границы монослоев;
- targetMonoCount, targetMonoWidth – номинальное количество монослоев и их ширина.

Возвращаемые параметры:

- borders, count – найденные границы и их число.

10.2. *buildMonolayers*

Данный блок кода выполняет алгоритмы поиска точек и построения границ монослоев для каждого изображения отдельно.

Входные параметры:

- img – обрабатываемое изображение;
- upCrop, downCrop, targetBorderCount, targetMonoWidth – аналогично getMonolayers.

Возвращаемые параметры:

- borders, count – найденные границы и их число.

10.3. *getPoints*

Эта функция извлекает все нужные точки для работы алгоритма построения монослоев. Извлекаются точки по включениям в материале, наиболее надежным

фрагментам границ и трещинам, которые наиболее вероятно являются верно определенными.

Входные параметры:

- img – обрабатываемое изображение.

Возвращаемые параметры:

- points – все извлеченные из изображения точки;
- avgAngle – примерный средний угол наклона границ монослоев.

10.4. *findBorder*

Эта функция отвечает за сам поиск и построение первичной аппроксимации всех границ монослоев.

Входные параметры:

- points – обрабатываемые точки;
- targetMonoWidth, avgAngle, upCrop, downCrop – аналогично предыдущим функциям;
- pointDistCoef – множитель ширины области поиска кандидатов на построение границы;
- xRes – разрешение границы;
- xResThresh – пороговое количество закрепленных точек границы, после которого она считается успешно построенной;
- imgW – ширина обрабатываемого изображения;
- snapThresh – пороговая длина закрепленных отрезков границы, после которого она считается успешно построенной.

Выходные параметры:

- newBorder – построенная граница;
- OOB – флаг выхода границы за пределы рассматриваемой области изображения.

10.5. *borderFromPoints*

Ключевой блок, выполняющий сам алгоритм построения границы по выбранным точкам.

Входные параметры:

- chosenPoints – точки, по которым выполняется построение границы;
- points – набор всех точек, извлеченных из изображения;
- baseSearchThresh – базовая область поиска новой точки границы;
- targetMonoWidth – аналогично предыдущим функциям.

Возвращаемые параметры:

- borderPoints – точки построенной границы;
- snapPointIndicesSet – множество индексов точек, на которых была закреплена граница.

10.6. *densityEvenOut*

Вспомогательная функция, которая реализует алгоритм удаления вариации плотности границ. Она ищет промежутки, в которых могут быть пропущенные границы, и удаляет лишние в местах высокой плотности.

Входные параметры:

- borders – обрабатываемые границы;

- points, targetMonoCount, targetMonoWidth, imgW – аналогично предыдущим функциям;
- thicknessReduce, thicknessAdd – пороговая ширина монослоя, в которой удаляются лишние или ищутся пропущенные границы соответственно.

Возвращаемые параметры:

- fixedBorders – улучшенные границы.

10.7. *mergeFromImages*

Последний крупный блок обработки границ. Если входные данные – пакет изображений, то данная функция совместит всю полученную информацию о слоях в один набор границ, в котором будет храниться усредненный характер расположения монослоев.

Входные параметры:

- imgBorders – пакет наборов границ каждого изображения;
- targetInd, targetMonoCount, targetMonoWidth – аналогично предыдущим функциям.

Выходные параметры:

- targetBorders, targetBordersCount – итоговые границы для выбранного изображения и их количество.

Упрощенная схема работы программы представлена на рис. 10.1.

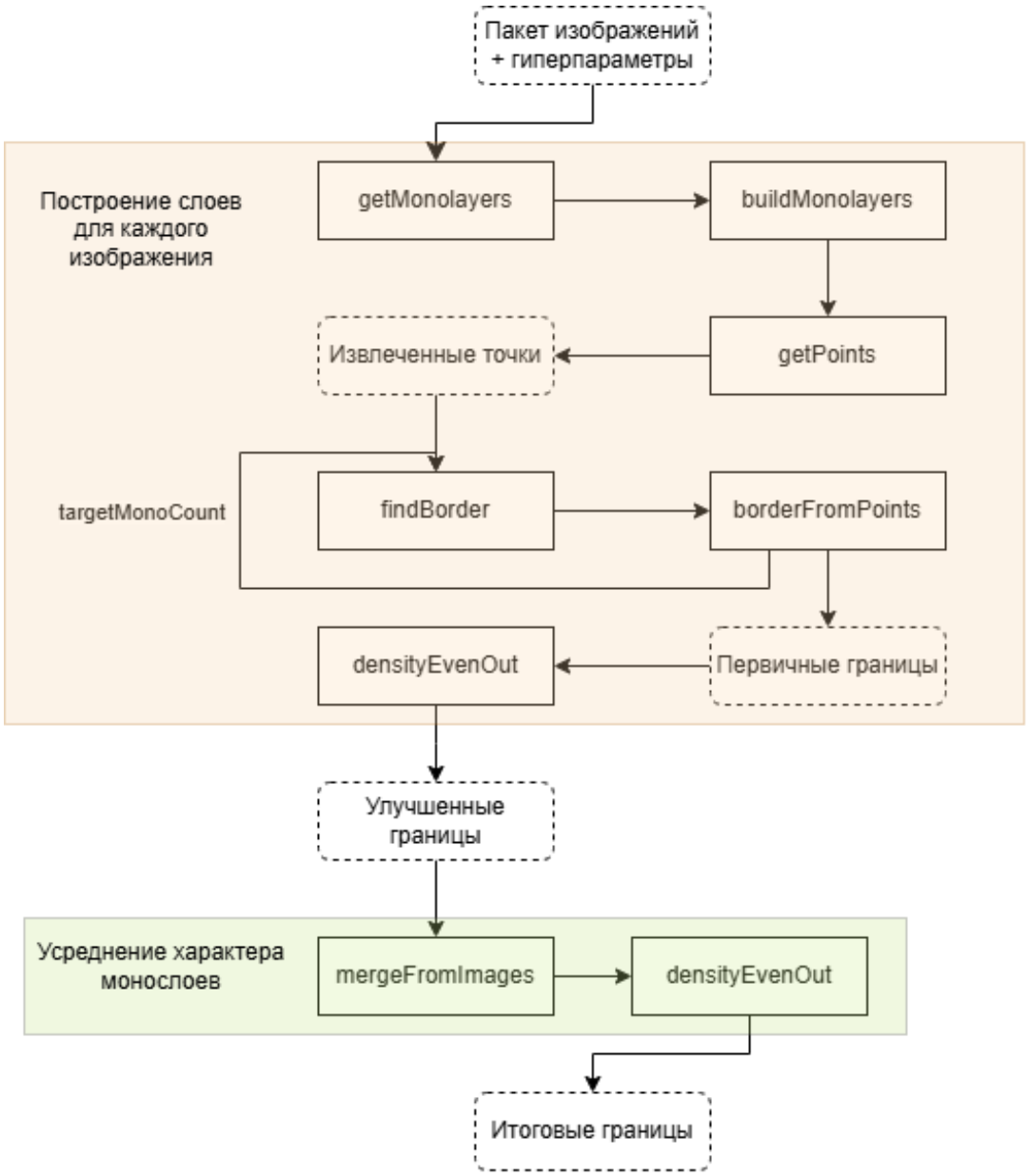


Рис. 10.1. Упрощенная схема работы программы, реализующей алгоритм поиска монослоев

11. Заключение

В статье предложен пошаговый алгоритм и описано созданное программное обеспечение на языке Python 3.12 в среде программирования PyCharm, предназначенное для автоматического определения границ монослоев в слоистых композитных материалах на основе

последовательности томографических изображений. Сформированное алгоритмическое и программное обеспечение позволяет решить задачу анализа микроструктуры композитов для последующего использования полученных данных в расчетных моделях.

Список источников

1. Hsien J. Computed tomography: principles, design, artifacts and recent advances.- SPIE Press, 2009. 387 p.

2. Banhart J. Advanced tomographic methods in materials research and engineering.- Oxford University Press, 2008. 462 p.

3. Maire E., Withers P. Quantitative X-ray tomography. International Materials Reviews, 2014. Vol. 59. No. 1. P. 1-43.

4. Georgantzinos S.K. Characterization and Modelling of Composites, 1st ed.; MDPI: Basel, Switzerland, 2022.

5. Brandon D., Kaplan W.D. Microstructural characterization of materials.- John Wiley ans Sons Ltd, 2013. 560 p.

6. Naresh K., Khan K.A. , Umer R., Cantwell W.J. The use of X-ray computed tomography for design and process modeling of aerospace composites: a review. Materials & Design, 2020. Vol. 190. No. 2. p.108553. https://doi.org/10.1016/j.matdes.2020.108553

7.Yu B. et al. A comparison of different approaches for imaging cracks in composites by X-ray microtomography. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 2016. Vol. 374. No. 2071. p. 20160037.

8. Heinzl C., Amirkhanov A., Kastner J. Processing, analysis and visualization of CT data. Industrial X-Ray Computed Tomography. 2018. P. 99–142.

9. Chen C-T., Gu G. Machine learning for composite materials. MRS Communications. 2019. V. 9. No. 2. P. 1–11. DOI:10.1557/mrc.2019.32

10. Шапиро Л., Стокман Дж. Компьютерное зрение: Пер. с англ. – 3-е изд. – М.: БИНОМ. Лаборатория знаний, 2015. – 763 с.

11. Гонсалес Р., Вудс Р. Цифровая обработка изображений: Пер. с англ. – 3-е изд. – М.: Техносфера, 2024. – 1104 с.

12. Suzuki S., K. Abe. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing. 1985. V.30. No.1. P. 32–46.

13. Крупенин А. М. Новый метод построения кинетической диаграммы по испытаниям на скорость роста трещины усталости // Труды МАИ, 2024, №136, http://mai.ru//upload/iblock/7b3/opvl5vr4421hcpqa9g5s7yj5pgurbsyr/04_Krupenin.pdf

14. Хомяков О.О., Панищев В.С., Титов В.С., Ватутин Э.И. Математическая модель и параллельный алгоритм обработки изображений, содержащих символьную информацию // Труды МАИ. 2024. № 137. URL: https://trudymai.ru/published.php?ID=181884

15. Трусфус М.В., Абдуллин И.Н. Алгоритм обнаружения маркерных изображений для вертикальной посадки беспилотного летательного аппарата // Труды МАИ. Выпуск № 116. http://trudymai.ru/ УДК 004.932 DOI: 10.34759/trd-2021-116-13

16. Пантелеев А.В., Турбин Н.В., Тучков Н.А., Талья Р.Л., Ахмедов И.А. Методика количественной оценки степени растрескивания слоистого композита по данным

компьютерной томографии // Труды МАИ. 2025. №143. URL: https://trudymai.ru/eng/published. php?ID=185657

17. NumPy documentation URL: https://numpy.org/doc/ (дата обращения: 19.07.2025)

18. OpenCV: OpenCV modules URL: https://docs.opencv.org/4.x/ (дата обращения: 20.07.2025)

19. Shapely // Shapely 2.1.1 documentation URL: https://shapely.readthedocs.io/en/stable/ (дата обращения: 24.07.2025).

20. SciPy // SciPy documentation URL: https://docs.scipy.org/doc/scipy/ (дата обращения: 29.07.2025).

21. Szeliski R. Computer Vision: Algorithms and Applications, 2nd ed. 2022, Springer: New York. 925 p. https://doi.org/10.1007/978-3-030-34372-9

References

1. Hsien J. Computed tomography: principles, design, artifacts and recent advances.- SPIE Press, 2009. 387 p.

2. Banhart J. Advanced tomographic methods in materials research and engineering.- Oxford University Press, 2008. 462 p.

3. Maire E., Withers P. Quantitative X-ray tomography. International Materials Reviews, 2014. Vol. 59. No. 1. P. 1-43.

4. Georgantzinos S.K. Characterization and Modelling of Composites, 1st ed.; MDPI: Basel, Switzerland, 2022.

5. Brandon D., Kaplan W.D. Microstructural characterization of materials.- John Wiley ans Sons Ltd, 2013. 560 p.

6. Naresh K., Khan K.A. , Umer R., Cantwell W.J. The use of X-ray computed tomography for design and process modeling of aerospace composites: a review. Materials & Design, 2020. Vol. 190. No. 2. p.108553. https://doi.org/10.1016/j.matdes.2020.108553

7. Yu B. et al. A comparison of different approaches for imaging cracks in composites by X-ray microtomography. Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences. 2016. Vol. 374. No. 2071. p. 20160037.

8. Heinzl C., Amirkhanov A., Kastner J. Processing, analysis and visualization of CT data. Industrial X-Ray Computed Tomography. 2018. P. 99–142.

9. Chen C-T., Gu G. Machine learning for composite materials. MRS Communications. 2019. V. 9. No. 2. P. 1–11. DOI:10.1557/mrc.2019.32

10. Shapiro L., Stokman Dzh. Komp'yuternoe zrenie: Per. s angl. [Computer vision]. M.: BINOM. Laboratoriya znanij, 2015. 763 p. (In Russ.).

11. Gonsales R., Vuds R. Cifrovaya obrabotka izobrazhenij: Per. s angl. [Digital image processing]. M.: Tekhnosfera, 2024. 1104 p. (In Russ.).

12. Suzuki S., K. Abe. Topological structural analysis of digitized binary images by border following. Computer Vision, Graphics, and Image Processing. 1985. V.30. No.1. P. 32–46.

13. Krupenin A. M. New method for construction kinetic diagram by fatigue crack growth rate tests. Trudy MAI, 2024. No 136. http://mai.ru//upload/iblock/7b3/opvl5vr4421hcpqa9g5s7yj5pgurbsyr/04_Krupenin.pdf

14. Khomyakov O.O., Panishchev V.S., Titov V.S., Vatutin E.I. Mathematical model and a parallel algorithm for processing images containing symbolic information about products. Trudy MAI, 2024, no. 137. URL: https://trudymai.ru/eng/published.php?ID=181884

15. Trusfus M.V., Abdullin I.N. Marker images detection algorithm for the unmanned aerial vehicle vertical landing. Trudy MAI. 2021. No. 116. http://trudymai.ru/ УДК 004.932 DOI: 10.34759/trd-2021-116-13

16. Panteleev A.V, Turbin N.V., Tuchkov N.A., Talya R.L., Akhmedov I.A. Method for tomographic images processing for matrix cracking quantification in composite material // *Trudy MAI*. 2025. No.143. (In Russ.). URL: https://trudymai.ru/eng/published.php?ID=185657

17. NumPy documentation URL: https://numpy.org/doc/ (дата обращения: 19.07.2025)

18. OpenCV: OpenCV modules URL: https://docs.opencv.org/4.x/ (дата обращения: 20.07.2025)

19. Shapely // Shapely 2.1.1 documentation URL: https://shapely.readthedocs.io/en/stable/ (дата обращения: 24.07.2025).

20. SciPy // SciPy documentation URL: https://docs.scipy.org/doc/scipy/ (дата обращения: 29.07.2025).

21. Szeliski R. Computer Vision: Algorithms and Applications, 2nd ed. 2022, Springer: New York. 925 p. https://doi.org/10.1007/978-3-030-34372-9