Труды МАИ. 2025. № 143
Trudy MAI. 2025. No. 143. (In Russ.)

Научная статья
УДК 004.932+519.254
URL: https://trudymai.ru/published.php?ID=185657
EDN: https://www.elibrary.ru/YPLTJX

МЕТОДИКА КОЛИЧЕСТВЕННОЙ ОЦЕНКИ СТЕПЕНИ РАСТРЕСКИВАНИЯ СЛОИСТОГО КОМПОЗИТА ПО ДАННЫМ КОМПЬЮТЕРНОЙ ТОМОГРАФИИ

Андрей Владимирович Пантелеев[1], Николай Васильевич Турбин[2]✉,
Николай Александрович Тучков[3], Роман Львович Талья[4],
Исак Азизович Ахмедов[5]

[1,2,3,4,5]Московский авиационный институт (национальный исследовательский университет),
Москва, Россия
[2]turbinnv@mai.ru✉

Аннотация. В работе предложен алгоритм и программное обеспечение для автоматического обнаружения и анализа поперечных микротрещин в слоистых композитных материалах по последовательности томографических изображений. Актуальность задачи обусловлена влиянием микротрещин на эксплуатационные и механические характеристики композитов, а также сложностью их выявления традиционными методами неразрушающего контроля. Предложенный двухэтапный алгоритм включает первичный анализ отдельных томографических срезов, основанный на комбинации методов фильтрации, адаптивной и простой бинаризации, выделения границ, серии морфологических преобразований и

алгоритма Сузуки для поиска контуров. Этот этап нацелен на идентификацию потенциальных областей трещин и их отделение от межслойных включений путем анализа и "вычитания" горизонтальных паттернов из вертикальных. Вторичный анализ направлен на уточнение и отслеживание трещин по последовательности срезов. Этот этап включает формирование последовательностей областей трещин, заполнение пропущенных элементов в этих последовательностях путем усреднения координат, коррекцию размеров областей с использованием метода простого скользящего среднего, проверку линейности смещения трещин с помощью аппроксимации полиномом второго порядка с применением метода наименьших квадратов, объединение разрозненных последовательностей, описывающих одну трещину, и добавление ранее неидентифицированных, но релевантных областей в существующие траектории на основе предсказанного положения. Сформированное программное обеспечение принимает на вход последовательность изображений микроструктуры композита. Результатом работы является матрица с координатами верхнего левого угла, высотой и шириной идентифицированных трещин для каждого проанализированного изображения, а также выходные изображения с визуализированными областями трещин. Предложенный процесс позволяет автоматизировать процесс контроля качества и исследования внутренней структуры композитных материалов.

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

***Для цитирования*:** 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/published.php?ID=185657

Original article

METHOD FOR TOMOGRAPHIC IMAGES PROCESSING FOR MATRIX CRACKING QUANTIFICATION IN COMPOSITE MATERIAL

Andrei V. Panteleev[1], Nikolay V. Turbin[2]✉, Nikolay A. Tuchkov[3], Roman L. Talya[4], Isak A. Akhmedov[5]

[1,2,3,4,5]Moscow Aviation Institute (National Research University),

Moscow, Russia

[2]turbinnv@mai.ru✉

Abstract. The paper proposes an algorithm and software for automatic detection and analysis of transverse microcracks in layered composite materials based on a sequence of tomographic images. The relevance of the problem is due to the influence of microcracks on the operational and fundamental mechanical characteristics of composites, as well as the complexity of their detection by traditional non-destructive testing methods. The proposed two-stage algorithm includes a primary analysis of individual tomographic slices based on a combination of filtering methods, adaptive and simple binarization, boundary detection, a series of morphological transformations, and the Suzuki algorithm for finding contours. This stage is aimed at identifying potential crack areas and separating them from interlayer inclusions by analyzing and "subtracting" horizontal patterns from vertical ones. Secondary analysis is aimed at refining and tracking cracks along a sequence of slices.

This step includes the formation of crack region sequences, filling in missing elements in these sequences by averaging coordinates, adjusting region sizes using the simple moving average method, checking the linearity of crack displacement using the second-order polynomial approximation using the least squares method, merging disparate sequences describing a single crack, and adding previously unidentified but relevant regions to existing trajectories based on the predicted position. The generated software accepts a sequence of composite microstructure images as input. The result of the work is a matrix with the coordinates of the upper left corner, the height and width of the identified cracks for each analyzed image, as well as output images with visualized crack regions. The proposed process allows automating the process of quality control and research of the internal structure of composite materials.

***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. 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

Список основных обозначений

$H^{(fin)}$ – изображение, содержащее горизонтальные паттерны,

$V^{(fin)}$ – изображение, содержащее вертикальные паттерны,

$(p_x, p_y), (q_x, q_y)$ – координаты пикселя,

CCr – матрица областей содержания трещин,

h_c − число анализируемых изображений,

w_c – число уникальных трещин,

x, y – координаты левого верхнего угла области содержания трещин,

w, h − ширина и высота области содержания трещин,

FCr − множество областей возможного содержания трещин на изображении, найденных в результате первичного анализа,

NCr − множество ложных областей содержания трещин на изображении,

PLF − массив, элементы которого описывают характер линейного смещения областей содержания трещин,

$\hat{k}_1, \hat{k}_0$ − значения коэффициентов аппроксимирующего полинома,

$Y1_0, Y1_{ed}$ − координаты по y левого верхнего угла первой и последней области последовательности.

Введение

Современное машиностроение, включая такие критически важные отрасли, как авиакосмическая, автомобильная и ветроэнергетическая промышленность, всё шире применяет слоистые композитные материалы [1].

Фундаментальные механические и эксплуатационные характеристики композитов неразрывно связаны с их микроструктурой. Сложная морфология, распределение размеров составляющих элементов и их локальное механическое поведение напрямую определяют глобальные свойства материала [1], например,

пористые структуры, являющиеся одним из типов микроструктур, демонстрируют уникальные физические свойства, такие как низкий вес, высокая проницаемость, большая площадь поверхности к объему, а также способность к поглощению энергии [2].

Однако в процессе производства, эксплуатации или под воздействием агрессивных факторов окружающей среды в композитах неизбежно возникают различные дефекты и повреждения. Эти изменения в микроструктуре напрямую влияют на механические свойства материала и его долговечность.

Таким образом, детальное изучение микроструктуры и её изменений под нагрузкой становится не просто академическим интересом, а критически важным для преодоления барьеров в инженерии и расширения областей применения композитов.

Среди различных дефектов, возникающих в композитных материалах, микротрещины, в особенности поперечные (то есть трещины внутри одного слоя, ориентированные перпендикулярно направлению нагрузки или по ширине), занимают особое место. Эти повреждения способны существенно снижать структурную целостность, уменьшать долговечность и служить предшественниками более серьёзных механизмов разрушения, таких как межслойное расслоение [3].

Обнаружение микротрещин представляет собой сложную задачу из-за анизотропии материала и малого размера повреждений. Для характеризации микроструктуры композитных материалов компьютерная томография (КТ) является мощным инструментом, позволяющим визуализировать и количественно оценивать внутреннюю трёхмерную структуру [4]. Методы сегментации, основанные на КТ-

данных, хорошо подходят для определения несплошности материала как интегральной характеристики. Однако традиционная сегментация томограмм сталкивается с серьёзными ограничениями при попытке классифицировать и отделить микротрещины. Значение контрастности пикселей микротрещин часто очень близко к значению окружающих фаз, что приводит к низкому общему контрасту и значительно затрудняет отделение трещин от сложной микроструктуры с помощью традиционных алгоритмов сегментации [4].

Для точного обнаружения и количественной характеристики микротрещин необходимо использовать непосредственно снимки сечений с томограмм (КТ) с высоким разрешением или микроскопические шлифы, полученные, например, с помощью сканирующего электронного микроскопа (СЭМ) [5]. Хотя КТ обеспечивает трёхмерную визуализацию и высокое разрешение, она не может конкурировать с СЭМ по максимальному разрешению для мельчайших деталей [5]. Тем не менее, СЭМ позволяет проводить прямые наблюдения и количественный анализ, например, размеров прекурсоров трещин [3].

Для автоматического обнаружения, классификации и подсчёта числа трещин в этих изображениях используются алгоритмы обработки изображений. Они включают в себя как традиционные методы (фильтрация, пороговая обработка, обнаружение границ, морфологические операции), так и, что более важно, современные подходы, основанные на машинном и глубоком обучении [6]. Проблема заключается не только в сложности обнаружения трещин, но и в неэффективности и субъективности ручного анализа. Традиционные методы НК часто требуют ручной интерпретации данных, что может быть трудоёмким и

субъективным [6]. Конвенциональная сегментация часто требует обширных технических знаний в области обработки изображений и человеческого вмешательства, что может приводить к неосознанным искажениям [7]. В [8–10] для решения прикладных задач обработки изображений эффективно применялись различные методы фильтрации и бинаризации.

Целью исследования является разработка алгоритма для автоматического определения областей поперечных микротрещин в слоистых композитах по данным томографических изображений. Алгоритм состоит из двух этапов: первичный анализ включает обработку изображений для выявления трещин с помощью фильтрации, бинаризации и морфологических операций; вторичный анализ уточняет области трещин, корректируя их размеры, проверяя линейность и объединяя разрозненные сегменты. Реализация выполнена на языке Python с использованием библиотек NumPy и OpenCV. Результатом является матрица с координатами и размерами областей трещин для каждого изображения, а также выделенные области на изображениях.

Изложение данной статьи организовано следующим образом: в разделах 1-3 представлена постановка задачи и краткое описание каждого из блоков алгоритма. Раздел 4 посвящен подробному описанию работы этапа первичного анализа изображений. В разделе 5 дается представление о работе каждого из блоков вторичного этапа анализа снимков. В разделе 6 представлено описание блоков программной реализации алгоритма и порядок их выполнения.

1. Техническая постановка задачи

Дана последовательность изображений в градациях серого, на каждом из которых представлен скан микроструктуры композита [11], полученный путем томографического анализа, и информация об этом скане. Томографические снимки 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 (рис. 1.1). Между слоями могут содержаться межслойные включения, такие как пустоты, воздух или остатки термовуали, которые отображаются как вкрапления черного цвета в материале. Внутри монослоя могут находиться трещины, отображающиеся узкими вертикальными черными полосами между границами монослоев.

Требуется разработать пошаговый алгоритм и программную реализацию алгоритма, которые на основе матричного представления изображения микроструктуры композита при помощи методов обработки и анализа определяют области с трещинами (рис. 1.2).

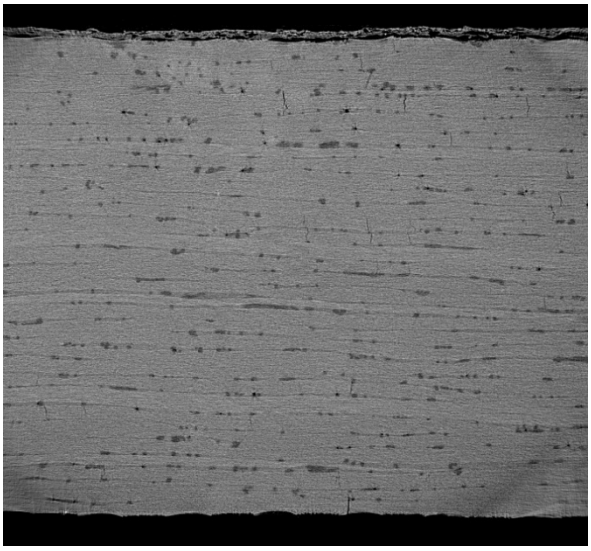


Рис. 1.1. Один из томографических сканов последовательности

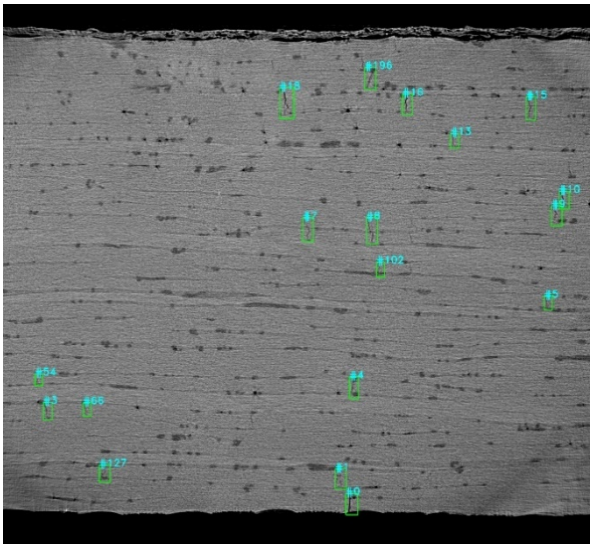


Рис. 1.2. Результат работы алгоритма поиска трещин

2. Математическая постановка задачи

Дана последовательность изображений в градациях серого, которые представляют собой матрицы $P^{(k)} = \left(P_{ij}^{(k)}\right)^{h \times w}$, $P_{ij}^{(k)} \in [0,255], k = \overline{0,n-1}$ где h – высота изображения, w – ширина, n – количество изображений, $P_{ij}^{(k)}$ – яркость пикселя с координатами (i, j) на k-м изображении.

Требуется разработать пошаговый алгоритм и программную реализацию алгоритма, которые на основе матричного представления изображения микроструктуры композита при помощи методов обработки и анализа определяют прямоугольные области содержания трещин, которые представляют собой множество упорядоченных чисел $(x, y, w, h)_{ij}$, где x_{ij}, y_{ij} – координаты пикселя, который является верхним левым углом j-й области на i-м изображении, h_{ij}, w_{ij} – высота и ширина в пикселях j-й области на i-м изображении.

3. Алгоритм обнаружения трещин

Алгоритм обнаружения трещин содержит два этапа.

3.1. Производится первичный анализ снимков. При помощи различных методов обработки изображений и морфологических сверток определяются прямоугольные области, где возможно содержатся трещины.

3.2. Производится вторичный анализ снимков. Ввиду особенности алгоритма данный этап не выполняется для первого и последнего изображения последовательности. Вторичный анализ снимков делится на следующие подпункты.

3.2.1. Сравниваются найденные области с областями на следующем и предыдущем изображениях последовательности – соседних изображениях. Формируются последовательности из координат областей, указывающих на расположение трещины, на разных снимках, причем каждой последовательности присваивается уникальный номер. Также для каждого изображения формируется множество, в котором содержатся ошибочно определенные области, внутри которых трещина отсутствует.

3.2.2. На основе информации из подпункта 3.2.1. вычисляются координаты областей, которые не были определены при первичном анализе.

3.2.3. Производится коррекция размеров областей с использованием метода простого скользящего среднего. Благодаря этому появление межслойных включений рядом с трещиной не будет влиять на размер области.

3.2.4. Последовательности координат областей с разными номерами, но описывающих одну и ту же трещину, объединяются в одну последовательность. Это делается на основе предположения о линейной скорости смещения трещин.

3.2.5. Среди областей из множества ошибочно определенных ищутся те, которые могут являться частью последовательностей, и записываются в них.

3.2.6. Если последовательность состоит из более 30 областей, описывающих линейное движение трещины, то границы областей этой последовательности окрашиваются в зеленый цвет и выводятся на изображение вместе с их уникальным номером.

4. Первичный анализ снимков

На данном этапе алгоритм определяет на изображении все области, где, возможно, содержится трещина. Трещины на изображении расположены вертикально, иногда отклоняясь от вертикального положения не более чем на 45 градусов. Таким образом, задача сводится к поиску скоплений черных пикселей, вытянутых вдоль оси *Oy* на изображении – вертикальных паттернов. Однако межслойные включения тоже имеют вертикальные паттерны, которые сильно влияют на точность работы алгоритма поиска трещин. Чтобы от них избавиться, алгоритм ищет на изображении все горизонтальные паттерны, представляющие скопления черных пикселей, вытянутых вдоль оси *Ox*, а затем «вычитает» их из изображения с вертикальными паттернами. Так как трещины практически не вытянуты вдоль оси *Ox*, то при поиске горизонтальных паттернов они захватываться не будут. Такой способ позволяет уменьшить количество ошибочно захваченных вертикальных паттернов межслойных включений. На основе исходного изображения алгоритм формирует два новых изображения: на изображении $H^{(fin)}$ будут содержаться горизонтальные паттерны, а на изображении $V^{(fin)}$ – вертикальные паттерны.

Рассмотрим работу алгоритма на примере сегмента одного из изображений последовательности (рис. 4.1).

4.1. Поиск горизонтальных паттернов.

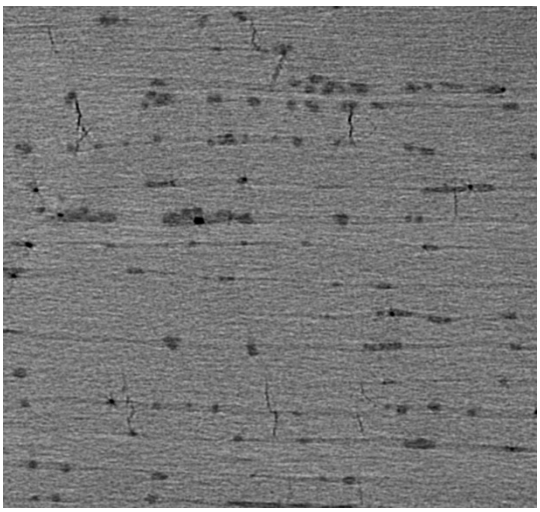


Рис. 4.1. Сегмент одного из изображений последовательности

4.1.1. Подготовим изображение для дальнейшего применения морфологических сверток. Применим билатеральный фильтр с параметрами $\sigma_1 = 75, \sigma_2 = 75, k_1 = 4$ к исходному изображению P, $P \in P^{(k)}$ для того, чтобы удалить лишний шум, сохранив при этом резкость границ межслойных включений (рис. 4.2):

$$H^{(1)}(p) = \frac{1}{W_p} \sum_{q \in S_1} G_{\sigma_1}\left(\|p - q\|\right) G_{\sigma_2}\left(\left|P(p) - P(q)\right|\right) P(q), \qquad (4.1.1)$$

$$W_p = \sum_{q \in S_1} G_{\sigma_1}\left(\|p - q\|\right) G_{\sigma_2}\left(\left|P(p) - P(q)\right|\right), \qquad (4.1.2)$$

$$G_{\sigma_i}(x) = \frac{1}{\sigma_i \sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma_i^2}\right), \; i = 1,2, \qquad (4.1.3)$$

где $p = (p_x, p_y)$, $q = (q_x, q_y)$ – координаты пикселя, $P(p), P(q)$ – яркость соответствующего пикселя, $S_1 = [p_x - k_1, \ldots, p_x + k_1] \times [p_y - k_1, \ldots, p_y + k_1]$ – ядро, центром которого является пиксель p; σ_1 – степень влияния пикселей друг на друга, σ_2 – степень смешения значений пикселей, k_1 – размер ядра.

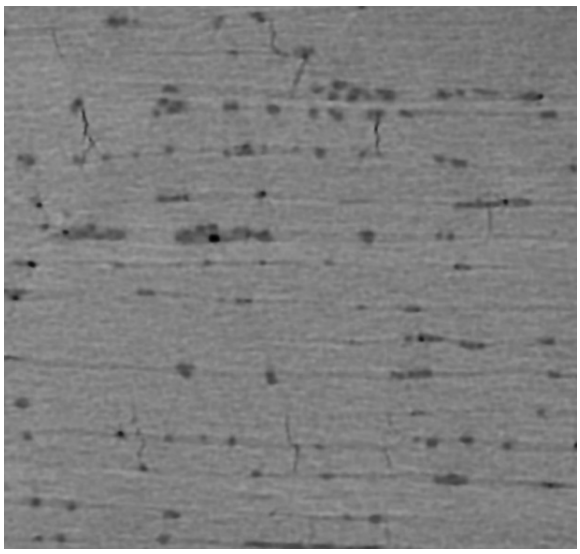
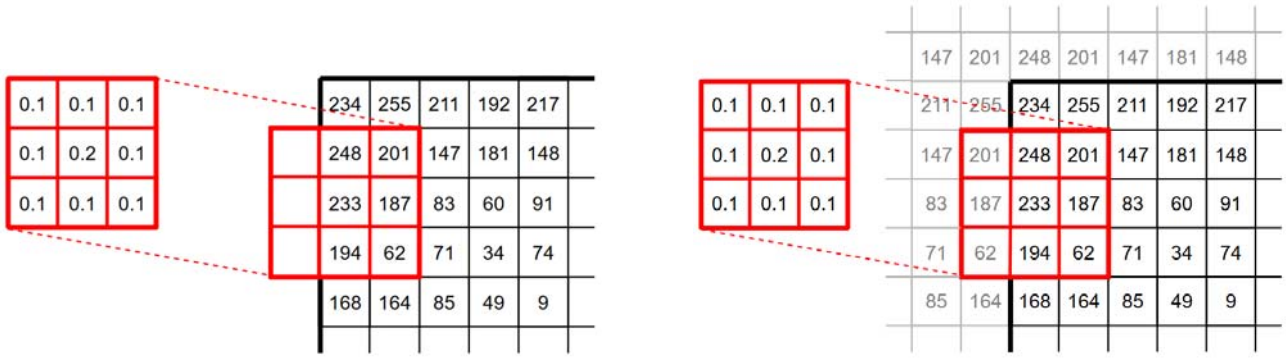


Рис. 4.2. Результат применения билатерального фильтра

Стоит отметить, что каждая операции над изображением в пунктах 4.1 – 4.8 выполняется для всех пикселей снимка, т.е. $\forall p \in [0,\ldots,w-1]\times[0,\ldots,h-1]$, если не указано иначе. Также рассмотрим ситуации, в которых при применении операции с ядром к граничному пикселю снимка ядро выходит за границу изображения. Пример такого случая представлен на рис 4.3, а, где демонстрируется применение ядра размером 3×3 к граничному пикселю в окрестности левого верхнего угла изображения. Чтобы выполнить операцию необходимо задать значения пикселей, соответствующих выходящим за границу элементам ядра. Один из способов определения этих значений заключается в зеркальном отражении пикселей изображения относительно пикселей на границе:

$$edcb\,|\,abcdefg\,|\,fedc,$$

где, | - граница изображения, $abcdefg$ – значения пикселей изображения, $edcb$ и $fedc$ – пиксели, полученные путем отражения значений пикселей изображения. На рис. 4.3, б представлено расширение границ для примера изображения с рис. 4.3, а.



а) б)

Рис. 4.3. (а) Пример случая, когда ядро выходит за границу изображения, (б) Расширение границы изображения путем зеркального отражения пикселей относительно пикселей на границе

4.1.2. Применим адаптивную бинаризацию с параметрами $k_2 = 15, C = 4$ (рис. 4.4). Это упростит дальнейший анализ и обработку изображения:

$$H^{(2)}(p) = \begin{cases} 255, & H^{(1)}(p) > A_p, \\ 0, & H^{(1)}(p) \le A_p; \end{cases} \tag{4.1.4}$$

$$A_p = \sum_{q \in S_2} G_{\sigma_3}(p-q) H^{(1)}(p) - C, \quad \sigma_3 = 0.3\big((k_2 - 1)0.5 - 1\big) + 0.8, \tag{4.1.5}$$

$$G_{\sigma_3}(x) = \frac{1}{\sigma_3\sqrt{2\pi}} \exp\left(-\frac{x^2}{2\sigma_3^2}\right), \tag{4.1.6}$$

где $S_2 = [p_x - k_2, \ldots, p_x + k_2] \times [p_y - k_2, \ldots, p_y + k_2]$ – ядро, центром которого является пиксель p; k_1 – размер ядра, C – константа.

Рис. 4.4. Результат применения адаптивной бинаризации

4.1.3. К полученному бинаризированному изображению применяется ряд морфологических преобразований [12], чтобы выделить на нем горизонтальные паттерны (рис. 4.5). В первую очередь выполним операцию *morph open* (*морфологическое открытие*) с ядром $Kr = \left(Kr_{ij} \right)^{2\times 2}$, $Kr_{ij} = 1,$ которая заключается в последовательном применении операций *erode* (*эрозия*) (4.1.7) и *dilate* (*наращивание*) (4.1.8).

Таким образом, можно избавиться от лишнего шума, созданного на предыдущем шаге:

$$H^{(3)}\left(p \right) = \min_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(H^{(2)}\left(p_x + \left(q_x - 1 \right), p_y + \left(q_y - 1 \right) \right) \right), \qquad (4.1.7)$$

$$H^{(4)}\left(p \right) = \max_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(H^{(3)}\left(p_x + (q_x - 1), p_y + \left(q_y - 1 \right) \right) \right). \qquad (4.1.8)$$

Рис. 4.5. Результат очистки изображения от шума

4.1.4. Выполним операцию *erode* (*эрозия*) с ядром $Kr = \left(Kr_{ij} \right)^{1\times 4}$, $Kr_{ij} = 1$, чтобы сжать все белые частицы на изображении по горизонтали. Это позволит удалить трещины на изображении, вытянутые по вертикали, никак не повлияв на межслойные включения (рис. 4.6):

$$H^{(5)}\left(p \right) = \min_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(H^{(4)}\left(p_x + q_x, p_y + \left(q_y - 2 \right) \right) \right). \qquad (4.1.9)$$

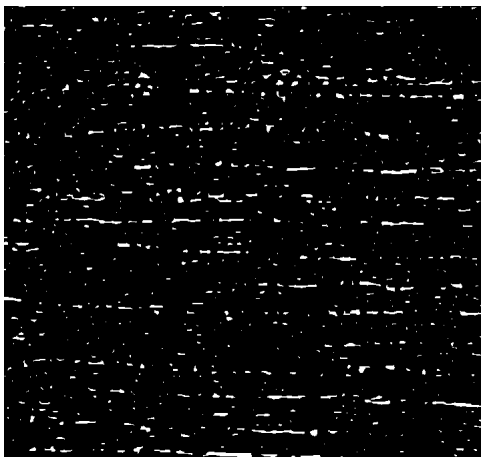


Рис. 4.6. Результат удаления вертикальных паттернов

4.1.5. Применим операцию *morph open* (*морфологическое открытие*) с ядром $Kr = \left(Kr_{ij} \right)^{2\times 2}$, $Kr_{ij} = 1$. Тем самым удаляется шум, созданный на предыдущем шаге:

$$H^{(6)}(p) = \min_{(q_x,q_y):\, Kr_{q_x,q_y}\neq 0} \left(H^{(5)}\left(p_x + (q_x - 1), p_y + \left(q_y - 1 \right) \right) \right), \qquad (4.1.10)$$

$$H^{(7)}(p) = \max_{(q_x,q_y):\, Kr_{q_x,q_y}\neq 0} \left(H^{(6)}\left(p_x + (q_x - 1), p_y + \left(q_y - 1 \right) \right) \right). \qquad (4.1.11)$$

Полученное изображение будет содержать исключительно горизонтальные паттерны, соответствующие межслойным включениям (рис. 4.7).

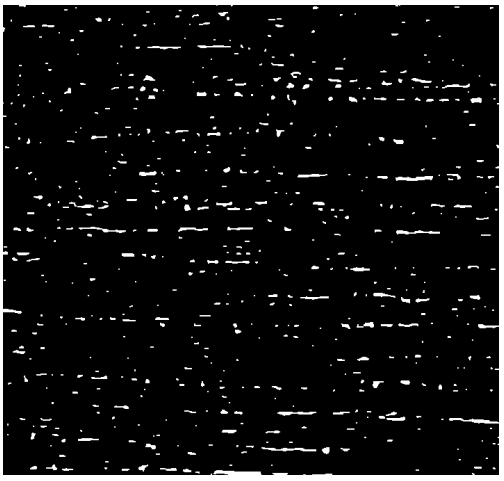


Рис. 4.7. Результат очистки изображения от шума после удаления вертикальных паттернов

4.1.6. Воспользуемся операцией *dilate (наращивание)* с ядром $Kr = \left(Kr_{ij} \right)^{6\times 5}$, $Kr_{ij} = 1$, чтобы восстановить изначальный размер горизонтальных паттернов и немного увеличить его (рис. 4.8):

$$H^{(fin)}(p) = \max_{(q_x,q_y):\, Kr_{q_x,q_y}\neq 0} \left(H^{(7)}\left(p_x + (q_x - 3), p_y + \left(q_y - 2 \right) \right) \right). \qquad (4.1.12)$$

В результате получим изображение $H^{(fin)}$, на котором белым цветом выделены горизонтальные паттерны исходного изображения. Далее алгоритм переходит к поиску вертикальных паттернов.

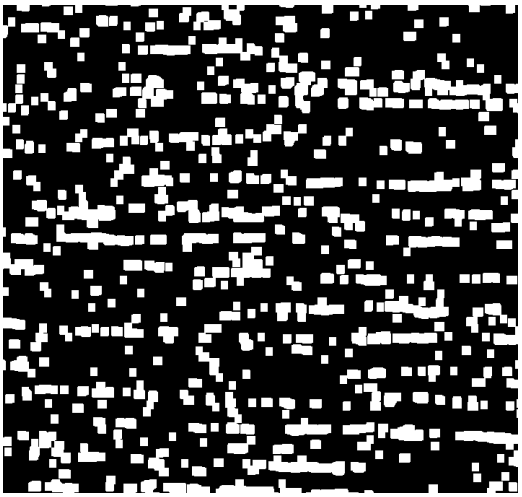


Рис. 4.8. Результат восстановления изначального размера горизонтальных паттернов

4.2. Поиск вертикальных паттернов

4.2.1. Применим размытие по Гауссу (фильтр Гаусса) с параметрами $\sigma_4 = 20,\ k_3 = 2$ для удаления шума с изображения (рис. 4.9):

$$V^{(1)}(p) = \sum_{q \in S_3} G_{\sigma_4}\left(\|p - q\|\right) P(q), \qquad (4.1.13)$$

где $S_3 = [p_x - k_3, \ldots, p_x + k_3] \times [p_y - k_3, \ldots, p_y + k_3]$ – ядро, центром которого является пиксель p; k_3 – размер ядра, σ_4 – степень влияния пикселей друг на друга.

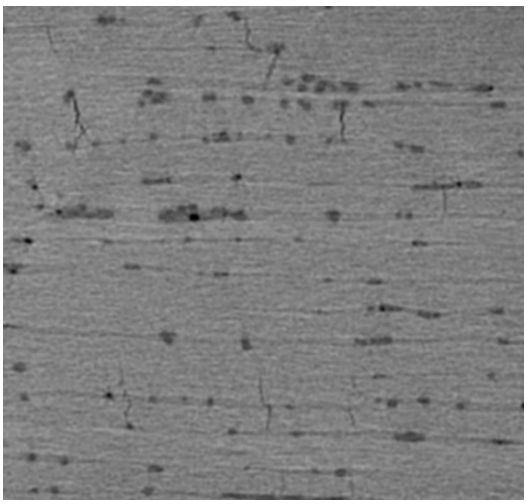


Рис. 4.9. Результат применения фильтра Гаусса

4.2.2. Воспользуемся фильтром Собеля [13] для поиска вертикальных границ объектов на изображении. Результатом его работы является матрица значений градиента в каждом пикселе изображения (рис. 4.10). Положительное значение градиента соответствует левой границе объекта, а отрицательное – правой границе объекта. Сохраним полученную матрицу градиентов в виде изображения. Если модуль значения в матрице градиентов превышает 255, то сохраняется значение 255, а иначе записывается само значение модуля:

$$V^{(Sb)}(p)=\sum_{q_x=-1}^{1}\sum_{q_y=-1}^{1}V^{(1)}(p+q)Sb_{q_x+1,q_y+1},\ V^{(Sb)}(p)\in[-1020,1020];\quad(4.1.14)$$

$$Sb=\begin{pmatrix}-1 & 0 & 1\\ -2 & 0 & 2\\ -1 & 0 & 1\end{pmatrix},\quad(4.1.15)$$

$$V^{(2)}(p)=\begin{cases}255, & \left|V^{(Sb)}(p)\right|>255,\\ \left|V^{(Sb)}(p)\right|, & \left|V^{(Sb)}(p)\right|\le 255.\end{cases}\quad(4.1.16)$$

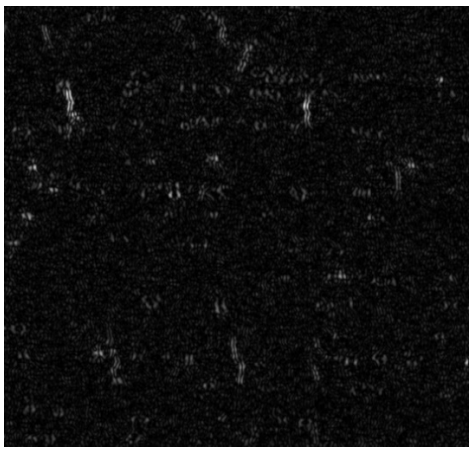


Рис. 4.10. Результат применения фильтра Собеля

4.2.3. Применим простую бинаризацию с пороговым значением $k_4 = 50$ для упрощения последующего анализа и обработки изображения (рис. 4.11):

$$V^{(3)}(p) = \begin{cases} 255, & V^{(2)}(p) > k_4, \\ 0, & V^{(2)}(p) \le k_4. \end{cases} \qquad (4.1.17)$$

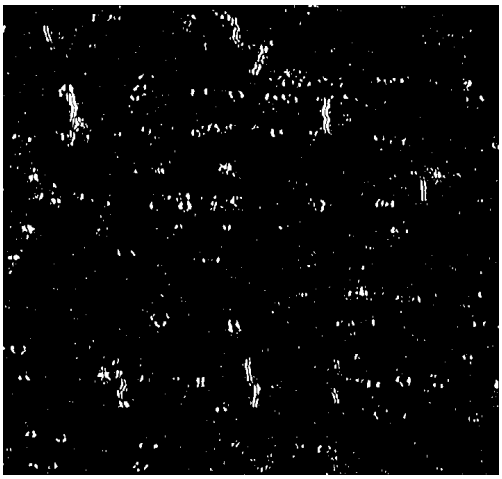


Рис. 4.11. Результат применения простой бинаризации

4.2.4. Применим операцию *morph open* (*морфологическое открытие*) с ядром $Kr = \left(Kr_{ij} \right)^{2\times 2}$, $Kr_{ij} = 1$. Таким образом, мы избавимся от лишнего шума, созданного после бинаризации изображения (рис. 4.12):

$$V^{(4)}\left(p \right) = \min_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(V^{(3)}\left(p_x + \left(q_x - 1 \right), p_y + \left(q_y - 1 \right) \right) \right), \qquad (4.1.18)$$

$$V^{(5)}\left(p \right) = \max_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(V^{(4)}\left(p_x + \left(q_x - 1 \right), p_y + \left(q_y - 1 \right) \right) \right). \qquad (4.1.19)$$

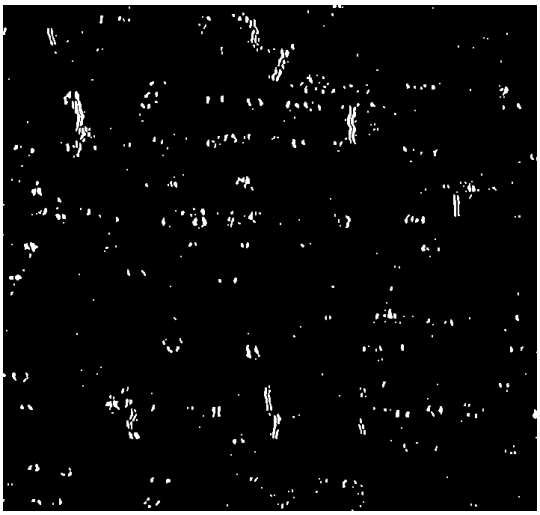


Рис. 4.12. Результат очистки изображения от шума

4.2.5. Воспользуемся операцией *dilate* (*наращивание*) с ядром $Kr = \left(Kr_{ij} \right)^{3\times 3}$, $Kr_{ij} = 1$ для увеличения площади белых частиц на изображении (рис. 4.13):

$$V^{(6)}\left(p \right) = \max_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(V^{(5)}\left(p_x + \left(q_x - 1 \right), p_y + \left(q_y - 1 \right) \right) \right). \qquad (4.1.20)$$

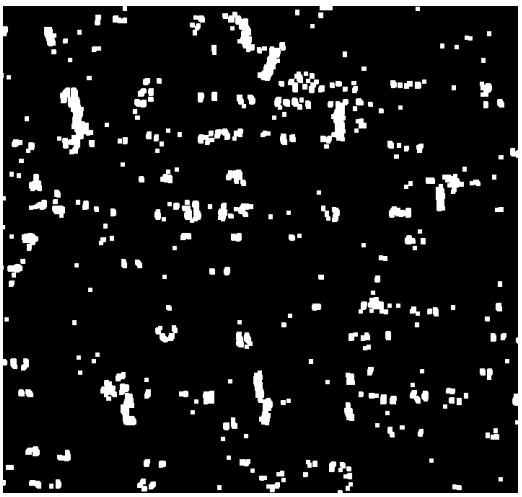


Рис. 4.13. Результат увеличения площади белых частиц на изображении

4.2.6. Используем операцию *erode* (*эрозия*) с ядром $Kr = \left(Kr_{ij}\right)^{4\times1}$, $Kr_{ij} = 1$ для сжатия белых частиц на изображении по вертикали. Это делается для удаления случайно захваченных горизонтальных паттернов. При этом изменения вертикальных паттернов будут минимальны (рис. 4.14):

$$V^{(7)}(p) = \min_{(q_x,q_y):\, Kr_{q_x,q_y}\neq 0} \left(V^{(6)}\left(p_x + (q_x - 2), p_y + q_y\right)\right). \qquad (4.1.21)$$

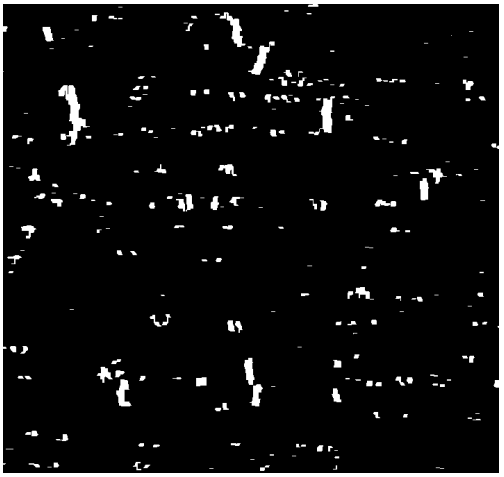


Рис. 4.14. Результат сжатия белых частиц по вертикали

4.2.7. Применим операцию *morph tophat* (*преобразование цилиндра*) с ядром $Kr = \left(Kr_{ij} \right)^{5\times 2}$, $Kr_{ij} = 1$ (рис. 4.15), которая заключается в последовательном применении операции *morph open* (*морфологическое открытие*) и операции побитового исключающего «или» над изображением с предыдущего этапа. Результатом является изображение с удаленными вертикальными паттернами:

$$V^{(8)}\left(p \right) = \min_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(V^{(7)}\left(p_x + \left(q_x - 2 \right), p_y + \left(q_y - 1 \right) \right) \right), \qquad (4.1.22)$$

$$V^{(9)}\left(p \right) = \max_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(V^{(8)}\left(p_x + \left(q_x - 2 \right), p_y + \left(q_y - 1 \right) \right) \right), \qquad (4.1.23)$$

$$V^{(10)}\left(p \right) = V^{(9)}\left(p \right) \oplus V^{(7)}\left(p \right). \qquad (4.1.24)$$

где $\oplus -$ операция побитового исключающего «или».

В результате имеем

$$V^{(10)}(p) = \begin{cases} 0, & V^{(9)}(p) = V^{(7)}(p), \\ 255, & V^{(9)}(p) \neq V^{(7)}(p). \end{cases} \qquad (4.1.25)$$

Рис. 4.15. Результат удаления вертикальных паттернов на изображении

4.2.8. Выполним операцию побитового исключающего «или» между итоговыми изображениями из двух предыдущих этапов. Полученное изображение будет содержать вертикальные паттерны с минимальным содержанием горизонтальных (рис. 4.16):

$$V^{(fin)}(p) = V^{(7)}(p) \oplus V^{(10)}(p). \qquad (4.1.26)$$

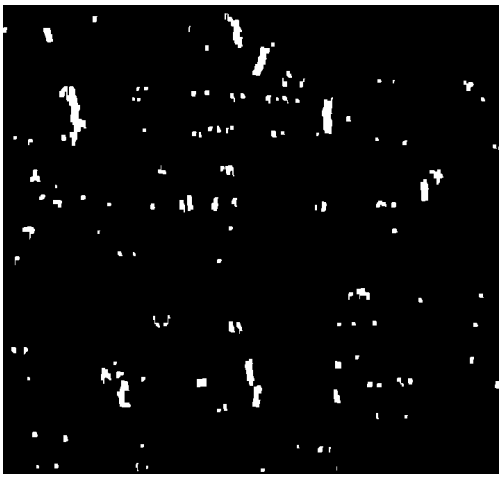


Рис. 4.16. Результат восстановления вертикальных паттернов

4.3. Изображения $H^{(fin)}$ и $V^{(fin)}$ содержат горизонтальные и вертикальные паттерны соответственно. Теперь «вычтем» изображение $H^{(fin)}$ из изображения $V^{(fin)}$, чтобы удалить случайно захваченные белые частицы. Для этого используем такие операции побитового сравнения, как исключающее «или» и отрицание (рис. 4.17):

$$I^{(1)}(p) = \overline{\overline{H^{(fin)}(p)} \oplus V^{(fin)}(p)}. \qquad (4.2.1)$$

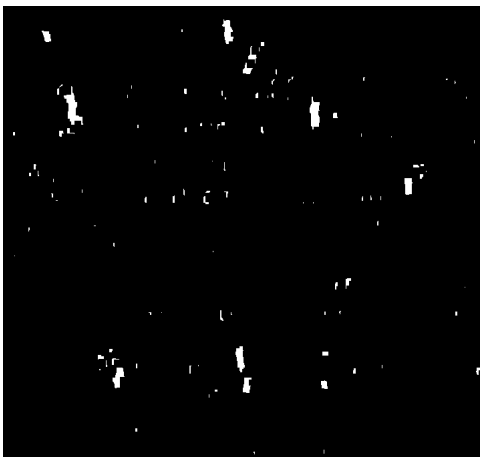


Рис. 4.17. Результат удаления горизонтальных паттернов на изображении

4.4. Применим операцию *morph open* (*морфологическое открытие*) с ядром $Kr = \left(Kr_{ij} \right)^{2\times 2}$, $Kr_{ij} = 1,$ чтобы избавиться от лишнего шума, созданного на предыдущем шаге (рис. 4.18):

$$I^{(2)}\left(p \right) = \min_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(I^{(1)}\left(p_x + \left(q_x - 1 \right), p_y + \left(q_y - 1 \right) \right) \right), \qquad (4.3.1)$$

$$I^{(3)}\left(p \right) = \max_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(I^{(2)}\left(p_x + \left(q_x - 1 \right), p_y + \left(q_y - 1 \right) \right) \right). \qquad (4.3.2)$$

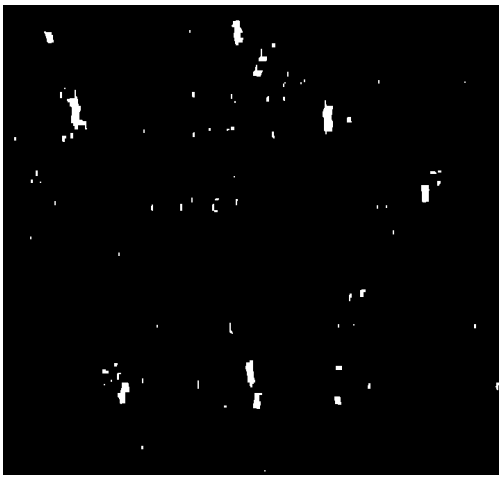


Рис. 4.18. Результат очистки изображения от шума

4.5. Воспользуемся операцией *dilate* с ядром $Kr = \left(Kr_{ij} \right)^{4\times4}$, $Kr_{ij} = 1$ для увеличения площади белых частиц на изображении (рис. 4.19):

$$I^{(4)}\left(p \right) = \max_{(q_x, q_y):\, Kr_{q_x, q_y} \neq 0} \left(I^{(3)}\left(p_x + \left(q_x - 2 \right), p_y + \left(q_y - 2 \right) \right) \right). \qquad (4.4.1)$$

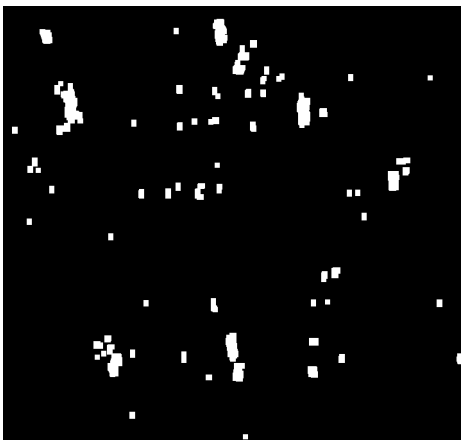


Рис. 4.19. Результат увеличения площади белых частиц

4.6. На полученном бинаризированном черно-белом изображении $I^{(4)} = \{ I^{(4)}(p), \forall p \in \left[0,\ldots, w-1 \right] \times \left[0,\ldots, h-1 \right] \}$ белый цвет соответствует областям, где возможно содержится трещина. Вычислим координаты этих областей при помощи алгоритма Сузуки для поиска контуров [14]. Контуры представляют из себя прямоугольные границы, внутрь которых вписаны области из белых пикселей.

В результате получим множество следующих упорядоченных чисел:

$$PCr_1 = \left\{ \left(x, y, w, h \right)_i \right\}, \quad i = \overline{0, N^{(pcr_1)} - 1}, \qquad (4.5.1)$$

где x_i, y_i – координаты пикселя, который является верхним левым углом прямоугольной области, w_i, h_i – высота и ширина прямоугольной области, $N^{(pcr_1)}$ – число контуров.

4.7. В виду того, что некоторые трещины практически сливаются с фоновым шумом внутри слоя, на полученном изображении $I^{(4)}$ две и более области могут соответствовать одной и той же трещине. Особенность таких областей заключается в том, что они по высоте меньше, чем минимальная ширина монослоя и находятся на достаточно близком расстоянии друг к другу, но при этом алгоритм Сузуки поиска контуров определяет их как разные области, каждой из которых соответствует своя трещина. Также на изображении все еще присутствуют небольшие ошибочно определенные области содержания трещин. Для устранения данных проблем алгоритм попарно сравнивает координаты областей. Если у области нет соседей сверху и снизу, а ее размер меньше, чем минимальная ширина монослоя, то эта область удаляется. Алгоритм выглядит следующим образом.

4.7.1. Задаем начальное значение $i = 0$.

4.7.2. Если $i = N^{(pcr_1)} - 1$, то завершаем работу алгоритма. В ином случае выбираем из множества PCr_1 элемент $(x, y, w, h)_i$. Задаем начальное значение $j = i + 1$.

4.7.3. Из множества PCr_1 выбираем элемент $(x, y, w, h)_j$. Если $j = N^{(pcr_1)}$, то переходим к шагу 4.7.5.

4.7.4. Производим сравнение двух элементов: если выполняются оба следующих условия,

$$a)\ \Big(\big(x_i < x_j\big) \wedge \big(x_i + w_i > x_j\big)\Big) \vee \Big(\big(x_j < x_i\big) \wedge \big(x_j + w_j > x_i\big)\Big), \qquad (4.6.1)$$

$$b)\ \Big(\big(y_i < y_j\big) \wedge \big(y_j - (y_i + h_i) > x_j\big)\Big) \vee \Big(\big(y_j < y_i\big) \wedge \big(y_i - \big(y_j + h_j\big) > x_i\big)\Big), \qquad (4.6.2)$$

то сохраняем j-й элемент в множество $CrCh$ и переходим к шагу 4.7.2, увеличивая значение i на единицу. Если условия не выполняются, то возвращаемся на шаг 4.7.3, увеличивая значение j на единицу.

4.7.5. Проверяем выполнение следующего условия:

$$\left(h_i < 25\right) \wedge \left(\left(x, y, w, h\right)_i \notin CrCh\right). \qquad (4.6.3)$$

Если условие выполняется, то закрашиваем все пиксели внутри границ i-го прямоугольника в черный цвет:

$$I^{(4)}\left(p_x, p_y\right) = 0,\ p_x = \overline{x_i, x_i + w_i}, p_y = \overline{y_i, y_i + h_i}. \qquad (4.6.4)$$

Возвращаемся к шагу 4.7.2, увеличив значение i на единицу.

Результатом работы алгоритма является изображение $I^{(5)}$, на котором удалены все одиноко стоящие маленькие области. При этом группы маленьких соседствующих областей сохранены в исходном виде (рис. 4.20).

Рис. 4.20. Результат работы алгоритма на шаге 4.7

4.8. Применим операцию *dilate* с ядром $Kr = \left(Kr_{ij}\right)^{5\times4}$, $Kr_{ij} = 1$, чтобы увеличить площадь оставшихся областей содержания трещин. Таким образом,

близко стоящие друг к другу маленькие области объединятся в одну большую область (рис. 4.21):

$$I^{(6)}(p) = \max_{(q_x,q_y):\, Kr_{q_x,q_y}\neq 0}\left(I^{(5)}\left(p_x+\left(q_x-2\right),p_y+\left(q_y-2\right)\right)\right). \qquad (4.6.5)$$

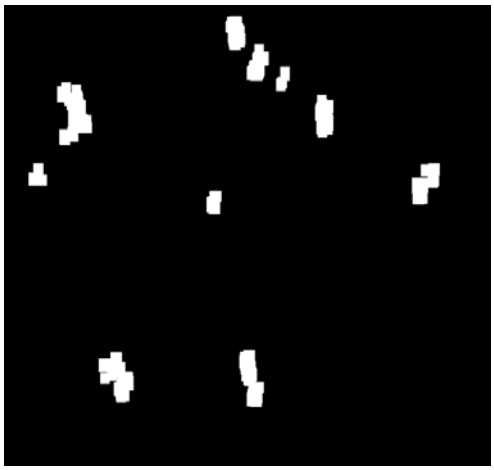


Рис. 4.21. Результат увеличения площади белых областей на изображении

Вновь воспользуемся алгоритмом Сузуки поиска контуров для определения координат новых областей. В результате получаем следующее множество:

$$PCr_2 = \left\{\left(x,y,w,h\right)_i\right\},\ i=\overline{0,N^{(pcr_2)}-1}, \qquad (4.6.6)$$

где $N^{(pcr_2)}$ – число контуров после применения алгоритма на шаге 4.7.

Однако среди оставшихся областей все еще остаются те, которые были определены ошибочно. Для того, чтобы отсеять наиболее явные из них, проверим выполнение следующего условия для каждой области:

$$\frac{h_i}{w_i} > 1.5,\ i=\overline{0,N^{(pcr_2)}-1}. \qquad (4.6.7)$$

Если условие для области выполняется, то записываем ее координаты в множество FCr. В результате, получаем множество, которое является результатом первичного анализа изображения:

$$FCr = \left\{ \left(x, y, w, h \right)_i \right\}, \ i = \overline{0, N^{(fcr)} - 1}, \tag{4.6.8}$$

где $N^{(fcr)}$ – число областей содержания трещин, найденных при первичном анализе снимка.

Первичный анализ выполняется для каждого изображения последовательности, формируя множество множеств ACr, на основе которого будет выполняться вторичный анализ последовательности снимков (рис. 4.22):

$$ACr = \left\{ \left\{ \left(x, y, w, h \right)_i \right\}_k \right\}, i = \overline{0, N_k^{(fcr)}}, k = \overline{0, n-1}. \tag{4.6.9}$$

где n – число изображений в последовательности.

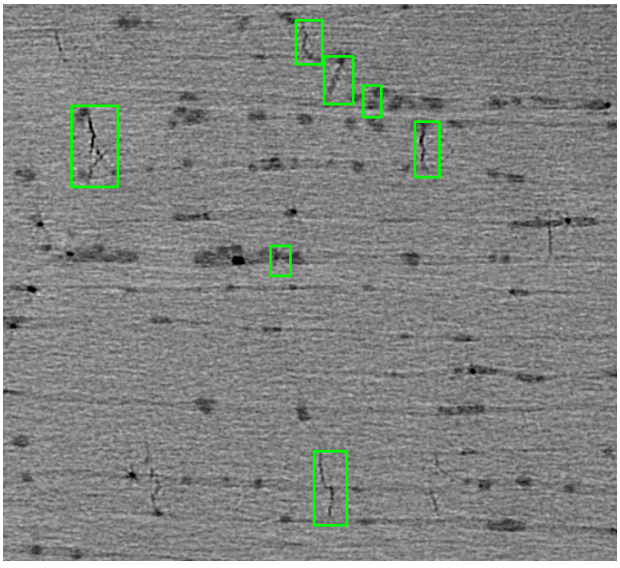


Рис. 4.22. Результат первичного анализа снимков

5. Вторичный анализ снимков

Задача этапа вторичной обработки томографических снимков заключается в определении, какие из найденных областей являются ложными, а какие нет. Ложными являются области, внутри которых отсутствуют трещины. Также, на основе результатов работы первого этапа производится коррекция уже расставленных областей и вычисление новых, неопределенных при первичной обработке. Алгоритм вторичного анализа снимков состоит из множества блоков, каждый из которых выполняет свою подзадачу. Рассмотрим подробнее работу каждого из блоков.

5.1. Формирование последовательностей из областей

В первую очередь на вторичном этапе обработки изображений производится формирование последовательностей областей, описывающих координаты каждой трещины, появляющейся на последовательности изображений. Последовательности областей представляют собой векторы следующего вида:

$$[coord_1, coord_2, ..., coord_{n-2}]^T, \qquad (5.1.1)$$

где $coord_i$, $i = \overline{1, n-2}$ – переменная, указывающая на наличие данной трещины на i-м изображении. Переменная $coord_i$ может принимать либо значение $(x, y, w, h)_i$, соответствующее координатам области содержания трещин, либо значение *None*, которое свидетельствует об отсутствии данной трещины на i-м изображении.

Из данных векторов формируется матрица $CCr = \left(CCr_{ij}\right)^{h_c \times w_c}$, где $h_c = n - 2$ – число изображений в последовательности, а w_c – число уникальных трещин,

найденных на всей последовательности изображений. Ввиду особенности алгоритма первое и последнее изображения последовательности не анализируются.

Опишем алгоритм формирования последовательностей областей содержания трещин.

5.1.1. Вначале определим, из каких областей можно сформировать последовательность, а из каких нет. Для этого обойдем каждое изображение, попарно сравнивая области содержания трещин с областями с предыдущего и следующего изображений последовательности. Если координаты областей отличаются незначительно, то это указывает на то, что они описывают одну и ту же трещину, поэтому из них можно составить последовательность.

5.1.1.1. Задаем начальные значения $k = 1,\ i = 0$.

5.1.1.2. Если $k = n - 1,$ то завершаем работу алгоритма. Иначе выбираем множество областей $FCr^{(k)}$, которое соответствует всем найденным областям содержания трещин на k-м изображении.

5.1.1.3. Если $i = N_k^{(fcr)}$, то переходим к шагу 5.1.2. Иначе выбираем область $\left(x, y, w, h\right)_i$ из множества $FCr^{(k)}$.

5.1.1.4. Задаем начальное значение $j = 0$.

5.1.1.5. Если $j = N_{k-1}^{(fcr)}$, то переходим к шагу 5.1.1.7. Иначе выбираем область $\left(x, y, w, h\right)_j$ из множества $FCr^{(k-1)}$.

5.1.1.6. Проверяем выполнение следующего условия:

$$\left(\left|x_j - x_i\right| < w_i\right) \vee \left(\left|y_j - y_i\right| < h_i\right). \qquad (5.1.2)$$

Если оно выполняется, то переходим на шаг 5.1.1.3, увеличив значение i на единицу. В ином случае возвращаемся на шаг 5.1.1.5, увеличив значение j на единицу.

5.1.1.7. Задаем начальное значение $j = 0$.

5.1.1.8. Если $j = N_{k+1}^{(fcr)}$, то переходим к шагу 5.1.1.10, а иначе выбираем область $\left(x, y, w, h\right)_j$ из множества $FCr^{(k+1)}$.

5.1.1.9. Проверяем выполнение следующего условия:

$$\left(\left|x_j - x_i\right| < w_i\right) \vee \left(\left|y_j - y_i\right| < h_i\right). \qquad (5.1.3)$$

Если оно выполняется, то переходим на шаг 5.1.1.3, увеличив значение i на единицу. В ином случае возвращаемся на шаг 5.1.1.7, увеличив значение j на единицу.

5.1.1.10. Если алгоритм перешел на данный шаг, то это значит, что область $\left(x, y, w, h\right)_i$ из множества $FCr^{(k)}$ не соответствует ни одной области из множества $FCr^{(k-1)}$ и $FCr^{(k+1)}$. Это может значить, что данная область является ложной, поэтому добавим ее в множество $NCr^{(k)}$ с неверно определенными областями. Возвращаемся на шаг 5.1.1.3, увеличив значение i на единицу.

5.1.2. Отсеяв все области, которые, скорее всего, не являются частью последовательностей, можно приступать к формированию этих самых последовательностей. Для этого вновь производится обход всех изображений и попарное сравнение областей с областями с предыдущего изображения. Если области отличаются незначительно, то она добавляется в последовательность. Если

для области с текущего изображения не была найдена соответствующая область на предыдущем изображении, то она становится началом для новой последовательности.

5.1.2.1. Создадим новое множество $FNCr^{(k)} = FCr^{(k)} \setminus NCr^{(k)}$, в котором будут храниться только те области из $FCr^{(k)}$, которых нет в $NCr^{(k)}$.

5.1.2.2. Если $k = 1$, то формируем из множества $FNCr^{(k)}$ массив, состоящий из областей содержания трещин на изображении $k = 1$:

$$Ccr^{(1)} = \left[\left(x, y, w, h\right)_0, \left(x, y, w, h\right)_1, \ldots, \left(x, y, w, h\right)_{n_k} \right], \qquad (5.1.4)$$

где n_k – количество областей в множестве $FNCr^{(k)}$. Возвращаемся на шаг 5.1.1.2, увеличив значение k на единицу.

5.1.2.3. Создадим массив $Ccr^{(k)}$, состоящий из значений *None*, длина которого равна длине $Ccr^{(k-1)}$:

$$Ccr^{(k)} = \left[None_1, None_2, \ldots, None_{n_{k-1}} \right]. \qquad (5.1.5)$$

5.1.2.4. Задаем начальное значение $i = 1$.

5.1.2.5. Если $i = N_k^{(fncr)}$, где $N_k^{(fncr)}$ – количество элементов в множестве $FNCr^{(k)}$, то переходим на шаг 5.1.3. В ином случае выбираем область $\left(x, y, w, h\right)_i$ из множества $FNCr^{(k)}$, задав начальное значение $j = 0$.

5.1.2.6. Если $j = n_{k-1}$, то переходим на шаг 5.1.2.8. В ином случае выбираем j-й элемент из массива $Ccr^{(k-1)}$. Если данный элемент равняется *None*, то проверяем этот же j-й элемент в массиве $Ccr^{(k-q)}, q = \overline{2, k-1}$ до тех пор, пока не

найдем первый элемент, не равный *None*. В итоге получаем координаты области $(x, y, w, h)_j$ из массива $Ccr^{(cn)}$, где cn – номер массива, где впервые был встречен элемент, не равный *None*.

5.1.2.7. Проверяем выполнение следующего условия:

$$\left(\left|x_i - x_j\right| < \max\left(\frac{w_i}{2}, \frac{w_j}{2}\right)\right) \wedge \left(\left|y_i - y_j\right| < \max\left(\frac{h_i}{2}, \frac{h_j}{2}\right)\right) \wedge (k - cn < 5). \quad (5.1.6)$$

Если условие выполняется, то записываем координаты области $(x, y, w, h)_i$ в массив $Ccr^{(k)}$ на место j-го элемента. Возвращаемся на шаг 5.1.2.5, увеличив значение i на единицу. Если условие не выполняется, то возвращаемся на шаг 5.1.2.6, увеличив значение j на единицу.

5.1.2.8. Если в алгоритме совершен переход к данному шагу, то это значит, что область $(x, y, w, h)_i$ не соответствует ни одной области из массива $Ccr^{(k-1)}$. Это может указывать на то, что эта область соответствует трещине, которая отсутствовала на предыдущих изображениях и появилась впервые на текущем. Добавим в конец массива $Ccr^{(k)}$ область $(x, y, w, h)_i$ и, чтобы размеры всех массивов Ccr совпадали, добавим *None* элементы в конец массивов $Ccr^{(q)}, q = \overline{1, k-1}$. Переходим на шаг 5.1.2.5, увеличив значение i на единицу.

5.1.3. Проверим, являются ли отсеянные на шаге 5.1.1 области частью уже сформированных последовательностей. При этом новых последовательностей на основе этих областей создаваться не будет.

5.1.3.1. Задаем начальное значение $i = 1$.

5.1.3.2. Если $i = N_k^{(ncr)}$, где $N_k^{(ncr)}$ – число элементов в множестве $NCr^{(k)}$, то переходим к шагу 5.1.3.5. В ином случае выбираем область $(x, y, w, h)_i$ из множества $NCr^{(k)}$, задав начальное значение $j = 0$.

5.1.3.3. Если $j = n_{k-1}$, то возвращаемся на шаг 5.1.3.2, увеличив значение i на единицу. В ином случае выбираем j-й элемент из массива $Ccr^{(k-1)}$. Если данный элемент равняется *None*, то проверяем этот же j-й элемент в массиве $Ccr^{(k-q)}, q = \overline{2, k-1}$ до тех пор, пока не найдем первый элемент, не равный *None*. В итоге получаем координаты области $(x, y, w, h)_j$ из массива $Ccr^{(cn)}$, где cn – номер массива, где впервые был встречен элемент, не равный *None*.

5.1.3.4. Проверяем выполнение следующего условия:

$$\left(\left|x_i - x_j\right| < \min\left(\frac{w_i}{2}, \frac{w_j}{2}\right)\right) \wedge \left(\left|y_i - y_j\right| < \min\left(\frac{h_i}{2}, \frac{h_j}{2}\right)\right) \wedge (k - cn < 5). \quad (5.1.7)$$

Если условие выполняется, то записываем координаты области $(x, y, w, h)_i$ в массив $Ccr^{(k)}$ на место j-го элемента. Также записываем данную область в множество $BCr^{(k)}$. Возвращаемся на шаг 5.1.3.2, увеличив значение i на единицу. Если условие не выполняется, то переходим на шаг 5.1.3.3, увеличив значение j на единицу.

5.1.3.5. Удаляем из множества $NCr^{(k)}$ множество $BCr^{(k)}$, которое состоит из элементов, оказавшихся в итоге частью последовательностей областей содержания трещин:

$$NCr^{(k)} = NCr^{(k)} \setminus BCr^{(k)}. \quad (5.1.8)$$

Возвращаемся на шаг 5.1.1.3, увеличив значение k на единицу.

Результатом работы данного блока являются массивы $Ccr^{(k)}, k = \overline{1, n-2}$, из которых формируется матрица $CCr = \left(CCr_{ij}\right)^{h_c \times w_c}$, где i-я строка матрицы соответствует массиву $Ccr^{(k+1)}$ и число столбцов матрицы совпадает с числом элементов в массивах.

5.2. Заполнение пустых элементов в последовательностях областей

На разных этапах вторичного анализа снимков в матрице прямоугольных областей $CCr = \left(CCr\right)_{ij}^{h_c \times w_c}, CCr_{ij} = \left(x, y, w, h\right)_{ij} \cup None$ значения *None* возникают внутри последовательности областей, соответствующих какой-либо трещине. Это свидетельствует о том, что алгоритм не нашел на определенных изображениях область, внутри которой находится данная трещина, но, так как значения *None* находятся внутри последовательности, то это значит, что трещина на этих изображениях все-таки есть. Требуется доопределить на данных снимках координаты прямоугольных областей и заменить ими значения *None*. Для этого обойдем каждый столбец матрицы, заменяя *None* элементы на усредненные координаты областей, вычисленных на основе ближайших не *None* значений. Опишем данный алгоритм.

5.2.1. Устанавливаем начальное значение $j = 0$.

5.2.2. Устанавливаем начальное значение $i = 0$. Задаем значения $k = -1$ и $flg = 0$. Переменная k хранит индекс последнего элемента последовательности, не

равного *None*. Значение flg указывает на наличие *None* элементов внутри последовательности: 1 – элементы присутствуют, 0 – отсутствуют.

5.2.3. Если $j = w_c$, то алгоритм завершает работу. Если $i = h_c$, то возвращаемся на шаг 5.2.2, увеличив значение j на единицу. В ином случае выбираем из матрицы CCr элемент $CCr_{ij} = (x, y, w, h)_{ij}$.

5.2.4. Проверяем выполнение следующего условия:

$$\left(CCr_{ij} \neq None\right) \wedge \left(flg = 1\right) \wedge \left(k \neq -1\right). \tag{5.1.9}$$

Если условие выполняется, то это значит, что между элементами последовательности CCr_{kj} и CCr_{ij} лежат *None* элементы. Заменим эти элементы приблизительными координатами области содержания соответствующей трещины. Для этого используем следующие формулы:

$$\begin{aligned} &x_{cj} = x_{k\,j} - \frac{x_{k\,j} - x_{ij}}{i-k-1}\left(c-k\right),\ y_{cj} = y_{k\,j} - \frac{y_{k\,j} - y_{ij}}{i-k-1}\left(c-k\right),\ c = \overline{k+1, i-1}, \\ &w_{cj} = w_{k\,j} - \frac{w_{k\,j} - w_{ij}}{i-k-1}\left(c-k\right),\ h_{cj} = h_{k\,j} - \frac{h_{k\,j} - h_{ij}}{i-k-1}\left(c-k\right),\ c = \overline{k+1, i-1}. \end{aligned} \tag{5.1.10}$$

Из полученных значений формируем кортежи $\left(x, y, w, h\right)_{cj}$ и присваиваем их соответствующим элементам матрицы CCr_{cj} вместо значений *None*. Меняем значение flg на 0, увеличиваем значение i на единицу и возвращаемся на шаг 5.2.3.

5.2.5. Проверяем выполнение следующего условия:

$$\left(CCr_{ij} \neq None\right) \wedge \left(flg = 0\right). \tag{5.1.11}$$

Если оно выполняется, то присваиваем переменной k значение j. Увеличиваем значение i на единицу и возвращаемся на шаг 5.2.3.

5.2.6. Проверяем выполнение следующего условия:

$$\left(CCr_{ij} = None\right) \wedge \left(k \neq -1\right). \qquad (5.1.12)$$

Если оно выполняется, то меняем значение flg на 1.

После проверки условия увеличиваем значение i на единицу и возвращаемся на шаг 5.2.3.

Результатом работы данного блока является матрица $CCr = \left(CCr_{ij}\right)^{h_c \times w_c}$ в которой *None* элементы не содержатся внутри последовательностей областей.

5.3. Коррекция размеров областей

В ходе проведения первичного анализа снимков внутрь областей возможного содержания трещин вместе с самими трещинами могут попадать межслойные включения. Это происходит в том случае, если они находятся на достаточно близком расстоянии друг от друга, в результате чего алгоритм воспринимает их как единый объект. Данный дефект проявляется в резком увеличении и затем уменьшении высоты и ширины области по сравнению с соседними областями последовательности, описывающей трещину. Для устранения дефекта применим метод простого скользящего среднего [15] к каждой из сформированных последовательностей. Опишем порядок применения данного метода к матрице областей содержания трещин $CCr = \left(CCr_{ij}\right)^{h_c \times w_c}, CCr_{ij} = \left(x, y, w, h\right)_{ij} \cup None$.

5.3.1. Устанавливаем начальное значение $j = 0$.

5.3.2. Если $j = w_c$, то алгоритм завершает свою работу. Иначе, извлекаем из матрицы CCr столбец под номером j:

$$col^{(1)} = CCr_{ij} = (x, y, w, h)_{ij} \bigcup None, i = \overline{0, h_c - 1}. \qquad (5.1.13)$$

В полученном массиве $col^{(1)}$ содержатся координаты областей, описывающих положение j-й трещины на всех изображениях.

5.3.3. Удалим из массива $col^{(1)}$ все *None* элементы. Также зафиксируем индекс ind первого элемента в массиве $col^{(1)}$, не равного *None*. В итоге получаем следующий массив:

$$col^{(2)} = (x, y, w, h)_{ij}, i = \overline{0, h_{col} - 1}, \qquad (5.1.14)$$

где h_{col} − число элементов массива $col^{(1)}$, не равных *None*.

5.3.4. Формируем четыре массива из координат областей, содержащихся в массиве $col^{(2)}$:

$$X1_i = x_{ij}, \quad Y1_i = y_{ij}, \quad X2_i = x_{ij} + w_{ij}, \quad Y2_i = y_{ij} + h_{ij}, \quad i = \overline{0, h_{col} - 1}. \quad (5.1.15)$$

5.3.5. К каждому из сформированных на предыдущем шаге массивов применяем простое скользящее среднее с параметром $wd = 7$:

$$SMA^{(X1)} = \frac{1}{wd} \sum_{k=0}^{wd-1} X1_{i+k}, \quad SMA^{(Y1)} = \frac{1}{wd} \sum_{k=0}^{wd-1} Y1_{i+k}, \quad i = \overline{0, h_{SMA} - 1}, \quad (5.1.16)$$

$$SMA^{(X2)} = \frac{1}{wd} \sum_{k=0}^{wd-1} X2_{i+k}, \quad SMA^{(Y2)} = \frac{1}{wd} \sum_{k=0}^{wd-1} Y2_{i+k}, \quad i = \overline{0, h_{SMA} - 1}, \quad (5.1.17)$$

где $h_{SMA} = h_{col} - wd$ – размер массива после применения метода простого скользящего среднего, wd – сглаживающий интервал.

5.3.6. Новые координаты областей записываем в CCr вместо старых:

$$CCr_{ij} = (SMA^{(X1)}, SMA^{(Y1)}, SMA^{(X2)}, SMA^{(Y2)}), i = \overline{st, h_{SMA} - 1}, \qquad (5.1.18)$$

где $st = ind + \left[\frac{wd}{2} \right]$.

5.3.7. Увеличиваем значение j на единицу и возвращаемся на шаг 5.3.2.

Результатом работы данного блока является матрица $CCr = \left(CCr_{ij} \right)^{h_c \times w_c}$ со скорректированными координатами областей содержания трещин.

5.4. Проверка линейного смещения областей

При просмотре последовательности снимков можно заметить, что на каждом следующем изображении положение трещин слегка отличается от положения на предыдущем снимке. Это указывает на наличие трещины в слое с ориентацией, отличной от нормали к сечению. Изображения ориентированы так, чтобы монослои располагались горизонтально, поэтому трещины будут двигаться вправо или влево. При этом все трещины на снимках смещаются с постоянной скоростью. Таким образом, зная координаты областей возможного содержания трещин на всех изображениях, можно проверить характер их движения. Если смещение областей не является равномерным, значит они были выделены ошибочно. Одним из способов проверки является построение аппроксимирующего полинома 2-го порядка с использованием метода наименьших квадратов [15] для x координат верхнего левого угла областей, содержащихся в одной последовательности. Если значение коэффициента при x^2 аппроксимирующего полинома превышает пороговое значение, то это указывает на неравномерное смещение областей. Опишем применение данного способа.

5.4.1. Устанавливаем начальное значение $j = 0$. Создаем массив PLF следующего вида: $PLF = [None_0, None_1 ... None_{w_c}]$, где w_c – число столбцов матрицы CCr.

5.4.2. Если $j = w_c$, то алгоритм завершает свою работу. Иначе извлекаем из матрицы CCr столбец под номером j:

$$col^{(1)} = CCr_{ij} = (x, y, w, h)_{ij} \bigcup None, i = \overline{0, h_c - 1}, \qquad (5.1.19)$$

где h_c – число строк матрицы CCr.

В полученном массиве $col^{(1)}$ содержатся координаты областей, описывающих положение *j*-й трещины на всех изображениях.

5.4.3. Удалим из массива $col^{(1)}$ все *None* элементы:

$$col^{(2)} = (x, y, w, h)_{ij}, i = \overline{0, h_{col} - 1}, \qquad (5.1.20)$$

где h_{col} – число элементов массива $col^{(1)}$, не равных *None*.

5.4.4. Формируем массив из x координат левого верхнего угла областей, содержащихся в $col^{(2)}$:

$$X1_i = x_{ij}, \quad Y1_i = y_{ij}, \quad i = \overline{0, h_{col} - 1}. \qquad (5.1.21)$$

5.4.5. Применим метод наименьших квадратов для аппроксимации значений массива $X1$ полиномом 2-й степени:

$$\hat{k} = \arg\min_k \left(\sum_{m=0}^{h_{col}-1} \left(p(m,k) - X1_m \right)^2 \right), \qquad (5.1.22)$$

где $p(m,k) = k_2 m^2 + k_1 m + k_0$, $\hat{k} = \left(\hat{k}_2, \hat{k}_1, \hat{k}_0 \right)^T$.

5.4.6. Проверяем выполнение следующего условия:

$$\hat{k}_2 < 0.008. \qquad (5.1.23)$$

Если условие выполняется, то заменяем $None_j$ элемент в массиве PLF на следующий набор значений:

$$PLF_j = \left(\hat{k}_1, \hat{k}_0, Y1_0, Y1_{ed}\right), \qquad (5.1.24)$$

где $ed = h_{col} - 1$. Данные параметры будут задействованы на следующих этапах алгоритма.

5.4.7. Увеличиваем значение j на единицу и возвращаемся на шаг 5.4.2.

Результатом работы данного блока является массив PLF, содержащий информацию о характере смещения областей каждой последовательности.

5.5. Объединение последовательностей

В ходе формирования последовательностей в пункте 5.1 возможны ситуации, когда одну и ту же трещину на снимках описывают две или более различных последовательностей координат областей. Это происходит из-за отсутствия областей, описывающих трещину на некотором ряде снимков, при этом на снимках до и после данного ряда области присутствуют. В результате из областей на снимках, предшествующих данному ряду, будет сформирована первая последовательность, а из областей на снимках, последующих данному ряду, – вторая. Данные последовательности необходимо объединить в одну. Основная трудность решения задачи состоит в определении среди множества всех последовательностей, описывающих трещины на снимках, тех, которые следует объединить. Один из способов ее решения заключается в сравнении характеристик смещения, полученных в пункте 5.4 и сохраненных в массив PLF. Если параметры

смещения отличаются незначительно, то это указывает на то, что последовательности описывают одну и ту же трещину. Опишем применение данного способа.

5.5.1. Задаем начальное значение $s = 0.$

5.5.2. Если значение $s = w_c,$ где $w_c -$ число столбцов матрицы $CCr,$ то алгоритм завершает свою работу. Иначе проверяем следующее условие:

$$PLF_i = None. \tag{5.2.1}$$

Если оно выполняется, то возвращаемся на шаг 5.5.2, увеличив значение s на единицу. В ином случае извлекаем параметры из *s*-го элемента массива:

$$\left(\hat{k}_1^{(s)}, \hat{k}_0^{(s)}, Y1_0^{(s)}, Y1_{ed}^{(s)}\right) = PLF_s. \tag{5.2.2}$$

5.5.3. Задаем начальное значение $j = s + 1.$

5.5.4. Если значение $j = w_c,$ то возвращаемся на шаг 5.5.2, увеличив значение s на единицу. Иначе проверяем следующее условие:

$$PLF_j = None. \tag{5.2.3}$$

Если оно выполняется, то возвращаемся на шаг 5.5.4, увеличив значение j на единицу. В ином случае извлекаем параметры из j -го элемента массива:

$$\left(\hat{k}_1^{(j)}, \hat{k}_0^{(j)}, Y1_0^{(j)}, Y1_{ed}^{(j)}\right) = PLF_j. \tag{5.2.4}$$

5.5.5. Проверяем выполнение следующего условия с пороговыми значениями $th1 = 0.15,\ th0 = 25,\ thY = 35:$

$$\left(\left|\hat{k}_1^{(s)} - \hat{k}_1^{(j)}\right| < th1\right) \wedge \left(\left|\hat{k}_0^{(s)} - \hat{k}_0^{(j)}\right| < th0\right) \wedge \left(\left|Y1_{ed}^{(s)} - Y1_0^{(j)}\right| < thY\right). \tag{5.2.5}$$

Если оно выполняется, то это указывает на то, что данные последовательности описывают одну и ту же трещину. Объединим их в одну, перенеся все элементы, не равные *None* из j -го столбца матрицы CCr в s -й столбец следующим образом:

$$CCr_{is} = \begin{cases} CCr_{ij}, & CCr_{ij} \neq None, \\ CCr_{is}, & CCr_{ij} = None; \end{cases} \quad i = \overline{0, h_c - 1}, \qquad (5.2.6)$$

где h_c – число строк матрицы CCr.

После этого все элементы столбца *j* матрицы CCr заменяются на *None*:

$$CCr_{ij} = None, i = \overline{0, h_c - 1}. \qquad (5.2.7)$$

Далее возвращаемся на шаг 5.5.4, увеличив значение *j* на единицу.

Результатом работы данного блока является массив $CCr = \left(CCr_{ij}\right)^{h_c \times w_c}$ с объединёнными последовательностями.

5.6. Добавление новых областей в последовательности

Зная характер смещения областей из каждой последовательности, можно предсказать их положение на тех снимках, где области не были найдены. Имея информацию о приблизительных координатах области можно сравнить их с координатами областей, которые содержатся в множестве множеств NCr с ошибочно определенными областями. Если координаты отличаются незначительно, значит, область является частью последовательности. Опишем работу алгоритма.

5.6.1. В первую очередь добавим в множество множеств NCr области, содержащиеся в последовательностях с малым количеством элементов и не описывающие линейное смещение.

5.6.1.1. Задаем начальное значение $i = 0$.

5.6.1.2. Задаем начальное значение $j = 0$.

5.6.1.3. Если $i = h_c$, то переходим к шагу 5.6.2. Если значение $j = w_c$, то возвращаемся на шаг 5.6.1.2, увеличив значение i на единицу. В ином случае проверяем выполнение следующего условия для j-го столбца матрицы CCr:

$$\left(PLF_j \neq None\right) \wedge \left(Num_j > 30\right), \qquad (5.3.1)$$

где Num_j – число элементов j-го столбца, не равных *None*. Если условие выполняется, то возвращаемся на шаг 5.6.1.3, увеличив значение *j* на единицу.

5.6.1.4. Если алгоритм перешел к данному шагу, то это значит, что для j-й последовательности условие (5.3.1) не выполняется. Соответственно области, содержащиеся в ней, являются ошибочными и должны быть добавлены в множество $NCr^{(i+1)}$. Извлечем текущий элемент $(x, y, w, h)_{ij}$ из матрицы CCr и добавим его в $NCr^{(i+1)}$. Далее возвращаемся на шаг 5.6.1.3, увеличив значение j на единицу.

5.6.2. После добавления всех недостающих элементов в NCr переходим к сравнению координат областей из данного множества с вычисленными координатами.

5.6.2.1. Задаем начальное значение $j = 0$.

5.6.2.2. Если $j = w_c$, то алгоритм завершает свою работу. В ином случае фиксируем индекс ind первого элемента в j-м столбце матрицы CCr, не равного *None*.

5.6.2.3. Задаем начальное значение $c = ind - 1$.

5.6.2.4. Проверяем выполнение следующего условия с параметром $dist = 24$:

$$(c < 0) \vee (|ind - c| \geq dist). \qquad (5.3.2)$$

Если условие выполняется, то переходим к шагу 5.6.2.10.

5.6.2.5. Извлекаем все элементы множества NCr^{c+1} в массив $arrNCr^{c+1}$:

$$arrNCr^{c+1} = \left[(x,y,w,h)_0,(x,y,w,h)_1,...,(x,y,w,h)_{edN}\right], \qquad (5.3.3)$$

где $edN = N_{c+1}^{(NCr)}$ – число элементов в множестве NCr^{c+1}.

5.6.2.6. Задаем начальное значение $t = 0$.

5.6.2.7. Если $t = N_{c+1}^{(NCr)}$, то переходим к шагу 5.6.2.4, уменьшив значение c на единицу. В ином случае извлекаем соответствующий элемент $(x,y,w,h)_t^{c+1}$ из массива $arrNCr^{c+1}$.

5.6.2.8. Вычислим на изображении c координаты области, которые могли бы быть частью последовательности. Для этого воспользуемся информацией о характере смещения областей j-й последовательности из массива PLF:

$$\left(\hat{k}_1^{(j)}, \hat{k}_0^{(j)}, Y1_0^{(j)}, Y1_{ed}^{(j)}\right) = PLF_j; \qquad (5.3.4)$$

$$\hat{x} = \hat{k}_1^{(j)} c + \hat{k}_0^{(j)}, \quad \hat{y} = Y1_0^{(j)}, \qquad (5.3.5)$$

где $\hat{x}$ и $\hat{y}$ – координаты верхнего левого угла предполагаемой области.

5.6.2.9. Сравним координаты верхнего левого угла областей, полученные на шаге 5.6.2.7 и 5.6.2.8. Для этого проверим следующее условие с пороговыми значениями $xThresh = 20$, $yThresh = 15$:

$$\left(\left|\hat{x} - x_t^{(c+1)}\right| < xThresh\right) \wedge \left(\left|\hat{y} - y_t^{(c+1)}\right| < yThresh\right). \qquad (5.3.6)$$

Если оно выполняется, значит область может быть продолжением j-й последовательности. Добавим ее в соответствующий столбец матрицы CCr:

$$CCr_{cj} = (x, y, w, h)_t^{c+1}. \qquad (5.3.7)$$

Также обновим значение ind, приравняв его к c. Возвращаемся к шагу 5.6.2.4, уменьшив значение c на единицу.

Если условие не выполняется, то возвращаемся к шагу 5.6.2.7, увеличив значение t на единицу.

5.6.2.10. Задаем начальное значение $c = ind + Num_j + 1$.

5.6.2.11. Проверяем выполнение следующего условия с параметром $dist = 24$:

$$\left(c \geq h_c\right) \vee \left(\left|ind + Num_j - c\right| \geq dist\right). \qquad (5.3.8)$$

Если условие выполняется, то переходим к шагу 5.6.2.2, увеличив значение j на единицу.

5.6.2.12. Извлекаем все элементы множества NCr^{c+1} в массив $arrNCr^{c+1}$:

$$arrNCr^{c+1} = \left[(x, y, w, h)_0, (x, y, w, h)_1, ..., (x, y, w, h)_{edN}\right], \qquad (5.3.9)$$

где $edN = N_{c+1}^{(NCr)}$ – число элементов в множестве NCr^{c+1}.

5.6.2.13. Задаем начальное значение $t = 0$.

5.6.2.14. Если $t = N_{c+1}^{(NCr)}$, то переходим к шагу 5.6.2.11, увеличив значение c на единицу. В ином случае извлекаем соответствующий элемент $(x, y, w, h)_t^{c+1}$ из массива $arrNCr^{c+1}$.

5.6.2.15. Вычислим на изображении c координаты области, которые могли бы быть частью последовательности. Для этого воспользуемся информацией о характере смещения областей j-й последовательности из массива PLF:

$$\left(\hat{k}_1^{(j)}, \hat{k}_0^{(j)}, Y1_0^{(j)}, Y1_{ed}^{(j)}\right) = PLF_j; \tag{5.3.10}$$

$$\hat{x} = \hat{k}_1^{(j)} c + \hat{k}_0^{(j)}, \quad \hat{y} = Y1_{ed}^{(j)}, \tag{5.3.11}$$

где $\hat{x}$ и $\hat{y}$ – координаты верхнего левого угла предполагаемой области.

5.6.2.16. Сравним координаты верхнего левого угла областей, полученных на шаге 5.6.2.14 и 5.6.2.15. Для этого проверим следующее условие с пороговыми значениями $xThresh = 20$, $yThresh = 15$:

$$\left(\left|\hat{x} - x_t^{(c+1)}\right| < xThresh\right) \wedge \left(\left|\hat{y} - y_t^{(c+1)}\right| < yThresh\right). \tag{5.3.12}$$

Если оно выполняется, значит область может быть продолжением j-й последовательности. Добавим ее в соответствующий столбец матрицы CCr:

$$CCr_{cj} = (x, y, w, h)_t^{c+1}. \tag{5.3.13}$$

Также обновим значение ind, приравняв его к c. Возвращаемся к шагу 5.6.2.11, увеличив значение c на единицу.

Если условие не выполняется, то возвращаемся к шагу 5.6.2.14, увеличив t на единицу.

Результатом работы данного блока является матрица $CCr = \left(CCr_{ij}\right)^{h_c \times w_c}$ с новыми добавленными областями.

5.7. Порядок работы блоков алгоритма

Итоговая последовательность выполнения блоков алгоритма вторичного анализа снимков выглядит следующим образом:

1) формирование последовательностей из областей (п. 5.1);

2) заполнение пустых элементов в последовательностях (п. 5.2);

3) коррекция размеров областей (п. 5.3);

4) проверка линейного смещения областей (п. 5.4);

5) объединение последовательностей (п. 5.5);

6) заполнение пустых элементов в последовательностях (п. 5.2);

7) добавление новых областей в последовательности (п. 5.6);

8) заполнение пустых элементов в последовательностях (п. 5.2).

Результатом работы алгоритма вторичного анализа томографических снимков (рис. 5.1) является матрица

$$CCr = \left(CCr_{ij} \right)^{h_c \times w_c}, CCr_{ij} = \left(x_{ij}, y_{ij}, w_{ij}, h_{ij} \right) \cup None,$$

где h_c – количество строк в матрице, соответствующее числу анализируемых изображений за исключением первого и последнего снимка исходной последовательности; w_c – количество уникальных последовательностей областей содержания трещин, сформированных в ходе работы алгоритма; x_{ij}, y_{ij} – координаты верхнего левого угла области на i-м снимке, описывающих j-ю

трещину; w_{ij}, h_{ij} – высота и ширина области на i-м снимке, описывающих j-ю трещину. Значение *None* в ячейке матрицы указывает на то, что j-я трещина на i-м изображении отсутствует.

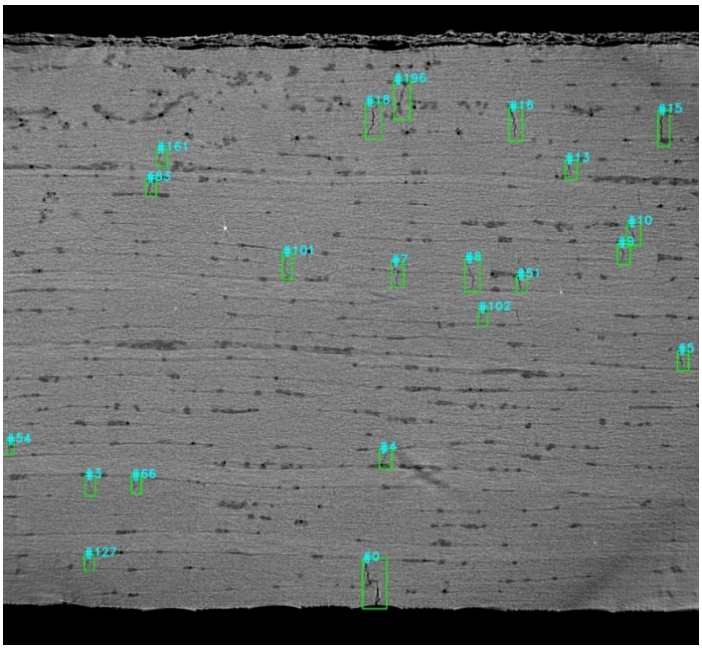


Рис. 5.1. Результат работы алгоритма вторичного анализа снимков для одного из изображений последовательности

6. Описание программной реализации

Программная реализация алгоритма поиска трещин выполнена на языке Python 3.10 в среде программирования PyCharm с использованием таких библиотек с открытым исходным кодом, как NumPy 2.0.2 [16] и OpenCV 4.10 [17]. На вход программе задается путь к папке с последовательностью изображений в формате JPG или PNG, которые требуется проанализировать. На выходе программа возвращает последовательность изображений в формате PNG с выделенными областями содержания трещин и сохраняет их в указанную пользователем папку с именами, соответствующими исходным изображениям.

Программная реализация состоит из следующих блоков и функций:

6.1. *getOutCords* – основной блок программы, в котором содержатся функции для подготовки изображений к анализу и обработке.

6.1.1. *getCoords*() – функция для подготовки изображений к первичному анализу.

6.1.2. *mainCracks*() – функция, внутри которой происходит вызов всех остальных функций программы.

6.2. *markCracks* – блок программы, в котором выполняется первичный анализ снимков.

6.2.1. *markCracks*() – функция, в которой выполняется первичный анализ снимков.

6.3. *markCorrection* – блок программы, в котором содержится большая часть функций, выполняющих вторичный анализ снимков.

6.3.1. *mCorrection*() – функция, в которой производится формирование последовательностей из областей.

6.3.2. *catCracks*() – функция, в которой выполняется заполнение пустых элементов в последовательностях.

6.3.3. *chCnCracks*() – функция, в которой производится объединение последовательностей.

6.3.4. *connCracks*() – функция, в которой выполняется добавление новых областей в последовательности.

6.4. *boxSizeCorrection* – блок программы, в котором производится коррекция размеров области содержания трещин и проверка линейного смещения.

6.4.1. *boxSizeCorr*() – функция, в которой производится коррекция размеров области содержания трещин (п. 5.3) и проверка линейного смещения (п. 5.4).

Порядок работы блоков и функций программы представлен на рис. 6.1.

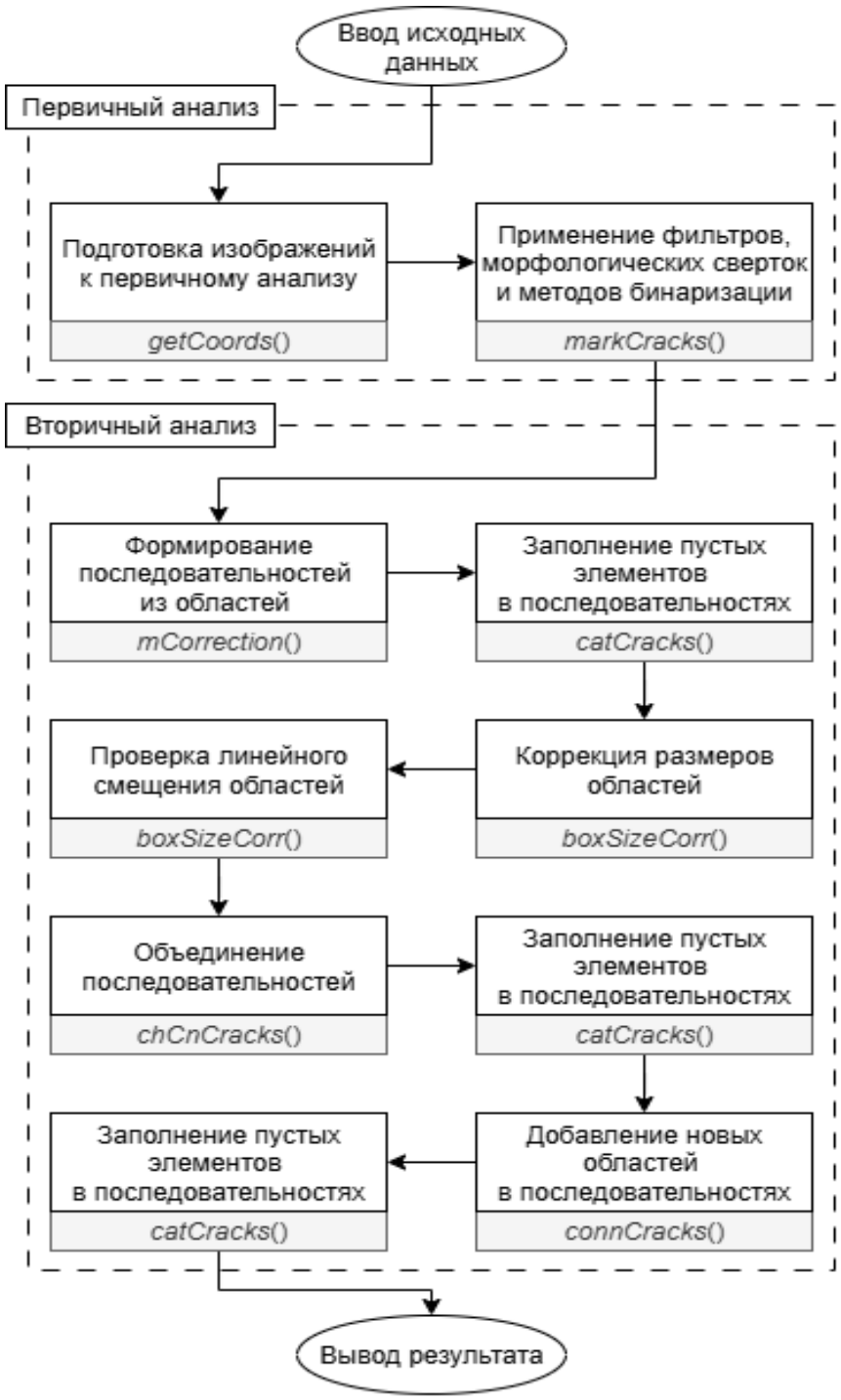


Рис. 6.1. Взаимодействие блоков и функций программы

Развитие описанных в статье методов обработки изображений приведено в [18–22]. Они могут быть использованы для улучшения качества обработки.

7. Заключение

В данной работе предложен алгоритм для автоматического обнаружения поперечных микротрещин в слоистых композитных материалах по томографическим изображениям. Двухэтапный подход включает первичный анализ с применением фильтров и методов бинаризации для формирования набора потенциальных областей трещин, и вторичный анализ для уточнения и отслеживания трещин с коррекцией размеров и интерполяцией пропущенных дефектов. Алгоритм реализован на Python с использованием библиотек NumPy и OpenCV, что позволяет эффективно обрабатывать изображения и генерировать структурированные данные для дальнейшего анализа. Разработанный инструмент улучшает объективность и скорость анализа микроповреждений, что важно для оценки целостности и долговечности изделий в авиа-, автомобилестроении и ветроэнергетике.

Список источников

1. Gibson L.J., Ashby M.F. Cellular Solids: Structure and Properties. Cambridge Solid State Science Series. Cambridge University Press, 1997. 380 p.

2. Fleck N.A., Deshpande V.S., Ashby M.F. Micro-Architected Materials: Past, Present and Future // Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences. 2010. Vol. 466, No. 2121. P. 2495–2516. DOI: 10.1098/rspa.2010.0215

3. Karapiperis K., Kochmann D.M. Prediction and Control of Fracture Paths in Disordered Architected Materials Using Graph Neural Networks // Communications Engineering. 2023. DOI: 10.1038/s44172-023-00085-0

4. Alireza Taherzadeh-Fard, Alejandro Cornejo et al. A Rule of Mixtures Approach for Delamination Damage Analysis in Composite Materials // Composites Science and Technology. 2023. Vol. 242, P. 110160. DOI: 10.1016/j.compscitech.2023.110160

5. Zhang P. et al. Quantitative Analysis of Micro-Crack Precursors in Carbon/Epoxy Composites Using SEM // Composites Science and Technology. 2023. Vol. 230, P. 109750.

6. Liu Y., St-Pierre L., Fleck N.A., Deshpande V.S., Srivastava A. High Fracture Toughness Micro-Architected Materials // Journal of the Mechanics and Physics of Solids. 2020. Vol. 143, P. 104060. DOI: 10.1016/j.jmps.2020.104060

7. Demeshko Y., Bobyr M.. Damage Models of Composite Materials // Journal Mechanics and Advanced Technologies. 2025. DOI: 10.20535/2521-1943.2025.9.1(104).313947

8. Крупенин А.М. Новый метод построения кинетической диаграммы по испытаниям на скорость роста трещины усталости // Труды МАИ. 2024. № 136. URL: https://trudymai.ru/published.php?ID=180667

9. 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

10. Трусфус М.В., Абдуллин И.Н. Алгоритм обнаружения маркерных изображений для вертикальной посадки беспилотного летательного аппарата // Труды МАИ. 2021. № 116. URL: https://trudymai.ru/published.php?ID=121099. DOI: 10.34759/trd-2021-116-13

11. Зотов А.А., Резниченко В.И. Композиционные материалы: классификация, состав, структура и свойства. - М.: Вузовская книга, 2023. – 132 с.

12. Шапиро Л., Стокман Дж. Компьютерное зрение. – М.: БИНОМ. Лаборатория знаний, 2015. – 763 с.

13. Гонсалес Р., Вудс Р. Цифровая обработка изображений. – М.: Техносфера, 2024. – 1104 с.

14. Suzuki S., K. Abe. Topological structural analysis of digitized binary images by border following // Computer Vision, Graphics, and Image Processing. 1985. Vol. 30, No.1. P. 32–46. DOI: 10.1016/0734-189X(85)90016-7

15. Макаров Р.И., Хорошева Е.Р. Методы анализа данных. – Владимир: Изд-во ВлГУ, 2021. – 216 с.

16. NumPy documentation. URL: https://numpy.org/doc/

17. OpenCV: OpenCV modules. URL: https://docs.opencv.org/4.x/

18. C.-C. Chen, C.-H. Peng. Topology-Preserving Downsampling of Binary Images. In book: Computer Vision – ECCV 2024. P. 416-431. DOI: 10.1007/978-3-031-72661-3_24

19. Jain R., Kasturi R., Schunck B. Machine Vision. Computer science series. McGraw-Hill, 1995. 549 p.

20. Abu-Ain W., Abdullah S.N.H.S., Bataineh B., Abu-Ain T., Omar K. Skeletonization algorithm for binary images // Procedia Technology. 2013. Vol. 11, P. 704–709. DOI: 10.1016/j.protcy.2013.12.248

21. Boudaoud L.B., Sider A., Tari A. A new thinning algorithm for binary images // 2015 3rd international conference on control, engineering & information technology (CEIT). 2015. DOI: 10.1109/CEIT.2015.7233099

22. Wang J., Kosinka J., Telea A. Spline-based medial axis transform representation of binary images // Computers & Graphics. 2021. Vol. 98, P. 165–176. DOI: 10.1016/j.cag.2021.05.012

References

1. Gibson L.J., Ashby M.F. *Cellular Solids: Structure and Properties*. Cambridge Solid State Science Series. Cambridge University Press, 1997. 380 p.

2. Fleck N.A., Deshpande V.S., Ashby M.F. Micro-Architected Materials: Past, Present and Future. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*. 2010. Vol. 466, No. 2121. P. 2495–2516. DOI: 10.1098/rspa.2010.0215

3. Karapiperis K., Kochmann D.M. Prediction and Control of Fracture Paths in Disordered Architected Materials Using Graph Neural Networks. *Communications Engineering*. 2023. DOI: 10.1038/s44172-023-00085-0

4. Alireza Taherzadeh-Fard, Alejandro Cornejo et al. A Rule of Mixtures Approach for Delamination Damage Analysis in Composite Materials. *Composites Science and Technology*. 2023. Vol. 242, P. 110160. DOI: 10.1016/j.compscitech.2023.110160

5. Zhang P. et al. Quantitative Analysis of Micro-Crack Precursors in Carbon/Epoxy Composites Using SEM. *Composites Science and Technology*. 2023. Vol. 230, P. 109750.

6. Liu Y., St-Pierre L., Fleck N.A., Deshpande V.S., Srivastava A. High Fracture Toughness Micro-Architected Materials. *Journal of the Mechanics and Physics of Solids.* 2020. Vol. 143, P. 104060. DOI: 10.1016/j.jmps.2020.104060

7. Demeshko Y., Bobyr M. Damage Models of Composite Materials. *Journal Mechanics and Advanced Technologies*. 2025. DOI: 10.20535/2521-1943.2025.9.1(104).313947

8. Krupenin A.M. New method for construction kinetic diagram by fatigue crack growth rate tests. *Trudy MAI*. 2024. No. 136. (In Russ.). URL: https://trudymai.ru/eng/published.php?ID=180667

9. 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. (In Russ.).

URL: https://trudymai.ru/eng/published.php?ID=181884

10. Trusfus M.V., Abdullin I.N. Marker images detection algorithm for the unmanned aerial vehicle vertical landing. *Trudy MAI*. 2021. No. 116. (In Russ.). URL: https://trudymai.ru/eng/published.php?ID=121099. DOI: 10.34759/trd-2021-116-13

11. Zotov A.A., Reznichenko V.I. *Kompozitsionnye materialy: klassifikatsiya, sostav, struktura i svoistva* (Composite materials: classification, composition, structure and properties). Moscow: Vuzovskaya kniga Publ., 2023. 132 p.

12. Shapiro L., Stokman Dzh. *Komp'yuternoe zrenie* (Computer vision). Moscow: BINOM. Laboratoriya znanii Publ., 2015. 763 p.

13. Gonsales R., Vuds R. *Tsifrovaya obrabotka izobrazhenii* (Digital image processing). Moscow: Tekhnosfera Publ., 2024. 1104 p.

14. Suzuki S., K. Abe. Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing*. 1985. Vol. 30, No.1. P. 32–46. DOI: 10.1016/0734-189X(85)90016-7

15. Makarov R.I., Khorosheva E.R. *Metody analiza dannykh* (Data Analysis Methods). Vladimir: Izd-vo VlGU Publ., 2021. 216 p.

16. *NumPy documentation*. URL: https://numpy.org/doc/

17. *OpenCV: OpenCV modules*. URL: https://docs.opencv.org/4.x/

18. C.-C. Chen, C.-H. Peng. *Topology-Preserving Downsampling of Binary Images. In book: Computer Vision – ECCV 2024*. P. 416-431. DOI: 10.1007/978-3-031-72661-3_24

19. Jain R., Kasturi R., Schunck B. *Machine Vision. Computer science series*. McGraw-Hill, 1995. 549 p.

20. Abu-Ain W., Abdullah S.N.H.S., Bataineh B., Abu-Ain T., Omar K. Skeletonization algorithm for binary images. *Procedia Technology*. 2013. Vol. 11, P. 704–709. DOI: 10.1016/j.protcy.2013.12.248

21. Boudaoud L.B., Sider A., Tari A. A new thinning algorithm for binary images. *2015 3rd international conference on control, engineering & information technology (CEIT)*. 2015. DOI: 10.1109/CEIT.2015.7233099

22. Wang J., Kosinka J., Telea A. Spline-based medial axis transform representation of binary images. *Computers & Graphics*. 2021. Vol. 98, P. 165–176. DOI: 10.1016/j.cag.2021.05.012

Статья поступила в редакцию 13.07.2025
Одобрена после рецензирования 25.07.2025
Принята к публикации 25.08.2025
The article was submitted on 13.07.2025; approved after reviewing on 25.07.2025; accepted for publication on 25.08.2025