УДК 004.932.

ОСОБЕННОСТИ СОЗДАНИЯ АВТОМАТИЗИРОВАННОЙ ПРОГРАММЫ ДЛЯ КОНТРОЛЯ СВОЙСТВ МАТЕРИАЛОВ С ИСПОЛЬЗОВАНИЕМ КОМПЬЮТЕРНЫХ ТЕХНОЛОГИЙ

С. И. РОГОВСКИЙ
(Представлено: канд. физ.-мат. наук, доц. С. А. ВАБИЩЕВИЧ)

Рассмотрены основные этапы обработки цифровых изображений, реализованные с использованием библиотеки OpenCV на языке программирования Python. Результаты создания автоматизированной программы обработки изображений могут быть полезны для совершенствования контроля качества материалов и оптимизации производственных процессов.

Введение. В современном мире информационные технологии прочно интегрированы в различные сферы человеческой деятельности, и они представляют уникальные возможности для оптимизации процессов, улучшения точности анализа данных и устранения субъективных факторов при оценке результатов. Одним из передовых направлений в информационных технологиях, имеющим значительное влияние на промышленность, является область компьютерного зрения. Компьютерное зрение открывает новые перспективы в интеграции автоматизированных систем для анализа и контроля качества в полупроводниковой промышленности. Эта технология позволяет машинам "видеть" и анализировать микроструктуры полупроводниковых материалов, что имеет важное значение для выявления дефектов, трещин и адгезионных проблем. Кроме того, компьютерное зрение способно обрабатывать большие объемы данных в реальном времени, что делает его мощным инструментом для оптимизации качества и производительности в полупроводниковой промышленности.

Цель работы состояла в разработке программного продукта для распознавания цифровых изображений, позволяющего проводить на основании анализа фотоснимков расчет прочностных характеристик материала [1]. В качестве среды разработки программного продукта был выбран язык программирования Python. Схема обработки и анализа данных цифровых изображений сводилась к следующим этапам: анализ входных данных изображения; обработка входных данных; использование медианного фильтра; сегментация изображения; поиск контуров изображения; отображение контуров на изображении; вычисление геометрических параметров объекта фотографирования.

На первом этапе в алгоритме подаются входные данные в виде цифрового изображения, представляющего микроструктуру поверхности полупроводникового материала после испытания на микротвердость (рис.). Следующим шагом является использование медианного фильтра на входном изображении. Этот фильтр помогает уменьшить шум на изображении, сохраняя при этом важные детали. После медианной фильтрации изображения выполняется сегментация. Этот шаг позволяет разделить искомые отпечатки от фона. Завершающим этапом алгоритма является поиск контуров объектов на сегментированном изображении. Это позволяет выделить границы объектов и дефектов, делая их доступными для анализа. Важно отметить, что этот этап помогает идентифицировать форму и размер объектов в микроструктуре.

Поиск контуров – это важный этап обработки цифровых изображений, который можно реализовать с использованием библиотеки OpenCV на языке программирования Python [2].

Для начала, изображение должно быть преобразовано в оттенки серого, так как большинство методов анализа изображений работают с контурами именно в этом цветовом режиме. Для перевода изображения в оттенки серого используется функция cv2.cvtColor(image, cv2.COLOR_BGR2GRAY). Эта функция предназначена для изменения цветового пространства изображения. В конкретном случае, функция cv2.cvtColor() используется для преобразования цветного изображения в оттенки серого, что обычно называется "градациями серого" или "монохромным изображением". Вот как устроена эта функция:

- image: исходное цветное изображение, которое вы хотите преобразовать в оттенки серого.
- cv2.COLOR_BGR2GRAY: константа, которая указывает, какое цветовое пространство должно быть применено к изображению. cv2.COLOR_BGR2GRAY используется для преобразования изображения в оттенки серого. Эта константа представляет собой код для переключения между различными цветовыми пространствами.

Когда функция cv2.cvtColor() вызывается с этими аргументами, она выполняет следующие действия:

- принимает исходное цветное изображение.
- применяет алгоритм, который конвертирует каждый пиксель изображения из цветного пространства (обычно BGR или RGB) в соответствующую яркость в оттенках серого. Это делается путем усреднения значений красной, зеленой и синей компонент цвета в каждом пикселе.
- возвращает новое изображение, где каждый пиксель представлен одним значением яркости (или оттенка серого) в диапазоне от 0 (черный) до 255 (белый).

После преобразования в оттенки серого, функция cv2.FindContours() применяется к изображению для поиска контуров. Функция cv2.findContours() является частью библиотеки OpenCV и используется для поиска контуров (границ) объектов на бинарных изображениях или изображениях в оттенках серого. Контур – это непрерывная кривая, которая образует границу объекта или области на изображении. Функция cv2.findContours() возвращает список контуров, которые могут быть далее использованы для различных аналитических и обработочных задач. Синтаксис функции: *contours, hierarchy = cv2.findContours(image, mode, method[, contours[, hierarchy[, offset]]]).*

- image: входное бинарное изображение (часто в оттенках серого или бинарное изображение после применения пороговой фильтрации), на котором будут искаться контуры.
- mode: параметр определяет, какие контуры будут найдены. Он может принимать одно из следующих значений:
  - cv2.RETR_EXTERNAL: Находит только внешние контуры.
  - cv2.RETR_LIST: Находит все контуры и хранит их в списке.
  - cv2.RETR_CCOMP: Находит все контуры и организует их в двухуровневую иерархию.
  - cv2.RETR_TREE: Находит все контуры и строит полную иерархическую структуру.
- method: этот параметр определяет способ аппроксимации контуров. Он может принимать одно из следующих значений:
  - cv2.CHAIN_APPROX_NONE: Сохраняет все точки контура.
  - cv2.CHAIN_APPROX_SIMPLE: Сжимает горизонтальные, вертикальные и диагональные сегменты и оставляет только их конечные точки.
- contours: переменная, в которой будет храниться список найденных контуров.
- hierarchy: переменная, в которой будет храниться информация об иерархии контуров (если используется параметр cv2.RETR_CCOMP или cv2.RETR_TREE).
- offset: необязательный параметр, который может использоваться для смещения контуров.

Полученные контуры, представленные в виде списка точек, могут быть отображены на исходном цветном изображении. Для этого используется функция cv2.drawContours(). Функция cv2.drawContours() в библиотеке OpenCV используется для рисования контуров на изображении. Она позволяет визуализировать контуры объектов, найденные с помощью функции cv2.findContours(). Эта функция обычно используется для отладки, визуализации результатов анализа изображений и создания аннотаций. Синтаксис функции cv2.drawContours(): cv2.drawContours(image, contours, contourIdx, color[, thickness[, lineType[, hierarchy[, maxLevel[, offset]]]]]).

- image: цветное изображение, на котором вы хотите нарисовать контуры.
- contours: список контуров, который вы хотите нарисовать на изображении. Этот список обычно возвращается функцией cv2.findContours().
- contourIdx: индекс контура в списке contours, который вы хотите нарисовать. Если вы хотите нарисовать все контуры, установите contourIdx равным -1.
- color: цвет, которым будут нарисованы контуры, обычно задается в формате BGR (Blue, Green, Red) или RGB, в зависимости от цветовой модели изображения.
- thickness (необязательный): толщина линии для рисования контура. По умолчанию установлено значение 1. Если вы хотите, чтобы линия была толще, увеличьте это значение.
- lineType (необязательный): тип линии, который будет использоваться для рисования контура. По умолчанию используется cv2.LINE_8, что означает восьми-связную линию. Вы также можете использовать cv2.LINE_4 или cv2.LINE_AA (антиалиасинг) для разных стилей линий.
- hierarchy (необязательный): параметр содержит информацию об иерархии контуров, если она была извлечена с помощью функции cv2.findContours().
- maxLevel (необязательный): максимальный уровень иерархии контуров, который будет рисоваться. По умолчанию установлено значение 0, что означает рисование всех уровней.
- offset (необязательный): смещение, которое можно применить к контурам перед рисованием.

Иногда на изображении могут обнаруживаться мелкие и ненужные контуры, которые не несут значимой информации и могут помешать дальнейшему анализу. Для фильтрации таких контуров можно использовать функцию cv2.contourArea(). Функция cv2.contourArea() в библиотеке OpenCV используется для вычисления площади контура. Эта функция принимает один аргумент – контур (список точек) и возвращает площадь ограниченной им области. Синтаксис функции cv2.contourArea(): *area = cv2.contourArea(contour).*

- contour: список точек, представляющих контур объекта на изображении. Обычно этот контур извлекается с помощью функции cv2.findContours().
- area: переменная, в которой будет храниться вычисленная площадь контура.

После вычисления площади контуры могут быть отсортированы по ее величине, и небольшие контуры могут быть исключены из рассмотрения, если их площадь слишком мала для значимого влияния на результаты анализа.

На рисунке продемонстрирован результат обработки изображения систематизированных в соответствии с уровнем нагрузки (50, 20, 10 и 5 граммов). Порядок расположения изображений начинается с верхних и заканчивается нижними изображениями в соответствии с уровнем нагрузки.

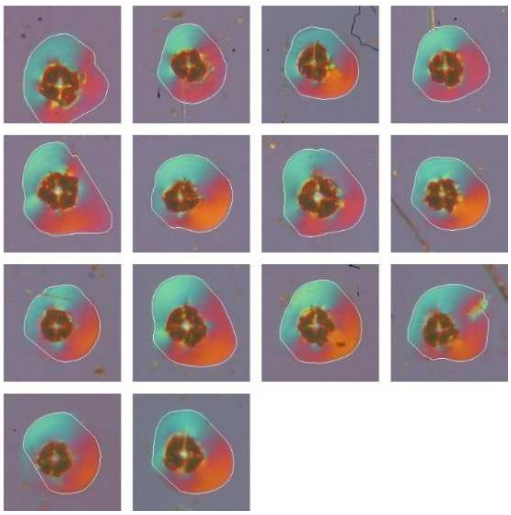

Рисунок – Результаты отображения контуров зоны деформации полимера после испытания на микротвердость

С развитием информационных технологий и компьютерного зрения, в частности, алгоритмы обработки цифровых изображений играют все более важную роль в полупроводниковой промышленности. Вот несколько перспектив использования такого алгоритма:

1. *Контроль качества материалов.* Алгоритм обработки цифровых изображений позволяет автоматически выявлять дефекты и адгезионные проблемы на поверхности полупроводниковых материалов. Это снижает риски производства бракованных изделий и повышает надежность полупроводниковых устройств.
2. *Оптимизация производства*. Интеграция алгоритма в производственные линии позволяет автоматизировать процессы контроля качества. Это снижает необходимость вручную проверять каждый образец и ускоряет производственные циклы.
3. *Повышение точности*. Алгоритмы обработки цифровых изображений способны вычислять параметры материала, такие как размеры трещин и характеристики адгезии, с высокой точностью и воспроизводимостью. Это позволяет более точно определять прочностные характеристики материалов.
4. *Анализ данных.* Полученные данные о микроструктуре и дефектах могут быть использованы для анализа и оптимизации процессов производства. Это позволяет выявлять тренды и понимать, какие параметры материала влияют на его качество.
5. *Сокращение затрат*. Автоматический контроль качества с использованием алгоритмов обработки изображений может снизить затраты на трудозатраты и уменьшить количество брака, что экономически выгодно для предприятий.

Заключение. Алгоритмы обработки цифровых изображений предоставляют большие перспективы для полупроводниковой промышленности, позволяя более точно контролировать качество материалов и оптимизировать производственные процессы. Это способствует повышению надежности полупроводниковых устройств и снижению затрат на производство.

ЛИТЕРАТУРА

1. Vabishchevich, S.A. Effect of ion implantation on the adhesion of positive diazoquinone-novolak photoresist films to single-crystal silicon/ S.A.Vabishchevich [et al.] // Journal of Surface Investigation. X-Ray, Synchrotron and Neutron Techniques – 2020. – V.14, № 6. – P.1351-1356. *DOI: 10.1134/S1027451020060476*
2. Bradski, G. Learning OpenCV. Computer vision with the OpenCV library/ G.Bradski, A.Kaehler //O'Reilly Media, Inc., – 2008.