Распознавание дефектов на металлических сплавах с помощью алгоритмов компьютерного зрения OpenCV

И.А. Балеев[1], А.Н. Земцов[2], М. И. Зыбин[3], В. А. Смирнов[4]

Волгоградский государственный технический университет

Аннотация: Целью данной работы является разработка алгоритма для распознавания усадочных дефектов на металлических сплавах. Описываются шаги по обработке изображения. При разборе алгоритма, для каждого шага представлено описание с последующей программной реализацией. Заключительным этапом, после использования методов алгоритма, является подсчет контуров дефектов. Для демонстрации работоспособности программного обеспечения были взяты случайные фотографии металлического тела в разрезе, где можно непосредственно наблюдать дефекты в виде газовых пор. Программное обеспечение при правильном подборе входных данных обрабатывает изображение с высокой точностью, минимизируя погрешности вычисления дефектов. Использование предложенных алгоритмов сокращает время диагностики. В статье была приведена сравнительная характеристика ручного и автоматизированного методов, показывающая эффективность второго метода по сравнению с первым. Для написания программного обеспечения был использован язык программирования Python 3.7, а также библиотека алгоритмов компьютерного зрения OpenCV.
Ключевые слова: обнаружение дефектов, алгоритмы компьютерного зрения, OpenCV, металлические дефекты, усадочные дефекты.

При обработке изображений одной из центральных является задача распознавания заданных объектов. Алгоритмы обработки изображений все чаще используются в научных и прикладных исследованиях в различных областях человеческой деятельности [1]. В связи с этим, задачи анализа изображений - одни из наиболее важных при обработке изображений. С помощью алгоритма распознавания круглых контуров можно выявить количество газовых пор в сварных швах на фотографии [2].

Анализ литейных предприятий показывает, что производство отливок терпит значительные убытки от брака литья - многомиллиардные [3].

Процесс разрушения деталей машин начинается с образования на их поверхностях микроскопических трещин [4]. В большинстве случаев дефекты имеют вид округлых пятен, но встречаются и крупные скопления дефектов разнообразной формы [5]. Человек, занимающийся подсчетом этих

пятен на фотографии, тратит большое количество времени. При этом, в силу человеческого фактора, возможна погрешность при расчете [6].

Использовав данную систему автоматизации, можно значительно сэкономить время, а также минимизировать количество ошибок.

Предложенный метод по обработке изображений будет состоять из нескольких основных этапов.

Этап 1. Преобразование цветового пространства – перевод изображения из RGB в оттенки серого, при этом на выходе получаем одноканальную версию изображения [7]. В нашем случае используем взвешенный метод. Использование данного метода необходимо для целесообразного распределения цветов на входящем изображении. Если мы будем использовать метод среднего значения – качество выходного изображения после обработки может быть несколько хуже.

Например, при средних значениях цветов, равных 33%, некоторые изображения могут казаться намного темнее, чем ожидалось, и поэтому, в нашем случае, некоторые газовые поры могут не проявиться после преобразования.

Исходя из этого, формула преобразования из RGB в оттенки серого имеет вид:

$$Gray = (0.299 * R) + (0.587 * G) + (0.114 * B),$$

где R, G, B – основные цвета аддитивной модели.

Эти три конкретных коэффициента представляют сосредоточение цветов на изображении в процентном соотношении.

Этап 2. Удаление шума с помощью пороговой функции [8]. После получения изображения с оттенками серого, необходимо отфильтровать изображение, убрав на нем шумы, то есть убрать слишком большие значения или слишком маленькие значения в одноканальном массиве [9].

Функция порогового значения имеет вид:

$$\mathrm{d}st(x,y) = \begin{cases} maxval & \text{если } src(x,y) > thresh \\ 0 & \text{в противном случае} \end{cases},$$

где maxval – это максимальное значение одноканального массива.

Функция применяет пороговое значение, и, если оно превысит данный порог, функция применит фильтр к нему.

Этап 3. Устранение дефектов изображения. Получив отфильтрованное изображение посредством пороговой функции, можно наблюдать в некоторых местах пересвечивание пикселей или же, наоборот, затемнение. Поэтому необходимо сгладить перепады соседних пикселей.

Для решения данной проблемы воспользуемся двухсторонним фильтром. Двухсторонний фильтр вычисляет средневзвешенное значение пикселей в окрестности, в котором вес уменьшается с расстоянием от центра соседства. Близкие пиксели, вероятно, будут иметь одинаковые значения, и поэтому целесообразно усреднять их вместе.

Двусторонний фильтр определяется как:

$$I^{filtered}(x) = \frac{1}{w_p}\sum_{x_y \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(||x_i - x||),$$

где W_p имеет вид:

$$W_p = \sum_{x_y \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(||x_i - x||),$$

где $I^{filtered}$ — это отфильтрованное изображение;

I – исходное входное изображение для фильтрации;

x – координаты текущего пикселя для фильтрации;

Ω – окно, сфокусированное на координате x;

f_r – ядро диапазона для сглаживания разностей интенсивностей;

g_s – является пространственным ядром для сглаживания различий в координатах;

W_p – срок нормализации (преобразование нескольких пикселей в один).

Этап 4. Бинаризация изображения – приведение изображения к такому виду, когда каждый пиксель кодируется либо единицей, либо нулем. Этот шаг необходим, так как некоторые последующие функции OpenCV используют бинарное изображение, как один из аргументов [10]. Выбор порога сегментации выполняется по методу Оцу. Так как после предыдущего шага на изображении присутствуют два класса пикселей: фоновые и объектные, – метод Оцу подходит для определения границы бинаризации лучше: выше будет межклассовая дисперсия. В итоге получим следующее изображение, которое продемонстрируем на рисунке 1.

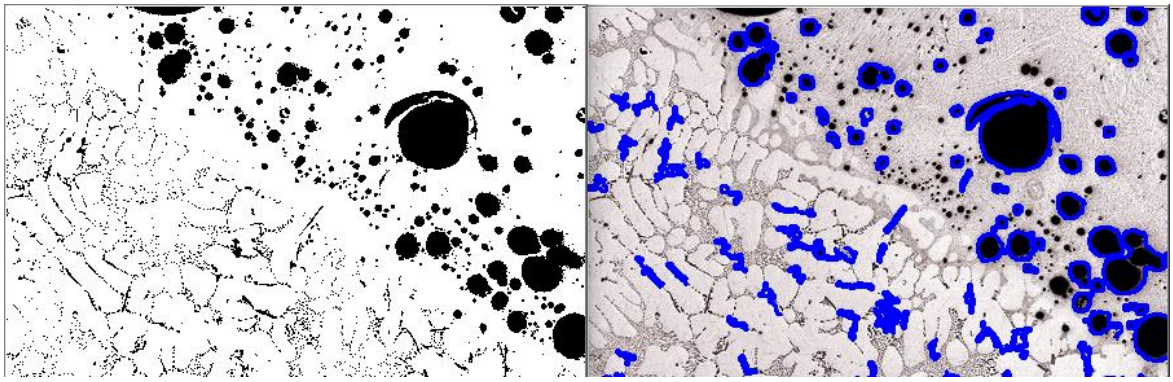

Рис. 1. - Бинаризация изображения.

Алгоритм выделения границ, или иначе - обнаружения границ работает после того, как качество изображения улучшено перечисленными выше методами. При обнаружении краев мы находим границы или края объектов в изображении, определяя, где яркость изображения резко меняется. Обнаружение краев может быть использовано для извлечения структуры объектов в изображении. Если мы заинтересованы в количестве, размере, форме или относительном расположении объектов на изображении, обнаружение краев позволяет нам сосредоточиться на наиболее полезных частях изображения, игнорируя части изображения, которые нам не помогут.

После того, как края объектов в изображении были найдены, можно использовать полученную информацию для поиска контуров изображения.

Этап 5. В OpenCV используется алгоритм топологического структурного анализа бинарных изображений. Алгоритм предполагает нахождение контуров с учетом вложенности, то есть способен определить, когда в контур одного объекта вложен другой.

С помощью полученных контуров можно вести подсчет количества объектов на изображении, проводить измерения размеров объектов, классифицировать их формы.

Для разработки приложения будем использовать библиотеку компьютерного зрения и машинного обучения с открытым исходным кодом OpenCV. Данная библиотека предполагает поддержку интерфейсов на различных языках, включая язык программирования Python.

Прежде всего необходимо выполнить установку библиотеки cv2 с помощью пакетного менеджера pip.

После установки удостоверимся в работоспособности библиотеки, импортировав ее, а далее применим функционал считывания изображения. Функция imread загружает изображение из указанного файла и возвращает его. В нашем случае файл будет храниться в корневой директории.

Отобразим наше изображение в десктопном окне, где будем проводить бинаризацию с помощью ползунка. У данного ползунка будет порог 255, потому что в дальнейшем будет использоваться пороговая функция threshold, все пиксели, которые темнее 127, заменены на 0, а все, которые ярче 127, - на 255. На выходе получим черно-белое изображение.

Далее, функция конвертирования цветов cvtColor () преобразует входное изображение из одного цветового пространства в другое. В случае преобразования из цветового пространства RGB, порядок каналов должен быть указан явно (RGB или BGR). Цветовой формат по умолчанию в OpenCV часто упоминается как RGB, но на самом деле это BGR (байты обращены). Таким образом, первый байт в стандартном (24-битном) цветном

изображении будет 8-битным синим компонентом, второй байт будет зеленым, а третий байт будет красным. Четвертый, пятый и шестой байты будут вторым пикселем (синий, зеленый, красный).

Далее нам следует установить порог серого изображения. Исходя из приведенных аргументов, будет проводиться бинаризация изображения.

Пройдя бинаризацию, необходимо воспользоваться алгоритмом выявления границ, так как в срезе металла возможны наложения дефектов друг на друга. OpenCV предоставляет функцию для устранения проблемы. Данная функция реализует многоступенчатый подход обнаружения краев. После применения алгоритма обнаружения краев произведем поиск контуров с помощью функции findContours (). Из функции следует, что мы собираемся сгруппировать контуры в многоуровневую иерархию и применить к ним метод аппроксимации.

Разработанный алгоритм по диагностике усадочных дефектов был протестирован и сравнен с ручным методом подсчета, при котором специалисты вручную подсчитывали количество дефектов на металлическом бруске на производстве. Также ручной метод был применен в идеальных условиях для абсолютно точного подсчета.

Было проведено несколько экспериментов, причем в ходе каждого эксперимента применялось три метода: ручной метод, ручной метод в идеальных условиях и метод распознавания с помощью компьютерного зрения. Для обычного ручного метода были взяты подсчеты трех разных специалистов и приведены к среднеарифметическому показателю.

Проведем сравнительную характеристику трех методов, отобразив результаты на графике. Для исследования были взяты детали с приблизительно одинаковой площадью поражения, чтобы избежать большой разницы между замерами. По оси у укажем количество дефектов, по оси х - количество замеров.

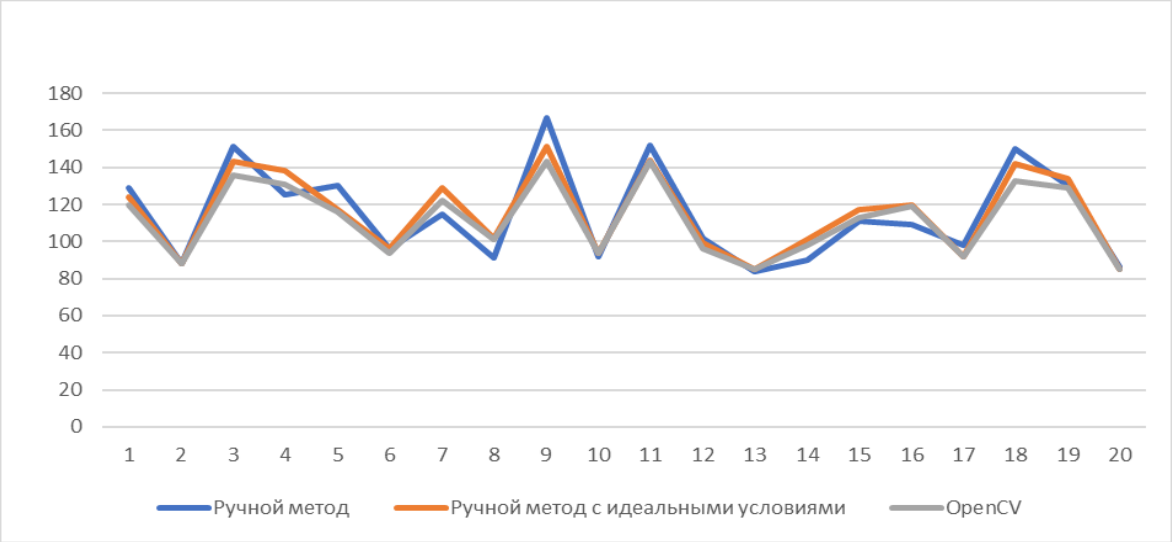

Рис. 2 – Сравнение методов по распознаванию

Исходя из графика, можно сделать вывод, что метод OpenCV в основном недосчитывал количество дефектов, в отличие от ручного метода с идеальными условиями. Прежде всего, это связано с засвечиванием пикселей на изображении при использовании фотооборудования. Что касается обычного ручного метода, здесь можно наблюдать как недосчет, так и пересчет усадочных изъян, при этом погрешность намного больше. Для ручного метода погрешность составляет 12.7%, для OpenCV – 3.4%. Алгоритм OpenCV показывает нам значительное повышение точности и минимизацию погрешностей.

Разработанный алгоритм распознавания дефектов на металлических телах позволяет определять контуры на изображении, что значительно облегчает процесс подсчета.

В нынешний момент алгоритм распознавания контуров библиотеки OpenCV отличается высокой точностью, что помогает избавиться от погрешности при анализе.

Эксперименты по проведению автоматизированного распознавания дефектов выполнялись с помощью программного обеспечения Python 3.7 с использованием библиотеки OpenCV. Результаты программной реализации

продемонстрировали работоспособность и эффективность данного алгоритма.

Литература

1. Хамухин А.В. Система тестирования алгоритмов компьютерного зрения // Актуальные проблемы современной науки. Москва, 2014. С. 170-174.

2. Игнатьев А.А., Бахтеев А.Р. Автоматизация распознавания дефектов шлифованных деталей в системе мониторинга технологического процесса производства подшипников // Вестник саратовского государственного технического университета. Саратов, 2006. С. 136-142.

3. Воронин Ю.Ф. Примеры распознавания и ликвидации дефектов отливок // Литейное Производство. Волгоград, 2016. С. 5-9.

4. Бойко Н.И., Фисенко К.С. Исследование качества поверхности наплавленного металла цилиндрической детали обработанной в горячем состоянии // Инженерный вестник Дона, 2012, № 2. URL: ivdon.ru/magazine/archive/n1y2012/618

5. Брюханова Е.В. Исследование состава и строения точечных поверхностных дефектов на отливках из нержавеющих сталей при литье по выплавляемым моделям (ЛВМ) // Инженерный вестник Дона, 2013, № 1. URL: ivdon.ru/uploads/article/pdf/IVD_65_Bruhanova.pdf_1543.pdf

6. Color conversions // docs.opencv.org. URL: docs.opencv.org /3.4/de /d25/imgproc_color_conversions.html.

7. Обработка изображения - пороговое преобразование // robocraft.ru. URL: robocraft.ru/blog/computervision/357.html.

8. Color Space Conversions // docs.opencv.org. URL: docs.opencv.org/3.4/d8/d01/ group__imgproc__color__conversions.html.

9. Грибков И.В., Захаров А.В., Кольцов П.П., Котович Н.В., Кравченко А.А. Сравнительное исследование методов анализа изображений // Автоматика и радиоэлектроника, Москва: НИИСИ РАН, 2005. С. 155.

10. Синюк В.Г., Батищев Д.С., Сойникова Е.С., Михелев В.М., Использование алгоритмов компьютерного зрения для выполнения гематологического анализа на основе кривой Прайс-Джонса // Научные ведомости белгородского государственного университета, 2018, №3, С. 537-546.

References

1. Hamuhin A.V. Aktual'nye problemy sovremennoj nauki. Moskva, 2014, pp. 170-174.

2. Ignat'ev A.A., Bahteev A.R. Vestnik saratovskogo gosudarstvennogo tehnicheskogo universiteta. Saratov, 2006. pp. 136-142.

3. Voronin Ju.V., Litejnoe Proisvodtstvo. [Foundry Production] Volgograd, 2016, pp. 5-9.

4. Bojko N.I., Fisenko K.S. Inzhenernyj vestnik Dona, 2012, № 2. – URL: ivdon.ru/magazine/archive/n1y2012/61.8

5. Brjuhanova E.V Inzhenernyj vestnik Dona, 2013, № 1. URL: ivdon.ru/uploads/article/pdf/IVD_65_Bruhanova.pdf _1543.pdf.

6. Color conversions. URL: docs.opencv.org/3.4/de/d25 /imgproc_color_conversions.html.

7. Obrabotka izobrazhenija - porogovoe preobrazovanie [Image processing - Threshold transformation]. URL: robocraft.ru/blog/computervision /357.html.

8. Color Space Conversions. URL: docs.opencv.org/3.4/d8/d01/ group__imgproc__color__conversions.html.

9. Gribkov I.V., Zaharov A.V., Kol'cov P.P., Kotovich N.V., Kravchenko A.A. Avtomatika i radiojelektronika, Moskva: NIISI RAN, 2005. P. 155.

10. Sinjuk V.G., Batishhev D.S., Sojnikova E.S., Mihelev V.M., Nauchnye vedomosti belgorodskogo gosudarstvennogo universiteta, 2018, №3, pp.537-546.