Автоматизация контроля поверхностных дефектов керамической плитки

Мохаммед Худаир Кадим
Санкт-Петербургский государственный
технологический институт
(технический университет)
mokadhim@yandex.ru

Л. А. Русинов
Санкт-Петербургский государственный
технологический институт
(технический университет)
lrusinov@yandex.ru

***Аннотация*. Несмотря на достаточно высокий уровень автоматизации производства керамической плитки, конечный ее контроль производится, практически, вручную. Предлагается метод автоматического контроля однотонных плиток на наличие дефектов механического происхождения. Метод состоит в обработке изображения плитки с отбраковкой по СКО интенсивности пикселей.**

Ключевые слова: дефекты керамической плитки, цифровая обработка изображений, контроль качества

I. Введение

Поверхностные дефекты керамической плитки достаточно разнообразны. Требования к ее качеству определяются международными и государственными стандартами [1]. Наиболее часто поверхность плитки повреждается от механических воздействий (далее, просто – механические дефекты), которые могут произойти на всех стадиях ее производства [2]. К ним, в частности, относятся трещины (посечки), царапины, волосяные трещины на глазури (цеки), ямки и т. п. [3]

В настоящее время контроль дефектов такого типа в большинстве случаев производится вручную, что приводит к многочисленным ошибкам вследствие большой нагрузки на зрение операторов, требований напряженного внимания, ограничения времени для принятия ими решения из-за движения конвейера, быстрого развития их усталости.

В статье предлагается метод автоматического обнаружения механических дефектов на лицевой поверхности гладких и однотонных керамических плиток с возможностью их отбраковки в реальном времени с конвейера. Метод предполагает получение изображения контролируемой плитки с последующей его обработкой. Основное требование к алгоритму, реализующему этот метод, касается прежде всего быстродействия.

II. Общее Описание алгоритма обработки

Изображение формируется цветной матричной камерой с разрешением не менее 2000x2000 пикселей в формате *jpg. Пример исходного изображения плитки приведен на рис. 1.

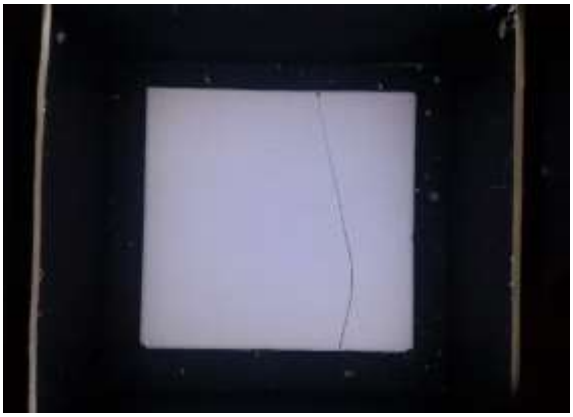

Рис. 1. Исходное изображение (видны вертикальные полосы – границы ленты конвейера)

Обработка производилась в среде открытой библиотеки OpenCV с привлечением других библиотек, которые используются для анализа 2D-изображений [4]. Программа была написана на языке Python 3.7. В рабочей среде PyCharm 2019.3.5 (community edition). Требования к ПЭВМ: процессор: Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80 GHz, установленная RAM: 8.00 GB (7.84 GB usable). Блок-схема предлагаемого алгоритма приведена на рис. 2.

Алгоритм состоит из двух взаимосвязанных частей. Первая часть (обведена пунктиром на рис. 2) предназначена для создания маски керамической плитки, с которой потом будет сравниваться реальное изображение той же плитки для выявления возможных механических дефектов. Особенностью алгоритма является создание маски не по отдельной эталонной плитке, а по рабочему изображению контролируемой плитки.

Вторая часть алгоритма производит сравнение маски с реальным изображением и позволяет выделить дефекты плитки и, таким образом, забраковать ее, не выясняя точный тип обнаруженного дефекта.

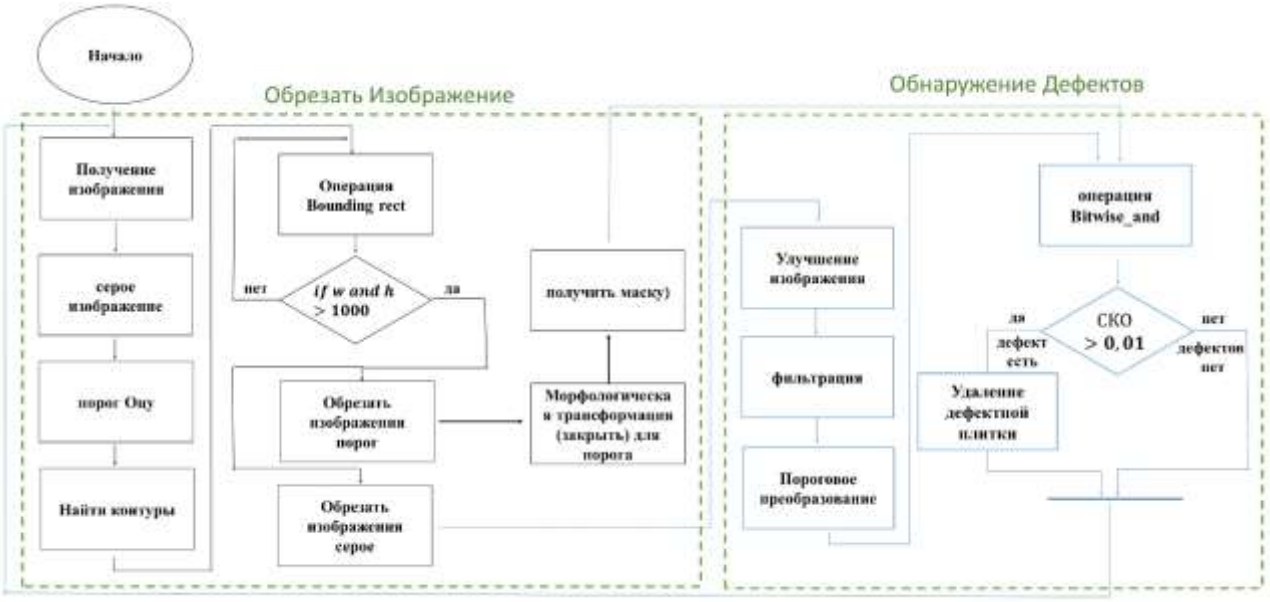

Рис. 2. Блок-схема алгоритма классификации керамической плитки на хорошие и дефектные плитки

A. Алгоритм создания маски

Первой операцией обработки является переход от полихромного изображения в монохромное полутоновое («оттенки серого»), что позволяет значительно ускорить дальнейшую обработку.

Затем следует целый ряд операций по формированию маски, сравнение с которой позволит выделить дефекты плитки. Для этого, прежде всего, производится бинаризация изображения, используя порог Отсу [5]. В этом случае, если интенсивность пиксела менее некоторого порогового значения, то пиксел обнуляется (становится черным), в противном случае – белым.

Для выделения на изображении области занимаемой собственно плиткой используется функция поиска контуров (cv2.findContours) [6]. При этом возможна реализация нескольких ее вариантов, определяющих режим (mode) и метод. Первый параметр определяет характер возвращаемого результата в том смысле, что учитывает или нет иерархию обнаруженных контуров. Учитывая, что в рассматриваемом случае интерес представляет контур плитки максимальный по площади (рис. 1), то можно не требовать выделение иерархии контуров полностью, а ограничиться максимальным внешним контуром, что даст также некоторую экономию по времени обработки. Тогда используем режим cv2.RETR_LIST:

Второй параметр функции поиска контуров определяет способ их аппроксимации. Учитывая, что плитка прямоугольная, а в рассматриваемом в работе случае - квадратная, запоминаем только конечные точки отрезков прямых контура (cv2.CHAIN_APPROX_SIMPLE). Далее выбирается максимальный по площади контур.

Для выделения интересующей области после получения контуров объекта на изображении используется функция cv2.boundingRect() [7]. Учитывая, что положение плитки на конвейере фиксируется, можно использовать вариант функции, строящей прямоугольник вокруг объекта, не учитывая возможные небольшие отклонения положения плитки относительно осей. Это естественно не гарантирует, что контур, охватывающий объект (в нашем случае, плитку), будет минимальным, но значительно ускоряет обработку. Если отклонения положения плитки большие, то используется другой вариант функции cv.minAreaRect(), которая вернет в числе прочих параметров угол необходимого поворота контура.

Задавая соответствующие параметры функции cv2.boundingRect(), можно вырезать из всего изображения область, занимаемую плиткой (маску) и, таким образом, ускорить обработку. Наконец, чтобы очистить поле маски, выполняется операция закрытия, убирающая мелкие темные шумовые точки, представляющая собой составной оператор, при котором сначала выполняется дилатация, а потом эрозия с тем же структурным элементом [5]. При этом использовался квадратный структурный элемент минимального размера 3x3. В результате получаем белую квадратную фигуру с определенными контурами, которая далее используется как маска (рис. 3). Из рисунка видно, что плитка на конвейере все-таки несколько отклоняется от нормального положения (неравная ширина черного контура по сторонам плитки).

B. Алгоритм выявления дефектов

Теперь алгоритм опять возвращается к серому исходному изображению плитки, но уже уменьшенного размера, соответствующего реальному изображению плитки (без фона), равному изображению маски.

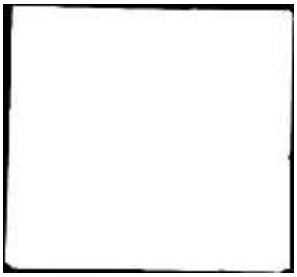

Рис. 3. Вид полученной маски

Т. к. исходное изображение, как правило, освещается недостаточно равномерно, контрастность в зонах дефектов также различна, то необходимо провести улучшение изображения, повысив яркость и контрастность.

Низкоконтрастные изображения могут возникнуть в нашем случае из-за небольшой разности в контрасте между дефектом и тоном поверхности плитки. Идея повышения контраста заключается в увеличении динамического диапазона уровней интенсивности пикселей (расширению активной зоны гистограммы) при обработке изображения. Естественно, если динамический диапазон изменения интенсивности равен диапазону кодирования (обычно используется 8-байтовое представление интенсивности, т.е. динамический диапазон равен 256), то методы, ориентированные на изменение гистограммы не пригодны.

Одним из распространённых методов увеличения контраста является эквализация гистограммы [5]. Фактически при этом достигается равномерная плотность распределения интенсивностей пикселей изображения по всему диапазону их изменения. Такой глобальный подход оказался полезным при обнаружении дефектов, достаточно сильно контрастирующих с тоном плитки (трещины, темные пятна и т.п.), но практически не влиял на контраст пятен с небольшими изменениями цвета по отношению к общему тону плиток.

В этом случае удобно использовать простой метод линейного контрастирования, на основе преобразования исходного изображения вида [8]:

$$\begin{cases} y_{max} = \alpha x_{max} + \beta \\ y_{min} = \alpha x_{min} + \beta \end{cases} \quad (1)$$

где y_{max} и y_{min} – максимальное и минимальное значения интенсивности преобразованного изображения (при 8-ми байтном кодировании 255 и 0 соответственно); x_{max} и x_{min} – максимальное и минимальное значения интенсивности пикселей исходного изображения; α и β – коэффициенты. При этом α – влияет на расширение динамического диапазона, а β – параметр его сдвига на оси гистограммы.

Улучшить результат контрастирования можно, если предварительно отсечь «хвосты», хорошо видимые на интегральной кривой гистограммы, где число пикселей мало (например, менее 1–2 % [9]). Это позволит определить xmax и xmin, найти из уравнений (1) α и β, и провести контрастирование для оставшейся части гистограммы. При этом отсечку удобно производить, задавшись процентной долей отсекаемой части интегральной кривой (clip_hist_percent). Если требуется повысить контраст низкоконтрастных участков плитки можно выделить и растянуть требуемую часть динамического диапазона исходного изображения. Участки, не попавшие в эту часть, будут черными. Достоинство метода – высокая скорость обработки.

Для удаления шума, часто имеющего в своем составе аномальные значения, в большинстве случаев используется нелинейная медианная фильтрация. В этом случае уровни яркости пикселей, которые принадлежат окну фильтра, ранжируются по возрастанию интенсивности (вариационный ряд). Центральному пикселу присваивается значение интенсивности среднего члена этого ряда (или полусуммы пары средних значений при четном числе пикселей в маске). Обычно окно выбирается размером 3x3 или 5x5 пикселей.

Для выделения дефектов также используется бинаризация изображения. Но в этом случае, учитывая для снижения требований к равномерности освещения плитки, предлагается использовать адаптивную бинаризацию с локальными порогами [10]. При этом в OpenCV предлагается для такой бинаризации использовать варианты вычисления локальных порогов по среднему значению или по средневзвешенному значению интенсивностей соседних пикселей. Во втором случае веса *w(i,j)* определяются функцией Гаусса:

$$w(i,j) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{u^2}{2\sigma^2}\right\} \quad (2)$$

Соседние пиксели определяется квадратным блоком с преобразуемым пикселем в его центре. Размеры блока обычно варьируются в широких пределах и определяют зону задания функции 6σ. Именно этот вариант вычисления порогов при бинаризации принят в предлагаемом алгоритме. OpenCV позволяет настраивать метод, вводя константу C. Тогда пороговое значение будет представлять собой взвешенную по Гауссу сумму значений интенсивностей соседних пикселей за вычетом константы C [10]. Экспериментально был подобран размер блока 39x39 и C=14.

Далее операцией попиксельного сравнения с маской с одновременным выполнением логической операции И (Bitwise_and) получаем изображение с выделенными дефектами на лицевой поверхности плитки (рис. 4).

Рис. 4. Обработанное изображение плитки с дефектом в виде неглубокой трещины

Для определения дефектная плитка или нет, алгоритм подсчитывает среднеквадратичное отклонение СКО и сравнивает с порогом γ:

$$\text{СКО} = \sqrt{\frac{1}{MN}\sum_{i=0}^{M-1}\sum_{j=0}^{N-1}(I_{(i,j)} - K_{(i,j)})^2} \geq \gamma \qquad (2)$$

где M, N – количество пикселей в строке и столбце соответственно; $I_{(i,j)}$ – интенсивности пикселей обработанного изображения, $K_{(i,j)}$ – среднее значение интенсивности пикселей, которое можно приближенно считать равным $K_{(i,j)} \approx 0$ [11]. Значение порога γ=0,01 было найдено экспериментально.

C. Экспериментальная проверка работы алгоритма

Для проверки алгоритма была использована база дефектных и нормальных плиток [12], из которой были выбраны 90 плиток, из них 40 штук были без дефектов, а остальные с различными механическими дефектами.

Все отобранные плитки были предъявлены алгоритму для определения бракованных. При этом число правильных обнаружений составило 98 %.

Как уже отмечалось выше, важным требованием к алгоритму обработки является время, затрачиваемое на прием и обработку изображения очередной плитки. Время, затрачиваемое на обработку изображения одной плитки, практически менее 175 мс (см. таблицу). Однако на построение маски требуется примерно 500 мс, но если учесть, что маска требуется при выявлении и других, а не только механических дефектов плитки, то суммарное время при расширении функций алгоритма изменится незначительно.

ТАБЛИЦА I ЗАТРАТЫ ВРЕМЕНИ НА ОБРАБОТКУ ИЗОБРАЖЕНИЯ ОДНОЙ ПЛИТКИ

Время (мс)*)	**Хорошие плитки**		**Дефектные плитки**	
	Кол-во плиток	**Процент**	**Кол-во плиток**	**процент**
Время $\leq$ **150**	14	35 %	17	34 %
150 < Время < 175	26	65 %	30	60 %
175 < Время < 200	0	0 %	3	6 %
*) Общее время обработки включает время на построение маски, которое составляет примерно 500 мс и должно быть добавлено к времени на выявление дефекта.				

III. ВЫВОДЫ

Предложен метод и скоростной алгоритм обработки изображений однотонной плитки, эффективно выявляющий поверхностные дефекты механического происхождения. Алгоритм реализован в среде OpenCV с привлечением других библиотек, которые используются для анализа 2D-изображений. Программа была написана на языке Python 3.7. В рабочей среде PyCharm 2019.3.5.

Метод отличается тем, что не требует предъявления эталонных плиток для выявления дефектных и при этом обеспечивает высокую скорость классификации плиток на годные и дефектные. Наибольшее время в алгоритме занимает построение маски плитки (порядка 500 мс), а среднее время последующего выявления возможного дефекта порядка 175 мс. Т. к. маска может быть использована и для алгоритмов поиска дефектов различного происхождения, то эффективность метода при расширении его функциональных возможностей дефекты другого генезиса будет только возрастать. К сожалению, до сих пор не было предложено общих алгоритмов, учитывающих все различные типы дефектов одновременно? Поэтому для каждой группы дефектов строится свой алгоритм [13].

СПИСОК ЛИТЕРАТУРЫ

[1] ГОСТ 13996–2019. Плитки керамически. Общие технические условия (ISO 13006:2018, NEQ). М.: Стандартинформ, 2019. 42 с.

[2] Hocenski Ž., Keser T. Failure Detection and Isolation in Ceramic Tile Edges Based on Contour Descriptor Analysis // 15th Mediterranean Conference on Control & Automation, Athens, 2007, 6 p. doi: 10.1109/MED.2007.4433713.

[3] ГОСТ Р 57141-2016 Плиты керамические (керамогранитные). Технические условия //URL: https://docs.cntd.ru/document/120014020 (дата обращения 01.07.2021г.)

[4] OpenCV Library URL: http://opencv.org. (дата обращения: 01.07.2021г.).

[5] Гонсалес Р., Цифровая обработка изображений. М.: Техносфера, 2012. 1104 с.

[6] cv2.findContours() function//URL: https://www.programmersought.com/article/82015048356/ (дата обращения 01.07.2021г.).

[7] Contour Features//URL: https://docs.opencv.org/3.4/dd/d49/tutorial_py_contour_features.html (дата обращения 01.07.2021г.)

[8] Грузман И.С. Цифровая обработка изображений в информационных системах: Учебное пособие. Новосибирск / И.С. Грузман, В.С. Киричук, В.П. Косых, Г.И. Перетягин, А.А. Спектор. Изд-во НГТУ, 2002. 352 с.

[9] Adaptive Brightness Contrast Adjustment // URL: https://www.programmersought.com/article/9935110603/ (Дата обращения 01.07.2021).

[10] Dnyandeo S.V., Nipanikar R.S. A Review of Adaptive Thresholding Techniques for Vehicle Number Plate Recognition // International Journal of Advanced Research in Computer and Communication Engineering, 2016. V. 5, No 4. P.944-946.

[11] Кокошкин А.В. Сравнение объективных методов оценки качества цифровых изображений / А.В.Кокошкин, В.А. Коротков, К.В. Коротков, Е.П. Новичихин // Журнал радиоэлектроники, 2015. N6. 11 с.

[12] Set1; set2 // URL: https://drive.google.com/drive/folders/0B8VsanE1mnJuak1RNWZnVTZabE0?resourcekey=0-YHuVwESfEAeL7aUVzYXeAQ (дата обращения 0.07.2021)

[13] Karimi M.H. Surface defect detection in tiling Industries using digital image processing methods:Analysis and evaluation / M.H. Karimi, D. Asemani // ISA Transactions,2014.-V.53. P.834–844