Введение

В рамках данного курсового проекта представлен процесс разработки информационной системы распознавания матричной маркировки на цифровых изображениях черных металлов. В настоящее время, в связи с необходимостью развития промышленности, требуется введение новых наукоемких технологий по контролю движения продукции. Реализация такого контроля возможна с помощью внедрения систем автоматической идентификации и распознавания. Для контроля движения продукции используются маркировки промышленных изделий. На новейших промышленных предприятиях контроль движения продукции осуществляется специализированными средствами - системами автоматической идентификации маркировки (САИМ). Основная сложность при использовании САИМ заключается в задаче локализации промышленной продукции, которую усложняют наличие неравномерной освещенности и резкости (засвечивание изображения искусственным и солнечным светом, неверная настройка фокуса), присутствие шумов и размытия, прочих лишних объектов. Автоматизация этой сферы деятельности является мало проработанной и, чаще всего, выполняется вручную. Этим обусловлена актуальность данной работы.

Цель данного курсового проекта – разработать информационную систему распознавания матричной маркировки на цифровых изображениях черных металлов.

Задачи данного курсового проекта:

a) провести анализ предметной области, проанализировать аналогичные алгоритмы, предъявить требования к разрабатываемой информационной системе;

b) произвести математическое моделирование разрабатываемой информационной системы;

c) проанализировать алгоритмы, используемые при разработке информационной системы, описать их и представить их блок-схемы;

d) реализовать информационную систему.

Таким образом, общая структура курсового проекта состоит из трех глав и выглядит следующим образом:

a) в первой главе производится анализ технического задания, а именно:

- обозревается предметная область;

- определяются исходные данные к проекту;

- кратко обозреваются некоторые аналогичные алгоритмы;

- выставляются требования к разрабатываемой информационной системе

b) во второй главе производится математическое моделирование разрабатываемой информационной системы, приводится описание используемых в разработке алгоритмов и их блок-схемы;

c) в третьей главе описывается процесс реализации разработанной информационной системы.

1. Анализ технического задания

1.1. Обзор предметной области

Маркировка красками или специальными чернилами - один из старейших способов маркировки. Наибольшее распространение сегодня маркировка красками имеет в легкой, электронной и пищевой промышленности, где используется оборудование типа IMAJE и DOMINO. Эти маркеры оборудованы специальными головками с соплами, с помощью которых возможно нанесение любого вида маркировки. Контрастность изображения обеспечивается различным количеством точек на единицу площади изображения. Новейшее оборудование для маркировки красками позволяет создавать разноцветные обозначения очень высокого качества.

Однако клеймение горячего металла красками невозможно: лучшие образцы лакокрасочной продукции выдерживают температуру не более 1000ºС. Плюс к этому в условиях металлургических предприятий и горячих цехов трудно обеспечить необходимую чистоту и приемлемый температурный режим, требуемые для безотказной работы краскоструйных сопел.

Затрудняет использование краскоструйных маркеров и характер поверхности многих металлоизделий. Окалина, пыль, неровности, влага и т.п. делают маркировку красками фрагментарной и недолговечной.

Все перечисленные факторы делают невозможным или экономически невыгодным использование краскоструйных маркеров в горячих цехах и при машинах непрерывного литья заготовок.

Однако маркировка в металлургии красками применяется - для обозначения холодных (остывших) металлоизделий. Единственное требование для использования маркировки красками - чистота и высокая адгезивность поверхности.

Краски используют для маркировки труб горячекатаных (на выходе из прокатного стана, когда температура изделия становится приемлемой), маркировки слябов и блюмов и т.п.

Оборудование для маркировки краской состоит из следующих узлов:

- головка с 7 или 9 соплами;

- манипулятор для перемещения головки с соплами;

- специальная панель для подготовки краски;

- пульт управления;

- терминал для оператора, с помощью которого можно вручную задавать параметры маркировки;

- электронная система управления (программное обеспечение).

Краскосопла имеют увеличенный диаметр отверстий, что препятствует их засорению, и приспособлены для операции самоочищения. Минимальная высота наносимой маркировки - 19 миллиметров, что может служить ограничением в использовании такого оборудования для некоторых видов металлоизделий.

С помощью оборудования для маркировки краской можно наносить не только буквенно-цифровые коды, но и штрих-коды. Однако, на данном оборудовании можно нанести только бинарные коды (содержащие минимум информации о металлоизделии), для считывания которых нужна специальная аппаратура (промышленные сканеры).

Оборудование для нанесения маркировки краской имеет высокую степень надежности. Однако, при использовании маркировки в металлургии с помощью красок и специальных чернил необходимо учитывать, что для полного высыхания краски требуется определенное время, при этом чем горячее металлоизделие, тем больший срок требуется для высыхания краски. При высокой температуре маркируемых металлоизделий также повышаются эксплуатационные расходы на краскоструйное оборудование: увеличивается расход красок, требуется больше времени для технического обслуживания агрегата и т.п.

Основные характеристики:

- постоянные метки для идентификации;

- высокая степень разборчивости символов для обычного или автоматического считывания в условиях неповреждения метки;

- разборчивая маркировка на грубых и неровных поверхностях;

- высокая скорость маркировки.

Основное оборудование, необходимое для нанесения маркировки – это многопистолетные установки нанесения краски или чернил.

В ходе работы над разработкой информационной системы, распознающей матричные маркировки, будут использоваться искусственно сгенерированные изображения. Пример такого изображения представлен на рисунке 1.



Рисунок 1 – пример тестового изображения

1.2. Исходные данные к проекту

В качестве входных данных были определены следующие:

a) набор смоделированных исходных изображений для проверки правильности работы разрабатываемой информационной системы;

b) среда программирования Visual Studio для программной реализации алгоритма и проведения экспериментальных исследований работы алгоритма;

c) язык программирования Python, так как данный язык является языком высокого уровня, что значительно упрощает работу над разработкой информационной системы, и позволяет реализовать возможность создания графического интерфейса, а также поддерживает работу с библиотеками OpenCV и Numpy;

d) библиотека OpenCV - это библиотека с открытым исходным кодом, разработанная на языке C++, которая используется разработчиками и исследователями при решении задач, связанных с компьютерным зрением;

e) библиотека Numpy – это также библиотека с открытым исходным кодом, которая позволяет работать с массивами и вычислять сложные арифметические операции.

Выходными данными в курсовом проекте будет являться текст распознанной матричной маркировки.

1.3. Обзор аналогичных алгоритмов

Задача распознавания матричной маркировки схожа с задачей распознавания почтовых индексов. В связи с этим, за основу был взят статистический метод распознавания рукописных цифр на почтовых индексах.

В соответствии с современными нормами оформления почтовых конвертов индексы могут записываться в произвольной форме, но в рамках заранее заданного поля определенного размера (отдельные ячейки для каждой цифры). Для автоматизации процесса сортировки писем почтовые индексы необходимо распознать. Поле индекса на конверте является единственным ограничением, но именно оно может быть взято за основу для унификации распознаваемых цифр. В случае с матричной маркировкой, ограничения по площади появляются после сегментации изображения (один сегмент – один символ).

Шрифтозависимые алгоритмы основываются на сопоставлении идентичных элементов в известных шрифтах и распознаваемых символах.

Первыми пунктами данного алгоритма являются выравнивание и масштабирование символов, но так как в случае с текущей темой все символы типизированы, этот пункт можно пропустить.

Далее производится обработка унифицированных изображений цифр имеющихся образцов почерков для получения их индивидуальных характеристик. С этой целью подсчитывается количество черных пикселей в каждом из столбцов *xi* (*i = 1…n*) и строк *yj* (*j = 1…m*) изображения. Полученные значения во многом зависят от толщины контура цифр. Для исключения влияния этого параметра выполняется нормирование с помощью общего количества черных пикселей для изображения рассматриваемой цифры и соответствующего почерка:

(1)

(2)

при этом, очевидно, что:

, (3)

и, соответственно, , .

Векторы и , элементами которых являются нормированные величины и , выступают в качестве характеристик соответствующих цифр. Верхний индекс *p* = 0,1,2,…,9 отмечает принадлежность вектора к соответствующей цифре, а индекс *l* = 1,2,… – к соответствующему почерку.

Выработка эталона для распознавания (обучение системы) производится путем усреднения элементов векторов и для каждой цифры при заданном количестве почерков. Можно предположить, что если нанесение матричных маркировок стандартизировано, то для распознавания данного типа меток достаточно выработать эталон для распознавания на нескольких возможных типах изображений:

- чистое изображение;

- изображение со слабыми шумами;

- изображение со средним уровнем шумов;

- изображение с сильными шумами;

- изображение с засветами (но существует вероятность, что данный тип изображений не подходит для распознавания, так как засветы придется распознать и удалять дополнительными алгоритмами).

Так, например, элемент эталонного (усредненного) вектора для цифры «0» по пяти почеркам определяется из соотношения

Аналогично вычисляется .

Распознавание цифры производится в следующем порядке:

1) считывание цифры;

2) бинаризация изображения;

3) нормирование изображения;

4) получение индивидуальной характеристики цифры (векторы и );

5) поочередное сравнение индивидуальной характеристики цифры с эталонными характеристиками всех цифр.

Поиск максимального совпадения производится путем нахождения рассогласований между индивидуальной характеристикой распознаваемой цифры и эталонными характеристиками всех цифр и выбора минимального из них. Определяются рассогласования отдельно для векторов и и, в качестве третьего параметра, вычисляется их сумма:

 (4)

 (5)

 (6)

Для облегчения анализа результатов при тестировании все полученные для какой-либо распознаваемой цифры величины нормируются с помощью , – с помощью , а – с помощью . В этом случае минимальные значения нормированных ,  и  равны единице и могут быть легко выделены. Рассматриваемая цифра считается распознанной при выполнении одного из логических выражений к величинам (4), (5) и (6) эталонной характеристики какой-либо цифры:

 (7)

 (8)

На основании выполненной работы автором статьи были получены следующие результаты:

- предлагаемый метод позволил с вероятностью ~ 0.97 распознать рукописные цифры;

- в 2,5 % однозначно определено, что цифра не распознана, не выполнено ни одно из логических выражений (7) к величинам (4), (5) и (6). Это всегда относилось к цифрам «2» и «8», что сужает круг дополнительного поиска и облегчает использование других методов распознавания, ориентированных именно на эти цифры;

- только в 0,5 % случаев происходит ошибочное распознавание. Это позволяет рекомендовать предложенный метод для распознавания рукописных цифр, в том числе в почтовых индексах.

В задаче распознавания матричной маркировки необходимо задействовать большее количество алгоритмов и предусмотреть корректное удаление дефектов изображения (предобработка изображения). Более подробно этот процесс описывается во второй главе пояснительной записки.

1.4. Требования к разрабатываемой информационной системе

Требования к разрабатываемой информационной системе следующие:

* разрабатываемая система должна оперировать со всеми входными данными и получать в результате текст распознанной матричной маркировки;
* каждый алгоритм разрабатываемой информационной системы должен состоять из отдельных элементарных шагов, работающих системно;
* последовательность шагов каждого алгоритма должна быть детерминирована, т.е. после каждого шага должно указываться, какой шаг следует выполнять дальше, либо указываться, когда следует работу алгоритма считать законченной;
* каждый алгоритм должен обладать результативностью, т.е. останавливаться после конечного числа шагов (зависящего от исходных данных) с выдачей результата;
* в отношении времени выполнения каждого алгоритма – он должен обладать высокой скоростью выполнения операций.

2. Разработка проекта ИС

2.1. Моделирование ИС

Разработка проекта информационной системы была начата с разработки модели информационной системы, которая была проведена в несколько этапов:

a) модель «черного ящика»;

b) модель состава;

c) структурная модель;

d) функциональная модель.

Модель «черного ящика» показывает следующие параметры: входные данные (изображение с матричной маркировкой), выходные данные (распознанная маркировка), сам «черный ящик» (процесс распознавания маркировки) и внешние факторы, влияющие на распознавание (шумы, световые блики, отсутствие резкости на изображении). Модель «черного ящика» представлена на рисунке 2.

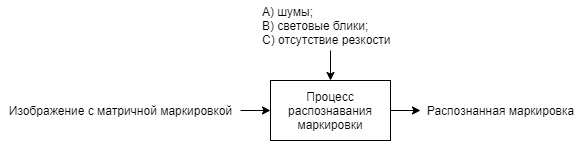


Рисунок 2 – модель «черного ящика»

Следующий этап – разработка модели состава. По сути, модель состава – это то, что находится внутри «черного ящика», но уже чуть более подробно описанное – в виде отдельных блоков. В модели состава, соответствующей данной информационной системе, должно присутствовать 4 следующих блока:

a) блок предобработки изображения – на данном этапе происходит фильтрация шума на изображении, удаление световых бликов, искусственное повышение резкости изображения (это необходимо для того, чтобы распознавание маркировки на последнем этапе произошло более качественно);

b) блок локализации маркировки – в данный блок включены алгоритмы, позволяющие определить координаты месторасположения маркировки на изображении, это нужно для ускорения работы программы, чтобы не приходилось задействовать всю область изображения;

c) блок сегментации маркировки – локализованная маркировка, обрабатываясь алгоритмами этого блока, разделяется на отдельные символы, что позволит в дальнейшем качественно произвести распознавание;

d) блок распознавания маркировки – в данном блоке происходит сравнение каждого поступающего на вход сегментированного элемента маркировки с уже имеющимся эталоном, соответственно, на выход поступает текст распознанной матричной маркировки.

Модель состава разработанной информационной системы представлена на рисунке 3.



Рисунок 3 – модель состава

Структурная модель представляет собой модель состава, дополненную стрелками, показывающими ход работы разрабатываемой информационной системы. Таким образом, на рисунке 4 видно, что на вход информационной системе проступает изображение с матричной маркировкой, где далее оно поступает в блок предобработки изображения, после чего маркировка локализуется, сегментируется на отдельные символы и наконец распознается. На выход поступает текст распознанной маркировки.

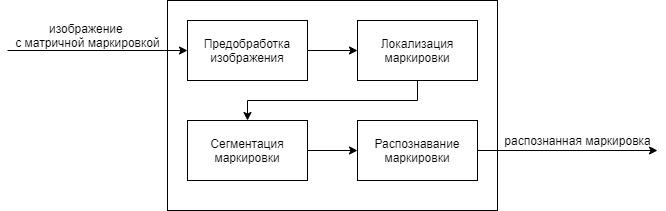


Рисунок 4 – структурная модель

Последним этапом в моделировании является построение функциональной схемы – схемы, которая подробно описывает принцип работы информационной системы. Каждый блок, описанный в модели состава, на функциональной схеме представлен в виде нескольких алгоритмов, которые последовательно обрабатывают изображение в целях эффективного получения выходных данных.

Таким образом, процесс распознавания маркировки был разделен на 16 алгоритмов, последовательно идущих друг за другом:

a) предобработка:

- перевод изображения в градации серого;

- устранение шумов: поиск искаженных пикселей;

- устранение шумов: восстановление изображения;

- бинаризация;

- устранение световых бликов на изображении;

- восстановление резкости изображения;

b) локализация:

- инверсия;

- построение графиков;

- определение координат;

c) сегментация:

- вычисление энергетической функции;

- построение всех путей прохождения;

- выбор путей с минимальной стоимостью;

- нахождение линий раздела;

d) распознавание:

- нормирование;

- получение индивидуальных характеристик цифры;

- поочередное сравнение индивидуальных характеристик с эталоном.

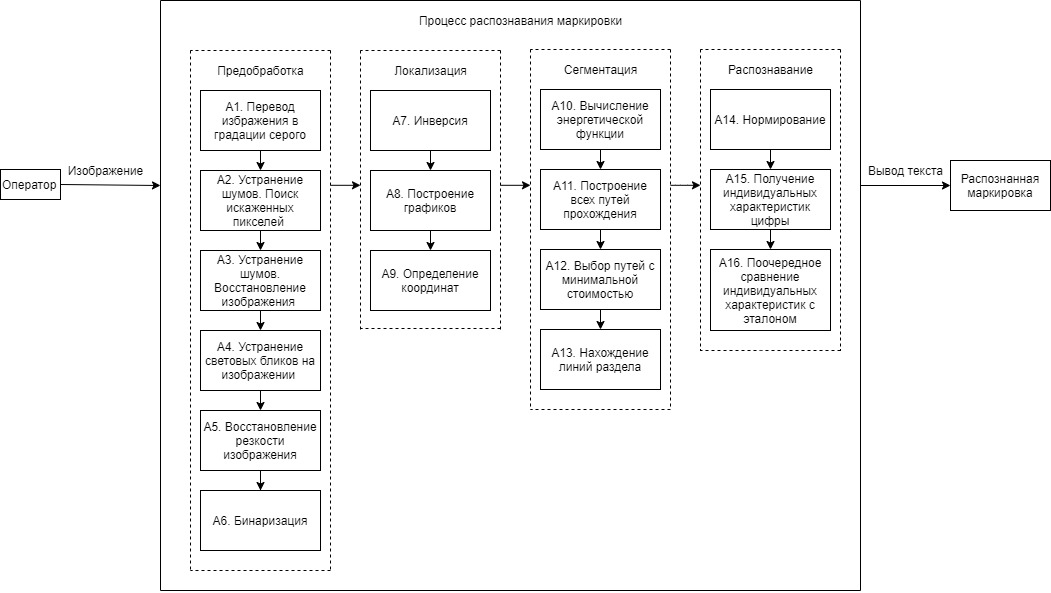


Рисунок 5 – функциональная модель

Для того, чтобы осуществить работу последнего алгоритма, необходимо дополнительно разработать небольшую систему, которая хранила бы в себе эталонные характеристики цифр, с которыми будет производиться сравнение реальных характеристик. На рисунке 6 изображена общая схема процесса распознавания маркировки.

Х – это характеристики, поступающие на вход для обучения информационной системы (база эталонных характеристик), и характеристики, формирующиеся в процессе работы алгоритма А15 (получение индивидуальных характеристик цифры). Обе совокупности этих характеристик начинают свою работу в системе сравнения, организуемой алгоритмом А16 (алгоритм обучения системы описан в пункте 2.2.4.3.1).

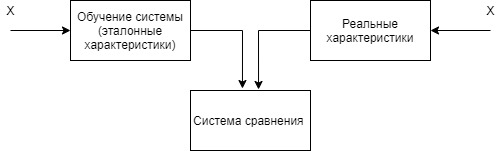


Рисунок 6 – общая схема процесса распознавания

2.2. Описания алгоритмов работы ИС

2.2.1. Алгоритмы блока предобработки изображений.

Данный блок необходим в разрабатываемой информационной системе для того, чтобы подготовить изображение к дальнейшей работе с ним. Его цель – максимально эффективно избавиться от внешних факторов, которые могут повлиять на качество распознавания маркировки (шумы, размытие изображения, световые блики).

2.2.1.1. Алгоритм А1. Преобразование в градации серого.

В дальнейшем изображение нужно будет бинаризовать, поэтому необходимо перевести его в градации серого. Это производится стандартным алгоритмом. Работа которого показана на блок-схеме 7.

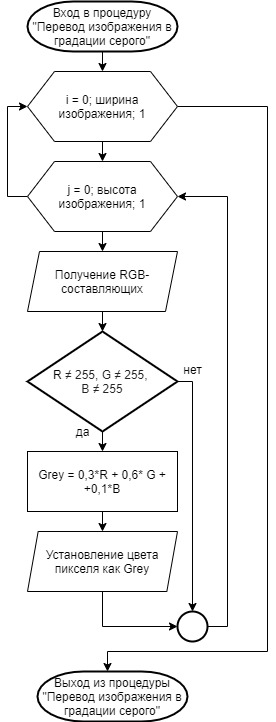


Рисунок 7 – перевод в градации серого

2.2.1.2. Алгоритм А2. Устранение шума. Поиск поврежденных пикселей.

Изображения, полученные при фотографировании листа металла с маркировкой, могут быть зашумлены в силу недостаточного количества света в цеху. Этот шум является импульсным (шум типа «соль и перец») – пиксели повреждаются точечно, поэтому восстановить их за счет использования яркости соседних пикселей не очень сложно. Алгоритм работы поиска поврежденных пикселей описан ниже.

Шаг 1. Инициализация окна (маски фильтра) размером 33 пикселя.

Шаг 2. Разделение уровней серого пикселей в окне на 4. Это деление необходимо для повышения надежности нахождения зашумленных значений.

Шаг 3. Поиск минимального и максимального значения яркости в окне (данные значения обозначаются как  и ).

Шаг 4. Определение того, зашумлен пиксель или нет, происходит путем следующей проверки: если уровень серого центрированного пикселя больше, чем , и меньше, чем , - в этом случае пиксель не зашумлен, и можно переходить к проверке следующего пикселя. Иначе – центральный пиксель зашумлен, и, основываясь на условиях проверки, описанных выше, необходимо найти другие зашумленные пиксели в окне.

Таким образом, необходимо провести данную маску фильтра по всему изображению, после чего нужно увеличить размер окна до 55 пикселей и повторить шаги 1-4.

Если все пиксели в окне зашумлены, а размер окна составляет 55 пикселей, тогда нужно использовать среднее значение трех ранее вычисленных соседних пикселей как значение уровня серого зашумленного пикселя, после чего можно совершить переход к следующему пикселю.

2.2.1.3. Алгоритм А3. Устранение шума. Восстановление изображения.

После того, как были определены все поврежденные пиксели, их необходимо восстановить следующим алгоритмом.

Шаг 1. Расчет среднего *AVG* и медианного *MED* значения окна.

Шаг 2. Путем расчета  и  по формулам (9) и (10) найти, какое из средних или медианных значений находится вблизи центра интервала [;]. Минимум  и  показывают предпочтительное значение.

 (9)

 (10)

Шаг 3. Зашумленный пиксель заменяется значением, полученным на шаге 2, если разница между его интенсивностью и значением, полученным на шаге 2, превышает пороговое значение . Этот порог рассчитывается согласно следующим уравнениям:

 (11)

 (12)

 (13)

 (14)

где *Zk* – значение интенсивности текущего пикселя;

*nk* – количество пикселей в изображении, значение интенсивности которых равно *k*;

*L* – динамический диапазон значений пикселей;

*m* – средняя интенсивность изображения;

 - дисперсия изображения;

 - порог, на который заменяется поврежденный пиксель.

Если разница между интенсивностью и значением, полученным на шаге 2, не превышает значение порога, яркость пикселя не меняется.

2.2.1.4. Алгоритм А4. Бинаризация.

Бинаризация изображения происходит в данном случае при помощи порогового метода Отцу.

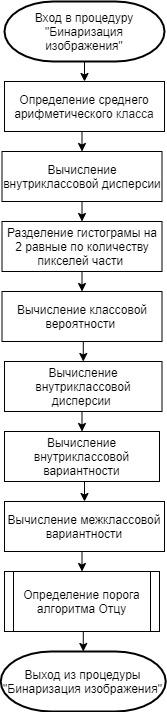


Рисунок 8 - бинаризация

На данном этапе производится присваивание пикселям одной из двух яркостей: либо 0, либо 255. Происходит это следующим образом:

a) определение среднего арифметического класса  по формуле 15:

 (15)

где *n* – общее количество пикселей изображения;

*i* – уровень яркости пикселя;

*p* – количество пикселей каждого уровня яркости.

b) вычисление внутриклассовой дисперсии  по формуле 16:

 (16)

c) разделение гистограммы на 2 класса, приблизительно равные по количеству пикселей (вдоль по оси *x*);

d) вычисление классовой вероятности для класса 1 (*W*1) и класса 2 (*W*2) по формуле 17:

; (17)

e) вычисление внутриклассовой дисперсии для каждого класса по формуле 16;

f) вычисление внутриклассовой вариантности  по формуле 18:

; (18)

g) вычисление межклассовой вариантности  по формуле 19:

; (19)

h) определение порога алгоритма Отцу (алгоритм представлен на рисунке 9), после чего процесс бинаризации изображения можно считать завершенным.



Рисунок 9 - определение порога Отцу

2.2.1.5. Алгоритм А5. Устранение световых бликов.

Блики на изображениях имеют ярко выраженный характер, занимают довольно большую часть изображения и могут существенно помешать качественному распознаванию маркировки. Работа над их устранением осуществляется следующим алгоритмом.

Шаг 1. Производится присвоение пикселям с яркостью 0 – ноль, а пикселям с яркостью 255 – единицу.

Шаг 2. Производится подсчет количества пикселей, к которым была присвоена 1. Результаты заносятся в отдельные массивы, соответственно близлежащих пикселей.

Шаг 3. Каждому пикселю в наибольшем массиве с единицами нужно присвоить яркость 0 (таким образом происходит цветокоррекция светового блика, закрашивание его черным цветом).

Выполнение этих вышеперечисленных манипуляций с изображением обеспечивает качественное удаление световых бликов на тестовых изображениях.

2.2.1.6. Алгоритм А6. Повышение резкости изображения.

После проведения фильтрации шумов и удаления бликов изображение размывается, поэтому для более качественного распознавания маркировки необходимо искусственно повысить резкость изображения. Это достигается при помощи использования алгоритма, представленного на рисунке 10.



Рисунок 10 - повышение резкости изображения

Данный метод позволяет найти приближенное значение невозмущенной функции путем простого вычитания из известного сигнала  восстанавливающей функции :

=-. (20)

На вход поступает полутоновое изображение шириной *М* (координата *x*, индекс *i*) и высотой *N* (координата *y*, индекс *j*) пикселей со значениями яркости *i,j*-го пикселя *fi,j*, подвергшееся воздействию неизвестного ядра размытия *hi,j*. Исходное искаженное изображение *gi,j*может быть представлено в виде результата применения операции дискретной свертки к невозмущенному изображению *fi,j* (21):

 (21)

Начало координат изображения находится в левом верхнем углу так, что индекс *i* по координате *х* возрастает слева направо от 1 до *М*, а индекс *j* по координате у возрастает сверху вниз от 1 до *N*.

Далее необходимо ввести отдельно для координат x и y наборы квадратных цифровых фильтров с нечетным количеством элементов *S**S* различного размера , элементы которых имеют следующий вид:

, (22)

, (23)

где  *p* и *q* – индексы по координатам *х* и *у* соответственно, диапазон изменения которых составляет от *–Sc*до *Sc*. В дальнейшем по аналогии с вейвлет-анализом величина *S* будет именоваться масштабом и обозначаться верхним индексом, заключенным в скобки.

Цифровые фильтры при S = 3 и S = 5 будут выглядеть следующим образом:

 (24)

 (25)

 (26)

 (27)

Для S = 3 фильтры  и  представляют собой известные операторы Prewit, которые используются для детекции границ объектов в цифровой обработке изображений.

Таким образом фильтры (22) и (23) удовлетворяют условию нулевого среднего:

 (28)

 (29)

Путем последовательного вычисления дискретной свертки исходного изображения *gi,j* с набором фильтров (22) и (23) различного масштаба *S,* получаются величины и :

 (30)

 (31)

которые называются нормированными дифференциальными откликами яркости цифрового изображения в точке с координатами *(i,j)* по соответствующей координате и масштабу *S*. В результате выполнения операции дискретной свертки, из-за граничных эффектов размеры массивов дифференциальных откликов уменьшаются и равны  для соответствующего масштаба *S*.

В выражениях (30) и (31) нормирующий множитель *A(S)* равен:

 (32)

Величина *A(S)* выбрана таким образом, чтобы для любого масштаба S было обеспечено равенство интегральных величин модулей дифференциальных откликов по всему изображению:

 (33)

 (34)

Величины *Cx* и *Cy* имеют смысл суммы абсолютных значений перепадов яркости на всех границах объектов на изображении.

В дальнейшем анализе потребуются также массивы производных по соответствующим координатам и масштабам от нормированных дифференциальных откликов, которые определяются следующим образом:

 (35)

 (36)

Следующий шаг – синтез восстанавливающей функции.

Восстанавливающая функция  в каждой точке изображения рассчитывается как сумма вкладов в нее  по соответствующим масштабам *S*:

 (37)

а качество восстановления *fi,j* в общем случае определяется числом учитываемых при работе масштабов. Перед вычислением вклада  в восстанавливающую функцию от каждого масштаба *S* необходимо создать массивы величин  и  путем транспозиции элементов массивов  и  по следующим правилам:

, (38)

, (39)

после чего вклады  по каждому из массивов вычисляются следующим образом:

 (40)

Восстановленное изображение  находится по формуле (20).

2.2.2. Алгоритмы блока локализации маркировки

2.2.2.1. Алгоритм А7. Инверсия.

Инвертировать изображение в рамках разработки данной информационной системы необходимо для ускорения работы программы. Для того, чтобы инвертировать изображение, нужно из 255 (максимальная яркость пикселя) вычесть его текущую яркость. Это действие необходимо повторить с каждым пикселем.

2.2.2.2. Алгоритм А8. Построение графиков.

На рисунке … изображен алгоритм построения графиков. Он необходим для того, чтобы следующим алгоритмом локализовать маркировку путем наложения одного графика на другого и определения граничных точек пересечения. Эти точки и будут являться координатами маркировки.

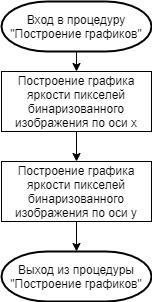


Рисунок 11 - построение графиков

2.2.2.3. Алгоритм А9. Определение координат.

Как было сказано выше, определение координат находится путем нахождения пересечений двух графиков, построенных в результате работы алгоритма А8. Более наглядно алгоритм представлен на рисунке 12.



Рисунок 12 - определение координат

После того, как были определены координаты *(x1;y1)* и *(x2;y2)*, локализованная маркировка выделяется рамкой (для наглядности).

Дальнейшая работа будет происходить внутри выделенной области.

2.2.3. Алгоритмы блока сегментации маркировки.

Эффективным решением задачи сегментации символов является алгоритм поиска разделяющих линий, основанный на методе минимизации стоимостной функции, который позволяет производить сегментацию цифрового изображения, учитывая его содержание. Данная функция является некой условной величиной, которая характеризует насколько важен каждый пиксель в изображении. Общую схему работы данного метода можно разделить на 4 алгоритма: вычисление энергетической функции для каждого пикселя изображения, построение всех путей прохождения, выбор путей с минимальной стоимостью (в соответствии с некоторыми условиями) и, наконец, определение линий раздела.

2.2.3.1. Алгоритм А10. Вычисление энергетической функции.

Существует достаточно большое количество вариантов энергетической функции. При вычислении функции (41) наиболее значимыми элементами на изображении будут являться структура и контуры объектов:

(41)

Данная функция должна быть вычислена для каждого пикселя изображения.

2.2.3.2. Алгоритм А11. Построение всех путей прохождения.

На изображении строятся все пути прохождения – 8-связные совокупности пикселей, построенные на изображении сверху вниз. Имея стоимостную функцию *e(I),* вычисляется стоимость таких путей (42):

 (42)

После этого можно переходить к следующему алгоритму, в котором будет производиться выбор путей с минимальной стоимостью для того, чтобы качественно произвести сегментацию.

2.2.3.3. Алгоритм А12. Выбор путей с минимальной стоимостью.

Выбор путей с минимальной стоимостью осуществляется по формуле (43):

 (43)

2.2.3.4. Алгоритм А13. Нахождение линий раздела.

Для того, чтобы узнать, является ли цепочка пикселей линией раздела, необходимо проверить ее на соответствие следующим условиям:

a) суммарная энергетическая функция всех пикселей вертикальной цепочки является минимальной;

b) цепочка пересекает изображение снизу вверх;

c) цепочка состоит только из 1-го пикселя из каждой строки;

d) соседние пиксели соединены сторонами и углами.

Если цепочка пикселей соответствует всем вышеперечисленным условиям - проводится линия раздела. Если нет – происходит переход к следующей цепочке пикселей на изображении.

2.2.4. Алгоритмы блока распознавания маркировки

2.2.4.1. Алгоритм А14. Нормирование.

В рамках работы алгоритма нормирования производится работа над каждым сегментированным элементом. Для каждого фрагмента в пределах имеющейся сетки  производится максимально возможное пропорциональное увеличение изображения с пропорциональным увеличением линий контура.

2.2.4.2. Алгоритм А15. Получение индивидуальных характеристик цифры.

Получение индивидуальных характеристик каждой цифры происходит по следующему алгоритму.

На первом этапе происходит подсчет количества черных пикселей в каждом из столбцов *xi* (*i=1…n*) изображения. Далее происходит подсчет количества черных пикселей в каждой строке *yi* (*i=1…m*) изображения.

Следующий шаг – нормирование полученных значений, это происходит по следующим формулам:

 (44)

 (45)

где *h* и *w* – высота и ширина изображения соответственно.

Полученные коэффициенты будут являться индивидуальными характеристиками цифр.

2.2.4.3. Алгоритм А16. Поочередное сравнение индивидуальных характеристик с эталоном.

2.2.4.3.1. Разработка эталона для распознавания.

Выработка эталона распознавания производится путем усреднения элементов векторов  и  для каждой цифры при заданном количестве элементов. Так, например, элемент эталонного (усредненного) вектора  для цифры «0» по пяти элементам определяется из соотношений (46) и (47):

 (46)

 (47)

где *l* – количество тестовых изображений цифры «0».

Векторы  и , элементами которых являются нормированные величины  и , выступают в качестве характеристик соответствующих цифр. Индекс *p* обозначает принадлежность вектора к соответствующей цифре, а индекс *l* – к соответствующему тестовому изображению.

Данные характеристики изображений должны соответствовать следующим условиям:

a) 

b) 

c) 

Поиск максимального совпадения производится путем нахождения рассогласований между индивидуальной характеристикой распознаваемой цифры и эталонными характеристиками всех цифр и выбора минимального из них. Определяются рассогласования отдельно для векторов  и  и в качестве третьего параметра вычисляется их сумма:

 (48)

 (49)

 (50)

Для облегчения анализа результатов при тестировании все полученные для какой-либо распознаваемой цифры величины  нормируются с помощью ,  - с помощью , а  - с помощью .

В этом случае минимальные значения нормируемых ,  и  равны 1 и могут быть легко выделены.

2.2.4.3.2. Распознавание маркировки.

Рассматриваемая цифра считается распознанной при выполнении одного из логических выражений к величинам (48), (49) и (50) эталонной характеристики какой-либо цифры:

 (51)

 (52)

Для более качественного распознавания необходимо проверить каждую цифру на соответствие хотя бы одному из условий (51) и (52).

В результате сравнения реальных характеристик с эталонными выводится результат распознавания.

3. Реализация ИС

В соответствии с пунктом «Исходные данные к проекту», изложенном в анализе технического задания, разработка информационной системы ведется на языке программирования Python. Также, для разработки потребуются библиотеки Numpy и OpenCV.

Перед тем, как начать работать с изображением в программе, его необходимо открыть. Для этого была написана функция uploadImage(self, path):

def uploadImage(self, path):

self.\_\_image = None

f = open(path, "rb")

chunk = f.read()

chunk\_arr = np.frombuffer(chunk, dtype=np.uint8)

self.\_\_image = cv2.imdecode(chunk\_arr, 0)

return False if self.\_\_image is None else True

Данная функция считывает путь изображения и открывает его в программе в виде двумерного массива.

Для отслеживания изменений изображения была написана функция \_\_debagShow(self,image):

def \_\_debagShow(self,image):

if self.\_\_debug:

cv2.imshow('debug', cv2.resize(image, (960, 540)))

cv2.waitKey()

Эта функция необходима лишь на этапе разработки (для того, чтобы фиксировать изменения на каждом этапе в пояснительной записке), в конечном варианте она будет отсутствовать.

Для работы данной функции необходимо проинициализировать переменную debug:

def \_\_init\_\_(self, debug=False):

self.\_\_debug = debug

Для отображения текущего состояния изображения используется функция showCurrentStateImage(self):

def showCurrentStateImage(self):

if self.\_\_image is not None:

cv2.imshow('image', cv2.resize(self.\_\_image, (960, 540)))

cv2.waitKey()

return True

else:

return False

Следующим этапом, согласно структурной модели, представленной на рисунке 4, идет предобработка изображения – этап, необходимый для подготовки изображения к дальнейшей работе, на котором минимизируется действие внешних факторов (таких как шумы, размытие изображения и световые блики).

Работа над очисткой изображения от шума и световых бликов осуществляется при помощи использования маски фильтра, который проходит по изображению и удаляет дефекты.

Функция \_\_masking(self, image, mask) выглядит следующим образом:

def \_\_masking(self, image, mask):

mask[mask>0] = 1

return cv2.bitwise\_and(image, image, mask=mask)

Световые блики занимают гораздо больше места на изображении, чем импульсный шум и какие-либо неровности в окраске металла, которые при прокате листа металла по конвейеру и фотографировании в это время, могут размываться. Исходя из этого было написано три функции: \_\_eraseBigAreas(self, mask), \_\_eraseSmallAreas(self, mask) и \_\_eraseNoise(self). Вторая функция была написана для того, чтобы можно было ее использовать в работе третьей функции следующим образом:

def \_\_eraseNoise(self):

\_, mask = cv2.threshold(self.\_\_image, 0, 255, cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

mask = cv2.medianBlur(mask, 3)

mask = cv2.dilate(mask, cv2.getStructuringElement(cv2.MORPH\_DILATE,(7,7)), iterations = 1)

mask = self.\_\_eraseBigAreas(mask)

self.\_\_image = self.\_\_masking(self.\_\_image, mask) # Big areas erased

\_, mask = cv2.threshold(self.\_\_image, 0, 255, cv2.THRESH\_BINARY+cv2.THRESH\_OTSU)

mask = cv2.dilate(mask, cv2.getStructuringElement(cv2.MORPH\_DILATE,(9,9)), iterations = 3)

self.\_\_debagShow(mask)

mask = self.\_\_eraseSmallAreas(mask)

self.\_\_image = self.\_\_masking(self.\_\_image, mask)

self.\_\_debagShow(self.\_\_image)

Код функции поиска зашумленных областей изображения приведен ниже:

def \_\_eraseSmallAreas(self, mask):

cnts = cv2.findContours(mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = cnts[0] if len(cnts) == 2 else cnts[1]

max\_area = 0

for c in cnts:

area = cv2.contourArea(c)

if area > max\_area:

max\_area\_c = c

max\_area = area

mask = np.zeros\_like(mask)

cv2.drawContours(mask, [max\_area\_c], -1, (255,255,255), -1)

return mask

Код поиска световых бликов на изображении и удаления их выглядит следующим образом:

def \_\_eraseBigAreas(self, mask):

cnts = cv2.findContours(mask, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

cnts = cnts[0] if len(cnts) == 2 else cnts[1]

max\_area = 5000

for c in cnts:

area = cv2.contourArea(c)

if area > max\_area:

cv2.drawContours(mask, [c], -1, (0,0,0), -1)

return mask