Правительство Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего профессионального образования

«Национальный исследовательский университет   
«Высшая школа экономики»

Факультет Бизнес-информатика

Отделение Программной инженерии

Кафедра Управление разработкой программного обеспечения

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА**

На тему: Программа распознавания QR кодов устойчивых к аффинным преобразованиям

Листов 34

|  |
| --- |
| Руководитель работы: доцент каф. УРПО  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Ахметсафина Р. З./  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. |
| Исполнитель: студент группы 471 ПИ  \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ /Пучковский А.И. /  «\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2013 г. |

Москва 2013 г

# Аннотация

Тема данной выпускной квалификационной работы – программа распознавания QR кодов устойчивых к аффинным преобразованиям. Результатом работы является реализация такой программы с полным заявленным функционалом: получение и обработка изображений, детектирование QR кода на изображении, декодирование QR кода.

В первой части работы приводится краткая история возникновения и использования QR кодов, актуальность данной работы, ее цели и задачи. Во второй части представлен обзор необходимых для достижения поставленных целей алгоритмов. Третья часть состоит из объяснения принципа работы программы распознавания и декодирования QR кодов. В заключении описаны результаты работы, область ее применения и направления для дальнейшей работы.

Оглавление

[Аннотация 2](#_Toc357867282)

[Введение 5](#_Toc357867283)

[1. Анализ предметной области 7](#_Toc357867284)

[1.1 Описание предметной области 7](#_Toc357867285)

[1.2 Стандарт QR кодов 8](#_Toc357867286)

[1.3 Распознавание QR кода 11](#_Toc357867287)

[2 Описание используемых алгоритмов 12](#_Toc357867288)

[2.1 Размытие изображения 12](#_Toc357867289)

[2.2 Бинаризация изображения 14](#_Toc357867290)

[2.3 Разметка (labeling) изображения 15](#_Toc357867291)

[2.4 Аффинные преобразования 17](#_Toc357867292)

[2.5 Кодирование QR кодов 19](#_Toc357867293)

[2.5.1 Генерация бинарной строки 19](#_Toc357867294)

[2.5.2 Генерация кодов Рида-Соломона для коррекции ошибок. 21](#_Toc357867295)

[2.5.3 Mask Patterns и Штрафные баллы 23](#_Toc357867296)

[2.5.4 Декодирования QR кодов 26](#_Toc357867297)

[3 Особенности реализации программы 27](#_Toc357867298)

[3.1 Выбор средств реализации 27](#_Toc357867299)

[3.2 Алгоритм обработки изображения 28](#_Toc357867300)

[3.3 Фильтрация областей 29](#_Toc357867301)

[3.4 Пример работы 30](#_Toc357867302)

[Заключение 34](#_Toc357867303)

[Список литературы 35](#_Toc357867304)

# Введение

Понятие “QR код” (Quick Response code) возникло впервые в 1994 году в Японии [1]. Этот стандарт штрих-кодов был разработан и представлен компанией “Denso-Wave” для отслеживания разных стадий производства продукции Toyota на cвоих предприятиях. В скором времени этот стандарт получил большую популярность среди рекламных и маркетинговых компаний, так как QR коды сделали процесс взаимодействия человека с объектом интерактивнее. Простота и популярность QR кодов сделали возможным использование этих двухмерных штрих кодов во всех сферах человеческой жизни, а так же промышленности.

QR код представляет из себя матричный (двухмерный) штих-код, который позволяет зашифровать гораздо больше информации, по сравнению с обычными одномерными штрих-кодами (например: “EAN/UPC”, “Code 128”), а также позволяют хранить различные типы данных (числа, символы, иероглифы, а так же смешанные типы данных). Этот стандарт штих-кодов получил большую популярность благодаря тому, что процесс распознавания не требует специального сканирующего оборудования такого, как специальный сканирующий луч, и может быть распознан при помощи любой камеры и установленного декодирующего программного обеспечения на устройстве. Благодаря этому, QR коды используются практически во всех сферах человеческой жизни, таких как: торговля, маркетинг, логистика, реклама, досуг, государственные услуги, туризм и многие другие. Кроме того, стандарт QR кодов имеет несколько уровней коррекции ошибок, что делает их использование еще более надежным и востребованным.

На сегодняшний день распознавание QR кодов не является идеальным. Для того чтобы QR коды были правильно распознаны, необходимо чтобы распознаваемый QR код был расположен параллельно объективу камеры. Кроме того, пользователь должен расположить специальную ограниченную область (чаще всего ограниченную нарисованными на экране считывающего устройства уголками) непосредственно над распознаваемым QR кодом. Возникает проблема, заключающаяся в том, что QR коды можно распознать только при определенном расположении камеры относительно самого штрих кода и при определенном расположении QR кода в пространстве.

Актуальной является задача создания упрощенной версии программного обеспечения для данного типа штрих кодов, при использовании которой пользователь не должен размещать код в «прицел» видоискателя. Код должен распознаваться вне зависимости от его расположения на изображении, размера, положения относительно осей координат. Эти преобразования относятся к группе аффинных преобразований, при которых сохраняется параллельность прямых и соотношение расстояний между точками лежащими на прямой [2]. Единственным требованием к пользователю является расположение объектива параллельно плоскости изображения.

Целью данной выпускной квалификационной работы является разработка программы распознавания QR кодов инвариантных к аффинным преобразованиям.

Для достижения цели работы необходимо решить следующие задачи:

1. Изучить алгоритмы преобразования геометрических объектов на плоскости, кодирования и декодирования QR кодов, распознавания и обработки изображений;
2. Разработать алгоритмы выделения QR кодов на изображении;
3. Разработать алгоритмы, необходимые для преобразования QR кодов к стандартному виду с помощью аффинных преобразований;
4. Реализовать алгоритм декодирования QR кода, содержащего символы ASCII;
5. Разработать программу распознавания и декодирования QR кодов

# 1. Анализ предметной области

## 1.1 Описание предметной области

Компьютерные и вычислительные технологии в 21 веке непрерывно совершенствуются и модернизируются. То, что было немыслимо 10 лет назад, сейчас является неотъемлемой частью современной жизни. В связи с этим, новые технологии разрабатываются для более мобильного и интерактивного взаимодействия человека с окружающим миром. Они упрощают процесс выполнения монотонной работы и сокращают много времени затрачиваемого на процессы, которые могут быть выполнены вычислительными системами за считанные секунды. Кроме того, в современном мире хранится очень много информации обо всем и возникает потребность в быстром поиске и обработке этой информации. Прогресс в данной области в значительной степени определяется развитием соответствующих технологий.

Одной из наиболее новых технологий, которая получила общественное признание, является технология кодирования QR кодов. О том, что такое QR код, в чем его особенность и как он может быть применен в повседневной жизни, описано в следующей главе.

## 1.2 Стандарт QR кодов

QR код (рис. 1) это разновидность двухмерного штрих кода, изначально спроектированный для производства автомобилей фирмы Toyota в Японии. Принцип использования QR кодов заключается в том, что распечатанный или нарисованный код помещается на объект, после чего он может быть считан и расшифрован при помощи устройства, у которого есть функционирующая камера и установленное программное обеспечение, которое декодирует сам QR код. Название QR код расшифровывается с английского языка как Quick Response code, то есть код быстрого отклика. Этот стандарт кодов пришел на смену обычного штрих-кода, который получил огромную популярность благодаря превосходным функциональным характеристикам, точностью содержащейся в нем информации и скорости ее считывания. Но главным недостатком обычного штрих-кода по сравнению с QR кодом является небольшой допустимый объем хранимых данных, а так же ограничения на типы данных, которые могут храниться в штрих-коде.



Рисунок 1: Пример QR кода

Информация в QR коде располагается в двух направлениях – горизонтально и вертикально. Благодаря такому расположению данных в QR коде, он способен хранить во много раз больше информации, чем его предшественники, включая разные типы данных: цифры, буквы, иероглифы, символы и т.д. Максимальный объем информации различных типов данных, который помещается в один QR код, представлен в таблице 1:

Таблица 1.

Типы данных и максимальный возможный объем в QR-кодах

|  |  |  |
| --- | --- | --- |
| Тип данных | Максимальный объем (символ) | Возможные символы |
| Числовые данные | 7 089 | 9,8,7,6,5,4,3,2,1,0 |
| Символьные данные | 4 296 | A–Z, $, %, \*, +, -, ., /, space, : |
| Бинарная информация, байт | 2 953 | JIS X 0201 |
| Иероглифы | 1 817 | JIS-X/0208 |

Еще одним преимуществом QR кода является его способность восстанавливать содержащуюся в нем информацию, даже если определенная часть символов на изображении QR кода были повреждены или не распознаны. Это стало возможным благодаря системе коррекции ошибок на базе кодов Рида-Соломона [6-8]. Максимальное количество кодовых слов, которое может быть восстановлено, составляет до 30%. В соответствии со спецификацией [3] QR код имеет 4 степени коррекции ошибок: L – 7%, M – 15%, Q – 25%, H – 30%. Чем выше степень коррекции ошибок, тем меньше данных можно зашифровать и поместить в QR код.

Сам QR код состоит из определенного набора меток и непосредственно пикселей, которые представляют собой закодированное сообщение, хранимое в QR коде. На любом QR коде обязательно должны присутствовать 6 видов меток :

1. Позиционирование

Область необходимая для детектирования кода

1. Номер версии

Определяет какая версия кода используется (от 1 до 40)

1. Синхронизация

Дублируется в двух направлениях, и позволяют понизить вероятность возникновения ошибок при считывании, системной информации (например, версия, тип данных и тп.)

1. Формат

Необходимы для определения типы данных закодированных в коде

1. Выравнивание

Используются для лучшего позиционирования кода во время обработки (при версии QR кода выше 1).

1. Уровень коррекции ошибки

Позволяют определить, какой уровень помехозащищённости был использован на этапе кодирования для правильного выбора способа обнаружения возможных ошибок в коде.

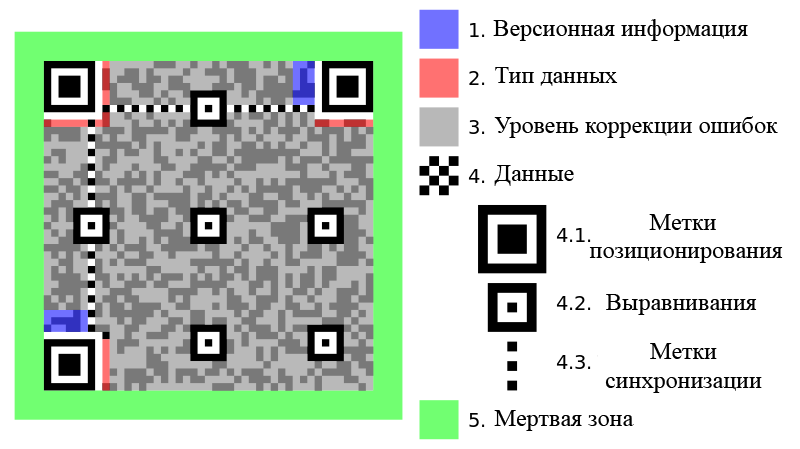


Рисунок 2: Метки и данные на QR коде

Более подробно узнать о каждой из меток можно в официальной спецификации ISO/IEC 18004:2006 [3].

Далее в работе будут обсуждаться только метки позиционирования, так как они будут необходимы для обнаружения QR кода на изображении.

## 1.3 Распознавание QR кода

Во всех существующих программах, которые считывают и декодируют QR коды, реализован простой алгоритм обнаружения QR кода на изображении, полученном с камеры. Затем реализована стандартная процедура декодирования информации из QR кода [3]. Однако этот алгоритм распознавания требует очень четкого позиционирования специально выделенной области на снимающем устройстве и определенного расположения QR кода в пространстве (рис. 2). После того, как специально определенная область на снимающем устройстве четко совпала с гранями QR кода, происходит поиск трех меток позиционирования, о которых говорилось в предыдущем разделе. Эти метки расположены строго в определенных местах в выделенной на снимающем устройстве области изображения. Недостаток такого подхода заключается в том, что QR код не может находиться в любой области на изображении, и пользователю необходимо самому предварительно фокусироваться на необходимом QR коде, и следить за тем, чтобы область для съемки QR кода совпадала с самим QR кодом, который необходимо декодировать.



Рисунок 3: Область позиционирования

# 2 Описание используемых алгоритмов

## 2.1 Размытие изображения

Перед тем как получить бинаризированное изображение, необходимо исходное цветное изображение конвертировать в изображение, состоящее из градации серых оттенков. Минимальная насыщенность пикселя в этом случае равняется самом темному цвету, который возможен, а максимальное значение пикселя равняется белому цвету. Метод, который используется для конвертирования цветного изображение в оттенки серого, базируется на разности освещенности пикселей. Данный метод является усовершенствованным аналогом метода усреднения значения трех компонент (R,G,B) каждого пикселя, по формуле:

Используемый метод, основанный на яркостной составляющей, так же усредняет значения пикселей, но в формулу вычисления значения пикселя добавлены экспериментально полученные коэффициенты для каждого компонента согласно человеческой цветочувствительности. Было выяснено, что человеческий глаз более чувствителен к зеленому цвету, по сравнению с остальными, поэтому у зеленого цвета полученный коэффициент наивысший. Существует много формул перевода цветного пикселя в серый согласно данному методу, но в них используются разные коэффициенты. Для данного проекта были выбраны коэффициенты для цветных компонентов, приведенные ниже:

В результате вычисления всех пикселей на изображении по приведенной формуле, будет получено новое изображение, представленное только в оттеках серого. Диапазон оттенков варьируется от 0 – черный цвет, до 255 – белый цвет.

## 2.2 Бинаризация изображения

Для того чтобы бинаризировать исходное изображение, использовался метод Оцу, предложенный ученым N. Otsu [4]. Суть метода заключается в том, что изображение, которое необходимо бинаризировать, состоит из двух классов пикселей (например, области интереса и фона) или образует так называемую бимодальную гистограмму. Для дальнейших действий вычисляется оптимальный порог, который разбивает эти два класса так, чтобы их внутригрупповая дисперсия была минимальной. Главное достоинство данного метода – это быстродействие.

Для бинаризации изображения можно так же использовать статическое значение порога (например, значение 113), но в этом случае результаты не всегда будут хорошими, так как разброс цветов различается от изображения к изображению. Метод Оцу, в отличие от статического метода, определяет порог динамически для каждого нового изображения, и разбивает эти два класса наиболее удачным образом. Математически данный метод описывается следующим образом:  
1. Пусть q1 и q2 – это вероятности каждого класса, вычисленные по формулам

2. Дисперсии для каждого из классов

3. Средние значения каждого класса равняются:

Для того чтобы получить результат необходимо максимизировать следующее уравнение:

В результате будет получен необходимый порог, по которому нужно разделить гистограмму на два класса и присвоить пикселям пикселям одного класса, меньшего, присваиваются нули, а пикселям другого класса - единицы. После этой процедуры получится полностью бинаризированное изображение (рис. 3)



Рисунок 4: Оригинальное и бинаризированное изображение

## 2.3 Разметка (labeling) изображения

Для обработки бинаризированного изображения и поиска на нем объектов необходимо выделить области (далее эти области будут называться «блобами»), образующие связанные участки на изображении. После разметки изображения каждому пикселу блоба будет присвоен один общий уникальный номер (label). В результате каждый блоб будет описан всеми пикселями, которым был присвоен номер данного блоба. Наглядно процесс разметки представлен на рисунке 4. На приведенном рисунке одному блобу принадлежат сразу несколько номеров. Это возникает из-за того, что описываемый далее алгоритм является однопроходным (один раз оценивает каждый пиксель), но в дальнейшем одному блобу присваивается только минимальный номер из тех, что попали в него.

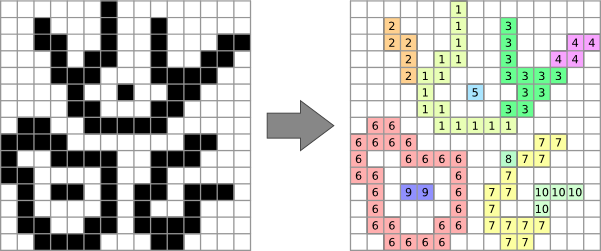


Рисунок 5: Разметка изображения.

Алгоритм, позволяющий произвести процесс разметки бинаризированного изображения, основан на том, что специальная маска (рис. 5) перемещается слева направо и сверху вниз, попадая на каждый пиксель. Одна позиция в маске соответствует исходному пикселю, а остальные представляют уже обработанные этой маской пиксели. Для того чтобы определить, какой номер необходимо поставить в исходный пиксель, осуществляется поиск минимального номера в маске у его соседей. Если такой был найден, то он становится номером маски, иначе исходному пикселу присваивается следующий по порядку номер, которого еще не было. В случае, когда в маске возникает конфликт, то есть на одном блобе обнаруживается сразу несколько номеров, выбирается минимальный из конфликтующих номеров, а остальные добавляются в специальный буфер и тем самым становятся связанными с минимальным.

Macintosh HD:Users:Puchok:Desktop:kernel.png

Рисунок 6. Маска для разметки (X – исходный пиксель;

A, B, C, D – связанные пиксели)

После того, как все пиксели были размечены, производится обработка блобов так, чтобы они содержали только один номер.

## 2.4 Аффинные преобразования

Ранее в работе уже упоминалось, что аффинные преобразования - это такие преобразования, которые сохраняют параллельность линий, а так же пропорции между точками, лежащими на прямых [2]. Данный вид преобразований является подмножеством проективных преобразований на плоскости. Проективные преобразования, в отличии от аффинных, не сохраняют параллельность прямых. При проективных преобразованиях сохраняется коллинеарность точек: три точки, лежащие на одной прямой, после преобразования остаются лежать на одной прямой; проективные преобразования связаны с отображением трехмерной информации на двумерную плоскость. Полная группа аффинных преобразований состоит из четырех трансформаций:

1. Поворот
2. Растяжение
3. Сдвиг
4. Масштабирование

Эти четыре типа преобразований представлены на рисунке 6. Видно, что все эти преобразования не изменяют форму преобразуемого объекта.

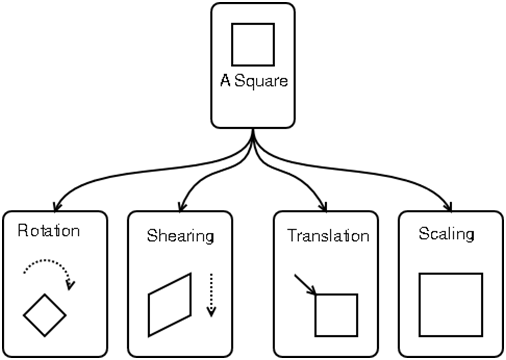


Рисунок 7: Группа аффинных преобразований

Преобразования данной группы могут быть представлены следующим образом:

Кроме того, эти преобразования могут быть представлены в матричном виде:

где

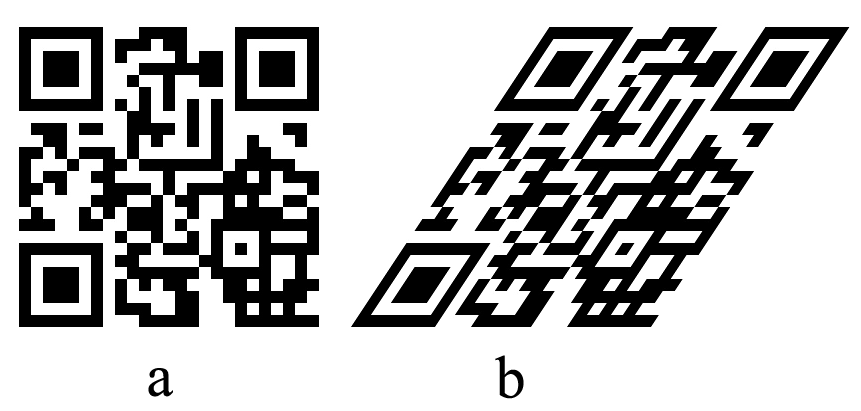


Рисунок 8: Пример операции растяжения на QR коде.

Любое аффинное преобразование можно представить в виде композиции последовательно выполненных простейших преобразований определенного типа. Зная параметры аффинного преобразования, можно так же вычислить параметры обратного преобразования для каждого случая.

## 2.5 Кодирование QR кодов

### 2.5.1 Генерация бинарной строки

Важной особенностью QR кодов является их симметричность относительно кодирования и декодирования. В связи с этим легче начинать рассмотрение с процесса кодирования, а процесс декодирования будет выполняться в обратном порядке.

Процесс кодирования QR кодов делится на несколько главных этапов.

Первый этап заключается в том, что строка с входными данными предварительно обрабатывается и приводится к стандартному для QR кодов виду.

После того, как получена строка с исходными данными, начинается процесс генерации бинарной строки. В начало бинарной строки X добавляются 4 технических бита, которые определяются на основе кодируемого типа данных. Список возможных комбинаций включает в себя: 0001 - числовые данные, 0010 – буквенно-символьные данные, 0100 - бинарная информация, 1000 - специальные японские иероглифы. В данной работе рассматривались данные только численного и символьного типа, для того, чтобы упростить процесс отображения результатов.

На следующем этапе генерации подсчитывается общее количество символов в исходной строке и десятичное число преобразуется в бинарное представление. (например, если исходное сообщение содержит 11 символов, включая пробелы, это будет значить то же самое, что бинарная строка 1011). Кроме того, необходимо определить длину генерируемой бинарной строки согласно версионному стандарту компании Denso Wave [4]. (например, по версии 1 QR кода символьная информации должна состоять из 9 бит). Таким образом, полученная ранее строка 1011 должна быть дополнена нулями с левой стороны, чтобы получилось 9 бит в итоге. После получения этих 9 бит, они конкатенируются со строкой X с правой стороны.

Для того чтобы закодировать символьный тип данных, исходная строка должна быть разбита на пары символов. Затем нужно узнать значение из ASCII таблицы для первого символа в каждой из пар, это значение умножается на число 45 (получено экспериментальным путем и признано стандартом согласно спецификации), затем берется значение согласно той же ASCII таблице для второго символа в каждой из пар. Эти два значения суммируются, и каждая пара символов имеет свое численное значение. Полученные числа должны быть последовательно представлены в виде 11-битных чисел в бинарном виде. Если в строке имеется нечетное количество символов, то берется значение для этого символа из ASCII и конвертируется в 6-битное бинарное число. Все полученные блоки также конкатенируются с генерируемой строкой X с правой стороны.

После выполнения этапов, описанных выше, необходимо опять обратиться к версионной таблице компании DENSO WAVE [4], и определить необходимую длину конечной бинарной строки. Если длина строки X меньше необходимой, с правой стороны добавляется от 1 до 4 бит в зависимости от того, сколько символов не хватает. Сгенерированная строка X разбивается на 8-битные токены (блоки) и, если последний блок меньше 8 бит, то к нему дописываются нули с правой стороны.

На заключительном этапе генерации бинарной строки, вспомогательные токены конкатенируются со строкой X. В случае, если сгенерированная строка меньше положенного (согласно версионной таблице) 2 типа 8 битных блоков добавляются в конец сгенерированной строки по очереди, пока не будет необходимого числа бит: 11101100 и 00010001. После того, как сгенерированная строка содержит необходимое число бит, бинарная строка считается окончательно сгенерированной и может быть использована на следующих этапах генерации QR кодов.

### 2.5.2 Генерация кодов Рида-Соломона для коррекции ошибок.

Ирвин Рид и Густав Соломон впервые представили помехозащищенные коды, впоследствии названные кодами Рида-Соломона, в 1959 году [7]. В общем случае при генерации кодов Рида-Соломона используется перемножение и деление полиномов друг на друга. Например, если полином A это кодовое слово (сгенерировано на основе полученной на предыдущем этапе бинарной строки), а B это неприводимый полином, который известен обеим сторонам, обменивающимся информацией, тогда их перемножение сгенерирует новый полином C. Формально все полиномы могут быть определены следующим образом:

Отсюда следует, что когда передаваемое сообщение будет получено, процесс его декодирования будет заключаться в делении полиномов друг на друга. Если при делении полинома C на полином B остается остаток, значит, переданное сообщение содержит ошибку. Остаток от деления образует синдром полинома, благодаря которому можно узнать поврежденные биты. Кроме того в некоторых случаях можно исправить определенное количество ошибок в полученном сообщении. Если степень полинома B отличается от степени полинома A как минимум на 2, тогда возможно не только определить наличие или отсутствие ошибки в передаваемом сообщение, но так же и исправить некоторые из них. В этом случае, приведенное далее уравнение (где k - это степень полинома; t - это максимальное значение возможных исправлений) показывает, что избыточность кодов Рида-Соломона составляет:

k = 2 \* t

Процесс генерации кодов Рида-Соломона, их кодирования и декодирования, описан в приложении спецификации QR кодов [3]. В качестве результата генерации строки с помощью кодов Рида-Соломона для исходного сообщения, будет сгенерирован набор десятичных значений. Их необходимо преобразовать в бинарный формат и конкатенировать с исходной строкой с правой стороны. Результатом всех предыдущих этапов будет строка в бинарном формате, содержащая в себе исходные данные сообщения плюс коды Рида-Соломона, для того чтобы обеспечить возможность коррекции ошибок в декодированном сообщении.

### 2.5.3 Mask Patterns и Штрафные баллы

Используя коды Рида-Соломона генерируются кодовые слова с целью восстановления ошибок при чтении информации из QR кода. Затем они объединяются с исходным сообщением, которые было сгенерировано на первом этапе алгоритма. В итоге получится одна строка, содержащая определенное количество бит согласно спецификации. На данном этапе QR код до сих пор не сформирован. В данном разделе речь пойдем о том, как генерируется двухмерное представление QR кода.

Для начала генерируется стандартный шаблон QR кода, в котором есть все необходимые технические биты, необходимые для определения вспомогательной информации. Такой стандартный шаблон является основой любого QR кода, он изображен на рисунке 8. Этот шаблон всегда включает в себя три позиционирующих маркера, расположенных в верхнем левом и правом углу, а так же в нижнем левом углу шаблона. Кроме того, все шаблоны содержат специальные Timing Patterns (поочередно чередующиеся белые и черные пикселы), которые служат для предварительного определения версии QR кода.

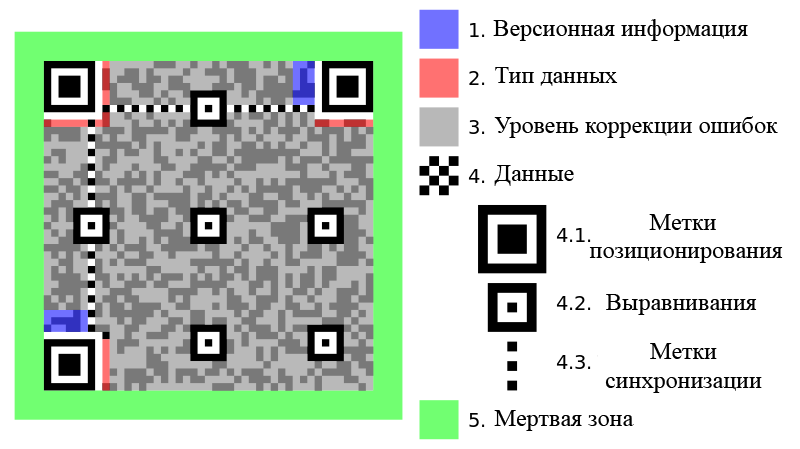


Рисунок 9: Метки и данные на QR коде [5]

Для версий кодов, отличных от первой, используются так же специальные выравнивающие примитивы, которые располагаются в заданных местах и служат для однозначного определения QR кода. Более подробно о них рассказано в спецификации [3]. При известных версии QR кода и уровня коррекции ошибок и при наличии всех технических примитивов в шаблоне, начинается заполнение шаблона QR кода сгенерированной ранее информацией. Процесс добавление битов исходного сообщения в матрицу осуществляется следующим образом: сначала первый пиксель (цвет выбирается согласно значению бита) располагается в самом правом нижнем углу шаблона. Второй пиксель добавляется слева от первого на той же линии. Следующая пара битов располагается сверху, аналогично первым двум. Так происходит заполнение всего столбика шаблона. Когда заполняемый столбец достиг верха шаблона, процесс начинается заново, но уже с первой свободной клетки расположенной справа сверху (тот же самый процесс, но только уже снизу вверх), так же как и первый столбец.

Крайне важно при заполнении шаблона не забывать пропускать технические биты. Таким образом, заполняется весь шаблон QR кода. Полученный QR код не является полностью законченным на данном этапе. Для того, чтобы завершить процесс генерации кода, необходимо применить специальные маски и посчитать у полученных QR кодов штрафные значения. Согласно спецификации, необходимо сгенерировать 8 разных видов QR кодов на основе только что сгенерированного. Каждый из этих 8 кодов определен своей формулой, которые представлены в таблице 2 [3].

Таблица 2

Mask pattern’ы. (i - индекс строки, j - индекс столбца)

|  |  |
| --- | --- |
| Mask Pattern | Условие |
| 0 | (*i + j*) mod 2 = 0 |
| 1 | *i* mod 2 = 0 |
| 2 | *j* mod 3 = 0 |
| 3 | (*i + j*) mod 3 = 0 |
| 4 | ((*i* div 2) + (*j* div 3)) mod 2 = 0 |
| 5 | (*i j*) mod 2 + (*i j*) mod 3 = 0 |
| 6 | ((*i j*) mod 2 + (*i j*) mod 3) mod 2 = 0 |
| 7 | ((*i j*) mod 3 + (*i+j*) mod 2) mod 2 = 0 |

Каждая из представленных формул используется для того, чтобы определить, следует ли заменить пиксель, расположенный с определенными индексам, на противоположное значение. Когда 8 кодов сгенерированы, необходимо подсчитать штрафные очки для каждого из них. В спецификации описаны 4 штрафных правила:

1. Добавить штрафные очки за каждый ряд из 5 пикслей одного и того же значения.
2. Добавить штрафные очки за каждый блок MxN одного и того же значения.
3. Добавить увеличенные штрафные очки за наличия областей похожих на выравнивающие примитивы.
4. Добавить штрафные очки, если больше половины пикселей одного и того же значения.

В результате считаются штрафные очки для каждого кода, и тот у которого будет наименьшее значение очков, будет итоговым QR кодом. Рисунок 9 демонстрирует процесс подсчета штрафных очков первых трех кодов (остальные не показаны).

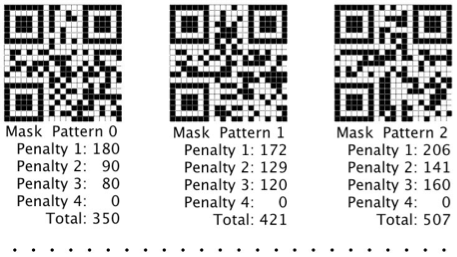


Рисунок 10: Штрафные очки [5].

Если выбирать из тех кодов, что представлены на Рисунке 9, то результирующим кодом будет “Mask Pattern 0”, потому что у него наименьшее значение штрафов. На данном этапе процесс генерации QR кода закончен.

### 2.5.4 Декодирования QR кодов

Согласно спецификации [3], процесс декодирования состоит из 8 этапов. Все этапы декодирования полностью симметричны процессу кодирования, только выполняются в обратном порядке, поэтому они будут рассмотрены в общих чертах. Процесс декодирования QR кодов состоит из следующих стадий:

1. Получить изображение QR кода и распознать черные и белые пикселы на нем. Затем заменить эти пикселы на 1 и 0 соответственно.
2. Получить информацию о версии QR кода.
3. Получить информацию о типе данных закодированных в коде
4. Провести обратный процесс маскирования, согласно той формуле, которая использовалась при кодировании.
5. Считать символы исходного сообщения и биты необходимые коррекции ошибок в сообщении.
6. Найти и исправить ошибки в полученном сообщение согласно кодам Рида-Соломона.
7. Разделить полученные биты на блоки (технические биты и биты исходного сообщения).
8. Декодировать информационные символы согласно типу данных, который использовался при кодировании.

# 3 Особенности реализации программы

## 3.1 Выбор средств реализации

Для написания программы распознавания QR кодов устойчивых к аффинным преобразованиям был выбран объектно-ориентированный язык программирования JAVA и среда разработки программных продуктов Eclipse 4.2.1. В процессе разработки для тестирования и проектирования дополнительно были использованы следующие сторонние библиотеки:

* JavaCV – обертка для языка Java библиотеки OpenCV написанной на C++;
* zxing – java библиотека содержащая методы необходимые для обработки штрих кодов;
* BoofCV – библиотека обработки изображений для машинного зрения и приложений связанных с программированием роботов.

Программная документация представлена в Приложениях.

## 3.2 Алгоритм обработки изображения

Этапы, которые необходимы для достижения поставленных целей, представлены далее в том порядке, в котором они реализованы в программе. При успешном выполнении всех перечисленных этапов можно достичь решения поставленных задач.

1. Получение изображение с веб-камеры или из файла
2. Перевод изображения из цветного в изображение, состоящее из оттенков серого
3. Бинаризация изображения, состоящего из оттенков серого цвета
4. Разметка изображения на обособленные области (блобы)
5. Обработка получившихся областей (вычисление числовых и геометрических характеристик)
6. Фильтрация полученных областей
7. Поиск областей, которые по характеристикам с большей вероятностью похожи на обязательные метки позиционирования на QR коде
8. Выбор трех наиболее вероятных областей
9. Анализ расположения QR кода на изображении
10. Выделение QR кода из всего изображения, для последующей обработки
11. Определение необходимых преобразований
12. Применение необходимых преобразований к QR коду
13. Декодирование QR кода
14. При удачном завершении этапа декодирования, вывод на экран закодированной информации

## 3.3 Фильтрация областей

Области, которые удалось выделить на изображении после предварительной обработки и разметки изображения, должны быть оценены. Оценка этих областей позволит с определенной долей уверенности сказать, является она потенциальной частью QR кода или нет. Для того чтобы можно было сравнить эти области, необходимо найти некоторый набор численных показателей, а так же особенности анализируемой области. В качестве параметров для оценки области были выбраны следующие численные характеристики:

Таблица 3

Характеристики областей

|  |  |
| --- | --- |
| Характеристика | Описание |
| Площадь | Площадь области в пикселах |
| Центр масс (X) | x-координата центра масс |
| Центр масс(Y) | y-координата центра масс |
| Проекция (X) | Проекция области на ось x-координат |
| Проекция (Y) | Проекция области на ось y-координат |
| Ширина | Ширина оцениваемой области |
| Высота | Высота оцениваемой области |
| Периметр | Длина внешнего контура области интереса |
| Отклонение от окружности | Величина численно равная наибольшему расстоянию от точек реального профиля до прилегающей окружности (изменяется от 0 до 1) |
| Сумма X | Сумма x-координат для всей области |
| Сумма Y | Сумма y-координат для всей области |

В процессе обработки областей некоторые из них отбрасываются в связи с ограничениями, накладываемыми снимающим оборудованием, качеством снимков, а так же ограничениями размеров, при которых информация с QR кода не может быть считана, так как теряется значительная часть информации, необходимой для декодирования.

## 3.4 Пример работы

1) Загрузка исходного изображения



Рисунок 11: Исходное изображение

2) Конвертация изображения в оттенки серого цвета



Рисунок 12: Изображение в оттенках серого

3) Бинаризация изображения



Рисунок 13: Бинаризированное изображение

4) Выделение областей (примеры областей)

1) Macintosh HD:Users:Puchok:Documents:workspace_Java7:ImageProcessing:blobs:2saved.jpg 2) Macintosh HD:Users:Puchok:Documents:workspace_Java7:ImageProcessing:blobs:4saved.jpg 3) Macintosh HD:Users:Puchok:Documents:workspace_Java7:ImageProcessing:blobs:9saved.jpg

Рисунок 14: Примеры найденных областей

5) Анализ областей

На данном этапе у всех полученных областей производится расчет численных параметров (периметр, площадь, отклонение от окружности и т.д.), а так же дополнительно проверяются признаки областей (например, область должна состоять как минимум из двух частей, и одна из этих областей должна находится внутри второй области). После чего, полученные значения сравниваются, и выбирается 3 области, наиболее схожие по характеристикам с метками позиционирования на QR коде (например, область 1 из 4 пункта «Пример работы»).

6) Определение расположения найденных областей

На этапе обнаружения QR кода, решается проблема масштаба и смещения, так как мы определили координаты точек с точностью до константы. После того, как на изображении 3 необходимые области были найдены, определяется их взаимное расположение на плоскости и вычисляется угол, на который был повернут QR код. Найденный угол поворота необходим для восстановления правильного расположения кода, чтобы его можно было декодировать. Вычисления углов подробно рассмотрены в приложении к выпускной квалификационной работе.

7) Преобразование

После того, как угол поворота найден, QR код располагается на плоскости так, чтобы он имел стандартную форму и расположение.

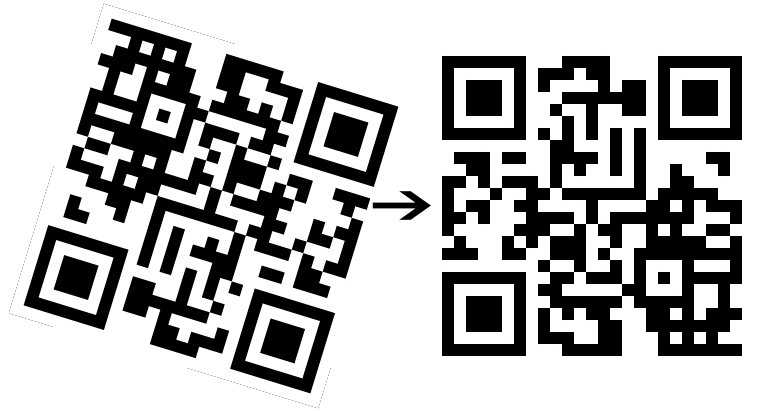


Рисунок 15: Восстановление расположения

8) Декодирование

Когда QR код был детектирован, правильно расположен и все его части распознаны, информация с QR кода заносится в матрицу. В результате декодирования, будет сгенерирована строка с исходным сообщением (например: «link: http://www.hse.ru»).

# Заключение

В работе описан подход к разработке программы распознавания QR кодов устойчивых к аффинным преобразованиям.

В результате были решены следующие задачи:

* изучены алгоритмы преобразования геометрических объектов на плоскости, кодирования и декодирования QR кодов, распознавания и обработки изображений;
* разработаны алгоритмы выделения QR кодов на изображении;
* разработаны алгоритмы, необходимые для преобразования QR кодов к стандартному виду с помощью аффинных преобразований;
* реализован алгоритм декодирования QR кода, содержащего символы ASCII;
* разработана программа распознавания и декодирования QR кодов;

В качестве программной реализации был разработан программный продукт «Программа распознавания QR кодов устойчивых к аффинным преобразованиям». В данной реализации использованы технологии обработки изображений и кодирования/декодирования QR кодов.

В качестве направления для дальнейшей работы можно выделить:

* + - 1. оптимизация способов распознавания QR кодов на изображении
      2. разработка программы распознавания QR кодов подверженных воздействию проективные преобразования
      3. разработка программы распознавания QR кодов в реальном времени (с видео потока)

# Список литературы

1. DENSO WAVE. (-). *About QR.com*. Retrieved February 15, 2013, from DENSO WAVE website: <http://www.qrcode.com/en/index.html>
2. Rogers, D. & Adams, A. (-). *Mathematical elements for computer graphics*. Second Edition. (Перевод Москва: издательство “Мир” 2001)
3. ISO/IEC 2000. (2000). *ISO/IEC 18004 Automatic identification and data capture techniques - Bar code symbology - QR code.*
4. DENSO WAVE. *Version capacity table*. Retrieved February 9, 2013, from DENSO WAVE website: <http://www.qrcode.com/en/vertable1.html>
5. Eby, Carolyn. (2012). *QR Code Tutorial*. Retrieved January 20, 2013, from Thonky website: <http://www.thonky.com/qr-code-tutorial/part-3-mask-pattern/>
6. Wicker, B. & Bhargava K. (1994). *Reed-Solomon Codes and Their Applications*. IEEE Press.
7. Семенов, А. (2007). *Введение в коды Рида-Соломона: принципы, архитектура и реализация*. Retrieved January 28, 2013, from Intuit website: <http://www.intuit.ru/department/network/algoprotnet/4/2.html>
8. Морелос-Сарагоса, Р. (2006). *Исскусство помехоустойчивого кодирования. Методы, алгоритмы, применение.* Москва: Техносфера.
9. Lowe, D. (2004). *Distinctive Image Features from Scale-Invariant Keypoints.* Vancouver: University of British Columbia.
10. Bay, H. & Tuytelaars, T & Van Gool L. (-). *SURF: Speeded Up Robust Features*. Zurich: Katholieke Universiteit Leuven
11. Edward Rosten & Tom Drummond (-). *Fusing Points and Lines for High Performance Tracking*. Cambridge: Department of Engineering, University of Cambridge

12. N. Otsu (1979). «A threshold selection method from gray-level histograms». *IEEE Trans. Sys., Man., Cyber.* **9**: 62-66.