

Министерство науки и высшего образования Российской Федерации  
Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий



## **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА**

**Разработка автоматизированной системы  
распознавания и идентификации продуктов  
питания**

Студент гр. 23541/1 А.А. Нагорнов

Санкт-Петербург  
2019



Министерство науки и высшего образования Российской Федерации  
Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
Кафедра компьютерных систем и программных технологий

Работа допущена к защите  
зав. кафедрой

\_\_\_\_\_ Б.М. Ицыксон  
«\_\_\_\_» \_\_\_\_\_ 2019 г.

## **ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА МАГИСТРА**

### **Разработка автоматизированной системы распознавания и идентификации продуктов питания**

по направлению 09.04.01 «Информатика и вычислительная техника»  
по образовательной программе  
09.04.01\_18 «Встраиваемые системы управления»

Выполнил студент гр. 23541/1 \_\_\_\_\_ А.А. Нагорнов  
Научный руководитель,  
к. т. н., доц. \_\_\_\_\_ К.В. Никитин  
Консультант по нормоконтролю,  
к. т. н., доц. \_\_\_\_\_ А.Г. Новопашенный

Санкт-Петербург  
2019



## РЕФЕРАТ

На 94 с., 75 рисунков, 5 таблиц, 1 приложение

### ВСТРАИВАЕМЫЕ СИСТЕМЫ, ОПРЕДЕЛЕНИЕ ФОРМЫ ОБЪЕКТОВ, ПОЛУЧЕНИЕ РАЗВЕРТКИ ЦИЛИНДРИЧЕСКИХ ОБЪЕКТОВ, ШАГОВЫЙ ДВИГАТЕЛЬ, КАМЕРА, РОЗНИЧНАЯ ТОРГОВЛЯ, ПРОДУКТЫ

Работа посвящена разработке аппаратно-программного комплекса для распознавания и идентификации продуктов питания в розничной торговли. В рамках работы был выполнен обзор и произведен анализ существующих систем, методов и решений, итогом которого стала разработка концепции и проектирование структуры программно-аппаратного устройства, реализующего решение поставленной задачи.

В результате был создан прототип системы для распознавания и идентификации продуктов питания, которая позволяет автоматизировать процесс оцифровки товара, снизить человеческие и финансовые затраты.

## THE ABSTRACT

94 pages, 75 pictures, 5 tables, 1 appendix

### EMBEDDED SYSTEMS, IDENTIFICATION OF THE SHAPE OF OBJECTS, PANORAMA OF THE CYLINDRICAL OBJECTS, A STEPPER MOTOR, A CAMERA, RETAIL, PRODUCTS

The diploma thesis is devoted to the development of the hardware and software complex for recognition and identification of foodstuffs in retail trade. Within the framework the review and the analysis of the existing systems, methods and solutions have been given, the outcome of which has been the development of the concept and the design of the

structure of the hardware and software device, implementing the solution of the assigned task.

As a result of the diploma thesis, a prototype of the system for recognition and identification of foodstuffs, which allows to automate the process of digitization of goods and to reduce human and financial cost, has been created.

## СОДЕРЖАНИЕ

<b>ВВЕДЕНИЕ . . . . .</b>	<b>7</b>
<b>1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЗОР СУЩЕСТВУЮЩИХ ПОДХОДОВ. ПОСТАНОВКА ЗАДАЧИ . . . . .</b>	<b>9</b>
1.1. Общие положения . . . . .	9
1.2. Существующие системы визуального контроля . . . . .	10
1.2.1. Системы визуального контроля фирмы Mettler Toledo . . . . .	10
1.2.2. Система линейного сканирования In-Sight 9902L . . . . .	11
1.2.3. Система OCR Data Traceability . . . . .	12
1.3. Современные подходы к построению систем распознавания и идентификации объектов . . . . .	13
1.3.1. Классификация объектов на основе нейронных сетей . . . . .	14
1.3.2. Классификация объектов на основе алгоритмов обработки 2D изображений . . . . .	15
1.3.3. Классификация объектов на основе 3D моделей . . . . .	19
1.4. Подходы к получению изображений объектов . . . . .	20
1.4.1. Обзор системы PhotoPizza . . . . .	23
1.4.2. Обзор системы Ciclop . . . . .	23
1.5. Постановка задачи . . . . .	25
<b>2. РАЗРАБОТКА СТРУКТУРНЫХ, СХЕМОТЕХНИЧЕСКИХ И КОНСТРУКТИВНЫХ РЕШЕНИЙ . . . . .</b>	<b>26</b>
2.1. Структурно-функциональная схема . . . . .	26
2.2. Разработка аппаратного обеспечения . . . . .	29
2.2.1. Управляющее устройство . . . . .	29
2.2.2. Разработка подсистемы управления приводным двигателем	32
2.2.3. Разработка подсистемы индикации . . . . .	35
2.3. Общая структура системы . . . . .	36
2.4. Разработка конструктивных решений . . . . .	36

<b>3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ</b>	40
3.1. Управление системой . . . . .	42
3.2. Сбор данных . . . . .	44
3.3. Определение формы объекта . . . . .	45
3.3.1. Удаление шумов . . . . .	45
3.3.2. Локализация объекта . . . . .	46
3.3.3. Морфологическое преобразование . . . . .	46
3.3.4. Нахождение контура . . . . .	48
3.3.5. Определение формы по контуру . . . . .	50
3.3.6. Определение положения объекта . . . . .	50
3.4. Алгоритм получения развертки цилиндрического объекта . . . . .	51
3.5. Распознавание текста . . . . .	55
<b>4. ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ</b> . . . . .	55
4.1. Метрики оценки качества . . . . .	56
4.1.1. Классификация формы . . . . .	56
4.1.2. Качество панорамных изображений . . . . .	57
4.2. Исследование объектов . . . . .	58
4.3. Использование реальной системы . . . . .	66
<b>ЗАКЛЮЧЕНИЕ</b> . . . . .	69
<b>СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ</b> . . . . .	71
<b>ПРИЛОЖЕНИЕ. ЛИСТИНГИ</b> . . . . .	74

## ВВЕДЕНИЕ

На сегодняшний момент нет единой сертифицированной базы данных, которая содержала бы полный перечень товаров. Существующие каталоги разрабатываются на частной основе, а их создатели не несут никакой ответственности за достоверность предоставляемых данных. Таким образом, продуктовые компании, а также производители тратят достаточно много времени и ресурсов для создания собственных каталогов.

Такие каталоги как правило содержат описание товара: перечень ингредиентов, сведения о пищевой ценности и содержании аллергенов; контактные данные производителя; масса-габаритные показатели; дату производства и срок годности. Все эти сведения должны быть тщательно проверены и соответствовать информации представленной на этикетках продукции.

Унификация и стандартизация справочной информации о товарах позволят бизнесу существенно сэкономить трудовые и финансовые ресурсы, которые сегодня уходят на многократное описание каждого наименования продукции в соответствии с требованиями различных организаций. Например, если сейчас какой-то производитель выпускает новый товар на рынке, то для того, чтобы завести товар в торговую сеть, например, в одну из сетей «Х5 Retail Group», «Metro», «Ашан», каждая из компаний проводит свои исследования. Производитель также должен тратить средства и силы, чтобы предоставить данные во множество источников [10].

Использование систем визуального контроля позволяет получить эту информацию за счет графической идентификации, что снижает риск человеческой ошибки, а также позволяет ускорить процесс. Продукты с недостоверной маркировкой вызывают недовольство потребителей, наносят репутационный вред организациям, а также могут навредить потребителю, например, из-за отсутствия информации о

содержании аллергенов.

Учитывая вышеперечисленное, центр развития перспективных технологий (ЦРПТ) и «Х5 Retail Group» в первой половине 2018 года решили приступить к созданию единого каталога товаров России [10].

В данной работе представлены результаты разработки прототипа системы распознавания и классификации товаров в сфере розничной торговли. Основной функцией данной системы является получение информации с этикеток продуктов. Такое решение может сэкономить трудовые и финансовые ресурсы, не ухудшая качества получаемой информации. Для достижения указанной цели были решены следующие задачи:

- составление обзора существующих систем, методов и подходов к решению поставленной задачи;
- аппаратная и программная реализация установки для вращения товара;
- разработка серверной части, в том числе алгоритмов обработки изображений для получения информации с этикетки товара;
- разработка мобильного приложения для автоматического фотографирования;
- разработка пользовательского интерфейса для управления установкой;
- распознавание текста и поиск штрих-кода на этикетках товара;
- проведение экспериментальных исследований разработанной системы и оценка качества.

## 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ И ОБЗОР СУЩЕСТВУЮЩИХ ПОДХОДОВ. ПОСТАНОВКА ЗАДАЧИ

В разделе 1 описывается общая концепция, а также рассматриваются существующие подходы к построению систем визуального контроля. Анализируются их преимущества и недостатки. Приводится обзор существующих методов построения систем распознавания и идентификации объектов, систем вращения объектов и алгоритмов получения развертки цилиндрических объектов.

### 1.1. Общие положения

Концепция единого каталога заключается в следующем: производитель предоставляет образец товара в контент-лаборатории, которые полностью оцифровывают этот товар и несут ответственность за качественное описание этого товара. Далее данные публикуются в едином каталоге, через который все ритейлеры, а также другие участники, могут эти данные получать [10].

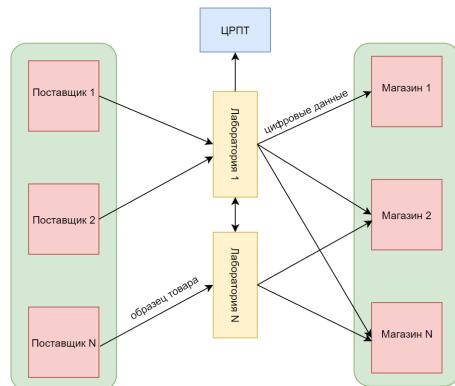


Рис.1.1. Обобщенная структура процесса

На данный момент оцифровкой товара занимаются люди, а так как при выполнении монотонной работы риск человеческой ошибки возрастает, возникает потребность в привлечении дополнительных человеческих ресурсов для выполнения функций контролера. Таким образом, при оцифровке товара задействованы минимум два человека.

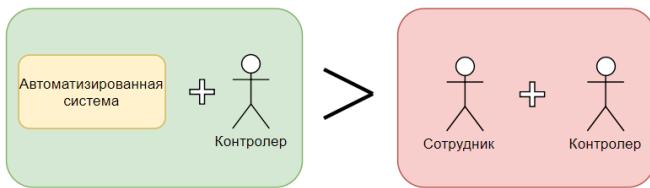


Рис.1.2. Сравнение эффективности

Идея предлагаемого подхода заключается в построении автоматизированной системы, которая бы упростила процесс оцифровки и снизила риск ошибки. Кроме того, использование автоматизированной системы является единовременной тратой, в то время как человеку придется выплачивать зарплату. Это также позволит снизить зависимость от больничных, отпусков и других непредвиденных ситуаций.

## 1.2. Существующие системы визуального контроля

### 1.2.1. Системы визуального контроля фирмы Mettler Toledo

Данная компания предлагает несколько систем, каждая из которых ориентирована на различные формы упаковки продуктов. Так, например, система Round Line V35 [7] подходит для проверки маркировки продуктов цилиндрической формы, обеспечивая полный 360-градусный обзор этикеты и упаковки. В свою очередь, система Flat Pack Line v33 [7] предназначена для сканирования верхней и нижней частей упаковок прямоугольной формы. Также, данная компания предлагает ряд измерительных приборов для точного определения

различных параметров исследуемых объектов.



Рис.1.3. Flat Pack Line v33



Рис.1.4. Round Line V35

Такие системы встраиваются непосредственно в цикл производства и осуществляют проверку качества и правильность нанесенной информации на упаковку продукта, что позволяет избежать отзывов продукции, штрафов или даже судебных исков.

Главным недостатком является отсутствие универсального решения, которое подходило бы под любую форму продукции. Так, например, для того чтобы обеспечить контроль продуктов любой формы необходимо приобрести несколько таких систем.

### 1.2.2. Система линейного сканирования In-Sight 9902L

Cognex Corporation — ведущий мировой поставщик систем машинного зрения, видеодатчиков и промышленных считывателей идентификационных кодов, используемых при автоматизации производства.



Рис.1.5. Образец 9902L

In-Sight 9902L - встроенная промышленная система линейного сканирования, которая обрабатывает изображения непосредственно на камере. Встроенная обработка исключает необходимость установки отдельного контроллера. Кроме того, корпус с классом защиты IP67 защищает систему от жидкостей и пыли [2].

Камеры линейного сканирования являются идеальным выбором для обнаружения крупногабаритных или цилиндрических объектов на быстроходных производственных линиях.

Однако, данное решение также ориентировано на производство и плохо подходит для одновременной работы с объектами разной формы. Для создания универсального решения на базе такой камеры возникает необходимость в модификации существующих алгоритмов, а также в создании специальной вспомогательной системы для вращения объектов.

### 1.2.3. Система OCR Data Traceability

Решения компании Datalogic, основанные на оптическом распознавании символов (OCR) также позволяют проверить маркировку продуктов и обеспечить их контроль на всех этапах производства и распространения. Один из наиболее известных продуктов – U-Camera, которая по словам производителя, обеспечивает сверхвысокую ско-

рость передачи изображений для решения даже самых сложных задач машинного зрения [12].



Рис.1.6. Система OCR Data Traceability

Компания не имеет универсального решения для нашей задачи, однако по информации с сайта производителя, она предлагает помочь в интеграции их оборудования в проекты заказчика.

### **1.3. Современные подходы к построению систем распознавания и идентификации объектов**

Теория распознавания — раздел информатики и смежных дисциплин, развивающий основы и методы классификации и идентификации предметов, явлений, образов, сигналов, ситуаций и т. п. объектов, которые характеризуются конечным набором некоторых свойств и признаков [8].

В контексте выполнения работы необходимым является определение некоторых свойств и признаков объекта, в частности – его форму. Сведения о форме исследуемого объекта позволяют нам лучше оцифровать объект. Так, например, для объектов цилиндрической формы целесообразно получить его развертку, когда как для объектов прямоугольной формы необходимо и достаточно извлечь сведения с четырех его граней.

Существует несколько различных подходов к решению задач распознавания формы объекта, каждый из которых имеет свои преимущества и недостатки. Наиболее распространеными являются алгоритмы на основе:

- нейронных сетей;
- обработки 2D изображений;
- обработки 3D моделей.

### 1.3.1. Классификация объектов на основе нейронных сетей

Одной из наиболее важных областей применения нейронных сетей является анализ изображений [20]. Для решения поставленной задачи лучше всего подходят сверточные нейронные сети. Такие сети применяются для оптического распознавания образов, классификации изображений, детектирования предметов [9].

Обычно задачи машинного обучения разделяют на четыре вида [24]:

- классификация (Object Recognition);
- семантическая сегментация (Class Segmentation);
- обнаружение объектов (Object Detection);
- сегментация объектов (Object Segmentation).

Распознавание объектов говорит нам, что находится на изображении, но не его положение. Семантическая сегментация добавляет информацию о принадлежности объектов к различным классам. Однако, если несколько объектов одного класса перекрываются, их пиксели никак не отделяются друг от друга. Задача обнаружения объектов отделяет каждый объект с помощью грубой ограничительной

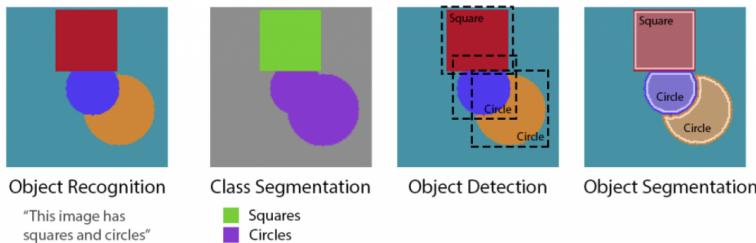


Рис.1.7. Типовые задачи машинного зрения

рамки. В свою очередь, сегментация объектов придает каждой форме четкую границу.

Если рассмотреть эти задачи с точки зрения типов нейронных сетей, то можно сделать следующие заключения: простые CNN сети хорошо подходят для распознавания объектов, но если мы хотим знать где находится находится объект, целесообразно использовать R-CNN(R-region) сети, которые могут определять ограничивающие рамки вокруг объектов. Mask R-CNN сети добавляют сегментацию на уровне пикселей, что позволяет находить точный контур объекта на изображении. Концепции, лежащие в основе в Mask R-CNN прошли поэтапное развитие через архитектуры нескольких промежуточных нейросетей, решавших разные задачи из приведённого выше списка [24].

К недостаткам использования нейро-сетевого подхода можно отнести длительное время обучения и необходимость подготовки огромного количества данных для обучения.

### **1.3.2. Классификация объектов на основе алгоритмов обработки 2D изображений**

Определение формы объекта можно также реализовать с использованием методов обработки изображений. Проанализировав работы

[16] и [25] можно описать общий подход к решению данной задачи. Ключевым моментом является определение контура объекта. Правильное извлечение контура увеличивает шансы правильно классифицировать исследуемый объект. Алгоритмы обнаружения контуров обычно подразделяют на три основных типа:

- следование по пикселям (Pixel following);
- следование по вершинам (Vertex following);
- следование по данным прогона (Run-data-based following).

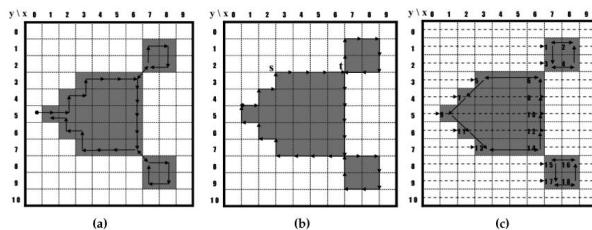


Рис.1.8. Алгоритмы обнаружения контуров

Среди них метод pixel fanning (следование по пикселям) является наиболее общим. Этот метод отслеживает пиксели контура в заранее определенном порядке, а затем сохраняет координаты в соответствии с порядком трассировки. То есть он последовательно ищет соседние черные пиксели текущего пикселя, используя относительный порядок направлений, такой как левый, передний левый, передний, передний правый, справа, сзади-справа и сзади.

Наиболее популярные методы следование пикселей:

- SBF (простой ограничитель границ);
- MNT (метод ближайшего соседа);
- RSA (алгоритм радиальной развертки).

Простой ограничитель границ (SBF) также известен как алгоритм черепахи Паперта или алгоритм квадратичной трассировки. Это самый простой алгоритм трассировки контуров. Первоначально местоположение трассировщика S сохраняется, и трассировщик перемещается в левом или правом направлении. В случае если текущий пиксель является контуром, движение продолжается влево; в противном случае движение продолжается вправо. Как показано на рис. 1.9, такой метод может пропускать отдельно взятые пиксели (а), поэтому существуют модифицированные версии этого алгоритма, принцип которых отражен на рис. 1.9 (б) [16].

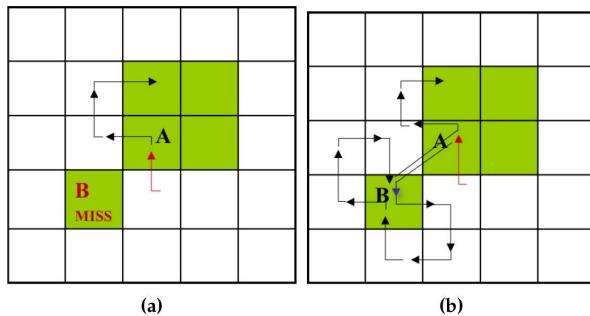


Рис.1.9. Демонстрация работы алгоритма SBF и его модифицированной версии

Метод ближайших соседей (MNT), представленный на рис. 1.10 (а) находит следующий пиксель контура, используя восемь ячеек, соединенных последовательной цепочкой по часовой стрелке, начиная с заднего пикселя трассера. То есть трассер сначала перемещается назад и находит следующий по часовой стрелке контурный пиксель, такой как левый задний, левый, передний, передний правый, правый и задний правый пиксели [16].

Алгоритм радиальной развертки (RSA) [16] похож на MNT, но у его трассировщика нет информации о направлении. Рис. 1.10 (б) иллюстрирует данный пример трассировки. На чертеже сначала ге-

нерируется вектор направления от  $P_i$  до  $P_{i-1}$ , и затем трассировщик ищет следующий пиксель контура, используя предыдущий пиксель  $P_{i-1}$  для направления вектора по часовой стрелке [16].

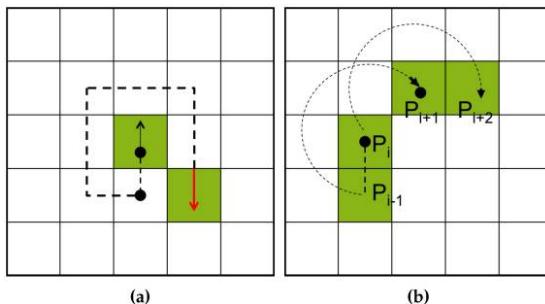


Рис.1.10. Демонстрация работы алгоритмов MNT и RSA соответственно

В библиотеке OpenCV есть встроенный метод, который реализует алгоритм обнаружения контура: `cv2.findContours()` [22]. Этот метод принимает в качестве входных данных три параметра: исходное изображение, режим поиска контура и метод аппроксимации контура.

После обнаружения всех контуров в изображении выбирается максимальный из них, после чего к нему применяется метод контурного приближения. Этот метод уменьшает количество точек на кривой. Обычно для такой цели используется алгоритм Ramer-Douglas-Peucker (алгоритм разделения и слияния). Контурное приближение основано на предположении, что кривая может быть аппроксимирована серией коротких отрезков. Контурное приближение также реализовано в OpenCV в `cv2.approxPolyDP()` методе [22].

Используя информацию о приближенном контуре, мы можем перейти к выполнению обнаружению формы. Важно понимать, что контур состоит из списка вершин. Мы можем использовать эту информацию для определения формы. Так, например, если приближенный контур имеет пять вершин, то это должен быть пятиугольник [19].

### 1.3.3. Классификация объектов на основе 3D моделей

В последнее время было опубликовано довольно много работ [23], [18], [13], направленных на использование трехмерной информации для распознавания формы объектов. Один из наиболее известных алгоритмов - Распознавание по компонентам (Recognition-by-Components), идея которого заключается в том, что большинство объектов могут быть представлены отдельными примитивными компонентами (геномами), которые могут быть получены из основных свойств ребер компонента: кривизна, симметрия, параллелизм и т.д. Эти свойства как правило не зависят от положения объекта и качества изображения, а следовательно, обеспечивают надежное восприятие объекта [13].

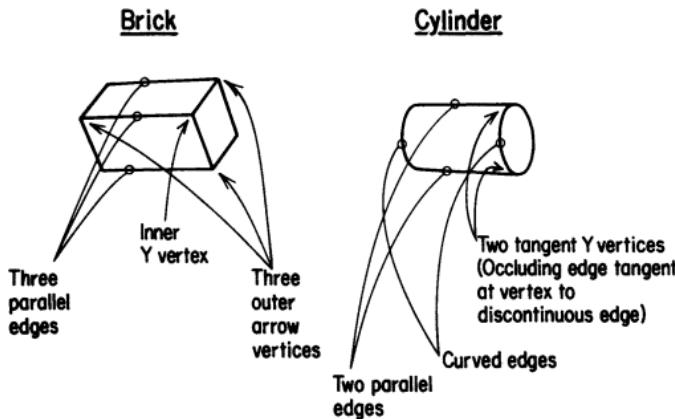


Рис.1.11. Демонстрация различий между двумя компонентами

Данный подход похож на метод, описанный ранее для двумерных изображений, однако обладает большой точностью и меньшей зависимостью от положения камеры. К недостаткам можно отнести трудоемкость реализации и ресурсоемкость алгоритма [13].

Другой подход, описанный в работе [18] основывается на сопоставлении точек, принадлежащих к объекту, с заранее подготовленным набором данных для различных форм.

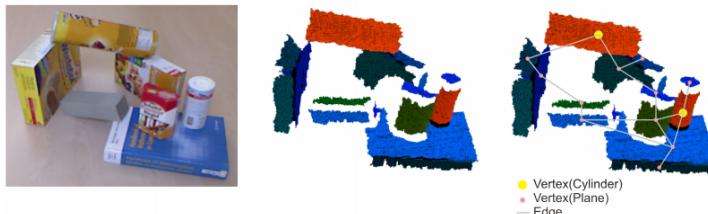


Рис.1.12. Демонстрация работы алгоритма

#### 1.4. Подходы к получению изображений объектов

Аналитическое агентство Smithers Pira представило отчёте «Будущее мировой упаковки до 2020 г. (Future of Global Packaging to 2020), в котором приводится прогноз объёма рынка упаковок на 2020г [6].

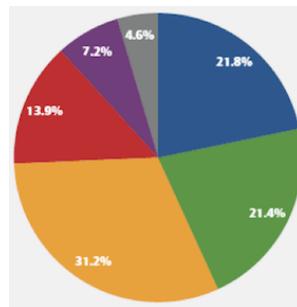


Рис.1.13. Прогноз Smithers Pira: синий – гибкая упаковка; жёлтый – картон; фиолетовый – стекло; зелёный – твёрдый пластик; красный – жесть; серый – другие виды упаковки

Как видно из диаграммы, наибольшую часть рынка занимают картонные, пластиковые, стеклянные и жестяные упаковки. Их об-

щая доля на рынке составляет порядка 74%. На основании статьи [1] и методом экспертных оценок, можно выделить две основные формы упаковок: цилиндрическая и прямоугольная, которые и будут рассмотрены в рамках данной работы.

В контексте решаемой задачи, для получения изображений исследуемых объектов, встает необходимость в получении развертки этикеток объектов цилиндрической формы и считывание информации с нескольких сторон для объектов прямоугольной формы.

Существует два основных способа получения развертки цилиндрических объектов. Первый способ (а) основывается на использовании одной камеры в фиксированном положении для съемки последовательности изображений с определенным углом поворота[15]. Очевидно, что этот метод является недорогим и хорошо подходит для создания статической панорамы.

Данный метод можно модифицировать основываясь на принципе действия линейных камер. Такой подход позволит избежать использования сложных алгоритмов обработки и выравнивания изображений цилиндрических объектов.

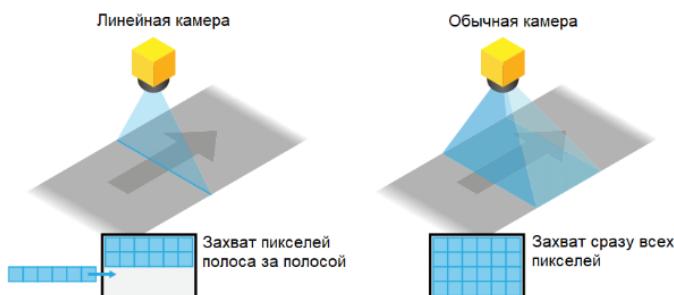


Рис.1.14. Принцип работы линейных и обычных камер

Для того, чтобы построить изображение линейной камерой необходимо движение объекта относительно камеры с небольшим шагом.

По мере прохождения объекта мимо камеры из изображений вырезаются центральные полосы. Программное обеспечение сохраняет в памяти эти полосы, а затем формирует готовое 2D изображение [17].

Другой способ (б) заключается в использовании панорамных устройств сбора, состоящих из нескольких камер [15]. Получение изображений может выполняться одновременно, тем самым повышается скорость работы. Такой способ хорошо подходит для систем реального времени, однако из-за большого числа камер является довольно дорогим.

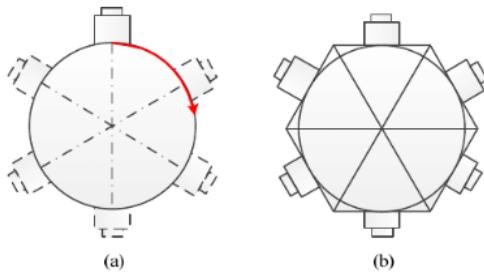


Рис.1.15. Способы получения панорамы цилиндрического объекта

Данные методы также могут быть применимы и к объекту прямоугольной формы. Так, например, для того чтобы получить информацию с нескольких сторон объекта необходимо либо разместить камеры напротив его сторон, либо повернуть его четыре раза так, чтобы каждая из сторон поочередно была повернута к камере.

Для реализации первого подхода возникает необходимость в создании специальной установки, которая была бы способна вращать объект на 360 градусов.

На сегодняшний день в области фото индустрии существует ряд решений, как профессиональных, так и любительских, позволяющих обеспечивать вращение объекта на 360 градусов. Далее будут рассмотрены некоторые из них.

#### 1.4.1. Обзор системы PhotoPizza



Рис.1.16. Образец PhotoPizza

PhotoPizza это "Open source" проект на базе микроконтроллера Arduino. Данная платформа способна вращать объекты массой до 200кг, а также в режиме реального времени отправлять снятые фотографии на сервер. Данное устройство можно собрать самостоятельно, следуя инструкциям [21] или приобрести у официального дилера. Стоимость такой установки 19000-39000 рублей (в зависимости от размеров). К преимуществам можно отнести прочность системы, возможность вращения крупных и тяжелых объектов, а также наличие активного сообщества и огромного числа инструкций по сборке. Стеклянный корпус позволит поместить дополнительную камеру снизу, что позволяет исключить мертвые зоны при сканировании объекта. Однако, данная система не предполагает работу с маленькими объектами и имеет фиксированное положение камеры.

#### 1.4.2. Обзор системы Ciclop

Данный проект был разработан в соответствии с философией сообщества RepRap: «устройства которые могут быть сделаны с помо-

щью 3D принтера» [14]. По этой причине большинство элементов данной системы можно распечатать на 3D принтере, а остальные детали приобрести в ближайших магазинах.



Рис.1.17. Образец Ciclop

Система обладает лицензией Creative Commons Attribution-ShareAlike 4.0 International License [11], что позволяет свободно распространять, изменять, а также использовать в коммерческих целях. К недостаткам данной системы можно отнести нерегулируемость положения камеры, а также привязанность системы к аппаратным решениям. Однако, данные недостатки можно легко нивелировать, модифицировав существующие модели, а также разработав соответствующую часть, отвечающую за движение камеры.

Системы Ciclop и PhotoPizza похожи с точки зрения функциональных возможностей, основные различия наблюдаются конструкторских решениях. Так как в данной работе разрабатывается опытный образец, целесообразно использовать наработки проекта Ciclop, так как исходные модели представлены в редактируемом формате

для 3D моделирования, а значит легко модифицируемы. Также, компоненты системы Ciclop обладают относительно низкой стоимостью.

### 1.5. Постановка задачи

Существующие на сегодняшний день системы по визуальному контролю продукции ориентированы преимущественно на производственный цикл, а следовательно, для нашей задачи имеют ряд недостатков, а именно:

- отсутствие универсального решения, которое подходило бы для объектов любой формы;
- высокая стоимость (для разных объектов - разные устройства);
- избыточность (нет необходимости в покупке многофункциональных устройств для решения частных задач);
- нет готового цельного решения.

С учетом отмеченных недостатков систем визуального контроля продукции следует сделать вывод о наличии потребности в разработке прототипа собственной системы, которая должна обладать следующими характеристиками:

- возможность оцифровки объектов как цилиндрической так и прямоугольной форм;
- низкая стоимость по сравнению с дорогостоящей аппаратурой систем визуального контроля;
- простота в использовании и модификации системы.

Таким образом, целью данной работы является разработка прототипа системы для распознавания и идентификации продуктов питания.

Для достижения указанной цели необходимо решить следующие задачи:

- аппаратная и конструктивная реализация системы вращения объекта;
- разработка структурных, схемотехнических и конструктивных решений;
- разработка программного обеспечения;
- проведение испытаний системы.

## **2. РАЗРАБОТКА СТРУКТУРНЫХ, СХЕМОТЕХНИЧЕСКИХ И КОНСТРУКТИВНЫХ РЕШЕНИЙ**

В разделе 2 предложена структурно-функциональная схема, приводится реализация аппаратного обеспечения и конструктивные решения для автоматизированной системы распознавания и идентификации продуктов питания.

### **2.1. Структурно-функциональная схема**

Изучив существующие решения становится понятно: для создания универсальной системы, которая была бы способна обрабатывать объекты любой формы необходимо разработать соответствующие решения на всех уровнях системы, от аппаратной части до пользовательского интерфейса. Обобщенная структура такой системы представлена на рис. 2.1.

Данная система должна работать в двух режимах: автоматизированном и ручном. В автоматизированном режиме участие оператора сводится к минимуму. Оператору необходимо поставить объект исследования на установку и дождаться окончания процесса оцифровки. В

ручном режиме, оператор может самостоятельно задавать команды управления через пользовательский интерфейс.



Рис.2.1. Общая структура системы

Для реализации системы было предложено разработать следующие элементы:

- серверная часть;
- управляющее устройство;
- подсистема управления шаговым двигателем;
- подсистема управления камерами;
- подсистема индикации.

На серверную часть возложены функции управления всей системой, обработки и хранения данных, представление результатов работы, а также взаимодействие с оператором.

Управляющее устройство в соответствии с поступающими командами от сервера осуществляет сбор данных и управление различными подсистемами.

Подсистемы управления шаговым двигателем и управления камерами осуществляют фотосъемку объекта.

Подсистема индикации предназначается для информирования оператора о текущем статусе работы системы.

Структурно-функциональная схема системы распознавания и идентификации продуктов питания изображена на рис. 2.2.



Рис.2.2. Структурно-функциональная схема системы распознавания и идентификации продуктов питания

## 2.2. Разработка аппаратного обеспечения

Разработка аппаратного обеспечения для автоматизированной системы распознавания и идентификации продуктов питания состоит из следующих частей:

- управляющее устройство;
- разработка подсистемы управления приводным двигателем;
- разработка подсистемы индикации.

### 2.2.1. Управляющее устройство

Выбор управляющего устройства является сложной задачей, поскольку рынок встраиваемых систем чрезвычайно широк: от простых микроконтроллеров до полноценных одноплатных микрокомпьютеров.

Микроконтроллеры хорошо подходят для управления различными электронными устройствами, обладают низкой стоимостью и довольно маленькими размерами. Основные периферийные устройства, ядро процессора и память представлены на одном кристалле, таким образом микроконтроллеры удобно использовать во встраиваемых системах.

В свою очередь, одноплатные микрокомпьютеры исполняют программы в рамках операционной системы и обладают большей производительностью. В отличии от микроконтроллера, компоненты микрокомпьютера не находятся на одном кристалле, что порой затрудняет их встраиваемость в другие системы. Широко распространены в различных бюджетных сегментах рынка, так как позволяют значительно сократить временные и финансовые затраты.

К основным преимуществам использования одноплатного микрокомпьютера можно отнести:

- снижение затрат на разработку и доработку ПО;
- создание универсального решения;
- вычислительная мощность;
- ценовая составляющая;
- удобство эксплуатации.

С учетом вышеперечисленного, автором было принято решение использовать одноплатный микрокомпьютер Raspberry Pi 3 B+ (рис. 2.3), который обладает следующими основными характеристиками:

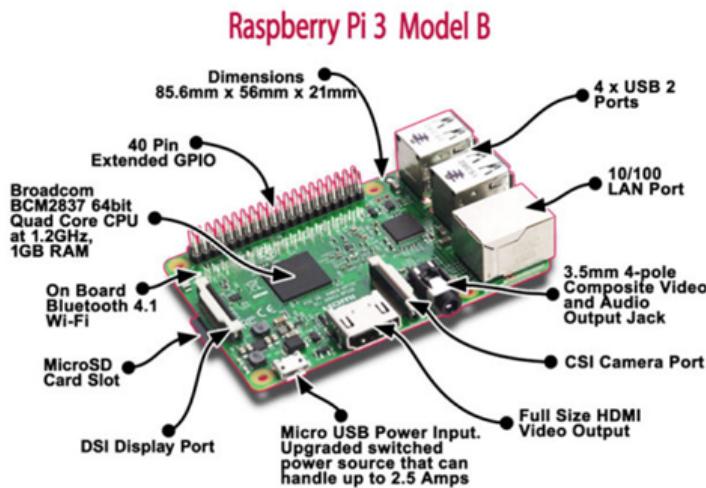


Рис.2.3. Образец Raspberry Pi

- 4-ядерный процессор, работающий на частоте 1.2GHz;
- встроенные WiFi и Bluetooth с низким энергопотреблением;
- 1 гигабайт встроенной оперативной памяти;

- 4 порта USB 2.0 и LAN порт для выхода устройства в интернет;
- полноразмерный HDMI разъем, а также аналоговый TV-выход;
- порт для подключения micro-SD для установки операционной системы;
- специальный коннектор CSI для подключения камеры;
- 40 GPIO;
- Стоимость: 35\$.

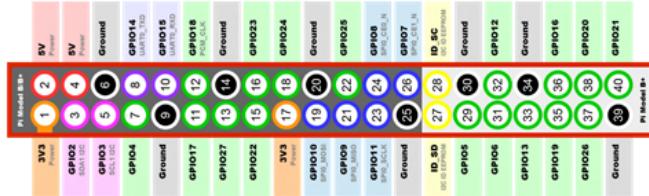


Рис.2.4. Порты ввода-вывода Raspberry Pi

Выбранный микрокомпьютер позволяет решить сразу несколько основных задач:

- управление шаговым двигателем для вращения установки;
- управление камерой;
- предварительная обработка данных;
- хранение данных;
- соединение по беспроводному каналу для связи с сервером;
- низкая стоимость, энергопотребление.

### 2.2.2. Разработка подсистемы управления приводным двигателем

Приводные устройства и приводные системы традиционно рассматриваются как один из важнейших базовых элементов технологических систем различного назначения, применяемых в большинстве отраслей промышленности. Во многом именно надежность приводов определяет степень производственной эффективности и уровень общей безопасности промышленной техники.

Одними из наиболее характерных представителей семейства современных приводных устройств являются сервоприводы и шаговые двигатели.

К преимуществам шаговых двигателей можно отнести:

- низкая стоимость;
- простота конструкции;
- управление через контроллер;
- более стабильная, чем у сервоприводов, работа в режиме удержания, обеспечивающаяся дискретностью ШД.

Однако у шаговых двигателей есть ряд недостатков: для крупных станков использовать шаговые двигатели не рекомендуется, так как они могут довольно сильно резонировать, что чревато пропуском шагов и падением точности; в сравнение с сервоприводом шаговый двигатель отличается довольно высоким уровнем шума.

Основными преимуществами сервопривода являются:

- высокий конечный КПД;
- практически не подвержены резонансу.

К минусам серводвигателей можно отнести их очень высокую стоимость, объясняющаяся конструктивной сложностью механизмов такого типа. Также следует отметить, что тонкие настройки сервопривода довольно сложны [3].

Таким образом, для создания прототипа системы ввиду низкой стоимости, целесообразно использовать шаговый двигатель. Также, следует отметить, что недостатки шагового двигателя не должны скаться на работе системы в целом.

Для управления шаговым двигателем был выбран аппаратный драйвер - A4988. Это довольно распространённый и недорогой драйвер для биполярного шагового двигателя, который имеет встроенную защиту от перегрузки и перегрева, возможность регулировать ток, а также изменять шаг.



Рис.2.5. Образец A4988

Использование радиатора позволяет устройству выдерживать ток до 2 ампер, в противном случае рекомендуется использовать источник питания с 1 ампером.

Драйвер чувствителен к скачкам напряжения по питанию двигателя, поэтому производитель рекомендует устанавливать электролитический конденсатор большой емкости по питанию VMOT для сглаживания скачков.

Принципиальная схема подключения шагового двигателя к Raspberry Pi представлена ниже:

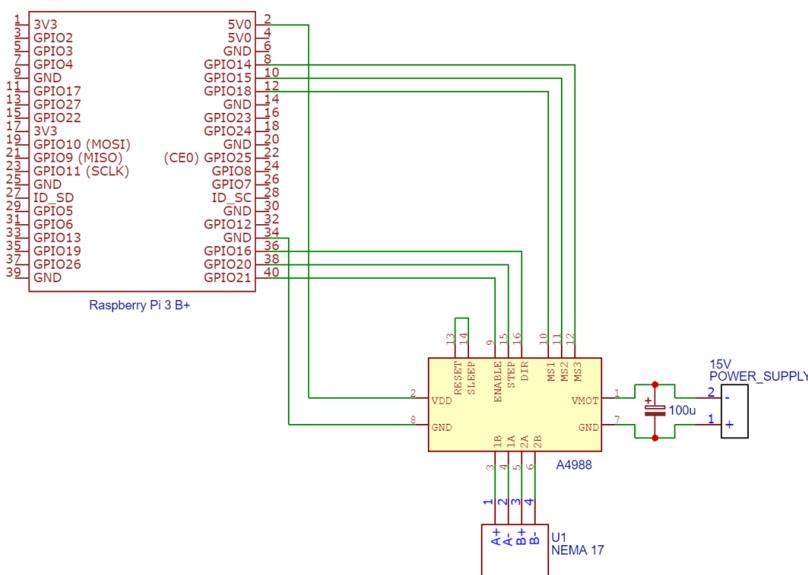


Рис.2.6. Принципиальная схема подключения шагового двигателя к Raspberry Pi

Таблица 2.1  
Описание контактов

Название контакта	Пояснение
VMOT	Напряжение питания двигателя (8В – 35В)
GND	Земля
2B, 2A	Соединение со второй катушкой биполярного двигателя
1A, 1B	Соединение с первой катушкой биполярного двигателя VDD Напряжение питания микросхемы (3В – 5В)
ENABLE	Активный уровень – 0. Включение/выключение драйвера
MS1, MS2, MS3	Задание шага (от полного шага до 1/16)
RESET	Активный уровень – 0. Сброс драйвера
STEP	Генерация импульсов для движения двигателя (один импульс – одному шагу)
DIR	Установка направления вращения. Высокий уровень – движение по часовой, низкий – против

### 2.2.3. Разработка подсистемы индикации

Для контроля состояния устройства используется подсистема индикации, основанная на RGB диодах. Визуальная диагностика способствует быстрому обнаружению неполадок в аппаратном и программном обеспечении. В таблице 2.2. перечислены возможные состояния устройства:

Таблица 2.2

Индикация состояния устройства

Состояние устройства	Цвет свечения светодиода
«Устройство готово к запуску»	Горит зеленый
«Устройство работает»	Горит желтый
«Устройство отработало без ошибок»	Мигает зеленый
«Сбой»	Мигает красный

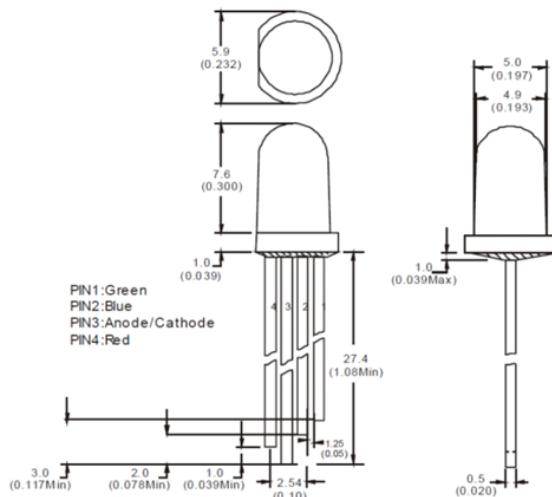


Рис.2.7. Образец RGB-диода

### 2.3. Общая структура системы

Таким образом, для обеспечения функциональных возможностей, указанных ранее, а также учитывая выбор аппаратных компонентов, была составлена общая структура системы для распознавания и идентификации продуктов питания (рис. 2.8).

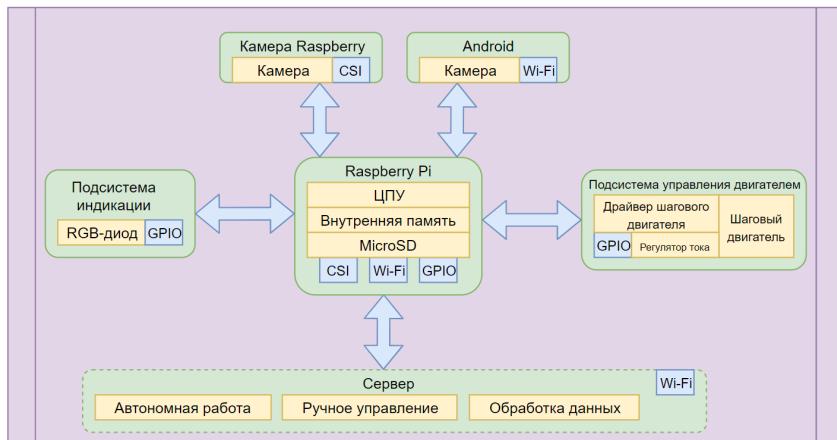


Рис.2.8. Общая структура системы

### 2.4. Разработка конструктивных решений

Учитывая тот факт, что разрабатываемая установка является опытным образцом, целесообразно использовать технологию 3D моделирования. Создание объёмной модели еще на стадии проектирования позволяет определить различного рода ошибки, например, нестыковки деталей или несоответствие размеров, провести проверку на собираемость изделий, а также оценить степень соответствия модели исходному замыслу.

Разработку установки необходимо вести с учетом размеров и требований к аппаратному обеспечению. Конструкция установки должна

быть устойчивой и обеспечивать функциональные возможности, заявленные во второй главе.

В качестве среды проектирования выбрана среда SolidWorks, которая имеет достаточно удобный и интуитивно понятный интерфейс, а также позволяет достаточно быстро проектировать как простые, так и более сложные объекты.

Разрабатываемую систему можно разделить на две основные части:

- система вращения объекта;
- система регулирования положения камеры.

Система вращения объекта — это модифицированная версия системы He3D Ciclop, которое состоит из четырех основных частей:

- база;
- диск;
- крепление для диска;
- переходник между диском и шаговым двигателем.

**База устройства.** Исходная модель базы из проекта He3D Ciclop необходимо адаптировать под размеры шагового двигателя и подшипника, выбранного автором. В данной работе используется шаговый двигатель - Nema17HS4401 и подшипник 8338-75 (ГОСТ).

Таблица 2.3

Значения размеров шагового двигателя в зависимости от модели

Model	Length
17HS2XXX	28 mm
17HS3XXX	34 mm
16HS4XXX	40 mm
16HS8XXX	48 mm

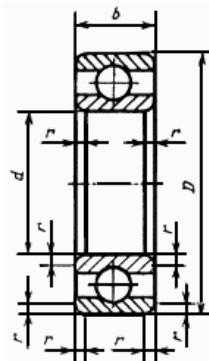


Рис.2.9. Чертеж подшипника.

Обозначения на чертеже, в соответствии с нашей моделью (7000114):

$d=70\text{мм}$  - номинальный диаметр отверстия внутреннего кольца;

$D=110\text{мм}$  - номинальный диаметр наружной цилиндрической поверхности наружного кольца;

$B=13\text{мм}$  - номинальная ширина подшипника;

$r=1\text{мм}$  - номинальная координата монтажной фаски.

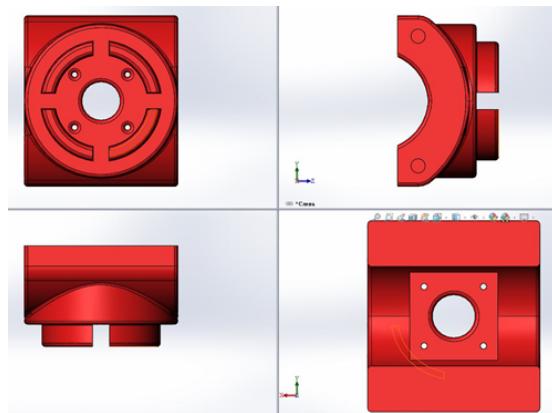


Рис.2.10. 3D модель базы устройства

Важную роль играет точность **крепления диска**. Необходимо убедиться, чтобы детали плотно прилегали друг другу, чтобы избежать нежелательных зазоров, которые могут привести к дополнительным колебаниям во время работы системы. Таким образом, согласно размерам нашего подшипника, была смоделирована точная его копия, что позволило убедиться в корректности выбранных размеров и возможность сстыковки.

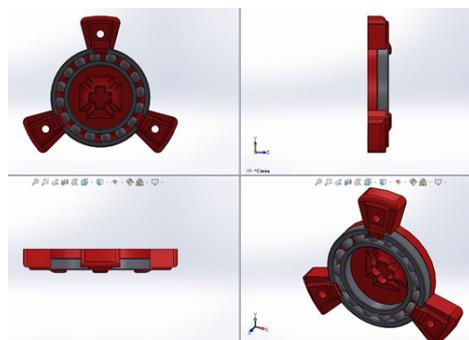


Рис.2.11. 3D модель крепления для диска

Для того чтобы обеспечить соединение шагового двигателя с диском, в проекте He3D Ciclop предлагается использование **переходника**. В данном случае используется оригинальная модель (рис. 2.12).

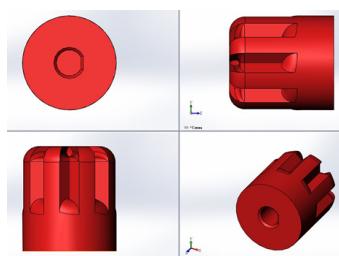


Рис.2.12. 3D модель переходника

**Система регулирования положения камеры** – это решение, основанное на принципе работы станков ЧПУ.

Устройство состоит из:

- шаговый двигатель (1);
- зубчатый ремень (2);
- линейная рельсовая направляющая (3);
- подшипник линейный (4);
- концевой выключатель (5).



Рис.2.13. Образец системы для регулирования положения камеры

### 3. РАЗРАБОТКА ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ

В соответствии с описанными во второй главе функциональными возможностями устройства, были разработаны следующие компоненты программного обеспечения для системы распознавания и идентификации продуктов питания:

- управление системой;
- сбор данных;
- определение формы объекта;
- получение развертки цилиндрического объекта;
- распознавание текста и поиск ключевых слов.

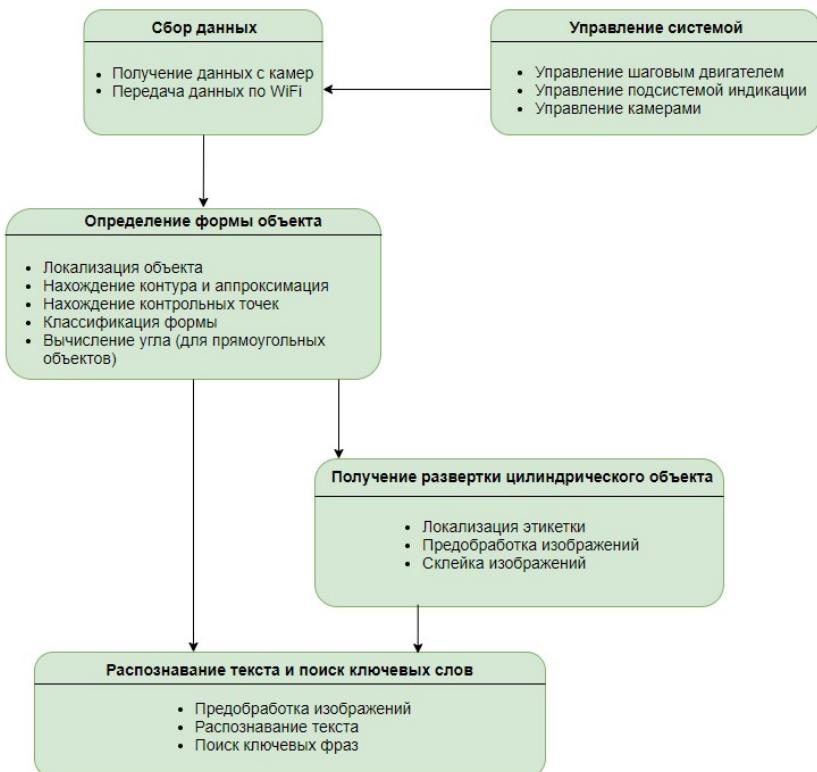


Рис.3.1. Диаграмма работы системы

### 3.1. Управление системой

Для демонстрации возможностей данного устройства было разработано два режима работы: автоматизированный и ручной. В автоматизированном режиме участие оператора сводится к минимуму. Оператору необходимо поставить объект исследования на установку и дождаться окончания процесса оцифровки. В ручном режиме, оператор может самостоятельно задавать команды управления через пользовательский интерфейс (рисунок 3.2).

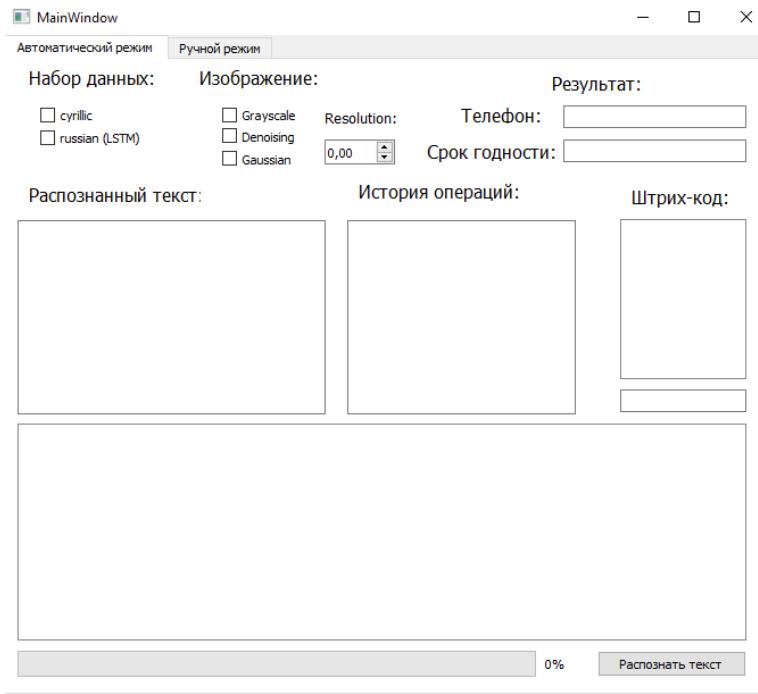


Рис.3.2. Пример главного окна приложения

Данное программное обеспечение предназначено для распознавания текста с этикеток продуктов. Предполагается использование сов-

местно с физической установкой.

Схема программы для управления системой представлена на рисунке 3.3.

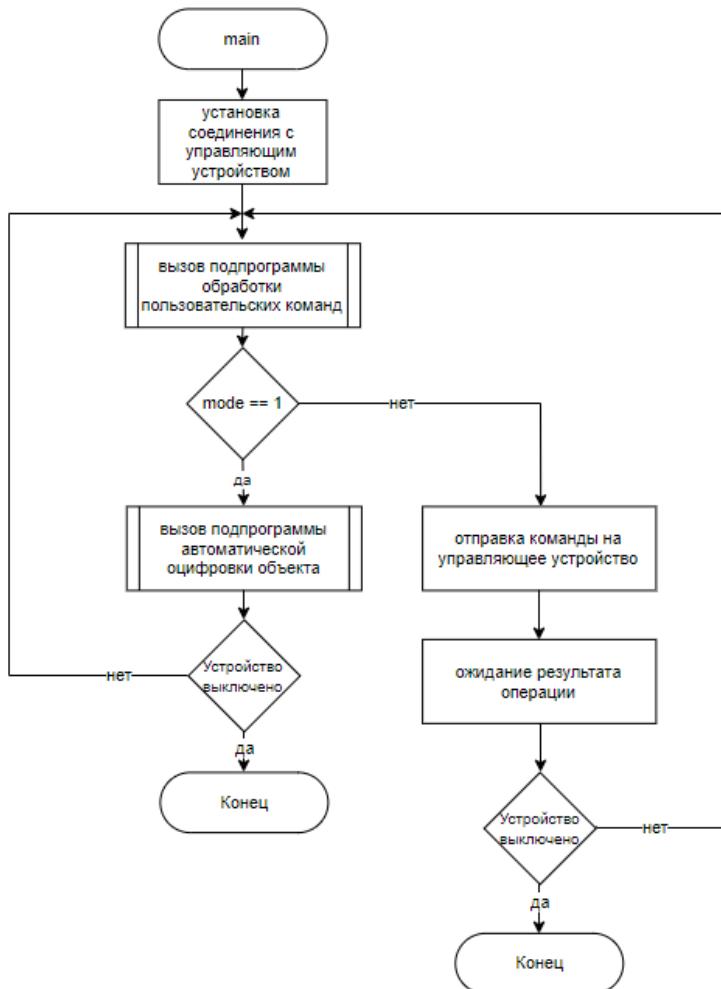


Рис.3.3. Схема программы для управления системой

### 3.2. Сбор данных

За управление камерами и сбор данных отвечает управляющее устройство - одноплатный микроконтроллер Raspberry Pi. В рамках данной работы под данными подразумеваются изображения (фотографии) исследуемого объекта, которые собираются с:

- камеры Raspberry;
  - камеры Android телефона.

Принцип работы программ для автоматического фотографирования в обоих случаях схож и может быть представлен следующим образом:

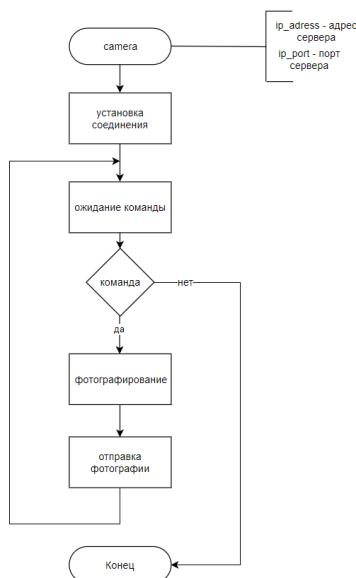


Рис.3.4. Схема программы автоматического фотографирования

### 3.3. Определение формы объекта

Предложенный автором подход, для решения поставленной задачи, состоит из нескольких частей:

- удаление шумов (тень объекта);
- локализация объекта;
- морфологические преобразования;
- нахождение контура;
- определение формы по контуру;
- определение положения объекта.

#### 3.3.1. Удаление шумов

Под шумами в рамках данной задачи, подразумевается тень от объекта. Если не выполнить предварительную обработку изображений (избавление от теней), то это может привести к тому, что выделенный объект будет включать в себя фрагмент тени, т.е. являться частью объекта. Для избавления от нежелательных теней была написана функция `_remove_shadows(image)`. Данная функция использует библиотечные функции OpenCV, такие как `cv2.medianBlur`, `cv2.normalize`. На рисунке 3.5 представлен пример работы медианного фильтра.

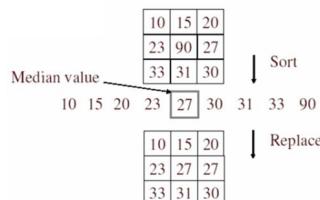


Рис.3.5. Пример работы медианного фильтра

### 3.3.2. Локализация объекта

Вычитание фона (Background Subtraction) является распространенным и широко используемым методом для выделение объекта на статичном фоне. Как понятно из названия метода, для выделения объекта (его маски), требуется выполнить вычитание изображения с объектом из изображения с фоном. Схема алгоритма представлена ниже.

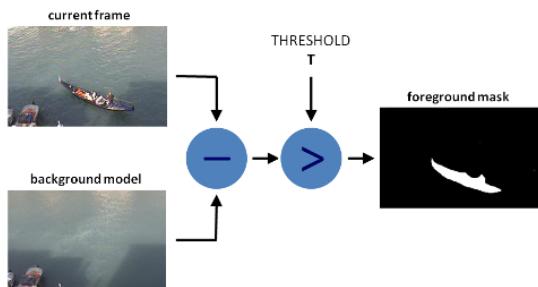


Рис.3.6. Алгоритм локализации объекта (его маски)

Данный алгоритм представлен в разработанном автором классе ShapeDetector, в методе `_get_diff(self, white=True, threshold)`.

Вычитание изображений выполняется с помощью библиотечной функции `cv2.absdiff()`. Далее идет сравнение пикселей с пороговым значением и определяется их принадлежность объекту (порог задается параметром `threshold`). Если аргумент функции `white=True` то маска объекта будет содержать только белые пиксели, иначе маска будет представлена в оттенках серого.

### 3.3.3. Морфологическое преобразование

К основным морфологическим преобразованиям относятся алгоритм размывания (операция сужения) и алгоритм растягивания (операция расширения). В данном случае, так как исследуемые обьек-

ты имеют различные текстуры, различную степень прозрачности и наличие трудно убираемых шумов, зачастую мы получаем искаженную маску объекта. Для того чтобы избавиться от этих негативных эффектов, автором работы был использован метод растягивания, который способствует объединению светлых областей изображения. Морфологические операции, чаще всего, применяются над двоичными изображениями, которые получаются после порогового преобразования (thresholding). Данный алгоритм представлен в методе `_dilate(self)` класса ShapeDetection и основан на функции из библиотеки OpenCv - `cv2.dilate(image, kernel, iterations)`. Настройка алгоритма осуществляется следующими параметрами: `kernel` - размер окна (ядро) и `iterations` - количество итераций.

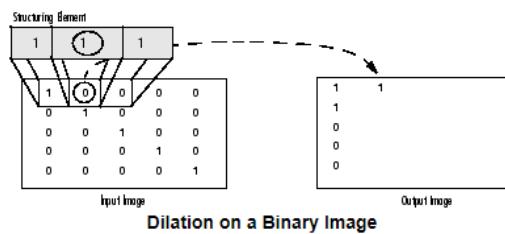


Рис.3.7. Пример морфологического преобразования



Рис.3.8. Результат морфологического преобразования

### 3.3.4. Нахождение контура

Для нахождения контура объекта реализован метод `_draw_contour(self, contour_type="largest")`. Алгоритм реализован следующим образом: на первом шаге с помощью функции `cv2.findContours()` находятся все возможные контуры, далее с помощью функции `cv2.contourArea()` вычисляется площадь каждого контура. Среди всех площадей выбирается наибольший и этот контур аппроксимируется функцией `cv2.approxPolyDP()`.

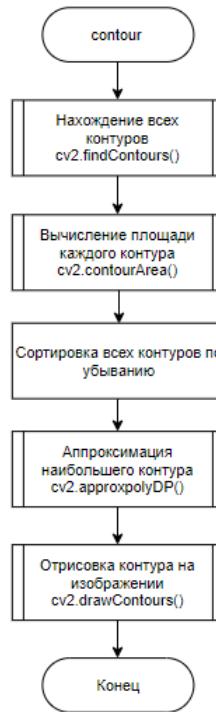


Рис.3.9. Алгоритм нахождения контура объекта

Принцип работы функции `cv2.findContours`(изображение, режим группировки, метод упаковки) подробно рассмотрен в первой главе.

Данная функция имеет следующие аргументы:

- режим группировки — один из четырех режимов группировки найденных контуров:
  - CV\_RETR\_LIST — выдаёт все контуры без группировки;
  - CV\_RETR\_EXTERNAL — выдаёт только крайние внешние контуры. Например, если в кадре будет пончик, то функция вернет его внешнюю границу без дырк;
  - CV\_RETR\_CCOMP — группирует контуры в двухуровневую иерархию. На верхнем уровне — внешние контуры объекта. На втором уровне — контуры отверстий, если таковые имеются. Все остальные контуры попадают на верхний уровень;
  - CV\_RETR\_TREE — группирует контуры в многоуровневую иерархию.
- метод упаковки — один из трёх методов упаковки контуров:
  - CV\_CHAIN\_APPROX\_NONE — упаковка отсутствует и все контуры хранятся в виде отрезков, состоящих из двух пикселей.
  - CV\_CHAIN\_APPROX\_SIMPLE — склеивает все горизонтальные, вертикальные и диагональные контуры.
  - CV\_CHAIN\_APPROX\_TC89\_L1 — применяет к контурам метод упаковки (аппроксимации) Teh-Chin.

Функция cv2.approxPolyDP() приближает форму контура к другой форме с меньшим количеством вершин в зависимости от заданной нами точности. Функция реализована на алгоритме Дугласа-Пекера.

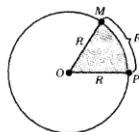
### 3.3.5. Определение формы по контуру

Алгоритм определения формы реализован в методе `_detect_shape(self)` класса `ShapeDetector` и основан на данных о количестве вершин аппроксимированного контура. Так, если количество вершин равно 4, считается что это прямоугольник, иначе - цилиндр.

### 3.3.6. Определение положения объекта

Если начальный радиус совершил полный оборот, то получится угол равный 360 градусам или 2 радианам. Из этого следует, что

$$\text{Радианская мера } 1^\circ \text{ равна } \frac{2\pi}{360} = \frac{\pi}{180} \text{ т.е } 1^\circ = \frac{\pi}{180} \text{ рад.}$$



Найдем градусную меру угла в 1 радиан. Так как дуга длиной  $\pi R$  (полуокружность) стягивает центральный угол в 180 градусов, то дуга длиной  $R$  стягивает угол в  $\pi$  раз меньший, т.е

$$1 \text{ рад} = \left( \frac{180}{\pi} \right)^\circ$$

Вычисление положения объекта (угла наклона) для объектов прямоугольной формы, происходит относительно горизонта.

$$\theta = \arccos \left( \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\| \|\vec{v}\|} \right) \cdot \frac{180}{\pi}$$

Алгоритм определения положения объекта представлен ниже.

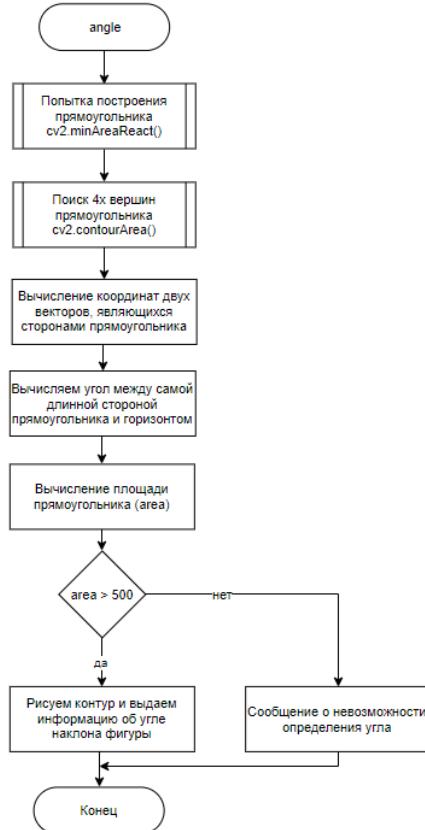


Рис.3.10. Алгоритм определения положения объекта

### 3.4. Алгоритм получения развертки цилиндрического объекта

Алгоритм получения развертки цилиндрического объекта, предложенный автором, можно разделить на следующие части:

- получение фотографий, сделанных с определенным (небольшим) шагом;

- вырезание линейной области из фотографий;
- нахождение особых точек;
- аффинное преобразование;
- склейка изображений

На сегодняшний день существует несколько хорошо зарекомендовавших себя на практике алгоритмов для нахождения особых точек: SIFT, SURF, FAST, HOG. Автором данной работы был выбран алгоритм SURF, основная задача этого алгоритма заключается в обнаружении локальных характерных особенностей изображения. В контексте нашей задачи основными достоинствами данного метода являются устойчивость к масштабированию, вращению и относительная устойчивость по отношению к аффинным преобразованиям. Алгоритм начинает свою работу с нахождения в изображении локальных экстремумов, на основе чего выделяются особые точки.

Особые точки определяются с помощью матрицы Гессе. Гессиан является симметрической квадратичной формой, описывающей поведение функции во втором порядке. Матрица этой функции образована её вторыми частными производными.

$$H(x, y, \sigma) = \begin{bmatrix} L_{xx}(x, y, \sigma) & L_{xy}(x, y, \sigma) \\ L_{xy}(x, y, \sigma) & L_{yy}(x, y, \sigma) \end{bmatrix},$$

Градиент в точке вычисляется с помощью фильтров Хаара



где черные области имеют значение -1, а белые +1. Это даст значение перепада градиента по осям.

Имея набор соответствующих друг другу особых точек на двух изображениях, нужно найти преобразование, которое переводило бы особые точки следующего кадра в точки предыдущего. Для этих целей принято использовать аффинное преобразование. В общем виде желаемое преобразование можно записывается следующим образом [5]:

$$H = [RS|t] = \begin{bmatrix} \cos(\theta)s & -\sin(\theta)s & tx \\ \sin(\theta)s & \cos(\theta)s & ty \end{bmatrix}.$$

Матрица имеет размерность 2x3 в силу того, что в компьютерном зрении точки изображения принято рассматривать в однородных координатах с последующей нормализацией. Обозначив

$$a = \cos(\theta)s, b = \sin(\theta)s, c = tx, d = ty, H = \begin{bmatrix} a & -b & c \\ b & a & d \end{bmatrix},$$

можно записать систему, которой должны удовлетворять одни и те же точки на двух кадрах:

$$\begin{bmatrix} a & -b & c \\ b & a & d \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \text{ или } \begin{aligned} ax - by + c &= x' \\ bx + ay + d &= y' \end{aligned}$$

Для того чтобы найти параметры a, b, c, d, необходимо иметь всего лишь две пары точек:

$$\begin{bmatrix} x_1 & -y_1 & 1 & 0 \\ y_1 & -x_1 & 0 & 1 \\ x_2 & -y_2 & 1 & 0 \\ y_2 & x_2 & 0 & 1 \end{bmatrix} \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \begin{bmatrix} x_1' \\ y_1' \\ x_2' \\ y_2' \end{bmatrix}.$$

Алгоритм получения развертки цилиндрического объекта представлен ниже.

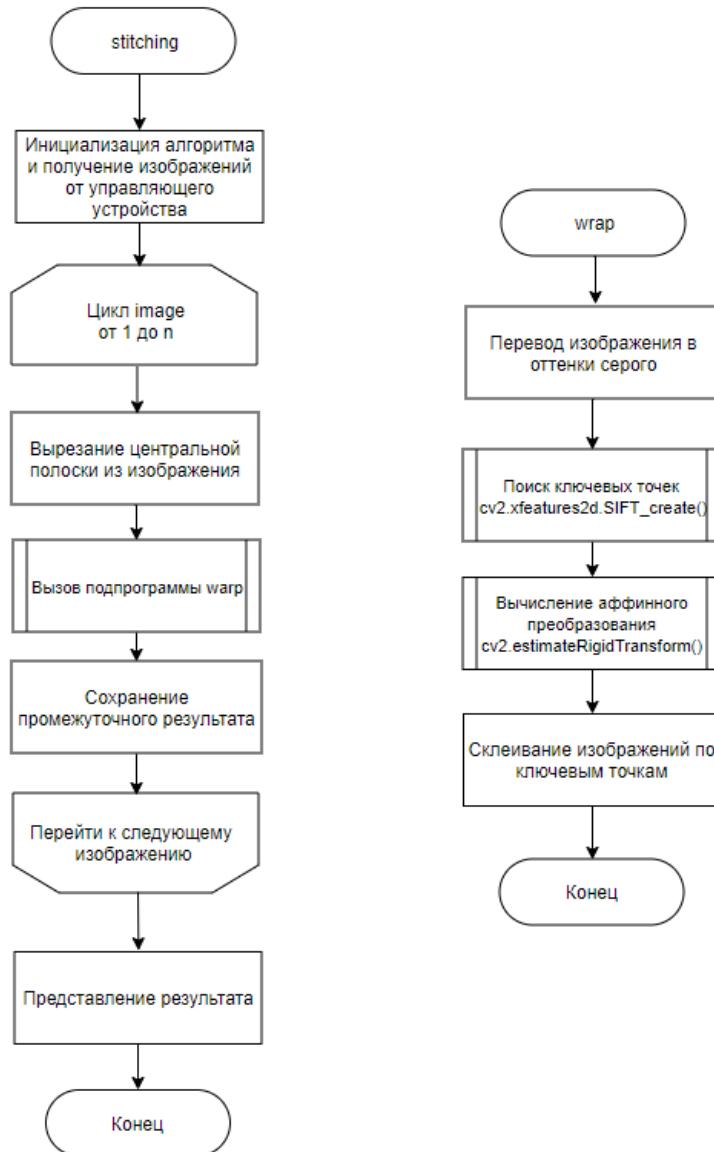


Рис.3.11. Алгоритм получения развертки изображения

### 3.5. Распознавание текста

Распознавание текста на этикетке товара можно проводить с помощью различных готовых библиотек. Одна из таких библиотек - бесплатная библиотека tesseract. Tesseract — свободная компьютерная программа для распознавания текстов, разрабатывавшаяся Hewlett-Packard с середины 1980-х по середину 1990-х, а затем 10 лет "пролежавшая на полке". В августе 2006 г. Google купил её и открыл исходные тексты под лицензией Apache 2.0 для продолжения разработки. В настоящий момент 10 версии программы уже работает с UTF-8, поддержка языков (включая русский с версии 3.0) осуществляется с помощью дополнительных модулей.

Пример использования этой библиотеки можно найти по адресу:  
<https://github.com/vtempest/tesseract-ocr-sample>

Исходный набор тестовых данных для тренировки библиотеки показывал довольно плохой результат, скорее всего это было связано с тем, что в качестве основного языка для распознавания использовался русский. Поэтому к данной библиотеке были добавлены два тестовых набора (взятых с GitHub): русские буквы и кириллица соответственно. Их можно использовать как вместе, так и по отдельности. Опытным путем было установлено, что их одновременное использование дает наилучший результат.

Опытным путем было установлено, что на качество распознавания наиболее сильно влияют:

- перевод изображения в оттенки серого;
- сегментация изображения с помощью порога Гаусса;
- избавление от шумов;
- увеличение размера изображения (dpi).

## 4. ЭКСПЕРИМЕНТАЛЬНЫЕ ИССЛЕДОВАНИЯ

### 4.1. Метрики оценки качества

#### 4.1.1. Классификация формы

Так как классы в тестовых выборках для наших задач локализации и классификации разбалансированы, такой широко используемый показатель качества как accuracy (доля объектов выборки, которым класс был присвоен верно) не подходит. В подобных случаях уместно использовать метрики, именуемые точностью (precision) и полнотой (recall).

Точность (precision) для класса – доля корректно классифицированных объектов среди всех объектов, отнесённых классификатором к данному классу. Более формально:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives} + \text{false positives}}$$

Полнота (recall) для класса – доля корректно классифицированных объектов среди всех объектов, принадлежащих данному классу:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives} + \text{false negatives}}, \text{ где}$$

true positives – количество объектов, верно отнесённых к данному классу

false positives – количество объектов, ошибочно отнесённых к данному классу

false negatives – количество объектов, ошибочно не отнесённых к данному классу

Понятно что чем выше точность и полнота, тем лучше. Но в реальной жизни максимальная точность и полнота не достижимы одновременно и приходится искать некий баланс. Поэтому, хотелось бы иметь некую метрику которая объединяла бы в себе информацию о точности и полноте нашего алгоритма. Именно такой метрикой является F-мера.

F-мера представляет собой гармоническое среднее между точностью и полнотой. Она стремится к нулю, если точность или полнота стремится к нулю.

$$F = 2 \frac{Precision \times Recall}{Precision + Recall}$$

#### 4.1.2. Качество панорамных изображений

Существующие методы оценки качества восстановленных и обработанных изображений можно разделить на два класса [4]:

- субъективные оценки (экспертные);
- математические оценки (метрики).

Для метода субъективных оценок необходима группа экспертов, выполняющая оценку изображений по определенному алгоритму. Например, существуют следующие алгоритмы: DSIS (Double Stimulus Impairment Scale), DSCQS (Double Stimulus Continuous Quality Scale), SCACJ (Stimulus Comparison Adjectival Categorical Judgment), SAMVIQ (Subjective Assessment Method for Video Quality Evaluation).

Данный способ позволяет получить оценку высокого качества, но его использование довольно затратно и длительно, особенно это важно при отладке и экспериментальной настройке алгоритмов восстановления, обработки и сжатия изображений.

## 4.2. Исследование объектов

Всего в исследовании задействовано 30 объектов: 15 цилиндрической формы и 15 прямоугольной формы. Ниже представлены типовые объекты (рис. 4.1)

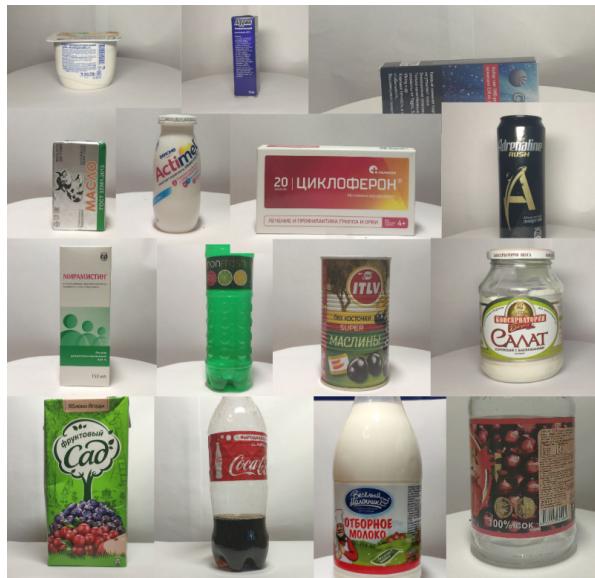


Рис.4.1. Пример исследуемых объектов

На основании полученных данных, была произведена оценка качества определения формы объекта. Результаты представлены в таблице 4.1.

Таблица 4.1

Оценка качества определения формы объекта (поменять данные)

Форма объекта	Точность	Полнота	F-мера
Цилиндрический объект	0.9231	0.8571	0.8889
Объект прямоугольной формы	0.9286	0.9286	0.9176

Проблемы с классификацией формы были вызваны:

- чувствительностью алгоритма к шумам (тени, отблески);
- наличием разнообразных текстур на упаковках товара;
- недостаточно хорошим качеством съемки (камера, освещение);
- наличием прозрачных упаковок.

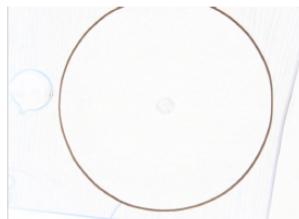
На примере товара "Африн" продемонстрируем определение формы объекта. На первом шаге произведем удаление шумов и нормализацию изображения.



(a) Изображение фона



(б) Изображение объекта на фоне



(в) Изображение фона (после удаления шумов и нормализации)

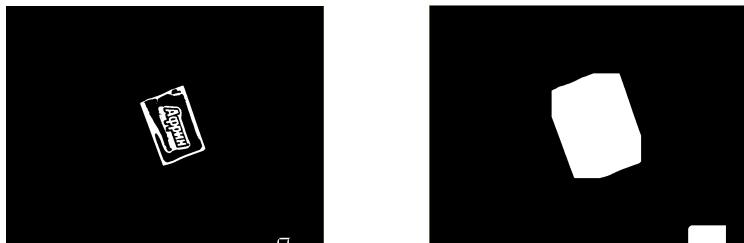


(г) Изображение объекта (после удаления шумов и нормализации)

Рис.4.2. Предварительная обработка изображения (удаление шумов и нормализация)

Далее выполним вычитание изображения с объектом из изображения фона. Это позволит локализовать объект (рис. 4.3. (а)), что

упростит его дальнейшее исследование. Так как объект имеет довольно сложные текстуры, функция нахождения оптимального (наибольшего) контура может дать сбой. Поэтому необходимо выполнить расширение (dilation) пикселей (рис. 4.3. (б)).



(a) Локализация объекта      (б) Расширение пикселей (dilation)

Рис.4.3. Нахождение маски объекта

Далее алгоритм находит оптимальный контур объекта, аппрокси- мирует фигуру и определяет его форму (рис. 4.4). В случае если объ- ект имеет прямоугольную форму производится расчет его положения относительно камеры. Это позволит правильно выставить объект при оцифровке.



(a) Нахождение контура

(б) Определение положения объекта

Рис.4.4. Определение формы и положения

Ниже представлены промежуточные результаты для типовых объектов:



Рис.4.5. Удаление шумов и нормализация

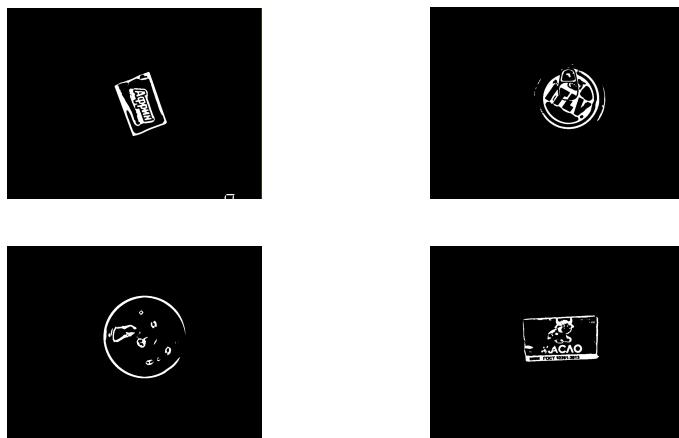


Рис.4.6. Локализация объектов

Конечный результат рассматриваемых объектов представлен на рис. 4.9.

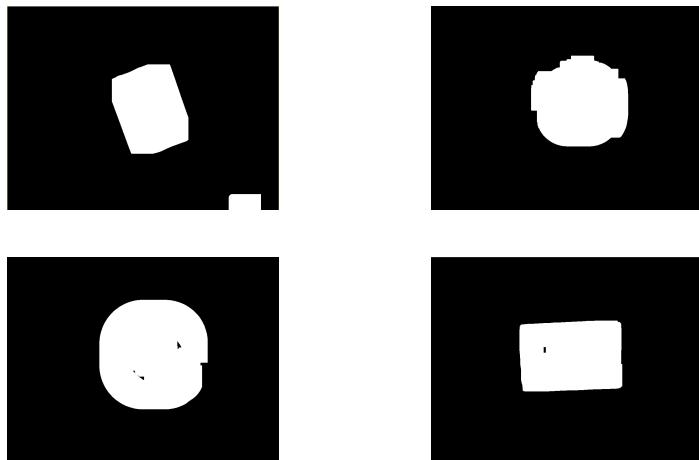


Рис.4.7. Нахождение масок объектов

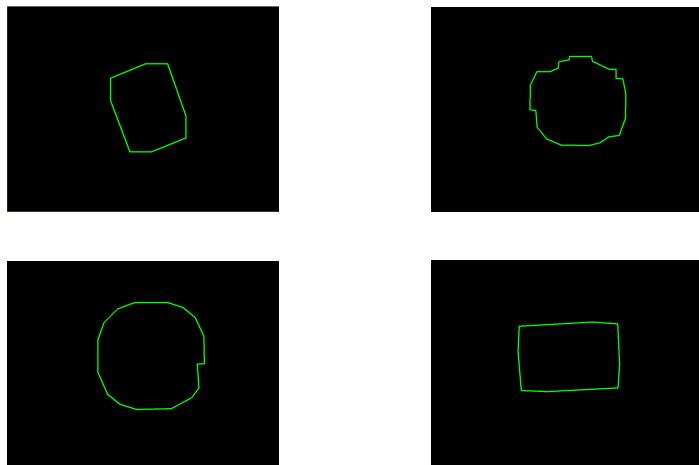


Рис.4.8. Нахождение контуров объектов

Для более сложных объектов, предполагается использование дополнительной камеры, что позволит лучше узнать об особенностях формы объекта. Ниже представлены результаты определения формы

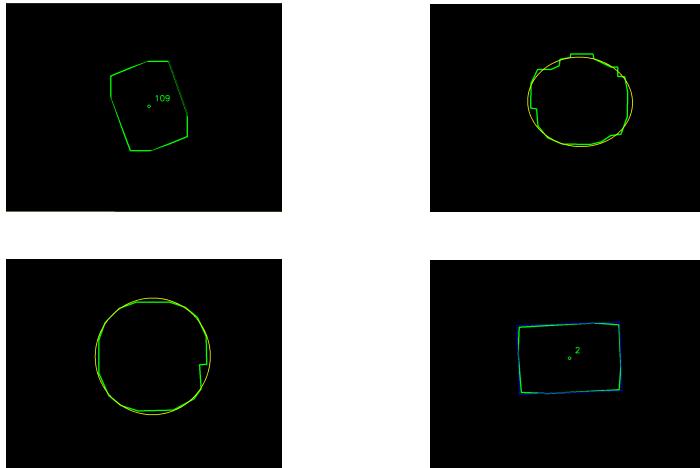


Рис.4.9. Определение формы и положения объектов

объекта при фронтальной съемки. Однако, в данном случае, усложняется алгоритм определения положения цилиндрических объектов, а также в некоторых случаях, повышается риск ошибки при определении формы объекта.

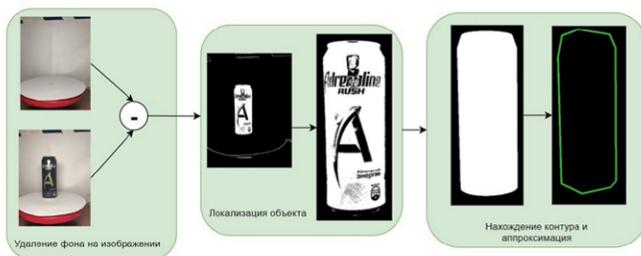


Рис.4.10. Пример исследуемых объектов

В случае если исследуемый объект имеет цилиндрическую форму, необходимо получить его развертку. Для тестирования алгоритма, возьмем некоторые изображения и обрежем их с нахлестом друг на

друга.



Рис.4.11. Пример нарезанных изображений

В алгоритме предусмотрена последовательная склейка всех изображений из папки img\_cut. Результат склейки представлен ниже.

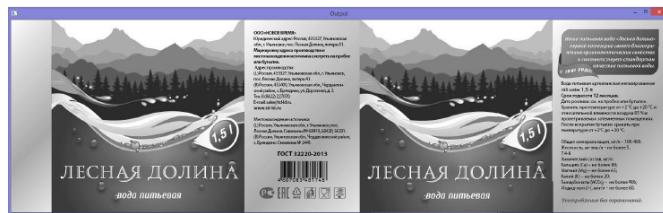


Рис.4.12. Результат склейки для идеального изображения

Однако, когда мы имеем дело с реальными объектами, качество склеивания ухудшается. Ниже приведен пример получения развертки для бутылки молока.



Рис.4.13. Часть изображений бутылки молока

В общей сложности, для оцифровки бутылки молока использовалось около 180 фотографий (шаг фотографирования - 2 градуса). Результат работы алгоритма для реального объекта представлен ниже.



Рис.4.14. Результат склейки для реального объекта

На качество полученного изображения повлияло плохое освещение (отблески), плохое качество изображения, а также неправильный совместный выбор таких параметров как величина шага и размер линейного окна. В дальнейшем предполагается улучшение данного ал-

горитма.

### 4.3. Использование реальной системы

Реальная модель физической установки представлена на рис. 4.15. В красном блоке находится управляющее устройство, подсистема индикации, подсистема управления шаговым двигателем; зеленые блоки указывают на места для камер; фиолетовым цветом помечена система для вращения объекта.

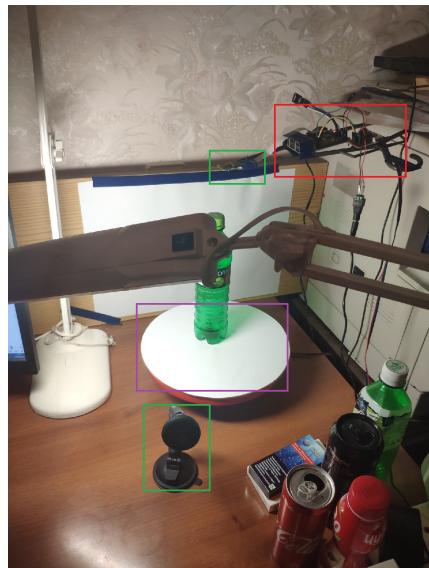


Рис.4.15. Прототип системы

Оцифровка товара происходит посредством пользовательского интерфейса. Вид главного окна представлен ниже (рис. 4.16). Перед тем как начать оцифровку товара, необходимо задать начальные параметры системы.

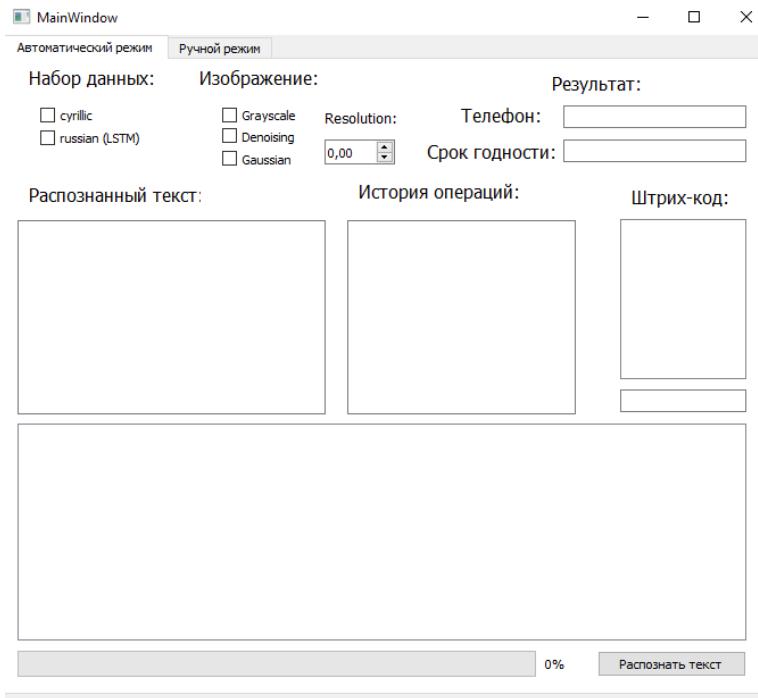


Рис.4.16. Пример главного окна приложения

В секции "Набор данных" пользователь может выбрать какой набор данных использовать при распознавании текста с этикетом. Опытном путем было установлено, что использование одновременно двух наборов данных улучшает качество распознавания.

Итоговое изображение, которое в последствии будет распознано с помощью библиотеки Tesseract, необходимо предварительно обработать. Настройки предварительной обработки находятся в разделе "Изображение".

Система завершит работу когда полоска индикации будет полностью заполнена. Результат работы системы будет отображен в следующих секциях: Результат, Распознанный текст, История операций,

Штрих-код и в нижнем блоке появится финальное изображение (полученное системой).

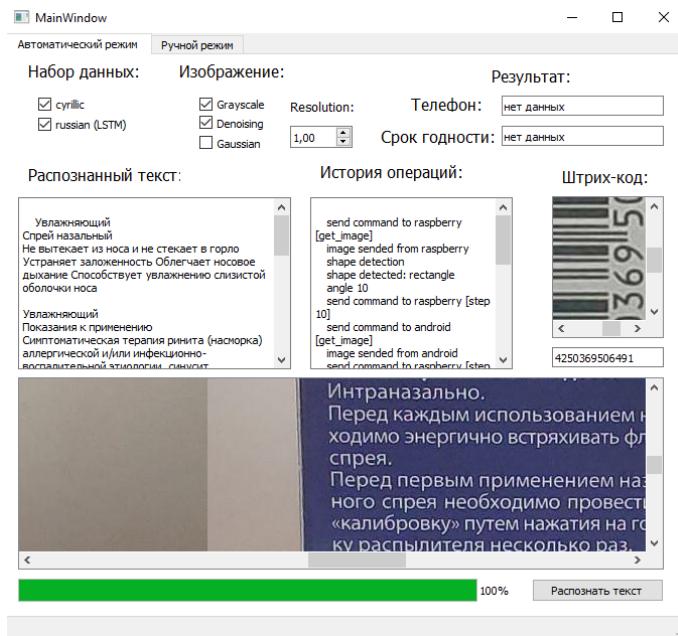


Рис.4.17. Результат работы программы

## ЗАКЛЮЧЕНИЕ

Использование систем визуального контроля позволяет получить информацию с этикеток товаров за счет графической распознавания, что снижает риск человеческой ошибки, а также позволяет ускорить процесс оцифровки товара.

В рамках данной работы был выполнен обзор и произведен анализ существующих систем, методов и решений, итогом которого стала разработка концепции и проектирование структуры программно-аппаратного устройства, реализующего решение поставленной задачи. Для демонстрации предлагаемого автором подхода, был создан прототип системы распознавания и идентификации продуктов питания, для которого было написано соответствующее инструментальное и прикладное программное обеспечение.

В работе были успешно решены следующие поставленные задачи:

- составление обзора существующих систем, методов и подходов к решению поставленной задачи;
- аппаратная и программная реализация установки для вращения товара;
- разработка серверной части, в том числе алгоритмов обработки изображений для получения информации с этикетки товара;
- разработка мобильного приложения для автоматического фотографирования;
- разработка пользовательского интерфейса для управления установкой;
- распознавание текста и поиск штрих-кода на этикетках товара;
- проведение экспериментальных исследований разработанной системы и оценка качества.

Испытания и тесты подтвердили работоспособность системы, а также позволили выявить ее слабые места. Дальнейшее развитие работы может быть связано с:

- исследование алгоритмов машинного обучения для решения задачи классификации формы;
- модификацией алгоритма получения развертки цилиндрических объектов для достижении должного качества;
- улучшение физической установки и пользовательского интерфейса.

## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Виды упаковки [Электронный ресурс]. — Режим доступа: [https://studme.org/63299/marketing/vidy\\_upakovki](https://studme.org/63299/marketing/vidy_upakovki) (дата обращения: 12.05.2019).
2. Обнаружение. Проверка. Классификация. Считывание [Электронный ресурс], Сайт производителя Cognex. — Режим доступа: <https://www.cognex.com> (дата обращения: 12.05.2019).
3. Шаговый двигатель или сервопривод? [Электронный ресурс]. — Режим доступа: <https://stepmotor.ru/servodvigateli-protiv-shagovyx> (дата обращения: 12.05.2019).
4. Филиппов А. О. и Филиппов А. О. Метрики для оценки качества восстановленного изображения // Научное сообщество студентов. Междисциплинарные исследования. — 2018. — С. 183–188.
5. Чиликин А. С. Автоматическое построение панорамного изображения по видеоряду. — 2018. — С. 49.
6. Прогноз: объём рынка упаковки вырастет к 2020 [Электронный ресурс]. — Режим доступа: [https://www.publish.ru/news/201607-08\\_20089353](https://www.publish.ru/news/201607-08_20089353) (дата обращения: 12.05.2019).
7. Контрольно-измерительные приборы и весовое оборудование METTLER TOLEDO [Электронный ресурс], Сайт производителя METTLER TOLEDO. — Режим доступа: <https://www.mt.com> (дата обращения: 12.05.2019).
8. А Горелик. Методы распознавания [Электронный ресурс]. — Москва: Высшая школа, 2004.
9. Сикорский О.С. Обзор свёрточных нейронных сетей для задачи классификации изображений // Новые информационные технологии в автоматизированных системах. — 2017. — № 20.
10. ЦРПТ и X5 Retail Group создадут единый каталог товаров России [Электронный ресурс], Веб-сайт проекта Retail Life. — Режим доступа: <https://retail-life.ru/>

crpt-i-x5-retail-group-sozdadut-edinyj-katalog-tovarov-rossii  
(дата обращения: 12.05.2019).

11. Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) [Электронный ресурс]. — Режим доступа: <https://creativecommons.org/licenses/by-sa/4.0/> (дата обращения: 12.05.2019).

12. Automatic data capture and process automation markets [Электронный ресурс], Datalogic. — Режим доступа: <https://www.datalogic.com> (дата обращения: 12.05.2019).

13. Biederman Irving. Recognition-by-components: a theory of human image understanding. // Psychological review. — 1987. — Vol. 94, no. 2. — P. 115.

14. Ciclop 3D Scanner [Электронный ресурс], Веб-сайт проекта RepRap. — Режим доступа: <https://reprap.org/wiki/Ciclop> (дата обращения: 12.05.2019).

15. Cylindrical panoramic image stitching method based on multi-cameras / Mingxiu Lin, Gang Xu, Xingning Ren, Ke Xu // 2015 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER) / IEEE. — 2015. — P. 1091–1096.

16. Fast contour-tracing algorithm based on a pixel-following method for image sensors / Jonghoon Seo, Seungho Chae, Jinwook Shim et al. // Sensors. — 2016. — Vol. 16, no. 3. — P. 353.

17. In-sight 9902L line scan vision system [Электронный ресурс], Веб-сайт компании Cognex. — Режим доступа: <https://www.cognex.com/products/machine-vision/2d-machine-vision-systems/in-sight-9902-line-scan-vision-system> (дата обращения: 12.05.2019).

18. Object recognition using constraints from primitive shape matching / Nikhil Somani, Caixia Cai, Alexander Perzylo et al. // International Symposium on Visual Computing / Springer. — 2014. — P. 783–792.

19. OpenCV shape detection [Электронный ресурс]. — Режим доступа: <https://www.pyimagesearch.com/2016/02/08/>

[opencv-shape-detection/](#) (дата обращения: 12.05.2019).

20. Pandelea Alexandrina-Elena, Budescu Mihai, Covatariu Gabriela. Image Processing Using Artificial Neural Networks // Buletinul Institutului Politehnic din Iasi. Sectia Constructii, Arhitectura. — 2015. — Vol. 61, no. 4. — P. 9.

21. PhotoPizza — поворотный стол для съемки 3D-фото 360 [Электронный ресурс], Веб-сайт проекта PhotoPizza. — Режим доступа: <http://makerdrive.org/project/item/photopizza/blog/10.html> (дата обращения: 12.05.2019).

22. Structural Analysis and Shape Descriptors [Электронный ресурс], OpenCV. — Access mode: [https://docs.opencv.org/3.1.0/d3/dc0/group\\_\\_imgproc\\_\\_shape.html](https://docs.opencv.org/3.1.0/d3/dc0/group__imgproc__shape.html) (online; accessed: 12.05.2019).

23. Taking in Shape: Detection and Tracking of Basic 3D Shapes in a Robotics Context / Richtsfeld Andreas, Mörwald Thomas, Zilllich Michael, Vincze Markus.

24. Train a Mask R-CNN model on your own data [Электронный ресурс], Веб-сайт проекта Waspinator. — Режим доступа: <https://patrickwasp.com/train-a-mask-r-cnn-model-on-your-own-dataset/> (дата обращения: 12.05.2019).

25. Xhensila Poda Olti Qirici. Shape detection and classification using OpenCV and Arduino Uno. — 2017.

## ЛИСТИНГИ

Листинг 1. Код клиента на Raspberry

```

1 import socket
2 from test_led import RGBLed
3 from test_motor import A4988Driver
4 from picamera import PiCamera
5 from time import sleep
6 import base64
7
8
9 HOST = '192.168.1.104'
10 PORT = 80
11
12 def take_foto():
13     with PiCamera() as camera:
14         camera.resolution = (1024, 768)
15         camera.awb_mode = 'sunlight'
16         camera.start_preview()
17         camera.capture('/home/pi/Desktop/image_to_send.jpg')
18         camera.stop_preview()
19
20 def convertImageToBase64():
21     with open("/home/pi/Desktop/image_to_send.jpg", "rb") as
22         image_file:
23             encoded = base64.b64encode(image_file.read())
24     return encoded
25
26 class RaspberryClient:
27     def __init__(self):
28         self.soc = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
29         self.soc.connect((HOST, PORT))
30         self.motor = A4988Driver(direction_pin=21, step_pin=20,
31                         step_type_pins=(11, 13, 15))
32         self.led = RGBLed(17, 27, 22)
33
34     def init_ping(self):
35         message = "raspberry"
36         self.soc.send(message.encode('ascii'))
37
38     def main_loop(self):
39         while True:
40             data = self.soc.recv(512)
41             if data:
42                 command = str(data.decode('ascii'))
43                 print('Received from the server : ' + command)
44                 command = command.split(' ')

```

```

43         if command[0] == "led":
44             self.led.turn_off_color('red')
45             self.led.turn_off_color('blue')
46             self.led.turn_off_color('green')
47             if command[1] == "red":
48                 self.led.turn_on_color('red')
49             elif command[1] == "blue":
50                 self.led.turn_on_color('blue')
51             elif command[1] == "green":
52                 self.led.turn_on_color('green')
53         elif command[0] == "step":
54             self.motor.go_steps(int(command[1]))
55         elif command[0] == "foto":
56             take_foto()
57             with open('/home/pi/Desktop/image_to_send.jpg',
58                         'rb') as fs:
59                 while True:
60                     data = fs.read(4096)
61                     self.soc.send(data)
62                     if not data:
63                         print('Foto sent')
64                         break
65                     self.soc.send(b'ENDED') # I used the same
66                     size of the BEGIN token
67                     fs.close()
68             elif command[0] == "stop":
69                 self.soc.close()
70                 break
71         if __name__ == '__main__':
72             ras = RaspberryClient()
73             ras.init_ping()
74             ras.main_loop()

```

Листинг 2. Код управления шаговым двигателем

```

1 import time
2 import RPi.GPIO as GPIO
3
4
5 class A4988Driver:
6     def __init__(self, direction_pin, step_pin, step_type_pins):
7         self.direction_pin = direction_pin
8         self.step_pin = step_pin
9         self.step_type_pins = step_type_pins

```

```

10     GPIO.setmode(GPIO.BCM)
11     GPIO.setwarnings(False)
12
13     step_types = {'Full': (0, 0, 0),
14                   'Half': (1, 0, 0),
15                   '1/4': (0, 1, 0),
16                   '1/8': (1, 1, 0),
17                   '1/16': (1, 1, 1)}
18
19     degree_values = {'Full': 1.8,
20                       'Half': 0.9,
21                       '1/4': .45,
22                       '1/8': .225,
23                       '1/16': 0.1125}
24
25     def set_step_type(self, step_type):
26         if step_type in self.step_types:
27             GPIO.output(self.step_type_pins, self.step_types[
28                 step_type])
29         else:
30             print("Unknown step type")
31
32     def get_degree(self, steps, step_type):
33         return steps * self.degree_values[step_type]
34
35     def _clean_GPIO(self):
36         GPIO.output(self.step_pin, False)
37         GPIO.output(self.direction_pin, False)
38         for pin in self.step_type_pins:
39             GPIO.output(pin, False)
40
41     def _set_up_GPIO(self):
42         GPIO.setup(self.direction_pin, GPIO.OUT)
43         GPIO.setup(self.step_pin, GPIO.OUT)
44         GPIO.output(self.direction_pin, False)
45         GPIO.setup(self.step_type_pins, GPIO.OUT)
46
47     def _run_steps(self, steps, step_type, step_delay, verbose):
48         for step in range(steps):
49             GPIO.output(self.step_pin, True)
50             time.sleep(step_delay)
51             GPIO.output(self.step_pin, False)
52             time.sleep(step_delay)
53             if verbose:
54                 print("Steps count {}".format(step))
55         if verbose:

```

```

55     print("\n Details: \n")
56     print("Step Type = {}".format(step_type))
57     print("Number of steps = {}".format(steps))
58     print("Size of turn in degrees = {}"
59           .format(self.get_degree(steps, step_type)))
60
61     def go_one_custom_step(self, step_type="Full", step_delay=0.005,
62                           init_delay=0.05,
63                           verbose=True):
64         self._set_up_GPIO()
65         self.set_step_type(step_type)
66         time.sleep(init_delay)
67         self._run_steps(1, step_type, step_delay, verbose)
68         self._clean_GPIO()
69
70     def go_steps(self, steps=200, step_type="Full", step_delay
71                  =0.005, init_delay=0.05,
72                  verbose=True):
73         self._set_up_GPIO()
74         self.set_step_type(step_type)
75         time.sleep(init_delay)
76         self._run_steps(steps, step_type, step_delay, verbose)
77         self._clean_GPIO()
78
79     def go_degree(self, degree=30, step_type="Full", step_delay
80                  =0.001, init_delay=0.05,
81                  verbose=True):
82         if step_type in self.step_types:
83             steps = int(degree / self.degree_values[step_type])
84             self._set_up_GPIO()
85             self.set_step_type(step_type)
86             time.sleep(init_delay)
87             self._run_steps(steps, step_type, step_delay, verbose)
88             self._clean_GPIO()
89
90     if __name__ == "__main__":
91         motor = A4988Driver(direction_pin=21, step_pin=20,
92                             step_type_pins=(11, 13, 15))
93         motor.go_steps(1600)

```

Листинг 3. Код управление RGB-светодиодом

```

1 import time, sys
2 import RPi.GPIO as GPIO
3

```

```
4
5 class RGBLed:
6     def __init__(self, red_pin, green_pin, blue_pin):
7         GPIO.setmode(GPIO.BCM)
8         GPIO.setwarnings(False)
9         self.colors = {'red': red_pin,
10                      'green': green_pin,
11                      'blue': blue_pin}
12         for value in self.colors.values():
13             GPIO.setup(value, GPIO.OUT)
14             GPIO.output(value, False)
15
16
17     def _get_pin(self, color):
18         if color in self.colors:
19             return self.colors[color]
20         else:
21             print('Unknown color!')
22             return None
23
24
25     @staticmethod
26     def _turn_on_pin(pin):
27         GPIO.setup(pin, GPIO.OUT)
28         GPIO.output(pin, True)
29
30     @staticmethod
31     def _turn_off_pin(pin):
32         GPIO.setup(pin, GPIO.OUT)
33         GPIO.output(pin, False)
34
35     def turn_on_color(self, color):
36         color_pin = self._get_pin(color)
37         if color_pin is not None:
38             self._turn_on_pin(color_pin)
39
40     def turn_off_color(self, color):
41         color_pin = self._get_pin(color)
42         if color_pin is not None:
43             self._turn_off_pin(color_pin)
44
45     def blink(self, color='red', delay=0.5):
46         color_pin = self._get_pin(color)
47         if color_pin is not None:
48             self._turn_on_pin(color_pin)
49             time.sleep(delay)
50             self._turn_off_pin(color_pin)
```

```

50         time.sleep(delay)
51
52
53 if __name__ == "__main__":
54     led = RGBLED(17, 27, 22)
55     while True:
56         led.blink()

```

Листинг 4. Код создания развертки

```

1 import glob
2 import sys
3 import numpy
4 import imutils
5 import cv2
6
7
8 def readImages(imageString):
9     images = []
10
11     # Get images from arguments.
12     for i in range(0, len(imageString)):
13         img = cv2.imread(imageString[i])
14         images.append(img)
15
16     return images
17
18
19 def findAndDescribeFeatures(image):
20     # Getting gray image
21     grayImage = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
22
23     # Find and describe the features.
24     # Fast: sift = cv2.xfeatures2d.SURF_create()
25     sift = cv2.xfeatures2d.SIFT_create()
26
27     # Find interest points.
28     keypoints = sift.detect(grayImage, None)
29
30     # Computing features.
31     keypoints, features = sift.compute(grayImage, keypoints)
32
33     # Converting keypoints to numbers.
34     keypoints = numpy.float32([kp.pt for kp in keypoints])
35
36     return keypoints, features

```

```
37
38
39 def matchFeatures(featuresA, featuresB):
40     # Slow: featureMatcher = cv2.DescriptorMatcher_create("BruteForce")
41     featureMatcher = cv2.DescriptorMatcher_create("FlannBased")
42     matches = featureMatcher.knnMatch(featuresA, featuresB, k=2)
43     return matches
44
45
46 def generateTransformation(allMatches, keypointsA, keypointsB, ratio):
47     if not allMatches:
48         return None
49     matches = []
50
51     for match in allMatches:
52         # Lowe's ratio test
53         if len(match) == 2 and (match[0].distance / match[1].distance) < ratio:
54             matches.append(match[0])
55
56     pointsA = numpy.float32([keypointsA[m.queryIdx] for m in matches])
57     pointsB = numpy.float32([keypointsB[m.trainIdx] for m in matches])
58
59     if len(pointsA) > 2:
60         transformation = cv2.estimateRigidTransform(pointsA, pointsB, True)
61         if transformation is None or transformation.shape[1] < 1 or
62             transformation.shape[0] < 1:
63             return None
64         return transformation
65     else:
66         return None
67
68 paths = glob.glob(("C:/Users/andre/Desktop/Panorama-master/result/*.jpg"))
69 images = readImages(paths)
70 result = images[0]
71
72 while len(images) > 1:
73     imgR = images.pop()
74     imgL = images.pop()
```

```

75     interestsR, featuresR = findAndDescribeFeatures(imgR)
76     interestsL, featuresL = findAndDescribeFeatures(imgL)
77     allMatches = matchFeatures(featuresR, featuresL)
78
79
80     transformation = generateTransformation(allMatches, interestsR,
81         interestsL, 0.35)
82     if transformation is None or transformation[0, 2] < 0:
83         images.append(imgR)
84         continue
85     transformation[0, 0] = 1
86     transformation[1, 1] = 1
87     transformation[0, 1] = 0
88     transformation[1, 0] = 0
89     transformation[1, 2] = 0
90     result = cv2.warpAffine(imgR, transformation, (imgR.shape[1] +
91         int(transformation[0, 2] + 1), imgR.shape[0]))
92     result[:, :imgL.shape[1]] = imgL
93     cv2.imshow("R", result)
94     images.append(result)
95     cv2.waitKey(1)
96
97     cv2.imwrite("Result2.jpg", result)
98     cv2.waitKey()

```

Листинг 5. Код нарезки изображений

```

1 import os
2 import re
3 from operator import itemgetter
4
5 import cv2
6 import glob
7
8 def sorted_aphanumeric(data):
9     convert = lambda text: int(text) if text.isdigit() else text.
10        lower()
11     alphanum_key = lambda key: [ convert(c) for c in re.split('
12        ([0-9]+)', key) ]
13     return sorted(data, key=alphanum_key)
14
15 pathOut = r"C:\Users\andre\Desktop\Panorama-master\result\milk"
16 if __name__ == "__main__":
17     num = 0
18     paths = glob.glob("C:/Users/andre/Desktop/Panorama-master/frames
19                      /milk_data/*.jpg")

```

```

17     #pdfList = sorted(paths, key=lambda x: x.rsplit('.', 1)[0])
18     dirlist = sorted_aphanumeric(paths)
19     for file in dirlist:
20         num = num + 1
21         print(file)
22         img = cv2.imread(file)
23         crop_img = img[535:1150, 355:430]
24         cv2.imwrite(pathOut + str(num).zfill(5) + ".jpg", crop_img)
25         #cv2.imwrite("cropped.jpg", crop_img)
26         #cv2.imshow("cropped", crop_img)
27         #cv2.waitKey()

```

Листинг 6. Код извлечения изображений из видео

```

1 import sys
2 import argparse
3
4 import cv2
5 print(cv2.__version__)
6
7 def extractImages(pathIn, pathOut):
8     count = 0
9     vidcap = cv2.VideoCapture(pathIn)
10    success,image = vidcap.read()
11    success = True
12    num = 0
13    while success:
14        num = num + 1
15        vidcap.set(cv2.CAP_PROP_POS_MSEC,(count*100))      # added
16                    this line
16        success,image = vidcap.read()
17        print ('Read a new frame: ', success)
18        cv2.imwrite( pathOut + "\\vlcsnap-" + str(num).zfill(5)+ "."
19                     jpg", image)      # save frame as JPEG file
20        count = count + 1
21
22 if __name__=="__main__":
23     print("aba")
24     a = argparse.ArgumentParser()
25     a.add_argument("--pathIn", default=r"C:\Users\andre\Desktop\
26                   Panorama-master\video_2019-05-05_23-36-15.mp4", help="path
27                   to video")
28     a.add_argument("--pathOut", default=r"C:\Users\andre\Desktop\
29                   Panorama-master\vlcsnaps", help="path to images")
30     args = a.parse_args()
31     print(args)

```

```
28     extractImages(args.pathIn, args.pathOut)
```

### Листинг 7. Определение формы объектов

```
1 import cv2
2 import numpy as np
3
4 BACKGROUND = "data/shapes_top/image0.jpg"
5 OBJECT = "data/shapes_top/image2.jpg"
6
7
8 def _remove_shadows(image):
9     rgb_planes = cv2.split(image)
10    result_planes = []
11    result_norm_planes = []
12    for plane in rgb_planes:
13        dilated_img = cv2.dilate(plane, np.ones((12, 12), np.uint8))
14        bg_img = cv2.medianBlur(dilated_img, 21)
15        diff_img = 255 - cv2.absdiff(plane, bg_img)
16        norm_img = cv2.normalize(diff_img, None, alpha=0, beta=255,
17                               norm_type=cv2.NORM_MINMAX, dtype=cv2.CV_8UC1)
18        result_planes.append(diff_img)
19        result_norm_planes.append(norm_img)
20    result = cv2.merge(result_planes)
21    result_norm = cv2.merge(result_norm_planes)
22    return result, result_norm
23
24 class ShapeDetector:
25     def __init__(self, background_image, object_image, threshold=65,
26                  step_image=True):
27         self.background_image, _ = _remove_shadows(background_image)
28         self.object_image, _ = _remove_shadows(object_image)
29         cv2.imwrite("test1.jpg", self.background_image)
30         cv2.imwrite("test2.jpg", self.object_image)
31
32         self.threshold = threshold
33         self.step_image = step_image
34         self.result = self.object_image
35
36     def get_shape(self):
37         self._get_diff()
38         self._dilate()
39         self._draw_contour()
40         shape = self._detect_shape()
```

```

41         cv2.imshow("Result", self.result)
42         cv2.waitKey()
43     return shape
44
45     def _get_diff(self, white=True):
46         diff = cv2.absdiff(self.background_image, self.object_image)
47         mask = cv2.cvtColor(diff, cv2.COLOR_BGR2GRAY)
48         i_mask = mask > self.threshold
49         canvas = np.zeros_like(self.object_image, np.uint8)
50         canvas[i_mask] = self.object_image[i_mask]
51         if white:
52             canvas[np.where((canvas != [0, 0, 0]).all(axis=2))] =
53                 [255, 255, 255]
54         if self.step_image:
55             cv2.imwrite("diff.jpg", canvas)
56         self.result = canvas
57
58     def _dilate(self):
59         kernel = np.ones((5, 5), np.uint8)
60         dilation = cv2.dilate(self.result, kernel, iterations=20)
61         if self.step_image:
62             cv2.imwrite("dilated.jpg", dilation)
63         self.result = dilation
64
65     def _draw_contour(self, contour_type="largest"):
66         num = 0 if contour_type == "largest" else 1
67         canvas = np.zeros((self.result.shape[0] + 15, self.result.
68             shape[1] + 15, 3), np.uint8)
69         self.result = cv2.cvtColor(self.result, cv2.COLOR_BGR2GRAY)
70         _, contours, hierarchy = cv2.findContours(self.result, cv2.
71             RETR_EXTERNAL,
72                                         cv2.
73                                         CHAIN_APPROX_SIMPLE
74 )
75         area_array = []
76         for i, c in enumerate(contours):
77             area = cv2.contourArea(c)
78             area_array.append(area)
79         sorted_data = sorted(zip(area_array, contours), key=lambda x
80             : x[0], reverse=True)
81         contour = sorted_data[num][1]
82         epsilon = 0.005 * cv2.arcLength(contour, True)
83         approx = cv2.approxPolyDP(contour, epsilon, True)
84         cv2.drawContours(canvas, [approx], -1, (0, 255, 0), 3)
85         if self.step_image:
86             cv2.imwrite("contours.jpg", canvas)

```

```
81         self.contour = contour
82         self.result = canvas
83
84     def _detect_shape(self):
85         peri = cv2.arcLength(self.contour, True)
86         approx = cv2.approxPolyDP(self.contour, 0.04 * peri, True)
87         if len(approx) == 4:
88             (x, y, w, h) = cv2.boundingRect(approx)
89             ar = w / float(h)
90             shape = "square" if 0.95 <= ar <= 1.05 else "rectangle"
91         else:
92             shape = "circle"
93         print(len(approx))
94         return shape
95
96     def find_rectangle(self):
97         rect = cv2.minAreaRect(self.contour)
98         box = cv2.boxPoints(rect)
99         box = np.int0(box)
100        center = (int(rect[0][0]), int(rect[0][1]))
101        area = int(rect[1][0] * rect[1][1])
102        edge1 = np.int0((box[1][0] - box[0][0], box[1][1] - box
103                      [0][1]))
104        edge2 = np.int0((box[2][0] - box[1][0], box[2][1] - box
105                      [1][1]))
106        usedEdge = edge1
107        if cv2.norm(edge2) > cv2.norm(edge1):
108            usedEdge = edge2
109        reference = (1, 0)
110        angle = 180.0 / np.math.pi * np.math.acos(
111            (reference[0] * usedEdge[0] + reference[1] * usedEdge
112            [1]) / (cv2.norm(reference) * cv2.norm(usedEdge)))
113        if area > 300:
114            cv2.drawContours(self.result, [box], 0, (0, 0, 0), 2)
115            cv2.circle(self.result, center, 5, (0, 255, 0), 2)
116            cv2.putText(self.result, "%d" % int(angle), (center[0] +
117                                              20, center[1] - 20),
118                                              cv2.FONT_HERSHEY_SIMPLEX, 1, (0, 255, 0), 2)
119            cv2.imshow('Rectangles', self.result)
120            cv2.waitKey()
121
122    def find_ellipse(self):
123        ellipse = cv2.fitEllipse(self.contour)
124        cv2.ellipse(self.result, ellipse, (0, 255, 255), 2)
125        cv2.imshow('Ellipse', self.result)
126        cv2.waitKey()
```

```

123
124
125 if __name__ == "__main__":
126     background_image = cv2.imread(BACKGROUND)
127     object_image = cv2.imread(OBJECT)
128     shape = ShapeDetector(background_image, object_image)
129     shape_type = shape.get_shape()
130     if shape_type == "rectangle" or shape_type == "square":
131         shape.find_rectangle()
132     else:
133         shape.find_ellipse()

```

Листинг 8. MainActivity

```

1 package pro.andreyn.pythonclient;
2
3 import android.Manifest;
4 import android.content.pm.PackageManager;
5 import android.graphics.SurfaceTexture;
6 import android.hardware.Camera;
7 import android.os.Build;
8 import android.os.Bundle;
9 import android.os.Handler;
10 import android.os.Looper;
11 import android.os.Message;
12 import android.support.annotation.NonNull;
13 import android.support.v4.app.ActivityCompat;
14 import android.support.v4.content.ContextCompat;
15 import android.support.v7.app.AppCompatActivity;
16 import android.util.Log;
17 import android.view.View;
18 import android.widget.Button;
19 import android.widget.EditText;
20 import android.widget.TextView;
21 import android.widget.Toast;
22
23 import java.io.BufferedReader;
24 import java.io.IOException;
25 import java.io.InputStreamReader;
26 import java.net.Socket;
27 import java.netSocketAddress;
28 import java.util.List;
29
30 public class MainActivity extends AppCompatActivity {
31
32     public final static String TAG = "DEBUG_TAG";

```

```
33     public static Handler handler = null;
34     private TextView textView = null;
35     private EditText editText;
36     private View view1, view2, view3;
37     private Socket s = null;
38     private android.hardware.Camera camera;
39     private Thread thread;
40
41
42     @Override
43     protected void onCreate(Bundle savedInstanceState) {
44         super.onCreate(savedInstanceState);
45         setContentView(R.layout.activity_main);
46
47         editText = (EditText) findViewById(R.id.edittext_server);
48         textView = (TextView) findViewById(R.id.status);
49
50         view1 = (View) findViewById(R.id.relative1);
51         view2 = (View) findViewById(R.id.relative2);
52         view3 = (View) findViewById(R.id.relative3);
53
54         view2.setVisibility(View.GONE);
55         view3.setVisibility(View.GONE);
56
57         editText.setText("192.168.1.104:80");
58
59         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.JELLY_BEAN)
60             {
61                 if (ContextCompat.checkSelfPermission(MainActivity.this,
62                     Manifest.permission.CAMERA)
63                     != PackageManager.PERMISSION_GRANTED) {
64                         ActivityCompat.requestPermissions(this,
65                             new String[]{Manifest.permission.CAMERA},
66                             1);
67                 } else {
68                     if (ContextCompat.checkSelfPermission(MainActivity.
69                         this, Manifest.permission.
70                         WRITE_EXTERNAL_STORAGE)
71                         != PackageManager.PERMISSION_GRANTED) {
72                         ActivityCompat.requestPermissions(this,
73                             new String[]{Manifest.permission.
74                                 WRITE_EXTERNAL_STORAGE},
75                             2);
76                 } else {
77                     getCamera();
78                 }
79             }
80         }
81     }
82
83     private void getCamera() {
84
85         camera = Camera.open();
86
87         if (camera != null) {
88             camera.setDisplayOrientation(90);
89             camera.setPreviewDisplay(textView.getHolder());
90             camera.startPreview();
91         }
92     }
93 }
```

```
74         }
75     }
76 }
77 }
78
79 ((Button) findViewById(R.id.button_connect)).
80     setOnClickListener(new View.OnClickListener() {
81     @Override
82     public void onClick(View v) {
83         if (editText.getText().length() > 0) {
84             String[] server = editText.getText().toString().
85                 split(":");
86             if (server.length == 2) {
87                 connectTo(server);
88             } else {
89                 Toast.makeText(MainActivity.this,
90                     getResources().getString(R.string.
91                         toast_not_wrong), Toast.LENGTH_SHORT).
92                     show();
93             }
94         }
95     }
96 });
97
98 ((Button) findViewById(R.id.button_exit)).setOnClickListener
99     (new View.OnClickListener() {
100     @Override
101     public void onClick(View v) {
102         try {
103             if (s != null) s.close();
104             view1.setVisibility(View.VISIBLE);
105             view2.setVisibility(View.GONE);
106             textView.setText("");
107         } catch (IOException e) {
108             e.printStackTrace();
109         }
110     }
111 });
112 // Получение разрешения на использования камеры и запись в память
```

```
    }

112    @Override
113    public void onRequestPermissionsResult(int requestCode, @NonNull
114        String[] permissions, @NonNull int[] grantResults) {
115        super.onRequestPermissionsResult(requestCode, permissions,
116            grantResults);

117        switch (requestCode) {
118            case 1: {
119                if (grantResults.length > 0 && grantResults[0] ==
120                    PackageManager.PERMISSION_GRANTED) {
121
122                    if (ContextCompat.checkSelfPermission(
123                        MainActivity.this, Manifest.permission.
124                        WRITE_EXTERNAL_STORAGE)
125                        != PackageManager.PERMISSION_GRANTED) {
126                        ActivityCompat.requestPermissions(this,
127                            new String[]{Manifest.permission.
128                                WRITE_EXTERNAL_STORAGE},
129                                2);
130                    } else {
131                        getCamera();
132                    }
133                }
134            case 2: {
135                if (grantResults.length > 0 && grantResults[0] ==
136                    PackageManager.PERMISSION_GRANTED) {
137                    getCamera();
138                } else {
139                    Toast.makeText(this, "Permission denied to read
140                        SMS", Toast.LENGTH_SHORT).show();
141                    finish();
142                }
143            }
144        }

145        private void connectTo(final String[] server) {
146
147            handler = new Handler(Looper.getMainLooper()) {
```



```
186                     break;
187                 }
188             }
189
190         } catch (Exception e) {
191             e.printStackTrace();
192             thread.run();
193         }
194
195         Log.d(TAG, "run: ");
196
197     }
198 });
199
200     thread.start();
201 }
202 }
203
204 public void makePhoto() {
205     SurfaceTexture st = new SurfaceTexture(MODE_PRIVATE);
206     try {
207         camera.setPreviewTexture(st);
208     } catch (IOException e) {
209         e.printStackTrace();
210     }
211     camera.startPreview();
212
213     camera.autoFocus(new Camera.AutoFocusCallback() {
214         @Override
215         public void onAutoFocus(boolean b, Camera cam) {
216             try {
217                 Thread.sleep(500);
218             } catch (InterruptedException e) {
219                 e.printStackTrace();
220             }
221             cam.takePicture(null, null, new PhotoHandler(s));
222         }
223     });
224 }
225
226 private void getCamera() {
227     // Проверяем есть ли камера на устройстве
228     if (!getPackageManager().hasSystemFeature(PackageManager.
229         FEATURE_CAMERA)) {
230         Log.d(TAG, "На этом устройстве не камеры: ");
231     } else {
```

```

231     // Получаем ID камеры
232     int cameraId = findFrontFacingCamera();
233     if (cameraId < 0) {
234         Log.d(TAG, "Фронтальная камера не найдена: " +
235             cameraId);
236     } else {
237         // Открываем камеру для съемки
238         Log.d(TAG, "Фронтальная камера найдена: " + cameraId
239             );
240         camera = android.hardware.Camera.open(cameraId);
241     }
242
243     Camera.Parameters params = camera.getParameters();
244     List<Camera.Size> supportedSizes = params.
245         getSupportedPictureSizes();
246     int max = -1, maxW = -1;
247     for (int i = 0; i < supportedSizes.size(); i++) {
248         if (maxW < supportedSizes.get(i).width) {
249             max = i;
250             maxW = supportedSizes.get(i).width;
251         }
252         Log.d(TAG, supportedSizes.get(i).height + "x" +
253             supportedSizes.get(i).width);
254     }
255
256     params.setPictureSize(supportedSizes.get(max).width,
257         supportedSizes.get(max).height);
258     camera.setParameters(params);
259 }
260
261 // Поиск камеры
262 private int findFrontFacingCamera() {
263     int cameraId = -1;
264
265     // Поиск Фронтальной камеры
266     int numberofCameras = android.hardware.Camera.
267         getNumberOfCameras();
268     for (int i = 0; i < numberofCameras; i++) {
269         android.hardware.Camera.CameraInfo info = new android.
270             hardware.Camera.CameraInfo();
271         android.hardware.Camera.getCameraInfo(i, info);
272         // CAMERA_FACING_BACK

```

```

269     // CAMERA_FACING_FRONT
270     if (info.facing == Camera.CameraInfo.CAMERA_FACING_FRONT
271         )
272         cameraId = i;
273         break;
274     } else {
275         if (info.facing == Camera.CameraInfo.
276             CAMERA_FACING_BACK) {
277             cameraId = i;
278             break;
279         }
280         // Возвращаем id найденной камеры
281     return cameraId;
282 }
283
284 @Override
285 protected void onPause() {
286     if (camera != null) {
287         camera.release();
288         camera = null;
289     }
290     super.onPause();
291 }
292
293 @Override
294 protected void onDestroy() {
295     try {
296         if (s != null)
297             s.close();
298     } catch (IOException e) {
299         e.printStackTrace();
300     }
301     super.onDestroy();
302 }
303 }
```

Листинг 9. PhotoHandler

```

1 package pro.andreyn.pythonclient;
2
3 import android.hardware.Camera;
4 import android.util.Log;
5
6 import java.io.DataOutputStream;
```

```
7 import java.io.OutputStream;
8 import java.net.Socket;
9
10 import static pro.andreyn.pythonclient.MainActivity.TAG;
11
12 public class PhotoHandler implements Camera.PictureCallback {
13
14     private Socket socket;
15
16     public PhotoHandler(Socket socket) {
17         this.socket = socket;
18     }
19
20     // Onnpasna fomoepeafuu
21     @Override
22     public void onPictureTaken(final byte[] data, Camera camera) {
23
24
25         new Thread(new Runnable() {
26             @Override
27             public void run() {
28                 try {
29                     OutputStream out = socket.getOutputStream();
30                     DataOutputStream dos = new DataOutputStream(out);
31                     ;
32
33                     dos.write(data);
34                     dos.flush();
35                     out.flush();
36
37                     dos.write("ENDED".getBytes());
38                     dos.flush();
39                     out.flush();
40                     Log.d(TAG, "PictureSEND: TRUE");
41                     MainActivity.handler.obtainMessage(1, "image
42                         send").sendToTarget();
43                 } catch (Exception e) {
44                     Log.d(TAG, "PictureSEND: FALSE");
45                     e.printStackTrace();
46                 }
47             }
48         }).start();
49     }
}
```