**«Санкт-Петербургский государственный электротехнический университет**

**«ЛЭТИ» им. В.И.Ульянова (Ленина)»**

**(СПбГЭТУ «ЛЭТИ»)**

|  |  |  |
| --- | --- | --- |
| **Направление** | 09.04.04 – Программная инженерия | |
| **Программа** | Разработка распределенных программных систем | |
| **Факультет** | КТИ | |
| **Кафедра** | МО ЭВМ | |
| *К защите допустить* |  | |
| Зав. кафедрой |  | Кринкин К. В. |

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

магистра

Тема: Разработка алгоритма распознавания объекта и определения его координат по изображению

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент |  |  |  | Левшин П. И. |
|  |  |  |  |  |
| Руководитель |  |  |  | Чернокульский В. В. |
|  |  |  |  |  |
| Консультанты | к.э.н., доцент |  |  | Ичкитидзе Ю. Р. |
|  |  |  |  |  |
|  | к.т.н., доцент |  |  | Родионов С. В. |
|  |  |  |  |  |

Санкт-Петербург

2019**ЗАДАНИЕ**

**на выпускную квалификационную работу**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Кринкин К. В. |
|  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Студент | Левшин П. И. | | | |  | Группа | 3303 | |
| Тема работы: Разработка алгоритма распознавания объекта и определения его координат по изображению | | | | | | | | |
| Место выполнения ВКР: СПБГЭТУ «ЛЭТИ», кафедра МО ЭВМ | | | | | | | | |
| Исходные данные (технические требования): Ubuntu, python, OpenCV | | | | | | | | |
| Содержание ВКР: аналитическая часть, моделирование компонентов разрабатываемого ПО, результаты разработки, составление бизнес-плана по коммерциализации результатов НИР магистранта | | | | | | | | |
| Перечень отчетных материалов: пояснительная записка, иллюстративный материал | | | | | | | | |
| Дополнительные разделы: Составление бизнес-плана по коммерциализации результатов НИР магистранта | | | | | | | | |
| Дата выдачи задания | | | Дата представления ВКР к защите | | | | | |
| «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. | | | | | |
| Студент | |  | | Левшин П. И. | | | |
| Руководитель | |  | | Чернокульский В. В. | | | |
|  | |  | |  | | | |
| Консультанты к.э.н., доцент | |  | | Ичкитидзе Ю. Р. | | | |
|  | |  | |  | | | |
| к.т.н., доцент | |  | | Родионов С. В. | | | |

**календарный план выполнения**

**выпускной квалификационной работы**

|  |  |
| --- | --- |
|  | Утверждаю |
|  | Зав. кафедрой МО ЭВМ |
|  | \_\_\_\_\_\_\_\_\_\_\_\_ Кринкин К. В. |
|  | «\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_ г. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Левшин П. И. |  | Группа | 3303 |
| Тема работы: Разработка алгоритма распознавания объекта и определения его координат по изображению | | | | |

|  |  |  |
| --- | --- | --- |
| № п/п | Наименование работ | Срок выполнения |
| 1 | Обзор литературы по теме работы | 06.02 – 07.03 |
| 2 | Аналитическая часть | 07.03 – 21.03 |
| 3 | Моделирование компонентов разрабатываемого ПО | 07.03 – 04.04 |
| 4 | Результаты разработки | 04.04 – 18.04 |
| 5 | Составление бизнес-плана по коммерциализации результатов НИР магистранта | 18.04 – 02.05 |
| 6 | Оформление пояснительной записки | 02.05 – 18.05 |
| 7 | Оформление иллюстративного материала | 18.05 – 22.05 |

|  |  |  |
| --- | --- | --- |
| Студент |  | Левшин П. И. |
| Руководитель |  | Чернокульский В. В. |
|  |  |  |

**РЕФЕРАТ**

Пояснительная записка 65 стр., 19 рис., 8 табл., 11 ист., 12 прил.

ПОИСК ОБЪЕКТОВ, КЛАССИФИКАЦИЯ ОБЪЕКТОВ, КОНТУРНЫЙ АНАЛИЗ, РАЗРАБОТКА ПО

Объектом работы является распознавание и классификация объектов на изображении.

Предметом является алгоритм распознавания объектов и определения его координат по изображению.

Цель работы – разработать алгоритм распознавания объектов и определения их координат по изображению.

Согласно выдвинутым требованиям, были проанализированы изображения, поступающие на анализ, выявляющий общие характеристики объектов на изображении и среды, в которой эти объекты находятся, а также принято решения оформления алгоритма в виде конфигурируемой библиотеки.

На основе этого анализа был сформулирован алгоритм поиска прямоугольных объектов на изображениях с помощью контурного анализа, а также алгоритм классификация найденных объектов по трем основным выявленным признакам, отличающим объекты друг от друга.

Были спроектированы модели действия каждого компонента разрабатываемой библиотеки. На основе полученных моделей, были разработаны все компоненты библиотеки.

В качестве дополнительного раздела было рассмотрено составление бизнес-плана по коммерциализации результатов. В нем были рассмотрены такие разделы как: анализ рынка, план производства, финансовый план и сделаны выводы на основании полученных данных.

**ABSTRACT**

The object of the work is the recognition and classification of objects in the image.

The subject is an algorithm for recognizing objects and determining its coordinates from the image.

The aim of the work is to develop an algorithm for recognizing objects and determining their coordinates from the image.

According to the proposed requirements, the images were analyzed for analysis, revealing the General characteristics of the objects in the image and the environment in which these objects are located, and decisions were made to design the algorithm in the form of a configurable library.

Based on this analysis, an algorithm for searching rectangular objects in images using contour analysis was developed, as well as an algorithm for classifying the found objects by three main identified features that distinguish objects from each other.

Models of action of each component of the developed library were developed. According to the developed models, all components of the library were developed.

As an additional section, the preparation of a business plan for the commercialization of the results was considered. It considered such sections as: market analysis, production plan, financial plan and made conclusions based on the data.

**СОДЕРЖАНИЕ**

[Определения, обозначения и сокращения 10](#_Toc9082868)

[Введение 11](#_Toc9082869)

[1. Аналитический раздел 13](#_Toc9082870)

[1.1. Мотивация к разработке программного обеспечения 13](#_Toc9082871)

[1.2. Требования предъявляемые к программному обеспечению 14](#_Toc9082872)

[1.3. Анализ требований 15](#_Toc9082873)

[1.3.1. Язык программирования 15](#_Toc9082874)

[1.3.2. Структура программного обеспечения 16](#_Toc9082875)

[1.3.3. Выявления характеристик продукции и его окружения на изображении для задачи поиска 17](#_Toc9082876)

[1.3.4. Выявление характеристик продукции для задачи классифицирования 19](#_Toc9082877)

[1.4. Исследование возможностей контурного анализа в OpenCV 20](#_Toc9082878)

[1.4.1 Подготовка изображения 21](#_Toc9082879)

[1.4.2 Применение функции findContours и поиск прямоугольников минимальной площади 22](#_Toc9082880)

[1.4.3 Получение прямоугольных областей с исходного изображения 25](#_Toc9082881)

[1.4.4. Алгоритм поиска прямоугольных объектов с помощью библиотеки OpenCV 27](#_Toc9082882)

[1.5. Целесообразность использования нейронных сетей в контексте рассматриваемой задачи 27](#_Toc9082883)

[1.6. Алгоритм классификации объектов 28](#_Toc9082884)

[1.6.1. Распознавание штрихкода на изображении 30](#_Toc9082885)

[1.6.2. Вычисление цветового признака изображения 34](#_Toc9082886)

[1.6.3. Сравнение вычисленного цветового признака с эталонными значениями классов 34](#_Toc9082887)

[1.6.4. Сравнение признака размера с эталонными значениями классов 36](#_Toc9082888)

[1.7. Окончательная структура программного обеспечения 37](#_Toc9082889)

[1.8. Выводы по разделу 38](#_Toc9082890)

[2. Моделирование компонентов разрабатываемого программного обеспечения 39](#_Toc9082891)

[2.1. Построение модели алгоритма поиска объектов на изображении 39](#_Toc9082892)

[2.2. Построение модели алгоритма выделения прямоугольной области с изображения 40](#_Toc9082893)

[2.3. Построение моделей алгоритмов вычисления признаков по изображению 41](#_Toc9082894)

[2.3.1. Модель алгоритма распознавания штрихкода по изображению 42](#_Toc9082895)

[2.3.2. Модель алгоритма вычисления цветового признака 43](#_Toc9082896)

[2.4. Построение модели алгоритма классификации объекта 43](#_Toc9082897)

[2.4.1. Модель алгоритма определения класса объекта по штрихкоду 45](#_Toc9082898)

[Алгоритм получает на вход: 45](#_Toc9082899)

[2.4.2. Модель алгоритма вычисления класса объекта по цветовому признаку 46](#_Toc9082900)

[2.4.2. Модель алгоритма определения класса объекта по признаку размера 46](#_Toc9082901)

[2.5 Модель общего алгоритма распознавания объектов на изображении и определения их координат 47](#_Toc9082902)

[2.6. Построение модели алгоритма создания и загрузки конфигурации программного обеспечения 48](#_Toc9082903)

[2.7. Выводы по разделу 49](#_Toc9082904)

[3. Результаты разработки 51](#_Toc9082905)

[3.1. Выбор технических средств 51](#_Toc9082906)

[3.2. Модуль поиска объектов на изображении 51](#_Toc9082907)

[3.3. Модуль вспомогательных функций 53](#_Toc9082908)

[3.3.1 Функция выделения прямоугольных областей с изображения crop\_item 53](#_Toc9082909)

[3.3.2 Функция выделения штрихкода на изображении search\_barcode 54](#_Toc9082910)

[3.3.3 Функция вычисления цветового параметра поп изображению 55](#_Toc9082911)

[3.4. Модуль классификации объектов classify 56](#_Toc9082912)

[3.4.1. Функция классификации classify\_item 56](#_Toc9082913)

[3.4.2. Функция классификации по штрихкоду classify\_barcode 57](#_Toc9082914)

[3.4.3. Функции классификации по цветовому признаку и признаку размера 57](#_Toc9082915)

[3.5. Модуль детектирования объектов 57](#_Toc9082916)

[3.6. Модуль конфигурации 58](#_Toc9082917)

[3.8. Использование библиотеки 59](#_Toc9082918)

[3.9. Тестирование разработанного программного обеспечения 59](#_Toc9082919)

[3.10. Выводы по разделу 59](#_Toc9082920)

[4. Составление бизнес-плана по коммерциализации результатов НИР 60](#_Toc9082921)

[4.1. Резюме 60](#_Toc9082922)

[4.2. Описание продукции 60](#_Toc9082923)

[4.2.1. Характеристики продукции 61](#_Toc9082924)

[4.2.2. Потребительские свойства 62](#_Toc9082925)

[4.2.3. Конкурентные преимущества продукции 63](#_Toc9082926)

[4.3. Анализ рынка сбыта 63](#_Toc9082927)

[4.4. План производства 63](#_Toc9082928)

[4.4.1. Расчет расходов на оплату труда 63](#_Toc9082929)

[4.4.2. Расчет накладных расходов 66](#_Toc9082930)

[4.4.3. Расходы по статье «Материалы» 66](#_Toc9082931)

[4.4.4. Издержки на амортизацию персонального компьютера и оргтехники 67](#_Toc9082932)

[4.4.5. Прочие прямые расходы 68](#_Toc9082933)

[4.4.6. Себестоимость выполнения ВКР 69](#_Toc9082934)

[4.4.7. Объем продаж 69](#_Toc9082935)

[4.5 Финансовый план 70](#_Toc9082936)

[4.5.1. План прибылей и убытков 70](#_Toc9082937)

[4.6. Выводы по разделу 72](#_Toc9082938)

[Заключение 73](#_Toc9082939)

[Список использованных источников 74](#_Toc9082940)

[ПРИЛОЖЕНИЕ А. Листинг кода эксперимента по выбору праметра упаковки контуров в findContours 76](#_Toc9082941)

[ПРИЛОЖЕНИЕ Б. Листинг кода эксперимента по анализу производительности поиска штрихкода 78](#_Toc9082942)

[ПРИЛОЖЕНИЕ В. Листинг кода эксперимента по выбору оптимального количество точек на изображении 79](#_Toc9082943)

[ПРИЛОЖЕНИЕ Г. Листинг кода модуля поиска объектов на изображении 81](#_Toc9082944)

[ПРИЛОЖЕНИЕ Д. Листинг кода функции выделения прямоугольных объектов с изображения. 84](#_Toc9082945)

[ПРИЛОЖЕНИЕ Е. Листинг кода функции выделения штрихкода на изображении search\_barcode 85](#_Toc9082946)

[ПРИЛОЖЕНИЕ Ж. Листинг кода функции вычисления цветового параметра поп изображению 88](#_Toc9082947)

[ПРИЛОЖЕНИЕ И. Листинг кода модуля классификации объектов 89](#_Toc9082948)

[ПРИЛОЖЕНИЕ К. Листинг кода функции детектирования объектов 92](#_Toc9082949)

[ПРИЛОЖЕНИЕ Л. Листинг кода функции получения конфигурации 94](#_Toc9082950)

Определения, обозначения и сокращения

В настоящей пояснительной записке применяют следующие термины с соответствующими определениями:

Вендинг (англ. Vending) — это продажа товаров и услуг с помощью автоматизированных систем (торговых автоматов).

ROS (Robot Operating System) – это гибкая основа для написания программного обеспечения робота. Это набор инструментов, библиотек и соглашений, направленных на упрощение задачи создания сложного и надежного поведения роботов на самых разных роботизированных платформах.

Снек – это в англоязычных странах общее название лёгких блюд, предназначенных для утоления голода между основными приёмами пищи.

EAN (European Article Number) — европейский стандарт штрихкода, предназначенный для кодирования идентификатора товара и производителя

Введение

В настоящее время активно происходит автоматизация различных производств с помощью передачи различным приборам и автоматическим устройствам с целью сокращения доли рабочих, занятых на разных его этапах, а также для значительного повышения производительности труда.

Одним из пример такого производства является расфасовка продукции для вендинговых автоматов. Продукцией в данном случае являются различные снеки и батончики.

Фирма, которая занимается обслуживанием вендинговых автоматов, планирует провести автоматизацию процесса расфасовки продукции, который на данный момент выполняется рабочими вручную.

Для автоматизации будет использован робот-манипулятор с фотокамерой. Изображения с фотокамеры проанализированы на наличия объектов на нем и классификации этих объектов.

**Целью** данной выпускной квалификационной работы является разработка алгоритма поиска объектов и их классификации по изображению.

Под поиском объектов подразумевается выделение на изображении прямоугольных областей минимальной площади, на которых расположен тот или иной объекта, а также определение реальных координат объекта относительно расположения объекта на изображении.

Под классификацией объектов подразумевается присвоение объекту того или иного класса. Классом является конкретное наименование продукции, например: “KitKat Trio”, “Mars Max”, “Twix”.

Для достижения цели необходимо решить следующие задачи:

* проанализировать требования к разрабатываемому программному продукту;
* проанализировать предметную область и изображения, поступающие на обработку;
* на основе анализа изображений сформулировать алгоритмы поиска и классификации;
* cоставить структуру разрабатываемого продукта и построить модель каждого её компонента, обеспечивающую выполнение всех требований к программному обеспечению;
* разработать программное обеспечение согласно составленной структуре и моделей ее компонентов;
* Протестировать разработанное программное обеспечение

**1. Аналитический раздел**

1.1. Мотивация к разработке программного обеспечения

Создаваемое программное обеспечение разрабатывается по запросу компании, занимающейся обслуживанием вендинговых автоматов.

В обслуживание входит: контроль над количеством продукции, загрузка новой продукции и выгрузка непроданной продукции из вендинговых автоматов.

В качестве продукции выступают снеки или шоколадные батончики.

После выгрузки непроданной продукции из вендингового автомата, ее необходимо сортировать по наименованию продукции. В данный момент для сортировки продукции используется ручной труд. Компания планирует провести автоматизацию данного процесса, заменив ручной труд на конвейер с роботом-манипулятором с фотокамерой.

На рис. 1.1 изображена диаграмма последовательности автоматизированного процесса сортировки продукции.

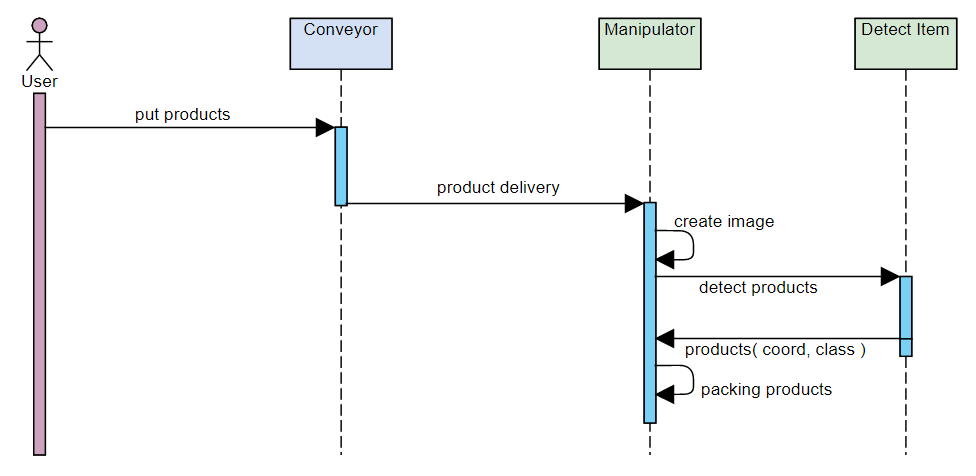


Рисунок 1.1 – Диаграмма последовательности автоматизированного процесса сортировки продукции

Рабочий загружает продукцию на конвейер, который доставляет продукцию к роботу-манипулятору. Робот-манипулятор, оснащенный фотокамерой, получает изображение конвейера с расположенными на нем продуктами. Далее изображение должно быть обработано для получения координат продуктов и их классификацию для дальнейшей расфасовки их роботом-манипулятором.

Под классами подразумеваются конкретные наименования продуктов, “Snickers Super”, “Bounty Trio”, “Kitkat” и так далее.

В данной работе описывается разработка программного обеспечения для детектирования продукции на изображении и их классификация.

1.2. Требования, предъявляемые к программному обеспечению

К разрабатываемому программному продукту предъявлены следующие требования:

* Программное обеспечение должно быть реализовано на одном из языков программирования, поддерживаемом операционной системой для управления роботами ROS.
* Обработка изображения должна быть произведена с помощью библиотеки OpenCV.
* Программное обеспечение должно быть оформлено в виде библиотеки, содержащей следующие основные модули:
  + модуль определения координат продукта на изображении;
  + модуль классификации найденных на изображении продуктов.
* Проанализировать продукты и выявить признаки определяющие, тот или иной продут.
* Программное обеспечение должно иметь возможность добавления новых классов продуктов, не требующую длительной подготовки для анализа.
* Программный продукт должен иметь возможность логирования хода обработки изображения и сохранения промежуточных этапов обработки на файловую систему.
* Программное обеспечение должно иметь следующие возможности настройки:
  + должна быть возможность использовать каждый модуль отдельно от других или совместно;
  + при классификации продуктов должна быть возможность указать, какие признаки анализировать, а какие нет;
  + должна быть возможность включения/отключения логирования хода обработки изображения.
* Программа должна тщательно быть задокументирована.

1.3. Анализ требований

В данном разделе будет проведен анализ требований к программному обеспечению.

1.3.1. Язык программирования

Основными языками программирования разработки под ROS являются python и C++. Для них написано большое количество модулей, а также они хорошо задокументированы.

В табл. 1.1 сведены плюсы и минусы использования языков python и C++. [2]

Таблица 1.1. – Анализ языков программирования ROS

|  |  |  |
| --- | --- | --- |
| **Метод** | **Языки** | |
| **python** | **C++** |
| Плюсы | * Быстрое построение прототипа * Быстрое изучение * В среднем программы короче * Проще прочитать и понять исходный код | * Работает намного быстрее * Нахождение ошибок на этапе компиляции |

Продолжение таблицы 1.1.

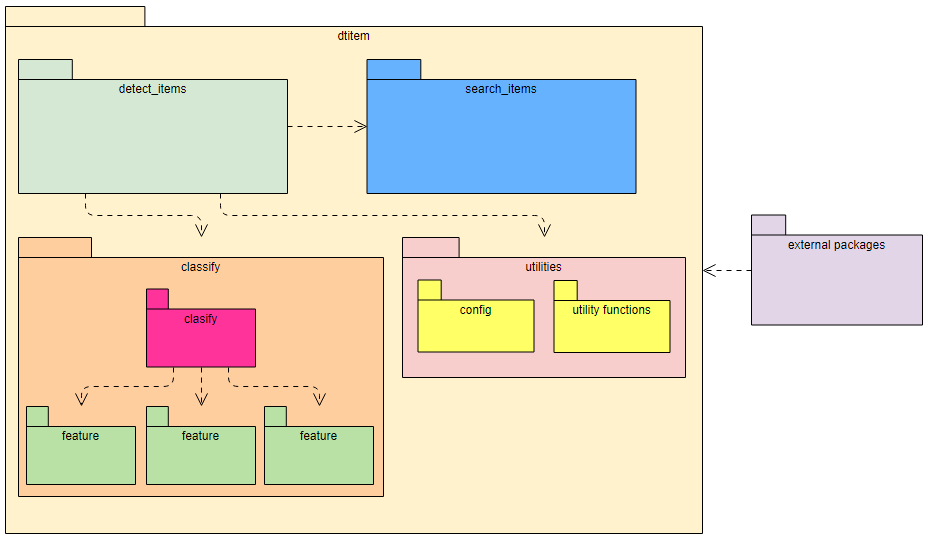
|  |  |  |
| --- | --- | --- |
| Общее | Оба имеют большое количество библиотек | |
| Минусы | * Работает медленнее * Нахождение ошибок на этапе выполнения | * Намного сложнее в изучении * Сложнее понять код |

На основе изложенных выше плюсов и минусов был выбран язык python, так как удобство и скорость разработки предпочтительнее.

1.3.2. Структура программного обеспечения

В соответствии с требованиями программное обеспечение должно быть оформлено в виде библиотеки, поделенной на модули.

Так как был выбран язык python, то программное обеспечение будет оформлено в виде пакета. На рис. 1.2 представлена внутренняя структура пакета.

  
Рисунок 1.2 – Внутренняя структура пакета

На представленной выше диаграмме пакетов можно увидеть четыре основных модуля:

* search\_items – модуль выделения продуктов на изображении;
* classify – модуль классификации продуктов. Модуль содержит подмодуль classify, который производит классификацию продуктов на основе признаков, выявленных с помощью подмодулей feature;
* utilities – модуль, содержащий подмодуль config, отвечающий за возможность настройки библиотеки и модуль utility\_functions, который содержит вспомогательные функции;
* detect\_items – модуль, позволяющий запустить процесс обработки изображения с возможность настройки.

1.3.3. Выявления характеристик продукции и его окружения на изображении для задачи поиска

В качестве продуктов выступают снеки. Основное свойство, объединяющее их от других объектов – это их прямоугольная форма, которая удобна для упаковки данной продукции в коробки для транспортировки.

На рис. 1.3 представлен пример распознаваемого продукта.

Рисунок 1.3 – Пример распознаваемого продукта

Так как робот-манипулятор фотографирует продукцию на конвейере, то фон изображению будет однотонным и прямоугольные объекты на таком фоне будут хорошо различимы, что позволит их легче отделить от основного изображения.

Также важно понимать, что в данном случае ось фотокамеры всегда перпендикулярна плоскости конвейера, что не влечет за собой искажение формы объектов.

На рис 1.4 представлен пример обрабатываемого изображения.

Таким образом, можно сделать вывод, что задача поиска объектов на изображении сводится к поиску прямоугольников, т.к. объекты других форм на изображения попасть не могут.

Рисунок 1.4 – Внутренняя структура пакета

Исходя из того, что задача сводится к поиску объектов определенной формы, то целесообразно использовать методы контурного анализа.

Методы контурного анализа будут рассмотрены далее.

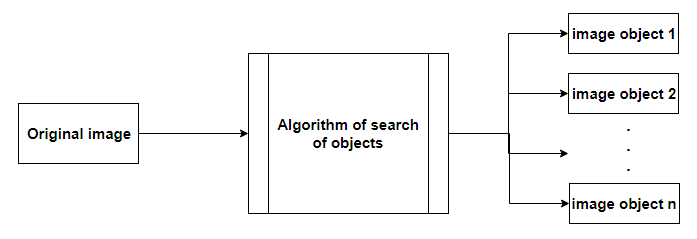
 На рис. 1.5 представлена схема, определяющая входные и выходные данные алгоритма поиска объектов на изображении.

Рисунок 1.5 – входные и выходные данные алгоритма поиска объектов

Входом для алгоритма является изображение с объектами, а выходом – прямоугольные фрагменты минимальной площади, полученные из исходного изображения, каждый из которых полностью вмещает в себя каждый найденный объект.

1.3.4. Выявление характеристик продукции для задачи классифицирования

В данной задаче классификации классами выступают конкретные наименования продуктов.

Для задачи классификации объектов необходимо выделить признаки, которые отличают друг от друга.

Основной признак каждого продукта это штрихкод, который представлен в формате EAN13, представляющем собой тринадцатиразрядную цифровую последовательность. [3] Штрихкод можно назвать идентификатором того или иного продукта и четко сопоставляется с тем или иным классом (наименованием).

Следующие два признака, отличающие продукты друг относительно друга имеют визуальных характер – это цвет и размер (ширина, длина).

Каждый класс, которым может быть отмечен объект, должен иметь в себе эталонную информацию о штрихкоде, цвете, размерах.

В свою очередь, для каждого объекта, который должен быть классифицирован, необходимо вычислить значения признаков.

Таким образом, задача классификации сводится к сравнению вычисленных признаков объекта с эталонными признаками каждого класса.

На рис. 1.6 представлена схема, определяющая входные и выходные данные алгоритма классификации.

На вход поступает изображение объекта и список классов с эталонными значениями следующих признаков: штрихкод, цвет, размер.

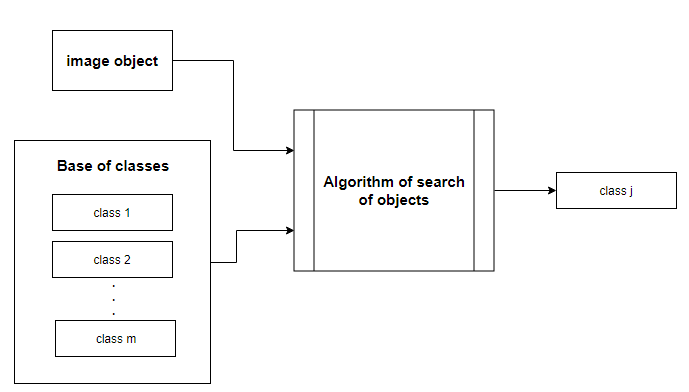


Рисунок 1.6 – входные и выходные данные алгоритма классификации объектов

1.4. Исследование возможностей контурного анализа в OpenCV

Согласно анализу обрабатываемых изображений, рассматриваемому в разделе 1.3.3., было выявлено, что искомые объекты имеют прямоугольную форму и располагаются на однотонном фоне, что позволяет применить алгоритмы контурного анализа.

Контурный анализ заключается в поиске контуров объектов на изображении.

Контурный анализ состоит из следующих частей: методы выделения, методы описания и методы преобразования контуров на изображениях. Форма полностью определяется контуром объекта на изображения. В контуре содержится вся информация, необходимая для распознавания объектов на изображении по их форме. [4]

OpenCV является библиотекой, которая содержит в себе ряд алгоритмов компьютерного зрения, алгоритмов обработки изображений. Данная библиотека является OpenSource проектом с открытым исходным кодом. Библиотека реализована на языке программировании C/C++, но существуют официальные версии библиотеки для таких языков программирования как: Python, Java, Ruby, Matlb, Lua и других. Важным плюсом данной библиотеки является то, что она может свободно использоваться в академических и коммерческих целях, так как распространяется в условиях лицензии BSD.

Библиотека OpenCV имеет в своем арсенале функцию findContours. [5] Функция принимает на вход три основных аргумента: изображение, режим группировки и метод упаковки.

Так как алгоритм также должен вычислить координаты объектов, то на вход алгоритму также передается коэффициент масштабирования, определяющий соотношение длин на изображении с реальными длинами объектов. В качестве координат объекта будет пониматься центр этого объекта, что в точности соответствует центру найденного прямоугольника.

Обобщенный ход алгоритма при применении данной функции:

1. Подготовить изображение.
2. Применить функцию findContours.
3. Для каждого найденного контура определить прямоугольник минимальной площади, который полностью его вмещает.
4. Вырезать из исходного изображения области, определенные найденными прямоугольниками.
5. Вычисление реальных размеров и центра объекта умножением размеров вырезанных областей на коэффициент масштаба.

1.4.1 Подготовка изображения

Для применения функции findContours, необходимо подготовить изображение, так как функция принимает на вход монохромное изображение, которое содержит в себе информацию о границах объектов на изображении, которые функция впоследствии объединит в контуры объектов.

Границы – это линии на изображении, в которых происходит резкое изменение яркости.

Чтобы получить монохромное изображение необходимо использовать оператор Кэнни.

Оператор Кэнни в дисциплине компьютерного зрения — оператор обнаружения границ изображения. Был разработан в 1986 году Джоном Кэнни и использует многоступенчатый алгоритм для обнаружения широкого спектра границ в изображениях.

Подготовка изображения:

1. Сжать изображение для уменьшения количества обрабатываемой информации.
2. Преобразовать изображение их трехканальной цветовой модели RGB в одноканальную цветовую модель.
3. Применить оператор Кэнни к одноканальному изображению для получения монохромного изображения, содержащего информацию о границах объектов.

1.4.2 Применение функции findContours и поиск прямоугольников минимальной площади

Помимо изображения функция принимает на вход еще два аргумента: режим группировки, метод упаковки.

Режим группировки, отвечает за то, какие границы искать и то, как найденные границы группируются друг относительно друга. Этот параметр может принимать следующие значения:

* CV\_RETR\_LIST — при данном значении функция выдает все найденные контуры и никак их не группирует между собой;
* CV\_RETR\_EXTERNAL — при данном значении функция выдает только внешние контуры объектов. Например, если на изображении будет тор, то функция вернет только внешний контур, не учитывая внутренний вырез;
* CV\_RETR\_CCOMP — при данном значении функция выдает все найденные контуры, а также группирует их в двухуровневую иерархию. На первом уровне находятся только внешние контуры, а на втором находятся внутренние, например контуры отверстий;
* CV\_RETR\_TREE — при данном значении функция выдает все найденный контуры, но при этом группирует контуры в многоуровневую иерархию, согласно вложенности контуров друг в друга.

Учитывая специфику задачи, необходимо выбрать значение CV\_RETR\_EXTERNAL, так как важны только внешние границы искомых объектов и не важна их внутренняя структура.

Метод упаковки, отвечает за то, как контуры хранятся в памяти, хранится ли каждая точка контура или происходит упаковка точек для уменьшения количестао памяти, занимаемой контуром. Этот параметр может принимать следующие значения:

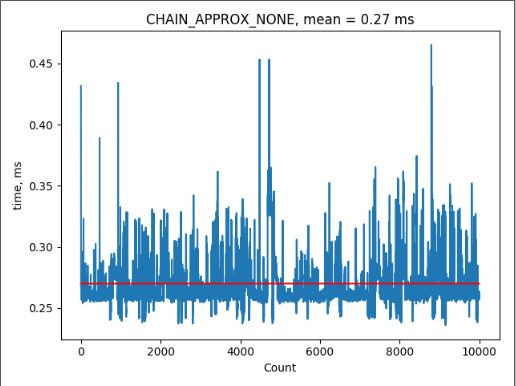
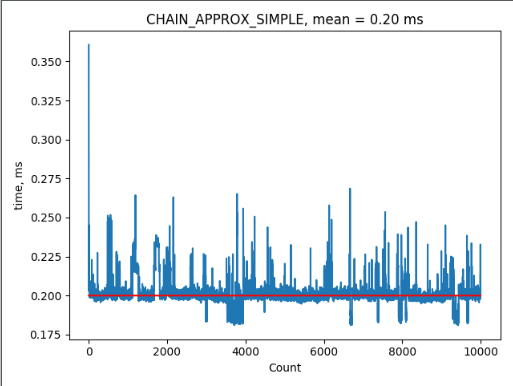
* CV\_CHAIN\_APPROX\_NONE — при данном значении функция никак не упаковывает контуры. В данном случае, контуры являются списком отрезков, состоящих из двух точек изображения;
* CV\_CHAIN\_APPROX\_SIMPLE — при данном значении функция группирует отрезки контур. Находящиеся рядом вертикальные, горизонтальные, и диагональные рядом стоящие отрезки объединяются в один отрезок;
* CV\_CHAIN\_APPROX\_TC89\_L1, CV\_CHAIN\_APPROX\_TC89\_KCOS — при данном значении функция применяет к найденным контурам метод упаковки Teh-Chin, который заключается в поиске доминирующих точек на кривых. [TEG-CHIN]

Данный параметр напрямую влияет на поиск минимального прямоугольника, так как на этом шаге идет перебор всех точек контура. С одной стороны значения для третьего параметра CV\_CHAIN\_APPROX\_TC89\_L1, CV\_CHAIN\_APPROX\_TC89\_KCOS позволят получить контуры с минимальным количеством точек, но в тоже время требуют значительных преобразования исходных контуров, что может повлиять на производительность.

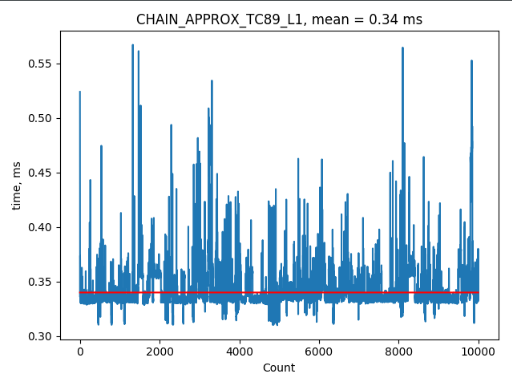
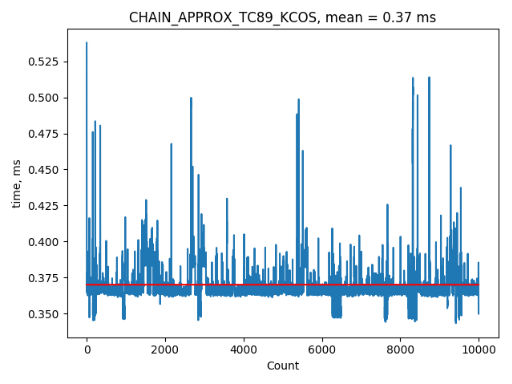
Для определения оптимального значения параметра был проведен эксперимент. Эксперимент состоял из следующих шагов:

* Подготовка изображения к поиску контуров согласно разделу 1.4.1.
* Последовательное применение функций findContours и minAreaRect, так же входящей в состав библиотеки OpenCV.

Результаты эксперимента показаны на рис. 1.7.

(a) (b)

(c) (d)

Рисунок 1.7 – Результаты эксперимента по выбору оптимального параметра упаковки контуров: (a) CHAIN\_APPROX\_NONE,

(b) CHAIN\_APPROX\_SIMPLE, (c) CV\_CHAIN\_APPROX\_TC89\_L1,

(d) CV\_CHAIN\_APPROX\_TC89\_KCOS

Согласно результатам эксперимента самым оптимальным является значение CV\_CHAIN\_APPROX\_SIMPLE. Результаты объясняются тем, что при значении CV\_CHAIN\_APPROX\_NONE функция findContours не упаковывает полученные контуры. Поэтому функция minAreaRect перебирает большое количество точек, в тоже время при значениях параметра CV\_CHAIN\_APPROX\_TC89\_L1, CV\_CHAIN\_APPROX\_TC89\_KCOS функция тратит большое количество времени на преобразование полученных контуров.

Листинг кода эксперимента находится в приложении А.

1.4.3 Получение прямоугольных областей с исходного изображения

Одним из последних этапов поиска объектов на изображении является выделение прямоугольных областей, полученных на предыдущем шаге.

Так как объекты на изображении могут быть расположены не параллельно сторонам изображение, то необходимо применить к исходному изображению аффинные преобразования.

Аффинное преобразование – отображение плоскости или пространства в себя, при котором параллельные прямые переходят в параллельные прямые, пересекающиеся — в пересекающиеся, скрещивающиеся — в скрещивающиеся.

Необходимые нам аффинные преобразования: параллельный перенос и поворот вокруг точки.

С предыдущего шага алгоритма нам известен центр прямоугольника, а также угол поворота относительно горизонтальной оси.

Пусть центр прямоугольника, высота и ширина прямоугольника соответственно, а также угол поворота относительно горизонтальной оси.

Пусть ширина и высота изображения соответственно.

Для применения аффинных преобразований стоит рассмотреть так называемые однородные точки. Если есть точка , то ее однородная равна . Для удобства .

Первый шаг – это параллельный перенос. Необходимо перенести центр прямоугольника в начало координат на вектор . Тогда параллельный перенос любой точки в однородных координатах производится по формуле (1.1)

(1.1)

Следующий шаг – поворот на заданный угол. Необходимо повернуть изображения относительно начала координат на угол . Поворот любой точки относительно начала координат в однородных координатах производится по формуле (1.2).

(1.2)

Общую формулу преобразования можно получить, объединив формулы (1.1) и (1.2). Общая формула с дополнительным переносом в центр координат показана в формуле (1.3).

(1.3)

Тогда точке в однородных координатах соответствует точка, определенная формулой (1.4).

(1.4)

Произведя соответствующие преобразования, остается только вырезать прямоугольную область размера с центром в точке .

1.4.4. Алгоритм поиска прямоугольных объектов с помощью библиотеки OpenCV

Исходя из разделов 1.4.1 и 1.4.2, был сформирован алгоритм поиска прямоугольных объектов:

1. Сжать изображение для уменьшения количества обрабатываемой информации.
2. Преобразовать исходное изображение из трехканальной цветовой модели RGB в одноканальную цветовую модель.
3. Применение оператора Кэнни.
4. Применение функции findContours с параметрами CV\_RETR\_EXTERNAL и CV\_CHAIN\_APPROX\_SIMPLE.
5. Применение функции minAreaRect для нахождения прямоугольников минимальной площади, вмещающих в себя найденные контуры.
6. Вырезать из исходного изображения области, определенные найденными прямоугольниками.
7. Вычисление реальных размеров и центра объекта умножением размеров вырезанных областей на коэффициент масштаба.

1.5. Целесообразность использования нейронных сетей в контексте рассматриваемой задачи

Искусственная нейронная сеть является математической моделью, построенной согласно принципу организации и функционирования биологических нейронных сетей, состоящих из нервных клеток живого организма.

На данный момент это один из самых популярных подходов в области обработки изображений. Нейронные сети позволяют качественно классифицировать объекты на изображении по заранее обученным классам.

В данной задаче, поставленной фирмой вендинговых автоматов, количество классов на которые надо будет натренировать нейронную сеть боле двадцати, что ведет к огромным затратам на подготовку данных и обучение.

Также важным моментом является то, что могут появляться новые продукты для автоматов, что ведет к появлению новых классов и соответственно требует заново обучать нейронную сеть.

Таким образом, выявлено, что использование нейронных сетей нецелесообразно.

1.6. Алгоритм классификации объектов

В соответствии с анализом характеристик классифицируемых объектов, как показано в разделе 1.3.2, основными признаками, определяющими класс, являются: штрихкод, цвет, размер.

Данные признаки не равносильны, так как штрихкод является признаком, четко определяющим тот или иной класс продукта, но из-за человеческого фактора объект может быть расположен на конвейере штрихкодом вниз, что не позволит его распознать.

Для таких случаев необходимо провести сравнения цвета и размеров объекта, которые в совокупности являются хорошими признаки, отличающими класс от другого, но следует учитывать, что цвет является более важным, чем размер.

Чтобы продемонстрировать, что признаки цвета и размера не равносильны, был проведен эксперимент. Были взяты объекты, являющиеся представителями самых популярных марок, и проведены вычисления их размера и цвета. Затем были вычислены коэффициенты вариации для обоих признаков.

Вариация — различие значений какого-либо признака у разных единиц совокупности (набора).

Коэффициент вариации — это отношение среднеквадратического отклонения к среднеарифметическому, рассчитывается в процентах. Коэффициент необходим для сравнения вариаций наборов, измеряемых разными шкалами. [6]

Вычисление коэффициентов вариации производится по формуле (1.5).

, (1.5)

где – коэффициент вариации признака ;

– среднеквадратичное отклонение признака;

– среднее значение признака .

Результаты эксперимента сведены в таблице 1.2.

Среднеквадратичные отклонения цвета и размера вычислены по формуле (1.6).

, (1.6)

где – среднеквадратичное отклонение признака;

– количество объектов в наборе;

– значение признака объекта ;

– среднее значение признака .

Средние значения признаков цвета и размера вычислены по формуле (1.7).

, (1.7)

где – среднее значение признака .

– количество объектов в наборе;

– значение признака объекта ;

По результатам эксперимента было выявлено, что коэффициент вариации цвета выше, чем коэффициент вариации размера. Это показывает, что разброс цвета исследуемых продуктов выше, чем разброс размера объектов.

Таблица 1.2. – Результаты эксперимента по вычислению коэффициента

|  |  |  |
| --- | --- | --- |
| **Продукт** | **Признаки** | |
| **Цвет (R, G, B)** | **Размер (ширина, длина), см** |
| Snickers Super | (88; 92; 107) | (18,2; 4,5) |
| Twix | (52; 75; 111) | (16; 8) |
| Twix Max | (75; 97; 131) | (18,5; 8) |
| KitKat | (91; 92; 134) | (14; 4) |
| KitKat Maxi | (98; 101; 136) | (18; 4) |
| Bounty | (131; 129; 126) | (16; 4) |
| Mars Max | (102; 116; 127) | (18,5; 4,5) |
|  | (91; 100; 124) | (17,03; 5,29) |
|  | 64,34 | 2,37 |
|  | 34,71 | 13,39 |

Таким образом, между признаками были расставлены приоритеты: Штрихкод, цвет, размер.

Алгоритм сводится к последовательному вычислению признаков и сравнению их с эталонными значениями.

Так как штрихкод является идентификатором продукта, то при его успешном распознавании вычисление остальных признаков нецелесообразно. Если штрихкод распознать не удалось, то следует учитывать цвет и размер.

1.6.1. Распознавание штрихкода на изображении

Алгоритм распознавания штрихкода основан на форме штрихкода. Все продукты маркируются штрихкодов стандарта EAN13, который кодируется 13 цифрами.

Данный штрих код является линейным, состоящим из параллельных линий разной толщины с разными промежутками между ними. На рис. 1.8 представлен пример штрихкода одного из исследуемых продуктов.

Преобразование самого штрихкода в цифровую последовательность будет производиться с помощью библиотеки pyzbar, одной из самых популярных библиотек по распознаванию штрихкодов различных типов.



Рисунок 1.8 – Штрихкод снека “SNICKERS SUPER”

На рис. 1.9 представлен пример изображения, поступающего на обработку.

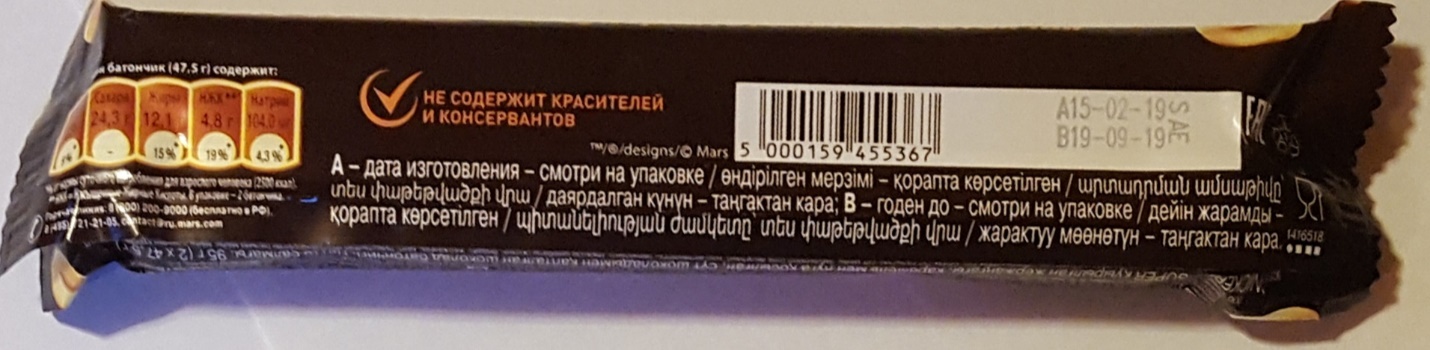


Рисунок 1.9 – Штрихкод снека “SNICKERS SUPER”

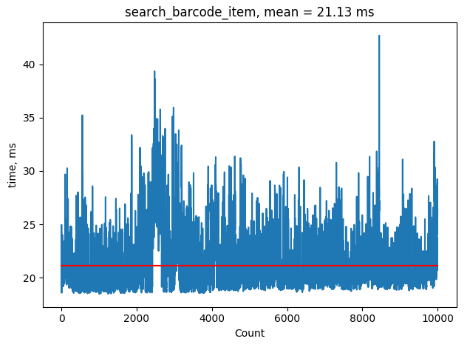
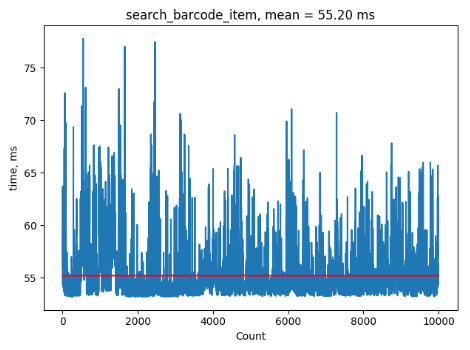
Для уменьшения времени распознавания штрихкода библиотекой pyzbar, его необходимо выделить с изображения. Данное решение принято на основе эксперимента, сравнения производительности распознавания по исходному изображению и по выделенному изображению. Результаты эксперимента отображены на рис. 1.10.

Листинг кода эксперимент указан в приложении Б.

Проанализировав множество изображений, полученных после алгоритма поиска, был сделан вывод, что штрихкод всегда располагается параллельно одной из осей. Что позволяет применить оператор Собеля.

Оператор Собеля — дискретный дифференциальный оператор, вычисляющий приближённое значение градиента яркости изображения.

Данный оператор необходимо применить к одноканальному изображению, поэтому исходное изображение необходимо преобразовать из трехканальной цветовой модели RGB в одноканальную модель.

(a) (b)

Рисунок 1.10 – Результаты сравнительного эксперимента

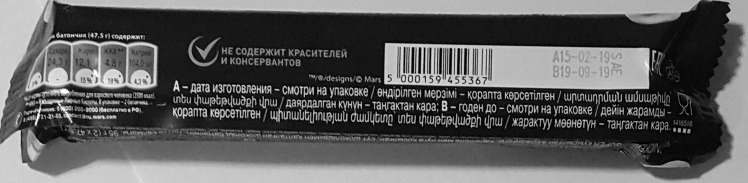
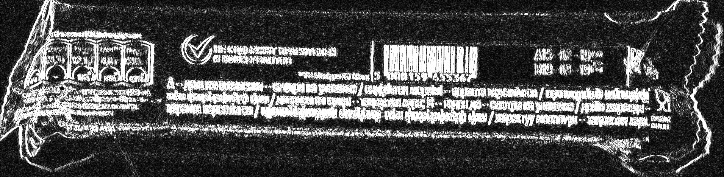
(а) с выделением штрихкода, (b) без выделения штрихкода

Данный оператор необходимо применить к одноканальному изображению, поэтому исходное изображение необходимо преобразовать из трехканальной цветовой модели RGB в одноканальную модель.

Далее необходимо убрать шум на изображении с помощью функции threshold, которая обнуляет точки изображения, значение которых меньше некоторого заданного значения.

На рис. 1.11 представлены преобразования цветовой модели и применение оператора Собеля над изображением с рис. 1.9.

Следующим шагом необходимо убрать лишние тонкие элементы с изображения. Для этого выполняются морфологические преобразования:

* Закрытие. Позволяет объединить близко стоящие фрагменты белого цвета, что позволит объединить полосы, из которых состоит штрихкод, в один большой общий блок
* Эрозия. Позволяет уменьшить фрагменты белого цвета, в результате чего большинство мелких элементов пропадет с изображения.
* Дилатация. Позволяет увеличить фрагменты белого цвета, в результате чего область штрихкода вернет свои изначальные размеры, которые были до операции эрозии.

(a) (b)

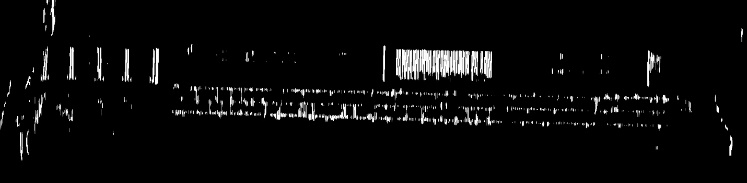
Рисунок 1.11 – Преобразования изображения:

(a) в одноканальную модель, (b) применение оператора Собеля

Данные преобразования позволят убрать тонкие элементы с изображения, в тоже время, оставив область штрихкода неизменной.

Последним шагом необходимо выполнить поиск контуров и прямоугольников минимальной площади, аналогично разделу 1.4. На изображении могут остаться несколько таких прямоугольников. Необходимо найти прямоугольник с максимальной площадью.

Пример преобразований над изображением с рис. 1.11(b) представлен на рис. 1.12.

(a) (b)

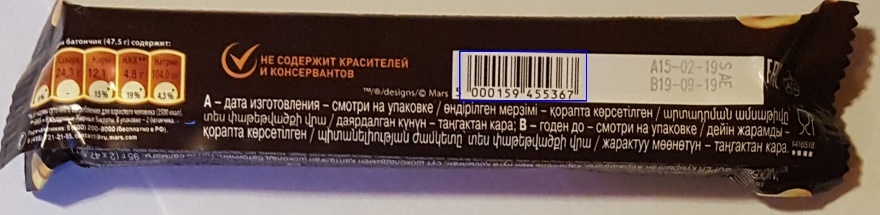
(c)

Рисунок 1.12 – Преобразования изображения:

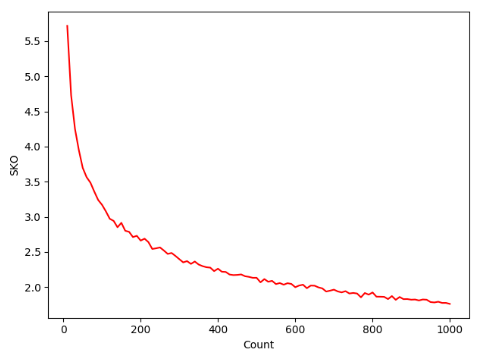
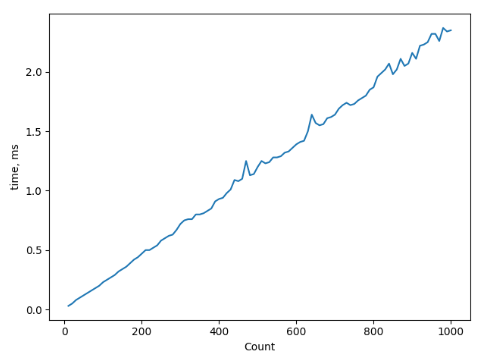
(a) фильтрация шума, (b) морфологические преобразования, (c) результат

1.6.2. Вычисление цветового признака изображения

Вычисление цветового признака производится на основе вычисления среднего значения цветов точек, взятых с изображения случайным образом с равномерным распределением.

Для получения количества точек, которых необходимо подсчитать для достижения достаточной точности и производительности вычислений был поставлен эксперимент.

В эксперименте были проведены расчеты времени вычисления и дисперсии получаемых цветов для количества точек от 10 до 10000. Дисперсия цветового признака рассчитана по формуле (1.6). Результаты эксперимента продемонстрированы на рис. 1.13.

(a) (b)

Рисунок 1.13 – Подбор количества точек в расчете цвета:

(a) зависимость среднеквадратичного отклонения от количества точек

(b) зависимость времени выполнения вычисления от количества точек

Эксперимент показал, что время вычислений растет линейно, а дисперсия падет по гиперболе, что означает, что нет смысла в количестве большем 1000.

Листинг кода эксперимента указан в приложении В.

1.6.3. Сравнение вычисленного цветового признака с эталонными значениями классов

Рассмотрев значение цветового признака, как точку в трехмерном пространстве, введем меру близости цветов. Мера близости рассчитывается, как декартово расстояние по формуле (1.8).

(1.8)

где – мера близости;

– значение соответствующего канал цвета.

Пусть имеется набор классов , каждый из которых имеет свое эталонное значение цветового параметра .

Пусть имеется вычисленное по изображению объекта значение цветового признака.

Тогда номер класса для объекта на изображении вычисляется по формуле (1.9)

(1.9)

Таким образом, производится классификация по цветовому признаку.

Также стоит учитывать, что могут быть классы, которые имеют близкие значения цветового признака, например для продуктов “Kitkat” и “Kitkat Super”. При таком случае необходима дополнительная проверка признака размера. Для того чтобы точно ли объект соответствует классу с минимальным значением меры близости, необходимо вычислить коэффициенты сходства вычисленные значения для каждого класса.

Имея множество декартовых расстояний полученных для каждого класса по формуле (1.8), можно получить коэффициенты масштабирования, показывающие во сколько раз мера близости того или иного класса выше минимальной меры близости, соответствующей классу , посчитанному по формуле (1.9). Коэффициент масштабирования рассчитывается по формуле (1.10).

, (1.10)

где – коэффициент масштабирования;

– значение меры близости класса ;

– минимальное значение меры близости, соответствующее классу .

Заметим, что для класса значение коэффициента масштабирования равно 1;

Имея множество коэффициентов масштабирования для каждого класса, можно вычислить коэффициенты сходства, решив систему уравнений (1.11):

(1.11)

Решение уравнения (1.11) представлено в системе (1.12).

(1.12)

Таким образом, были вычислены коэффициенты сходства цветового признака относительно эталонных цветовых признаков каждого класса .

Наибольший коэффициент сходства соответствует классу с наименьшей мерой близости.

Взяв отношение двух самых больших коэффициентов, можно принять решение, необходима ли проверка признака размера, сравнив его с заранее заданным пороговым значением.

1.6.4. Сравнение признака размера с эталонными значениями классов

Для признака размера аналогично цветовому признаку, рассмотренному в разделе 1.5.3, вводится мера близости, которая рассчитывается в двумерном евклидовом пространстве по формуле (1.13).

(1.13)

где – мера близости;

– ширина, высота объекта.

Пусть – эталонный признак размера для класса, а значение признака класса, вычисленного по изображению.

Тогда номер лучшего класса определяется по формуле (1.9), а коэффициента сходства по формулам (1.10) и (1.12).

1.7. Окончательная **структура программного обеспечения**

В соответствии с требованиями, предъявленными к программному обеспечению и анализом задач поиска и классификации, была доработана структура программного обеспечения и сформулирован алгоритм использования данного программного обеспечения. Структура программного обеспечения показана на рис. 1.14.

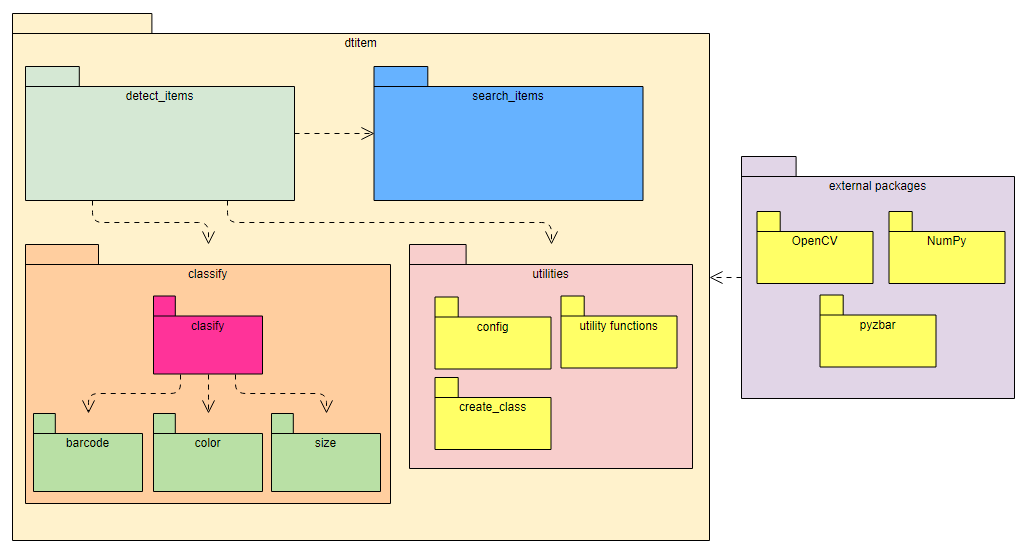


Рисунок 1.14 – Структура программного обеспечения

Порядок использования разрабатываемого программного обеспечения:

1. Подготовка стенда. Заключается в установке фотокамеры в месте, в котором она будет использоваться.
2. Подготовка классов продуктов. Заключается в создании базы классов, каждый из которых содержит: наименование, штрихкод, цвет, размеры.
3. Использование по назначению, описанному в разделе 1.1

1.8. Выводы по разделу

В данном разделе были проанализированы технические требования к разрабатываемому программному обеспечению.

Был проведен анализ изображений, поступающих на обработку, из которого был сделан вывод, что оптимальным решением задачи поиска объектов на изображениях, будет контурный анализ, основанный на форме искомых объектов.

Также были проанализированы исходные объекты для выявления признаков, отличающих их друг от друга для задачи классификации. Были выявлены следующие признаки: штрихкод, цвет, размер. Для каждого из признаков были сформулированы алгоритмы их вычисления.

В итоге была разработана структура разрабатываемого программного обеспечения и алгоритм его использования.

2. Моделирование компонентов разрабатываемого программного обеспечения

2.1. Построение модели алгоритма поиска объектов на изображении

На основе анализа, проведенного в разделе 1.4, была разработана модель действия алгоритма поиска объектов прямоугольной формы на изображении.

На рис. 2.1 представлена диаграмма действий алгоритма поиска объектов на изображении.

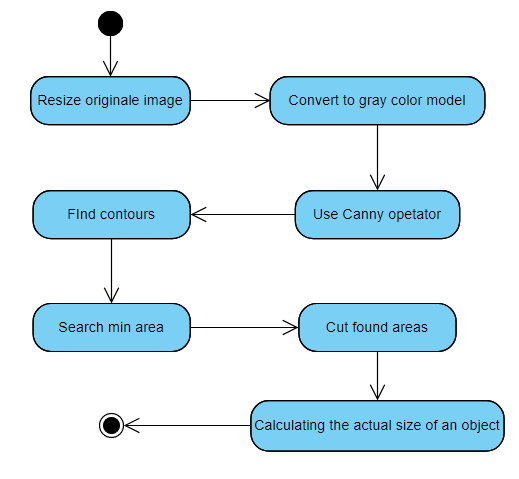


Рисунок 2.1 – Диаграмма действий алгоритма поиска объектов на изображении

Алгоритм принимает на вход следующие аргументы:

* исходное изображение для анализа;
* параметры, зависящие от фотокамеры:
  + коэффициент сжатия.
  + минимальное и максимальное пороговые значения оператора Кэнни;
  + минимальный порог площади прямоугольника;

Выходом алгоритма является набор областей с исходного изображения, соответствующих найденным объектам.

Вначале алгоритм производит уменьшение исходного изображения. Затем производится преобразование трехканальной цветовой модели RGB в одноканальную цветовую модель. Следующим шагом является применение оператора Кэнни. И наконец производится поиск контуров с последующим поиском прямоугольников минимальной площади для каждого найденного контура. На основе полученных прямоугольников вычисляются и

2.2. Построение модели алгоритма выделения прямоугольной области с изображения

На основе анализа, проведенного в разделе 1.4.3, была разработана модель действия алгоритма выделения прямоугольной области на изображении.

На рис. 2.2 представлена диаграмма действий алгоритма выделения.

Алгоритм принимает на вход следующие аргументы:

* исходное изображение (размером ;
* прямоугольник, содержащий:
  + ширину и высоту прямоугольника ;
  + центр прямоугольника ;
  + угол поворота прямоугольника относительно горизонтальной оси координат.

Алгоритм принимает на вход следующие аргументы:

* исходное изображение (размером ;
* прямоугольник, содержащий:
  + ширину и высоту прямоугольника ;
  + центр прямоугольника ;
  + угол поворота прямоугольника относительно горизонтальной оси координат.

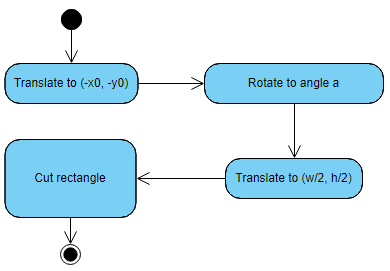


Рисунок 2.2 – Диаграмма действий алгоритма выделения прямоугольной области с изображения

Выходом алгоритма является область исходного изображения, соответствующая заданному прямоугольнику.

Алгоритм производит три последовательных аффинных преобразования:

* параллельный перенос на вектор ;
* поворот относительно центра координат на угол ;
* параллельный перенос на вектор (в центр изображения).

Далее производится вырезание прямоугольной области с центром в точке и размером .

2.3. Построение моделей алгоритмов вычисления признаков по изображению

На основе анализа признаков объектов, рассмотренного в пункте 1.3.4, и рассмотренных алгоритмов вычисления признаков, описанных в разделах 1.6.1 и 1.6.2, сформированы модели алгоритмов вычисления признаков штрихкода, цвета и размера.

2.3.1. Модель алгоритма распознавания штрихкода по изображению

На основе алгоритма, рассмотренного в пункте 1.6.1, была сформирована диаграмма действий алгоритма распознавания штрихкода по изображению.

На рис. 2.3 представлена диаграмма действий алгоритма распознавания штрихкода по изображению.

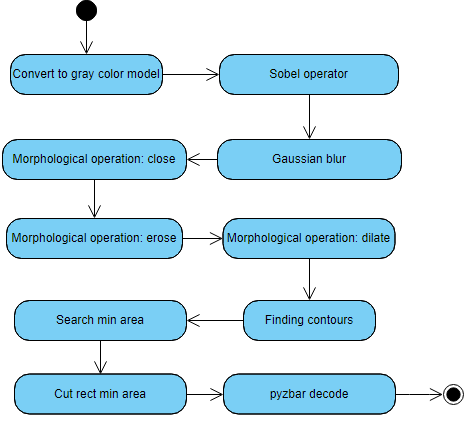


Рисунок 2.3 – Диаграмма действий алгоритма распознавания штрихкода по изображению

Алгоритм принимает на вход:

* изображение объекта, полученное от алгоритма поиска прямоугольных объектов;
* расширение границ найденного штрихкода;
* параметры, зависящие от фотокамеры:
  + ядро размытия Гаусса;
  + минимальный и максимальный порог обнуления шума;
  + ядро морфологической операции закрытия;
  + количество итераций морфологических операций эрозии и дилатации;

Выходом алгоритма является строка из 13 символов цифр, соответствующая формату EAN13.

2.3.2. Модель алгоритма вычисления цветового признака

На рис. 2.4. представлена диаграмма действия алгоритма вычисления цветового признака по изображению.

Алгоритм получает на вход:

* – количество итераций.

Выходом алгоритма является среднее значение признака цвета, вычисленное по изображению, взятием случайным образом по равномерному закону точек.

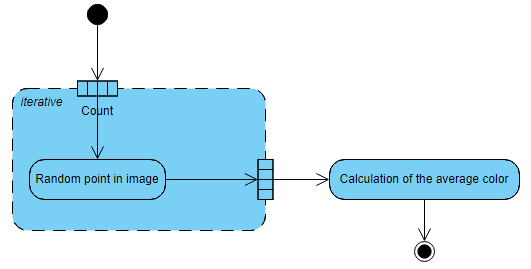


Рисунок 2.4 – Диаграмма действий алгоритма вычисления цветового признака по изображению

2.4. Построение модели алгоритма классификации объекта

На основе анализа, проведенного в разделе 1.6, была разработана модель действия алгоритма классификации объекта по трем признакам: штрихкод, цвет, размер.

На рис. 2.5 представлена диаграмма действий алгоритма классификации объекта.

Алгоритм получает на вход:

* штрихкод, полученный с помощью алгоритма, описанного в разделе 2.3.1;
* цветовой признак, полученный с помощью алгоритма, описанного в разделе 2.3.2;
* признак размера, полученный с помощью алгоритма поиска объектов на изображении, описанного в разделе 2.1;
* список классов с эталонными значениями признаков.

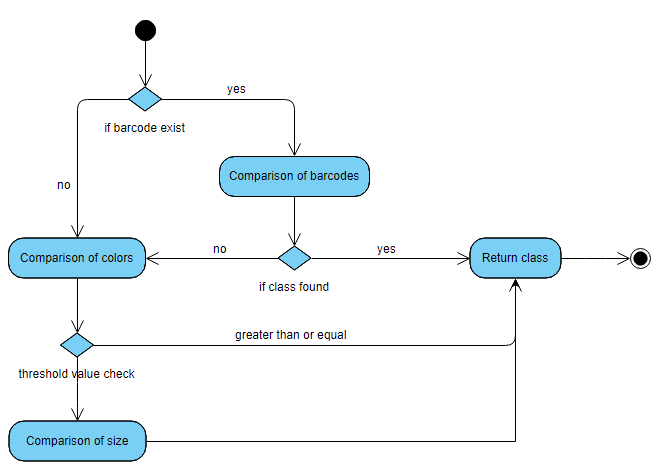


Рисунок 2.5 – Диаграмма действий алгоритма классификации объекта

Выходом алгоритма является номер класса, присвоенный объекту.

Данный алгоритм начинается с проверки распознавания штрихкоды. В случае, если штрихкод распознан, то проверяется определение класса по штрихкоду. Если класс определен, то алгоритм завершает действие. В случае, если штрихкод не распознан или не найден класс с таким штрихкодом, то начинается проверка цветового признака.

Вычислив коэффициенты сходства цветового параметра, производится сравнение значения отношения двух самых наибольших коэффициентов с пороговым значением. Если отношение больше или равно порогового значения, то алгоритм возвращает класс, имеющий наибольшее значение коэффициента сходства. Если отношение меньше порогового значения, то производится сравнение признака размера.

Рассчитав коэффициенты сходства по признаку размера, вычисляются средние значения коэффициентов сходства между признаками цвета и размера. Результатом является класс с наибольшим коэффициентом сходства.

2.4.1. Модель алгоритма определения класса объекта по штрихкоду

На рис. 2.6 представлена диаграмма действий алгоритма определения класса объекта по штрихкоду.

Алгоритм получает на вход:

* штрихкод, вычисленный по изображению;
* список классов с соответствующими эталонными значениями штрихкодов.

Выходом алгоритма является, либо номер класса, либо None, если класс не найден.

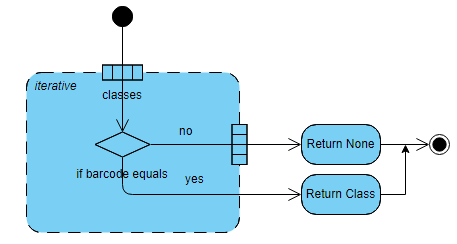


Рисунок 2.6 – Диаграмма действий алгоритма определения класса объекта по штрихкоду

Алгоритм заключается в последовательном обходе списка классов с проверкой равенства штрихкода с изображения и штрихкода класса на текущей итерации. Если штрихкоды равны, то возвращается номер класса текущей итерации, иначе происходит переход на следующую итерацию. Если список пройден, и класс не определен, алгоритм возвращает None.

2.4.2. Модель алгоритма вычисления класса объекта по цветовому признаку

На основе анализа, проведенного в разделе 1.6.3, была разработана модель действия алгоритма определения класса объекта по цветовому признаку.

На рис. 2.7 представлена диаграмма действий алгоритма определения класса объекта по цветовому признаку.

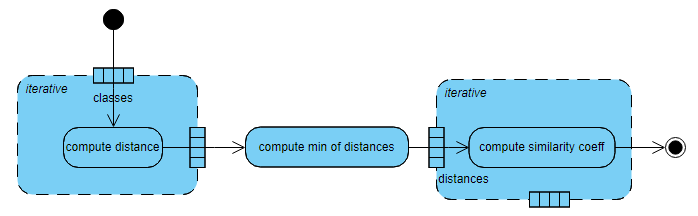


Рисунок 2.7 – Диаграмма действий алгоритма определения класса объекта по цветовому признаку

Алгоритм получает на вход:

* цветовой признак, вычисленный по изображению;
* список классов с соответствующими эталонными значениями цветового признака.

Выходом алгоритма является список классов с соответствующими коэффициентами сходства. Приоритетным классом для анализируемого объекта является класс с наибольшим коэффициентом сходства.

Первым шагом алгоритма является вычисление мер близости для каждого класса по формуле (1.8). Затем производит вычисление коэффициентов сходства по формулам (1.10) и (1.12).

2.4.2. Модель алгоритма определения класса объекта по признаку размера

Модель действий алгоритма вычисления класса объекта по признаку размера аналогична модели представленной на рис. 2.7. За исключением того, что вычисление мер близости производится по формуле (2.1).

, (2.1)

где функция, описанная формулой (1.13);

размерный обратный размеру , т.е. если , то .

Выбор данной функции основан на том, что объект на изображении может быть сориентирован не по той же оси, по которой сориентирован размер проверяемого класса.

2.5 Модель общего алгоритма распознавания объектов на изображении и определения их координат

На основе моделей, описанных в разделах 2.1 – 2.5, была сформирована модель действия общего алгоритма распознавания объектов на изображении и определения их координат.

На рис. 2.8 представлена диаграмма действия общего алгоритма распознавания объектов на изображении и определения их координат.

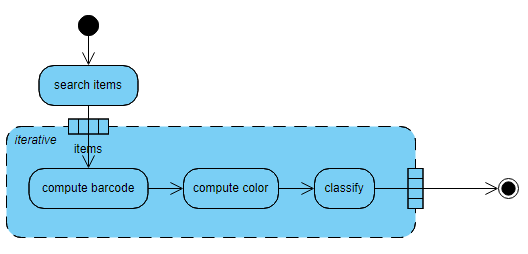


Рисунок 2.8 – Диаграмма действия общего алгоритма распознавания объектов на изображении и определения их координат

Алгоритм принимает на вход:

* исходное изображение для анализа;
* список классов с эталонными значениями признаков: штрихкод, цвет, размер;
* дополнительные параметры для внутренних подмодулей, описанных в предыдущих разделах.

Выходом алгоритма является список найденных на изображении предметов с присвоенными им классами.

Первым шагом алгоритма является поиск прямоугольных объектов на исходном изображении с помощью алгоритма, описанного в разделе 2.1. Затем для каждого найденного объекта на изображении:

* вычисляются значения штрихкода и цветового признака по алгоритмам, описанным в разделах 2.3.1 и 2.3.2;
* производится классификация по алгоритму, описанному в разделе 2.4.

2.6. Построение модели алгоритма создания и загрузки конфигурации программного обеспечения

Одним из требований к разработке программного обеспечения была возможность конфигурирования алгоритма. Помимо основных аргументов алгоритма таких как: исходное изображения и список классов, есть параметры, зависящие от качества фотографий, которые необходимо подбирать под каждый случай отдельно.

Было решено вынести данные параметры в отдельный конфигурационный файл. Алгоритм может быть запущен с конфигурационным файлом по умолчанию или с файлом, переданным пользователем.

На рис. 2.9 представлена модель действий алгоритма создания и загрузки конфигурации.

Алгоритм принимает на вход:

* путь к файлу конфигурации с настройками расположенный в файловой системе.

Выходом алгоритма является конфигурационный объект, содержащий в себе необходимые параметры.

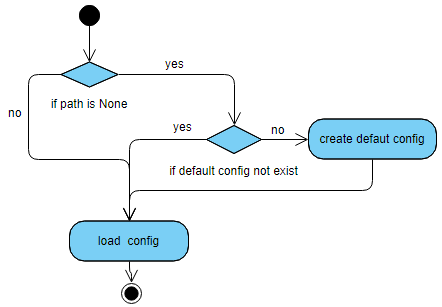


Рисунок 2.9 – Диаграмма действий алгоритма создания и загрузки конфигурации

Если алгоритму был передан пустой путь, то производится проверка существования конфигурационного файла с настройками по умолчанию. В случае существования данного файла он загружается и возвращается в качестве результата, иначе в файловой системе создается конфигурационный файл по умолчанию, загружается и возвращается качестве результата.

Если алгоритму был передан путь к существующему файлу, то производится загрузка данного файла и возврат его в качестве результата.

2.7. Выводы по разделу

В данном разделе были разработаны модели действий алгоритма поиска прямоугольных объектов на изображении и алгоритма классификации объектов по трем признакам: штрихкод, цвет, размер.

На основе этих моделей была разработана модель общего алгоритма распознавания объектов на изображении и определения их координат

С учетом того, что алгоритма должен иметь возможность конфигурации параметров, требуемых для его составных частей был сформулирована и разработана модель действий алгоритма создания и загрузки файла конфигурации программного обеспечения.

Каждая модель представлена диаграммой действий построенная с унифицированного языка моделирования.

3. Результаты разработки

На основе моделей, описанных в разделе 2, были разработаны все компоненты разрабатываемого программного обеспечения. Далее рассматриваются основные моменты разработки каждого компонента ПО.

3.1. Выбор технических средств

В соответствии с требованиями каждый компонент разработан на языке программирования python.

Для обработки изображений используется библиотека OpenCV.

Для работы с многомерными массивами используется библиотека NumPy, так как библиотека OpenCV представляется изображения в формате двумерных массивов точек в формате NumPy, поэтому для совместимости была выбрана эта библиотека.

Для распознавания штрихкода используется библиотека pyzbar.

Для работы с графиками используется библиотека mathplotlib.

Для реализации логирования используется встроенный модуль logging.

Для работы с конфигурационными файлами используется встроенный модуль ConfigParser.

3.2. Модуль поиска объектов на изображении

Согласно модели, описанной в разделе 2.1, был разработан модуль поиска объектов на изображении.

Модуль содержит в себе импорт основных требуемых библиотек: OpenCv, NumPy, logging, ConfigParser, os, utills. Модуль содержит одну функцию – search\_items.

Функция search\_items сначала проверят файл конфигурации, и выгружает из него необходимые для алгоритма настройки с помощью библиотеки ConfigParser. Затем она выполняет шаги алгоритма описанного на рис. 2.1. Для каждого шага алгоритма, если было указано в файле конфигурации:

* добавляется запись в лог, который сохраняется в файл c помощью библиотеки logging;
* сохраняется промежуточный результат в виде изображения в формате “.jpg”;
* отображение промежуточного результата на экран в режиме отладки.

Основными шагами функции являются:

* запуск функции cv.resize с параметром image\_scale\_factor, полученным из файла конфигурации, для изменения размера исходного изображения;
* запуск функции cv.cvtColor с параметром cv.COLOR\_BGR2GRAY, для конвертации цветовой модели изображения;
* запуск функции cv.Canny с параметрами lower и upper, полученных с конфигурационного файла, для преобразования оператором Кэнни;
* запуск функции cv.findContours с параметрами cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_SIMPLE, для поиска внешних контуров объектов;
* поиск прямоугольников минимальной площади для каждого найденного контура с помощью функции cv.minAreaRect;
* отсеивание прямоугольников по пороговому значению minArea, полученному с конфигурационного файла;
* преобразование оставшихся прямоугольников к исходному размеру изображения с помощью параметра image\_scale\_factor;
* Вычисление реальных размеров объекта и его центра с помощью параметра scale, полученного с конфигурационного файла;
* выделение полученных областей с исходного изображения с помощью функции crop\_item из модуля utills;
* возврат списка найденных объектов.

Полный код модуля представлен в приложении Г.

3.3. Модуль вспомогательных функций

Модуль вспомогательных функций состоит из следующих функций, который были построены согласно разделам 2.2 и 2.3:

* функция выделения прямоугольных областей с изображения crop\_item;
* функция выделения штрихкода на изображении search\_barcode;
* функция вычисления цветового признака search\_color.

Модуль содержит в себе импорт основных требуемых библиотек: OpenCv, NumPy, logging, ConfigParser. Модуль содержит одну функцию – search\_items.

3.3.1 Функция выделения прямоугольных областей с изображения crop\_item

Данная функция реализована в соответствии алгоритмом, построенным в разделе 2.2.

Основными шагами функции являются:

* запуск функции cv.warpAffine с параметром матрицы равным [[1, 0, ], [0, 1, ], для выполнения параллельного переноса в начало координат;
* запуск функции cv.warpAffine с параметром матрицы, полученным с помощью функции cv.getRotationMatrix2D для угла , для поворота изображения относительно начала координат;
* запуск функции cv.warpAffine с параметром матрицы равным [[1, 0, ], [0, 1, ]], для выполнения параллельного переноса в центр изображения;
* запуск функции cv.getRectSubPix с параметрами size, и center, полученными из параметров прямоугольника, для вырезания необходимой области с исходного изображения;
* возврат вырезанной области.

Листинг кода функции представлен в приложении Д.

3.3.2 Функция выделения штрихкода на изображении search\_barcode

Функция search\_barcode сначала проверят файл конфигурации, и выгружает из него необходимые для алгоритма настройки с помощью библиотеки ConfigParser. Затем она выполняет шаги алгоритма, описанного на рис. 2.3.1. Для каждого шага алгоритма, если было указано в файле конфигурации:

* добавляется запись в лог, который сохраняется в файл c помощью библиотеки logging;
* сохраняется промежуточный результат в виде изображения в формате “.jpg”;
* отображение промежуточного результата на экран в режиме отладки.

Основными шагами функции являются:

* запуск функции cv.cvtColor с параметром cv.COLOR\_BGR2GRAY, для конвертации цветовой модели изображения;
* запуск функции cv.Sobel, для нахождения градиента яркости на изображении;
* запуск функции cv.threshold с параметрами up и low, полученными с файла конфигурации, для очистки изображения от шума;
* запуск функции cv.morphologyEx с параметром cv.MORPH\_CLOSE и параметром kernel, полученным из конфигурационного файла, для выполнения морфологического преобразования закрытия над изображением;
* запуск функций cv.erode и cv.dilate с параметром iterations, полученным из конфигурационного файла, для выполнения морфологических преобразований эрозии и дилатации над изображением;
* запуск функции cv.findContours для получения контуров на изображении;
* для каждого найденного контура выполняется поиск прямоугольника минимальной площади с помощью функции cv.minAreaRect;
* поиск прямоугольника с максимальной площадью;
* выделение области найденного прямоугольника с исходного изображения;
* запуск функции decode из библиотеки pyzbar над изображением, полученным на предыдущем шаге;
* возврат результата;

Листинг кода функции представлен в приложении Е.

3.3.3 Функция вычисления цветового параметра поп изображению

Функция search\_barcode сначала проверят файл конфигурации, и выгружает из него необходимые для алгоритма настройки с помощью библиотеки ConfigParser.

Данная функция вычисляет цветовой параметр, вычисляя среднее значения точек на исходном изображении, выбранных с помощью функции uniform библиотеки random. Количество точек, определяется параметром count\_samples, получаемым с конфигурационного файла.

Листинг кода функции представлен в приложении Ж.

3.4. Модуль классификации объектов classify

Модуль содержит в себе импорт библиотеки NumPy и четыре функции, работающие в соответствии с разделом 2.4:

* функция классификации classify\_item;
* функция классификации по штрихкоду classify\_barcode;
* функция классификации по цветовому признаку classify\_color;
* функция классификации по признаку размера classify\_size.

Листинг с кодом данных функций представлен в приложении И.

3.4.1. Функция классификации classify\_item

Данная функция выполняет комплексную классификацию объекта по трем признакам в соответствии с моделью действий, представленной на рис. 2.5.

Функция search\_barcode сначала проверят файл конфигурации, и выгружает из него необходимые для алгоритма настройки с помощью библиотеки ConfigParser.

Основные шаги функции:

* классификацию по штрихкоду с помощью функции classify\_barcode. Если класс определен, то функция завершается, иначе переход на следующий шаг;
* классификация по цветовому признаку с помощью функции classify\_color. Из полученных коэффициентов схожести и параметра threshold определяется необходимо ли проводить классификацию по размеру. Если отношение двух наибольших коэффициентов схожести больше или равно порогу threshold, то функция завершает работу, возвращая в качестве результата класс с наибольшим коэффициентом схожести, иначе происходит классификация по цвету;
* классификация по признаку размера с помощью функции classify\_size. Коэффициенты схожести по размеру усредняются с коэффициентами схожести по цвету. Результатом является класс с наибольшим коэффициентом.

3.4.2. Функция классификации по штрихкоду classify\_barcode

Данная функция выполняет классификацию объекта по штрихкоду, модель действий, которой представлена на рис. 2.6.

Функция производит последовательное сравнение штрихкода со штрихкодами каждого класса. Если штрихкод найден, то выдается соответствующий ему класс, иначе None.

3.4.3. Функции классификации по цветовому признаку и признаку размера

Функции classify\_color и classify\_size выполняют классификацию объекта по цвету и размеру соответственно моделям действий, представленным на рис. 2.7.

Основные шаги:

* вычисление меры близости proximity для каждого класса;
* вычисление коэффициентов масштабирования scales;
* вычисление коэффициентов схожести probabilities

3.5. Модуль детектирования объектов

Модуль содержит в себе импорт библиотек: OpenCV, logging и одну функцию detect\_items, работающую в соответствии с диаграммой действий описанной на рис. 2.8.

Данная функция получает на вход изображения и набор классов и проводит полный алгоритм по поиску объектов и их классификации, а на выходе выдает список найденных объектов, промаркированных соответствующим классом.

Функция detect\_items сначала проверят файл конфигурации, и выгружает из него необходимые для алгоритма настройки с помощью библиотеки ConfigParser.

Каждый шаг функции логируется в текстовый файл с помощью библиотеки logging.

Основные шаги функции:

* выполнение функции search\_items над исходным изображением, для получения областей, на которых расположены объекты;
* для каждого найденного объекта выполняется вычисление признаков штрихкода и цвета c помощью функций search\_barcode и search\_color соответственно;
* выполняется классификация каждого объекта

Листинг с кодом функции представлен в приложении К.

3.6. Модуль конфигурации

Модуль содержит в себе импорт библиотек: ConfigParser и одну функцию get\_config, работающую в соответствии с диаграммой действий описанной на рис. 2.9.

Данная функция получает на вход путь к конфигурационному файлу в файловой системе и выдает объект с настройками. Функция проверяет существование файла по переданному пути и возвращает настройки, если такой файл существует. В случае, если файл не существует, то производится возврат настроек по умолчанию.

Листинг с кодом функции представлен в приложении Л.

Конфигурационный файл по умолчанию сформирован на основе проведенных опытов над исходными изображениями, полученными для работы над ВКР.

3.8. Использование библиотеки

Порядок использования программного обеспечения следующий:

* подготовка стенда. Необходимо вычислить параметр scale для стенда. Он определяется отношением реального размера объекта в сантиметрах к размеру объектов в пикселях;
* подготовка классов. Для этого необходимо подготовить изображения, для каждого продукта. Над каждым изображением запустить функции search\_items, search\_barcode, search\_color, что позволит вычислить все признаки изображения.
* запустить функцию detect\_items

Истин кода с пример использования библиотеки представлен в приложении М.

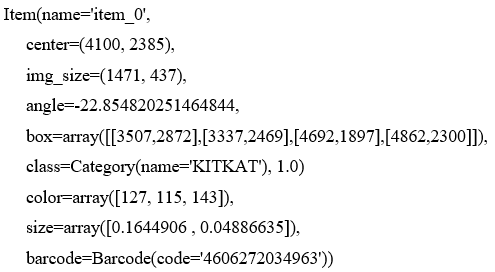
На рис. 3.1. представлен пример получаемого ответа от функции detect\_items.

Рисунок 3.1 – Пример ответа функции detect\_items

Функция detect\_item вернула список объектов Item, пример которого показан на рисунке 3.1.

На нем видны следующие параметры найденного объекта:

* center – определяет центр объекта на изображении;
* img\_size – определяет размеры области объекта в пикселях;
* angle – угол поворота относительно горизонтальной оси;
* box – координаты вершин прямоугольник, ограничивающего область объекта;
* class – идентификатор класса, присвоенный объекту;
* color – вычисленный цветовой признак;
* size – вычисленные реальные размеры объекта в метрах;
* barcode – распознанный штрихкод.

3.9. Тестирование разработанного программного обеспечения

Все функции покрыты модульными тестами. При модульном тестировании тестируются минимально возможные компоненты: отдельные классы и функции. Этим методом была проверена работоспособность функций поиска объектов search\_item, функция классификации classify\_item, функции вычисления признаков search\_barcode, search\_color, а также функция detect\_item.

В качестве примера в приложении Н приведен пример тестирования функции detect\_item с конфигурацией классификации только по признаку изображения. Данный тест необходим, т.к. цветовой признак имеет случайный характер. Тест проводился по нескольким изображения. Каждое изображение детектировалось несколько раз. В 100% случаев класс объекта был определен верно.

Также было проведено тестирование производительности детектирования. При тестировании, на исходном изображении были изображены пяти объектов. Классификация производилась по семи классам.

На рисунке 3.2. представлен график производительности детектирования объектов в случае классификации по всем признакам.

Из рисунка видно, что время обработки изображения держится на уровне 195 мс.

Рисунок 3.2 – График производительности детектирования объектов

3.10. Выводы по разделу

В данном разделе была приведена программная реализация всех разработанных компонентов библиотеки. Каждая функция была тщательно протестирована с помощью модульных тестов.

4. Составление бизнес-плана по коммерциализации результатов НИР

4.1. Резюме

Выпускная квалификационная работа описывает разработку программного продукта, позволяющего определять местоположения объекта на изображении и идентификации продукта питания (снеки, батончики).

Бизнес-план предназначен для экономического обоснования разработанного программного обеспечения.

Были рассчитаны полные затраты на проект и прибыль за первой год.

Стоимость единицы продукции – 60000 руб.

В первом квартале продажи продукции не производится. В нем идет только разработка программного продукта. В следующие 3 квартала производится продажа продукции. Первая прибыль появляется на первый квартал.

4.2. Описание продукции

В качестве продукции выступает библиотека для языка python включающая в себя модули, позволяющие определять местоположения объекта на изображении и идентифицировать продукт питания (снеки, батончики).

Согласно ГОСТ 19781 – 90:

Прикладная программа - Программа, предназначенная для решения задачи или класса задач в определенной области применения системы обработки информации. [23]

Таким образом, продукция является прикладным программным обеспечением, нацеленным на автоматизацию обработки изображений различных продуктов питания.

4.2.1. Характеристики продукции

Согласно ГОСТ Р ИСО/МЭК 9126 – 93 программная продукция должна иметь следующие характеристики [24]:

1. Функциональные требования

Данная продукция имеет следующие функциональные возможности:

* Определение координат центра масс продукта и его угол поворота относительно оси абсцисс.
* Классификация продукта методом сравнение размеров продукта
* Классификация продукта методом сравнения цвета продукта
* Классификация продукта методом поиска штрихкода
* Классификация продукта с использованием обученной нейросети.

1. Практичность

Разработанное программное обеспечение является практичным за счет ее деление на модули, в соответствии с функциональными возможностями.

Каждый модуль может использоваться отдельно от других или в совокупности для достижения более точных результатов.

Каждый модуль имеет настройки по умолчанию, с возможностью внесения изменений с помощью речного ввода настроек или с помощью конфигурационного файла.

1. Надежность

Разработанное программное обеспечение было протестировано с помощью:

* Тестирование производительности. Результаты тестирования производительности приведены в разделе 3.9.
* Модульное тестирование. Результаты модульного тестирования, приведены в разделе 3.9.

1. Эффективность

Программное обеспечение разработано для быстрого и удобного поиска продуктов питания (Снеком, Батончиков) с возможностью настройки конфигурации его модулей.

1. Сопровождаемость

Сопровождаемость достигается за счет деления библиотеки на модули, что позволяет упростить разработку новых модулей при появлении новых функциональных требований, а также за счет наличия документации.

1. Мобильность

Программный продукт разработан на языке python, что позволяет использовать на любых операционных системах, для которых реализован интерпретатор python

4.2.2. Потребительские свойства

Потребитель, при использовании разработанного программного, продукта уменьшает затраты на выпуск своего ПО из-за отсутствия необходимости разрабатывать свое решение с нуля и возможности настройки конфигурации модулей разработанного программного продукта.

4.2.3. Конкурентные преимущества продукции

На данный момент нет реализованных комплексных программных продуктов, которые имели бы тот же функционал и возможность комбинации результатов модулей.

4.3. Анализ рынка сбыта

Основными потребителя являются компании, которым по тем или иным причинам проводить анализ изображения для определения типа продукции. Например, компании обслуживающие автоматы с батончиками, для которых необходимо проводить сортировка батончиков или снеков, идущих по конвейеру.

4.4. План производства

В данном разделе производится расчёт себестоимости продукции и объема продаж

4.4.1. Расчет расходов на оплату труда

Расчет расходом затраченных на оплату труда сотрудников касающихся непосредственной разработки программного обеспечения производится на основе трудоемкости выполненных ими работ. Расходы, затраченные на оплату труда одной из основных статей расчета себестоимости разработки.

Трудоемкости рассчитывается по перечню этапов разработки программного обеспечения каждым сотрудником и оценивается человеко-днях.

Ставка каждого сотрудника в расчете на день определяется отношением средней заработной платы в месяц к числу рабочих дней в месяце.

На основе данных, взятых с сайт russia.trud.com сделан вывод, что по состоянию на 20.04.19, по профессии программист-стажер открыто 983 вакансий в России. В 49.4% вакансий, работодателями указана заработная плата в размере 0 - 22 000 руб. 35% объявлений с зарплатой 22 000 - 44 000 руб., 9.9% с зарплатой 44 000 - 66 000 руб., и 5.6% с зарплатой более 88 000 руб.

Возьмем среднюю зарплату, как взвешенную сумму зарплат по процентам открытых вакансий (4.1):

Тогда по формуле (4.2) можно рассчитать ставку заработной платы в день.

(4.2)

Согласно приложению 1 к приказу от 03.05.2017 №1372 «Размеры должностных окладов ППС с учетом повышающих квалификационных коэффициентов», размер заработной платы доцента с наличием ученой степени кандидата наук составляет 22 200 рублей [9].

Тогда по формуле (4.2) можно рассчитать ставку заработной платы в день.

(4.3)

Списки работ программиста и его руководителя представлены в табл. 4.1. и табл. 4.2. соответственно.

Таблица 4.1. – План работ программиста

|  |  |  |
| --- | --- | --- |
| **Этапы и содержание выполняемых работ** | **Трудоемкость, чел./день.** | **Ставка, руб./день** |
| Разработка ТЗ | 4 | 1302,71 |
| Анализ предметной области | 5 | 1302,71 |
| Разработка методов выделения объекта на изображении и их классифицирования | 10 | 1302,71 |
| Разработка приложения | 50 | 1302,71 |
| Отладка и тестирование программного обеспечения | 10 | 1302,71 |
| Согласование с руководителем и уточнение недочетов | 3 | 1302,71 |

Продолжение таблицы 4.1

|  |  |  |
| --- | --- | --- |
| Создание технико-экономического обоснования | 4 | 1302,71 |
| Оформление пояснительной записки | 10 | 1302,71 |
| Сдача проекта | 1 | 1302,71 |

Таблица 4.2. – План работ руководителя

|  |  |  |
| --- | --- | --- |
| **Этапы и содержание выполняемых работ** | **Трудоемкость, чел./день** | **Ставка, руб./день** |
| Разработка ТЗ | 2 | 1057,14 |
| Анализ предметной области | - | 1057,14 |
| Разработка методов выделения объекта на изображении и их классифицирования | 2 | 1057,14 |
| Согласование с руководителем и уточнение недочетов | 6 | 1057,14 |
| Сдача проекта | 1 | 1057,14 |

Основная и дополнительная заработная плата сотрудников участвующих в разработке, т.е. программиста и его руководителя, рассчитываются на основании вышеприведенных таблиц:

• Трудоемкость выполнения работ разработчика по табл. 4.1.

• Трудоемкость выполнения работ руководителя по табл. 4.2.

• Ставка разработчика и ставка руководителя указаны в табл. 4.1. и табл. 4.2. соответственно

• Процент дополнительной заработной платы 12%

• Процент отчислений на социальные нужды 30,2%

Основная заработная плата исполнителей рассчитывается по формуле (4.4).

(4.4)

Дополнительная заработная плата составляет 12% от основной заработной платы и рассчитывается по формуле (4.5):

(4.5)

Отчисления на социальные нужды составляют 30,2% от суммы основной и дополнительной заработной платы вычисляется по формуле (4.6).

(4.6)

4.4.2. Расчет накладных расходов

В рамках данной работы накладные расходы устанавливаются в размере 40% от суммы основной и дополнительной заработной платы рассчитывается по формуле (4.7).

(4.7)

4.4.3. Расходы по статье «Материалы»

Расчет количества и стоимости материалов с учетом транспортно-заготовительных расходов: на статью «Материалы» относятся расходы на основные и вспомогательные материалы и комплектующие изделия, которые могут понадобиться при выполнении разработки. Калькуляция расходов по статье «Материалы» с учетом транспортно-заготовительных расходов приведена в табл. 4.3.

Стоимость материалов определена по данным, находящимся на сайте ulmart.ru.

Таблица 4.3. – расходы по статье «Материалы»

|  |  |  |  |
| --- | --- | --- | --- |
| **Материалы** | **Кол-во** | **Цена, руб.** | **Сумма, руб.** |
| Бумага для печати, пачка | 1 | 210 | 210 |
| Картриджи для принтера  Canon G3400 (зеленый, синий, красный, черный) | 1 | 1960 | 1960 |
| Канцелярские товары |  |  | 300 |
| ИТОГО: | | | 2470 |
| Транспортные расходы (15%) | | | 370.5 |
| ВСЕГО: | | | 2840,5 |

4.4.4. Издержки на амортизацию персонального компьютера и оргтехники

Амортизация оборудования, к которому относится один персональный компьютер и принтер определяется линейным методом [10]. Для вычисления суммы амортизации за год необходимо использовать функцию (4.8).

(4.8)

где стоимость оборудования;

норма амортизации.

Стоимость используемых персональных компьютеров:

* ноутбука 70 000 руб.

Стоимость используемой оргтехники:

* принтер Canon G3400 11 720 руб.

Стоимости принтера и ноутбука взяты с сайта ulmart.ru

Далее необходимо рассчитать норму амортизации используемого оборудования.

В соответствии с постановлением правительства РФ от 01.01.2002 N 1 (ред. от 07.07.2016) "О Классификации основных средств, включаемых в амортизационные группы" персональные компьютеры и печатающие устройства входят во второй группу амортизации, то есть являются имуществом со сроком полезного действия от двух до трех лет [11]. Данное оборудование является новым и с известными производителями, что позволяет его считать надежным. Поэтому был выбран срок полезного действия – три года.

Тогда норма амортизации рассчитывается по формуле (4.9).

(4.9)

Далее произведены расчеты годовой суммы амортизации. В расчет принимались ноутбук и принтер:

За один рабочий день сумма амортизации оборудования рассчитывается по формуле (4.10).

,0 (4.10)

где количество рабочих дней в 2019 году, равное 247 дням.

Амортизация оборудования за время разработки программного обеспечения, рассматриваемого в ВКР, рассчитывается по формуле (4.11):

(4.11)

4.4.5. Прочие прямые расходы

К прочим прямым расходам относятся:

* затраты на приобретение технической литературы;
* ежемесячные затраты на платежи за использование средств связи и коммуникации (Интернет).

Смета по данной статье, включающая в себя вышеперечисленные затраты приведена в табл. 4.4.

Таблица 4.4. – прочие прямые расходы

|  |  |  |  |
| --- | --- | --- | --- |
| **Наименование** | **Кол-во** | **Цена, руб.** | **Сумма, руб.** |
| Доступ в интернет, месяц | 3 | 650 | 1950 |
| ИТОГО: | | | 1950 |

4.4.6. Себестоимость выполнения ВКР

Себестоимость разработки программного обеспечения, рассмотренного в ВКР, представлена в табл. 4.5.

Таблица 4.5. – Смета затрат на ВКР

|  |  |
| --- | --- |
| **Статья затрат** | **Сумма, руб.** |
| Материалы | 2 840,5 |
| Расходы на оплату труда | 137 991,41 |
| Дополнительная заработная плата | 16 558,96 |
| Отчисления на социальные нужды | 46 674,21 |
| Издержки на амортизацию ПК и оргтехники. | 10 696,42 |
| Прочие прямые расходы | 1950 |
| Накладные расходы | 61 820,15 |
| ИТОГО: | 278 531,65 |

Из таблицы видно, что наибольшие затраты идут на оплату труда сотрудникам.

4.4.7. Объем продаж

Программный продукт, было решено продавать по квартальной подписке стоимостью 60 000 р. Чтобы получить первую прибыль во втором квартале необходимо, с учетом себестоимости продукции, продавать 5 единиц продукции в квартал.

4.5 Финансовый план

4.5.1. План прибылей и убытков

План прибылей и убытков представляет собой финансовый документ, в котором отражаются доходы, расходы и финансовые результаты за период реализации проекта. Основная задача формирования данного документа состоит в том, чтобы рассчитать величину и структуру себестоимости продукции, соотношение затрат и результатов деятельности за определенный период, по которым можно будет судить о рентабельности проекта, его окупаемости.

Таблица 4.6. – План прибылей и убытков

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Показатели** | **Квартал** | | | | **Всего** |
| **I** | **II** | **III** | **IV** |
| Объем продаж | 0 | 5 | 7 | 10 | 22 |
| Стоимость продукции | 0 | 60 000 | 60 000 | 60 000 | 180 000 |
| Выручка-нетто (без учета НДС) от реализации, руб. | 0 | 254 237,28 | 355932,20 | 508474,57 | 1118644,06 |
| Валовая прибыль, руб. | (278 531,65) | 254 237,28 | 355932,20 | 508474,57 | 840112,41 |
| Прибыль от продаж, руб. | (278 531,65) | 254 237,28 | 355932,20 | 508474,57 | 840112,41 |
| Прибыль до налогообложения, руб. | (278 531,65) | 254 237,28 | 355932,20 | 508474,57 | 840112,41 |
| Налог на прибыль | 0 | 17796,61 | 24915,25 | 35593,22 | 78305,08 |
| Чистая прибыль, руб. | (278 531,65) | 236440,67 | 331016,93 | 472881,35 | 1040338,97 |

Налог на прибыль взят 7%, т.к. форма организации подразумевается ИП.

План прибылей и убытков составлен в соответствии с формулами (4.12), (4.13), (4.14), (4.15), (4.16).

* Выручка-нетто (без учета НДС) от реализации:

(4.12)

где V – объем реализованной продукции;

P – цена за единицу продукции.

* Валовая прибыль ()

(4.13)

где В – выручка-нетто;

С – себестоимость продукции.

* Прибыль от продаж

(4.14)

где валовая прибыль;

– коммерческие расходы;

управленческие расходы.

* Прибыль до налогообложения

(4.15)

где – прибыль от продаж;

ДО и РО - доходы и расходы от обычных видов деятельности;

прочие доходы и расходы.

* Чистая прибыль

(4.16)

где – прибыль до налогообложения;

Н – налог на прибыль.

4.6. Выводы по разделу

В результате составления бизнес-плана по коммерциализации разработки программного обеспечения была определена сумма, необходимая для разработки программного продукта, а также был рассчитан план прибылей и убытков, составлен план производства и организационный план.

Была выбрана форма организации – индивидуальное предпринимательство из-за небольшого числа людей участвующих в разработке.

Согласно производственному плану, разработка программного обеспечения ведется в первый квартал.

Согласно финансовому плану, рассчитанному на следующие три квартала. Первая прибыль поступит после второго квартала.

Себестоимость продукции составляет 278 531,65 руб.

Чистая прибыль за первый год 1040338,97 руб.

Заключение

Список использованных источников

1. About ROS. [Электронный ресурс] // https://www.ros.org/ официальный сайт Open Source Robotics Foundation: https://www.ros.org/about-ros/ (дата обращения: 07.02.2019).
2. Should I learn ROS with Python or with C++? [Электронный ресурс]// http://www.theconstructsim.com/ сайт обучения и прикладной разработки под ROS: http://www.theconstructsim.com/learn-ros-python-or-cpp/ (дата обращения: 07.02.2019).
3. ГОСТ ISO/IEC 15420-2010. Межгосударственный стандарт. Автоматическая идентификация. Кодирование штриховое. Спецификация символики штрихового кода EAN/UPC – Москва: Изд-во стандартов, 2011. – 56 с.
4. Сакович И.О., Белов Ю.С. Обзор основных методов контурного анализа

для выделения контуров движущихся объектов. Инженерный журнал: наука и

инновации, 2014, вып. 12. URL: http://engjournal.ru/catalog/it/hidden/1280.html

1. Structural Analysis and Shape Descriptors [Электронный ресурс] // https://docs.opencv.org/ официальный сайт документации библиотеки компьютерного зрения OpenCV. (дата обращения: 09.02.2019).
2. Ивченко Г. И., Медведев Ю. И. Введение в математическую статистику. — М. : Издательство ЛКИ, 2010.
3. C. The, R. Chin. On the Detection of Dominant Points on Digital Curves // IEEE transactions on pattern analysis and machine intelligence. – 1989. – Vol II, No. 8 – стр. 859
4. ГОСТ 19781 – 90. Обеспечение систем обработки информации программное. Термины и определения. – Москва: Изд-во стандартов, 1992. – 45 с.
5. ГОСТ Р ИСО/МЭК 9126 – 93. Информационная технология. Оценка программной продукции. Характеристики качества и руководства по их применению. – Москва: Изд-во стандартов, 1994. – 17 с.
6. Размеры должностных окладов ППС с учетом повышающих квалификационных коэффициентов: Приложение 1 к приказу № 1372 от 03.05.2017: о повышении уровня оплаты труда отдельных категорий работников университета
7. Налоговый кодекс Российской Федерации (часть вторая) от 05.08.2000 № 117-ФЗ (ред. от 03.04.2017) // Собрание законодательства Российской Федерации. 07.08.2000. N 32. Ст. 259.1. С. 309 – 310.
8. О Классификации основных средств, включаемых в амортизационные группы: постановление Правительства Российской Федерации от 01.01.2002 N 1 (ред. от 07.07.2016) // Собрание законодательства Российской Федерации. 07.07.2016. N 640. С. 4

ПРИЛОЖЕНИЕ А. Листинг кода эксперимента по выбору праметра упаковки контуров в findContours

**Файл experiment\_chain.py**

|  |
| --- |
| import cv2 as cv  import numpy as np  import dtitem  from ast import literal\_eval as make\_tuple  import timeit  import matplotlib.pyplot as plt  fn = './samples/multi\_4.jpg'  config = dtitem.get\_config()  image\_scale\_factor= config["SearchItem"].getfloat("image\_scale\_factor")  gaussian\_kernel = make\_tuple(config["SearchItem"]["gaussian\_kernel"])  sigma = config["SearchItem"].getfloat("sigma")  close\_kernel = make\_tuple(config["SearchItem"]["close\_kernel"])  max\_area = make\_tuple(config["SearchItem"]["max\_area"])  img\_resize = cv.resize(cv.imread(fn), (0, 0), fx=0.1, fy=0.1)  gray = cv.cvtColor(img\_resize, cv.COLOR\_BGR2GRAY)  v = np.median(gray)  lower = int(max(0, (1.0 - sigma) \* v))  upper = int(min(255, (1.0 + sigma) \* v))  edged = cv.Canny(gray, lower, upper)  kernel = cv.getStructuringElement(cv.MORPH\_RECT, close\_kernel)  closed = cv.morphologyEx(edged, cv.MORPH\_CLOSE, kernel)  count = 10000  times = np.zeros(count)  for i in range(count):  start = timeit.default\_timer()  contours0, hierarchy = cv.findContours(closed.copy(), cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_TC89\_KCOS)  for cnt in contours0:  rect = cv.minAreaRect(cnt) # пытаемся вписать прямоугольник  times[i] = (timeit.default\_timer() - start)\*1000  mean = round(times.mean(), ndigits=2)  plt.plot([i for i in range(count)], times) |

**Окончание приложения А**

|  |
| --- |
| plt.plot([i for i in range(count)], [mean for i in range(count)], "r")  plt.xlabel("Count")  plt.ylabel("time, ms")  plt.title("CHAIN\_APPROX\_TC89\_KCOS, mean = %2.2f ms" % mean)  plt.show() |

ПРИЛОЖЕНИЕ Б. Листинг кода эксперимента по анализу производительности поиска штрихкода

**Файл experiment\_barcode.py**

|  |
| --- |
| import cv2 as cv  import numpy as np  import dtitem  import timeit  import matplotlib.pyplot as plt  from pyzbar.pyzbar import (decode, ZBarSymbol)  fn = './samples/multi\_6.jpg'  config = dtitem.get\_config()  items = dtitem.search\_items(cv.imread(fn))  item = items[1]  count = 10000  times = np.zeros(count)  for i in range(count):  start = timeit.default\_timer()  barcode = decode(item.img, [ZBarSymbol.EAN13])  times[i] = (timeit.default\_timer() - start)\*1000  mean = round(times.mean(), ndigits=2)  plt.plot([i for i in range(count)], times)  plt.plot([i for i in range(count)], [mean for i in range(count)], "r")  plt.xlabel("Count")  plt.ylabel("time, ms")  plt.title("search\_barcode\_item, mean = %2.2f ms" % mean)  plt.show() |

ПРИЛОЖЕНИЕ В. Листинг кода эксперимента по выбору оптимального количество точек на изображении

**Файл experiment\_color.py**

|  |
| --- |
| import cv2 as cv  import numpy as np  import dtitem  import timeit  import matplotlib.pyplot as plt  import random  import math  fn = './samples/multi\_6.jpg'  config = dtitem.get\_config()  items = dtitem.search\_items(cv.imread(fn))  item = items[1]  count = 1000  count\_samples = 100  times = np.zeros(count)  means = np.zeros(count\_samples)  sko = np.zeros(count\_samples)  height = item.img.shape[0]  width = item.img.shape[1]  for i in range(count\_samples):  print(i)  colors = []  color\_mean = np.zeros(3)  for j in range(count):  avg\_color = np.zeros(3)  d = (i + 1) \* 10  start = timeit.default\_timer()  for z in range(d):  w = int(random.uniform(0, width))  h = int(random.uniform(0, height))  avg\_color += item.img[h][w]  avg\_color /= d  color\_mean += avg\_color  times[j] = (timeit.default\_timer() - start)\*1000  colors.append(avg\_color) |

**Окончание приложения В**

|  |
| --- |
| np\_colors = np.asarray(colors)  color\_mean /= count  m = 0  for j in range(count):  np\_colors[j] -= color\_mean  m += np.linalg.norm(np\_colors[j])  m /= count  m = math.sqrt(m)  means[i] = round(times.mean(), ndigits=2)  sko[i] = m  plt.plot([(i + 1) \* 10 for i in range(count\_samples)], means)  #plt.plot([i for i in range(count)], [mean for i in range(count)], "r")  plt.xlabel("Count")  plt.ylabel("time, ms")  #plt.title("search\_barcode\_item, mean = %2.2f ms" % mean)  plt.show()  plt.plot([(i + 1) \* 10 for i in range(count\_samples)], sko, "r")  #plt.plot([i for i in range(count)], [mean for i in range(count)], "r")  plt.xlabel("Count")  plt.ylabel("SKO")  #plt.title("search\_barcode\_item, mean = %2.2f ms" % mean)  plt.show() |

ПРИЛОЖЕНИЕ Г. Листинг кода модуля поиска объектов на изображении

**Файл search\_item.py**

|  |
| --- |
| import numpy as np  import cv2 as cv  import logging  import datetime  import os  from dtitem.utility import (Item)  from dtitem.config import (get\_config)  from dtitem.crop\_rect import (crop\_rect)  from ast import literal\_eval as make\_tuple  def search\_items(img, config=None):  if config is None:  config = get\_config()  logger = None  text\_log = config["Logging"].getboolean("text\_log")  img\_log = config["Logging"].getboolean("img\_log")  show\_img = config["Logging"].getboolean("show\_img")  config["Logging"]["log\_path"] = config["Logging"]["log\_path"].format(date\_time=str(datetime.datetime.today()))  log\_path = config["Logging"]["log\_path"]  image\_scale\_factor = config["SearchItem"].getfloat("image\_scale\_factor")  gaussian\_kernel = make\_tuple(config["SearchItem"]["gaussian\_kernel"])  sigma = config["SearchItem"].getfloat("sigma")  close\_kernel = make\_tuple(config["SearchItem"]["close\_kernel"])  max\_area = make\_tuple(config["SearchItem"]["max\_area"])  if text\_log or img\_log:  if not os.path.exists(log\_path):  os.makedirs(log\_path)  if text\_log:  logger = logging.getLogger("SEARCH\_ITEMS")  logger.info("Module SEARCH\_ITEMS started")  def logging\_action(name, image):  if show\_img:  cv.imshow(name, image)  if img\_log: |

**Продолжение приложения Г**

|  |
| --- |
| cv.imwrite(log\_path + name + ".jpg", image)  if text\_log:  logger.info(repr(name) + " saved on " + repr(log\_path + name + ".jpg"))  items = []  logging\_action("1-source", img)  img\_resize = cv.resize(img, (0, 0), fx=image\_scale\_factor, fy=image\_scale\_factor)  logging\_action("2-resize", img\_resize)  gray = cv.cvtColor(img\_resize, cv.COLOR\_BGR2GRAY)  logging\_action("3-gray", gray)  v = np.median(gray)  lower = int(max(0, (1.0 - sigma) \* v))  upper = int(min(255, (1.0 + sigma) \* v))  edged = cv.Canny(gray, lower, upper)  logging\_action("4-edged", edged)  kernel = cv.getStructuringElement(cv.MORPH\_RECT, close\_kernel)  closed = cv.morphologyEx(edged, cv.MORPH\_CLOSE, kernel)  logging\_action("5-closed", closed)  contours0, hierarchy = cv.findContours(closed.copy(), cv.RETR\_EXTERNAL, cv.CHAIN\_APPROX\_SIMPLE)  total = 0  for cnt in contours0:  rect = cv.minAreaRect(cnt)  area = int(rect[1][0] \* rect[1][1])  rect = ((int(rect[0][0] / image\_scale\_factor), int(rect[0][1] / image\_scale\_factor)),  (int(rect[1][0] / image\_scale\_factor), int(rect[1][1] / image\_scale\_factor)),  rect[2])  box = cv.boxPoints(rect) # поиск четырех вершин прямоугольника  box = np.int0(box) # округление координат  if area > max\_area:  img\_crop = crop\_rect(img, rect)  if text\_log: |

**Окончание приложения Г**

|  |
| --- |
| logger.info("Object " + repr("item\_" + str(total)) + " found: " + repr(rect))  logging\_action("item\_" + str(total), img\_crop)  if img\_log:  cv.drawContours(img, [cnt \* int(1/image\_scale\_factor)], 0, (255, 0, 0), 2)  items.append(Item("item\_" + str(total), rect, box, img\_crop))  total = total + 1  logging\_action("6-dist", img)  if img\_log:  cv.waitKey()  cv.destroyAllWindows()  if text\_log:  logger.info("Module SEARCH\_ITEMS has been successfully completed")  return items |

ПРИЛОЖЕНИЕ Д. Листинг кода функции выделения прямоугольных объектов с изображения.

**Файл crop\_item.py**

|  |
| --- |
| import cv2 as cv  import numpy as np  def crop\_rect(img, rect):  center, size, angle = rect[0], rect[1], rect[2]  center, size = tuple(map(int, center)), tuple(map(int, size))  height, width = img.shape[0], img.shape[1]  translation = np.float32([[1, 0, width/2 - center[0]], [0, 1, height/2 - center[1]]])  img\_tr = cv.warpAffine(img, translation, (width, height))  rotation = cv.getRotationMatrix2D((width/2, height/2), angle, 1)  img\_rot = cv.warpAffine(img\_tr, rotation, (width, height))  img\_crop = cv.getRectSubPix(img\_rot, size, (width/2, height/2))  if img\_crop.shape[0] > img\_crop.shape[1]:  img\_crop = cv.rotate(img\_crop, cv.ROTATE\_90\_CLOCKWISE)  return img\_crop |

ПРИЛОЖЕНИЕ Е. Листинг кода функции выделения штрихкода на изображении search\_barcode

**Файл search\_barcode.py**

|  |
| --- |
| import numpy as np  import cv2 as cv  import os  import logging  import datetime  from pyzbar.pyzbar import (decode, ZBarSymbol)  from dtitem.config import (get\_config)  from dtitem.utility import (Barcode)  from dtitem.crop\_rect import (crop\_rect)  def search\_barcode\_item(item, config = None):  if config is None:  config = get\_config()  logger = None  text\_log = config["Logging"].getboolean("text\_log")  img\_log = config["Logging"].getboolean("img\_log")  show\_img = config["Logging"].getboolean("show\_img")  config["Logging"]["log\_path"] = config["Logging"]["log\_path"].format(date\_time=str(datetime.datetime.today()))  log\_path = config["Logging"]["log\_path"]  margin\_x = config["Barcode"].getint("margin\_x")  margin\_y = config["Barcode"].getint("margin\_y")  if img\_log:  if not os.path.exists(log\_path + item.name + "/"):  os.makedirs(log\_path + item.name + "/")  if text\_log:  logger = logging.getLogger("SEARCH\_BARCODE")  logger.info("Module SEARCH\_BARCODE started")  def logging\_action(name, image):  if show\_img:  cv.imshow(name, image)  if img\_log:  cv.imwrite(log\_path + item.name + "/" + name + ".jpg", image)  if text\_log:  logger.info(repr(name) + " saved on " + repr(log\_path + item.name + "/" + name + ".jpg")) |

**Продолжение приложения Е**

|  |
| --- |
| gray = cv.cvtColor(item.img, cv.COLOR\_BGR2GRAY)  logging\_action("1-gray.jpg", gray)  grad\_x = cv.Sobel(gray, ddepth=cv.CV\_32F, dx=1, dy=0, ksize=-1)  logging\_action("2-dx.jpg", grad\_x)  gradient = cv.convertScaleAbs(grad\_x)  logging\_action("3-gradient.jpg", gradient)  blurred = cv.GaussianBlur(gradient, (1, 45), 0)  (\_, thresh) = cv.threshold(blurred, 225, 255, cv.THRESH\_BINARY)  logging\_action("4-thresh.jpg", thresh)  kernel = cv.getStructuringElement(cv.MORPH\_RECT, (17, 1))  closed = cv.morphologyEx(thresh, kernel)  logging\_action("5-closed.jpg", closed)  closed = cv.erode(closed, None, iterations=4)  closed = cv.dilate(closed, None, iterations=4)  logging\_action("6-dilate.jpg", closed)  (cnts, \_) = cv.findContours(closed.copy(), cv.RETR\_EXTERNAL,  cv.CHAIN\_APPROX\_SIMPLE)  max\_rect = cv.minAreaRect(cnts[0])  max\_area = int(max\_rect[1][0] \* max\_rect[1][1])  for cnt in cnts:  rect = cv.minAreaRect(cnt) # пытаемся вписать прямоугольник  area = int(rect[1][0] \* rect[1][1])  if area > max\_area:  max\_area = area  max\_rect = rect  max\_rect = ((max\_rect[0][0], max\_rect[0][1]),  (max\_rect[1][0] + 2 \* margin\_x, max\_rect[1][1] + 2 \* margin\_y),  max\_rect[2])  box = cv.boxPoints(max\_rect) # поиск четырех вершин прямоугольника  box = np.int0(box) # округление координат  img\_copy = item.img.copy()  if img\_log: |

**Окончание приложения Е**

|  |
| --- |
| cv.drawContours(img\_copy, [box], 0, (255, 0, 0), 2)  img\_crop = crop\_rect(item.img, max\_rect)  logging\_action("7-barcode.jpg", img\_crop)  barcode = decode(img\_crop, symbols=[ZBarSymbol.EAN13])  logging\_action("8-dest.jpg", img\_copy)  if show\_img:  cv.waitKey()  cv.destroyAllWindows()  if len(barcode) > 0:  item.barcode = Barcode(barcode[0].data.decode("utf-8"), max\_rect, box, img\_crop)  else:  item.barcode = Barcode(None, max\_rect, box, img\_crop)  return item |

ПРИЛОЖЕНИЕ Ж. Листинг кода функции вычисления цветового параметра поп изображению

**Файл search\_color.py**

|  |
| --- |
| import numpy as np  import random  import os  import logging  import datetime  from dtitem.config import (get\_config)  def search\_color\_item(item, config=None):  if config is None:  config = get\_config()  logger = None  text\_log = config["Logging"].getboolean("text\_log")  img\_log = config["Logging"].getboolean("img\_log")  config["Logging"]["log\_path"] = config["Logging"]["log\_path"].format(date\_time=str(datetime.datetime.today()))  log\_path = config["Logging"]["log\_path"]  count\_samples = config["Color"].getint("count\_samples")  if text\_log or img\_log:  if not os.path.exists(log\_path + item.name + "/"):  os.makedirs(log\_path + item.name + "/")  if text\_log:  logger = logging.getLogger("SEARCH\_COLOR")  logger.info("Module SEARCH\_COLOR started")  def logging\_action(color):  if text\_log:  logger.info("The calculated average color:" + str(color))  height = item.img.shape[0]  width = item.img.shape[1]  avg\_color = np.zeros(3)  for i in range(count\_samples):  w = int(random.uniform(0, width))  h = int(random.uniform(0, height))  avg\_color += item.img[h][w]  avg\_color /= count\_samples  logging\_action(avg\_color)  item.color = avg\_color.astype(np.int)  return item |

ПРИЛОЖЕНИЕ И. Листинг кода модуля классификации объектов

**Файл classify.py**

|  |
| --- |
| from dtitem.features import \*  from dtitem.config import (get\_config)  import numpy as np  def classify\_barcode(item, cb):  probabilities = np.zeros(len(cb.categories))  i = 0  count = 0  for category in cb.categories:  feature = category.get\_feature(BarcodeFeature)  if feature is None:  raise Exception("BarcodeFeature not found")  proximity = feature.proximity(item)  if proximity:  probabilities[i] = 1.0  count += 1  i += 1  if count != 0:  probabilities /= count  return probabilities  def classify\_color(item, cb):  probabilities = np.zeros(len(cb.categories))  i = 0  used\_proximity = []  for category in cb.categories:  feature = category.get\_feature(ColorFeature)  if feature is None:  raise Exception("ColorFeature not found")  probabilities[i] = feature.proximity(item) + 0.1  used\_proximity.append(probabilities[i])  i += 1  if len(used\_proximity) > 0:  probabilities /= min(used\_proximity)  k = 0  for p in probabilities:  k += 1 / p  k = 1 / k  for i in range(probabilities.size):  probabilities[i] = k / probabilities[i] |

**Продолжение приложения И**

|  |
| --- |
| return probabilities  def classify\_size(item, cb):  probabilities = np.zeros(len(cb.categories))  i = 0  used\_proximity = []  for category in cb.categories:  feature = category.get\_feature(SizeFeature)  if feature is None:  raise Exception("SizeFeature not found")  probabilities[i] = feature.proximity(item) + 0.1  used\_proximity.append(probabilities[i])  i += 1  if len(used\_proximity) > 0:  probabilities /= min(used\_proximity)  k = 0  for p in probabilities:  k += 1 / p  k = 1 / k  for i in range(probabilities.size):  probabilities[i] = k / probabilities[i]  return probabilities  def classify\_item(item, cb, config=None):  if config is None:  config = get\_config()  search\_barcode\_enabled = config["Barcode"].getboolean("enabled")  search\_color\_enabled = config["Color"].getboolean("enabled")  search\_size\_enabled = config["Size"].getboolean("enabled")  search\_neural\_enabled = config["Neural"].getboolean("enabled")  threshold\_ration = config["General"].getfloat("threshold\_ration")  count\_methods = 0  if search\_barcode\_enabled:  barcode\_proximity = classify\_barcode(item, cb)  if barcode\_proximity.max() != 0:  item.cl = sorted(list(zip(cb.categories, barcode\_proximity)), key=lambda x: x[1], reverse=True)  return item  proximity = np.zeros(len(cb.categories))  if search\_color\_enabled:  count\_methods += 1  proximity += classify\_color(item, cb) |

**Окончание приложения И**

|  |
| --- |
| max\_item = proximity.max()  max\_near = np.delete(proximity, proximity.argmax()).max()  if max\_item / max\_near >= threshold\_ration:  item.cl = sorted(list(zip(cb.categories, proximity.round(3))), key=lambda x: x[1], reverse=True)  return item  if search\_size\_enabled:  count\_methods += 1  proximity += classify\_size(item, cb)  proximity /= count\_methods  item.cl = sorted(list(zip(cb.categories, proximity.round(3))), key=lambda x: x[1], reverse=True)  return item |

ПРИЛОЖЕНИЕ К. Листинг кода функции детектирования объектов

**Файл search\_color.py**

|  |
| --- |
| import os  import logging  import cv2 as cv  import datetime  from dtitem.config import (get\_config)  from dtitem.utility import (CategoryBase)  from dtitem.search\_items import (search\_items)  from dtitem.search\_barcode import (search\_barcode\_item)  from dtitem.search\_color import (search\_color\_item)  from dtitem.search\_size import (search\_size\_item)  from dtitem.classify import (classify)  def detect\_items(img\_path, cb: CategoryBase, config=None):  if config is None:  config = get\_config()  logger = None  text\_log = config["Logging"].getboolean("text\_log")  img\_log = config["Logging"].getboolean("img\_log")  config["Logging"]["log\_path"] = config["Logging"]["log\_path"].format(date\_time=str(datetime.datetime.today()))  log\_path = config["Logging"]["log\_path"]  search\_barcode\_enabled = config["Barcode"].getboolean("enabled")  search\_color\_enabled = config["Color"].getboolean("enabled")  search\_size\_enabled = config["Size"].getboolean("enabled")  search\_neural\_enabled = config["Neural"].getboolean("enabled")  if text\_log or img\_log:  if not os.path.exists(log\_path):  os.makedirs(log\_path)  if text\_log:  logging.basicConfig(filename=log\_path + "log",  level=logging.INFO,  format=u'[%(asctime)s][%(name)12s]# %(message)s',  datefmt="%Y-%m-%d %H:%M:%S")  logger = logging.getLogger("DETECT\_ITEMS")  img = cv.imread(img\_path)  if text\_log: |

**Окончание приложения К**

|  |
| --- |
| logger.info("Source image is loaded from " + repr(img\_path))  items = search\_items(img, config)  if text\_log:  logger.info("Module SEARCH\_ITEMS found " + str(len(items)) + " objects")  for item in items:  if text\_log:  logger.info("Module SEARCH\_BARCODE started on " + repr(item.name))  if search\_barcode\_enabled:  item = search\_barcode\_item(item, config)  if search\_color\_enabled:  item = search\_color\_item(item, config)  if search\_size\_enabled:  item = search\_size\_item(item, config)  item = classify(item, cb)  if text\_log:  logger.info("Processing the image is complete")  return items |

ПРИЛОЖЕНИЕ Л. Листинг кода функции получения конфигурации

**Файл config.py**

|  |
| --- |
| import configparser as cp  import os  default\_path = "config.ini"  def get\_config(path=None):  config = cp.ConfigParser()  if path is not None:  config.read(path)  else:  if not os.path.exists(default\_path):  config["General"] = {  "threshold\_ration": "2.0"  }  config["Logging"] = {  "text\_log": "True",  "img\_log": "True",  "show\_img": "False",  "log\_path": "../log/{date\_time}/"  }  config["SearchItem"] = {  "image\_scale\_factor": "0.1",  "max\_area": "3000.1",  "gaussian\_kernel": "(3, 3)",  "close\_kernel": "(7, 7)",  "sigma": "0.1"  }  config["Barcode"] = {  "enabled": "True",  "margin\_x": "15",  "margin\_y": "15"  }  config["Color"] = {  "enabled": "True",  "count\_samples": 1000  }  config["Size"] = {  "enabled": "True",  "scale": "0.0001118223" |

**Окончание приложения Л**

|  |
| --- |
| }  config["Neural"] = {  "enabled": "True"  }  with open(default\_path, "w") as config\_file:  config.write(config\_file)  config.read(default\_path)  return config |

ПРИЛОЖЕНИЕ М. Листинг кода примера использования библиотеки

**Файл main.py**

|  |
| --- |
| import dtitem  def main():  # fn = './samples/snickers\_1.jpg'  fn = './samples/multi\_6.jpg'  c1 = dtitem.Category("SNICKERS SUPER",  dtitem.BarcodeFeature("5000159455367"),  dtitem.ColorFeature(88, 92, 107),  dtitem.SizeFeature(18.2, 4.5))  c2 = dtitem.Category("KITKAT",  dtitem.BarcodeFeature("4606272034963"),  dtitem.ColorFeature(91, 92, 134),  dtitem.SizeFeature(14, 4))  c3 = dtitem.Category("KITKAT BIG",  dtitem.BarcodeFeature("4606272035465"),  dtitem.ColorFeature(98, 101, 136),  dtitem.SizeFeature(18, 4))  c4 = dtitem.Category("MARS MAX",  dtitem.BarcodeFeature("4607065001490"),  dtitem.ColorFeature(102, 116, 127),  dtitem.SizeFeature(18.5, 4.5))  c5 = dtitem.Category("BOUNTY",  dtitem.BarcodeFeature("4011100977624"),  dtitem.ColorFeature(131, 129, 126),  dtitem.SizeFeature(16, 4))  c6 = dtitem.Category("TWIX",  dtitem.BarcodeFeature("4011100977952"),  dtitem.ColorFeature(52, 75, 111),  dtitem.SizeFeature(16, 8))  c7 = dtitem.Category("TWIX BIG",  dtitem.BarcodeFeature("5000159390729"),  dtitem.ColorFeature(75, 97, 131),  dtitem.SizeFeature(18.5, 8))  cb = dtitem.CategoryBase(c1, c2, c3, c4, c5, c6, c7)  detected = dtitem.detect\_items(fn, cb)  print(\*detected, sep="\n")  if \_\_name\_\_ == "\_\_main\_\_":  main() |

ПРИЛОЖЕНИЕ Н. Листинг кода тестирования детектирования объектов по цвету

**Файл test\_detect\_items\_color.py**

|  |
| --- |
| from unittest import TestCase  import dtitem  import cv2 as cv  class TestDetect\_items(TestCase):  def test\_detect\_items(self):  fn = '../samples/multi\_6.jpg'    cb = dtitem.CategoryBase(c1, c2, c3, c4, c5, c6, c7)  count = 100000  config\_color = dtitem.get\_config(“config\_color.ini”)  items = dtitem.detect\_item(fn, config\_color)  for i in range(count):  items = dtitem.detect\_item(fn, config\_color)  self.assertEqual(item.cl[0][0].name, 'BOUNTY') |