

МИНОБРНАУКИ РОССИИ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ им. Р.Е. АЛЕКСЕЕВА»
(НГТУ)

Институт (факультет) Институт радиоэлектроники и информационных технологий (ИРИТ)

Направление подготовки (специальность) 09.03.01 Информатика и вычислительная техника

Направленность (профиль) образовательной программы Вычислительные машины, комплексы, системы и сети

Кафедра Вычислительные системы и технологии

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

бакалавра

Студента Никифорова Николая Андреевича группы 14-В-1

на тему Программная система обнаружения объектов

СТУДЕНТ

КОНСУЛЬТАНТЫ:

Никифоров Н.

1. По _____

А.

(подпись)

(фамилия, и., о.)

(подпись)

(фамилия, и., о.)

РУКОВОДИТЕЛЬ

2. По _____

Гай В.Е.

(подпись)

(фамилия, и., о.)

(подпись)

(фамилия, и., о.)

3. По _____

(подпись)

(фамилия, и., о.)

ЗАВЕДУЮЩИЙ КАФЕДРОЙ

Кондратьев В.В.

ВКР защищена _____

(подпись)

(фамилия, и., о.)

(дата)

протокол № _____

с оценкой _____

Оглавление

Введение.....	3
1. Постановка задачи.....	4
1.1. Назначение разработки и область применения	4
1.2. Технические требования.....	4
2. Анализ поставленной задачи.....	5
2.1. Выбор операционной системы для разработки	5
2.2. Выбор языка программирования	6
2.3. Выбор среды разработки	8
2.4. Обзор существующих систем обнаружения объектов на изображении	9
2.5. Выбор подхода к решению задачи обнаружения объектов	11
3. Разработка системы на структурном уровне	13
3.1. Разработка информационной модели системы обнаружения объектов	13
3.2. Разработка общей структуры системы.....	15
3.3. Подсистема обучения	16
3.4. Подсистема применения.....	17
3.5. Подсистема классификации	19
3.6. Разработка алгоритма принятия решения.....	21
4. Разработка программных средств.....	24
4.1. Разработка системы анализа качества обнаружения	28
5. Тестирование системы.....	30
5.1. Пользовательский интерфейс.....	30
5.2. Описание набора данных.....	33
Результаты вычислительного эксперимента	35
Заключение	40
Список литературы	41

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)		
Изм.	Лист	№ докум.	Подпись	Дата			
Разраб.		Никифоров Н.А.			Программная система обнаружения объектов		
Провер.		Гай В.Е.					
Реценз.							
Н. Контр.							
Утверд.							
						Лит.	Лист
							Листов
							3
							42
						НГТУ им. Р.Е. Алексеева	

Введение

В настоящее время широкое распространение получили системы обнаружения объектов на изображении. Задача обнаружения объектов на изображении – одна из фундаментальных проблем в области компьютерного зрения и обработки изображений. Практическое применение данная задача находит в системах автопилотирования, индексирования изображений и видео, охранных системах.

Заметим, что с задачей распознавания, в более широком ее понимании, сталкивается каждый человек. Читаем ли мы текст, переходим ли улицу, смотрим ли телевизор, слушаем ли музыку - в каждый момент времени мы решаем задачу распознавания.

Одним из примеров, где каждодневно люди сталкиваются с задачей распознавания образов, является криминалистика и решается она путем составления словесного признакового описания искомого объекта. Сопоставив это описание с реальным объектом человек принимает решение о том, является ли данный объект его целью.

Приведенный пример позволяет заметить принятое в теории рас познавания образов разделение процесса распознавания на два этапа. На первом этапе происходит формирование некоего набора свойств и правил, описывающих искомый объект. Данный набор называют признаками объекта. На втором этапе производится сопоставление того что человек наблюдает с теми признаками, которые были получены на предыдущем шаге. То есть человек принимает решение о том, принадлежит ли данный объект классу, который был описан в признаковом описании. Этот процесс в направлении распознавание образов называют классификацией.

Следовательно, структуру системы обнаружения объектов на изображении можно представить в виде совокупности трёх блоков: предварительная обработка изображения, формирование признакового описания, принятие решения.

Целью данной работы является разработка системы обнаружения объектов на основе теории активного восприятия, разработанной профессором Нижегородского Технического Университета им. Р.Е. Алексеева Утробиним В.А., и её тестирование.

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		4

1. Постановка задачи

1.1. Назначение разработки и область применения

Разрабатываемая система предназначена для обнаружения отдельных объектов на изображении на основе теории активного восприятия. Для работы системы достаточно иметь персональный компьютер с установленным на нем программным обеспечением.

Области применения разрабатываемой системы:

1. Оценка плотности транспортного потока.
2. Обнаружение посторонних объектов на железнодорожных переездах.
3. Обнаружение объектов для автомобилей с автопилотированием.

Данная система предназначена для использования на портативных и стационарных компьютерах.

1.2. Технические требования

Опишем требования, предъявляемые разрабатываемой системой к ЭВМ:

1. Операционная система *Microsoft Windows XP* и выше
2. Требования к аппаратному обеспечению определяются операционной системой
3. Мышь, дисплей, клавиатура

Рассмотрим, каким функционалом должна обладать разрабатываемая система обнаружения объектов на изображении:

1. Система должна предоставить возможность пользователю выбрать изображение, на котором будет искать объект.
2. Позволить указать обучающую выборку или выбрать уже обученную модель.
3. Реализовать поиск объектов методом скользящего окна.
4. Вывести результирующее изображение.

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		5

2. Анализ поставленной задачи

2.1. Выбор операционной системы для разработки

На этапе начала разработки важным является выбор операционной системы, так как каждая из них имеет свои особенности, влияющие на разработку. Рассмотрим наиболее распространенные операционные системы: *Mac OS, Linux, Windows*:

1. Windows - семейство операционных систем корпорации *Microsoft*, ориентированных на применение графического интерфейса при управлении. Первая версия *Windows*, выпущенная в 1985 году, была простым графическим интерфейсом, поставляемым как расширение существующей операционной системы от *Microsoft* – *MS DOS*. В последующих версиях появилась большая функциональность, включая собственные диспетчер файлов *Windows*, диспетчер программ и программы печати, а также более динамичный интерфейс. *Microsoft* также разработала специализированные пакеты *Windows*, включая сетевые ОС для рабочих групп и мощную *Windows NT*, ориентированную на бизнес. В 2015 году *Microsoft* выпустила *Windows 10*, которая поставляется вместе с *Cortana*, цифровым персональным помощником, таким как *Apple Siri*, и веб-браузером *Microsoft Edge*, который заменил *Internet Explorer*. *Microsoft* также объявила, что *Windows 10* будет последней версией *Windows*, а это означает, что пользователи получают регулярные обновления ОС. Примерно 90 процентов пользователей ПК сейчас используют одну из версий *Windows*. [1].

2. *MAC OS* - операционная система, разработанная *Apple Inc.* Предназначена для использования на серии персональных компьютеров *Macintosh* с 1984 года. Является второй по распространенности операционной системой. Главным отличием от *Windows* считается то что *Mac OS* основана на системе *Unix* [1].

3. *Linux* - название семейства *Unix* – подобных операционных систем на основе ядра *linux* и включающих в себя библиотеки и системные программы проекта *GNU*. Особенностью является открытость кода. Также *Linux* создается и распространяется как свободное и открытое ПО, посредством готовых дистрибутивов, настроенных для решения определенных задач [1].

Для создания системы обнаружения объектов была выбрана система *Windows*, так как данная операционная система обладает наибольшим выбором программного обеспечения для разработки, а также является более распространенной.

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		6

2.2. Выбор языка программирования

Одним из важнейших этапов разработки программного обеспечения является выбор языка программирования. Для того, чтобы определиться с тем, на каком языке будет написана наша программная система, предлагаем рассмотреть особенности и направленность нескольких популярных языков программирования: *C++*, *Java*, *Python* и *R*.

1. *C++* - является языком программирования общего назначения. *C++* - один из самых популярных языков, в основном используемых с системным / прикладным программным обеспечением, драйверами и клиент-серверными приложениями. *C++* считается языком промежуточного уровня, поскольку он инкапсулирует как языковые функции высокого, так и низкого уровня. Первоначально язык назывался «*C with classes*», так как он обладал всеми свойствами языка *C* с дополнительным понятием «классы». Однако в 1983 году он был переименован в *C++*. [2].

2. *Java* – это язык программирования и вычислительная платформа, впервые выпущенная *Sun Microsystems* в 1995 году. Существует множество приложений и веб-сайтов, которые не будут работать, если у вас нет установленной *Java*. Программы, написанные на *Java* транслируются в байт-код и выполняются на виртуальной машине – *JVM*, обрабатывающей байт-код и передающей инструкции оборудованию. В отличие от текста байт-код обрабатывается намного быстрее, что является достоинством [3].

3. *Python* – интерпретируемый, объектно-ориентированный, высокоуровневый язык программирования с динамической семантикой. Его высокоуровневые встроенные структуры данных в сочетании с динамической типизацией и динамической привязкой делают его очень привлекательным для *Rapid Application Development*, а также для использования в качестве языка сценариев или для соединения существующих компонентов. Преимуществом *Python* является наличие множества расширений и пакетов. Базовые пакеты присутствуют сразу после установки *Python*, при наличии интернета дополнительные пакеты можно установить командой *pip install*. [4]

4. *R* – язык программирования, использующийся для статистической обработки данных и работы с графикой. *R* предоставляет широкий спектр статистических (линейное и нелинейное моделирование, классические статистические тесты, анализ временных рядов, классификация, кластеризация, ...) и графических методов и является весьма расширяемым. Есть возможность реализации функций на *C*, *C++*, для более сложных вычислительных задач [5].

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист 7
Изм.	Лист	№ докум.	Подпись	Дата		

Для разработки программной системы был выбран язык Python, так как в нем реализованы необходимые нам библиотеки для работы с графическими файлами, такие как:

1. *OpenCV* — библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом.

2. *scikit-learn* - это модуль *Python* для машинного обучения, построенный поверх *SciPy* и распространяемый по лицензии BSD 3-Clause.

3. *SciPy* - это основанная на *Python* экосистема программного обеспечения с открытым исходным кодом для математики, науки и техники.

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		8

2.3. Выбор среды разработки

Современный выбор сред разработки для языка *Python* достаточно обширен. В него входят как непосредственно среда выполнения, поставляющаяся вместе с языком (в консольном и графическом вариантах), так и средства, поставляющиеся сторонними разработчиками. Самые заметные из них - *PyCharm* (от известного разработчика *JetBrains*), *Eric*, *Spyder(IDE* с открытым исходным кодом), *Komodo IDE*(среда для языков с динамической типизацией), *PyScripter* (находящийся в свободном доступе бесплатный IDE), *Eclipse* (платформа общего назначения).

Среди всего этого многообразия для разработки данной программной системы нами была выбрана интегрированная среда разработки *PyCharm*. Она сочетает интуитивный пользовательский интерфейс с мощной консолью, и её основным преимуществом перед остальными интегрированными средами разработки является её бесплатность для студентов в сочетании с ее возможностями.

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		9

2.4. Обзор существующих систем обнаружения объектов на изображении

Задача создания бесшовных изображений не является сравнительно новой, существует несколько работ по данной теме, различные подходы и алгоритмы.

В [6] рассмотрена задача детектирования объекта, представляющего собой несколько соединенных сфер одинакового радиуса. В решении поставленной задачи определен ряд этапов: выделение границ объекта с использованием детектора Канни, поиск структурных элементов на основе преобразования Хафа, определение положения объекта с использованием распознавания по эталону и поиска изменений по кадрам. К недостаткам предложенного метода распознавания по эталону можно отнести частный характер задачи (должна быть известна структура объекта), значительные затраты времени и вычислительных ресурсов на формирование базы данных всевозможных изображений объекта.

В работе также рассмотрен подход к решению задачи определения положения объекта с использованием нейросети. К достоинствам использования неокогнитрона при распознавании относятся отсутствие необходимости этапов поиска границ и выделения окружностей, а также возможность внесения изменений для каждой новой задачи. К недостаткам следует отнести сложность структуры сети, порождающую большой объем вычислений и отсутствие возможности самообучения.

В [7] предложен метод распознавания, относящийся к классу обобщающих методов. Объект рассматривается как совокупность элементов, в качестве которых выбраны линии границ. Для обучения используется эталонное изображение объекта, на основе которого строится эталонный каркас. Исследуемое изображение обрабатывается фильтром Хаара с последующим выделением и соединением точек. Обнаружение заключается в построении каркаса исследуемого изображения, поиск наилучшего совмещения с эталоном, принятие решения об обнаружении путем сравнения количества совпавших линий с пороговым значением. К недостаткам данного метода можно отнести то, что он применим только для ограниченного класса объектов характерной формы.

В [8] описан подход к распознаванию объекта на изображении с использованием алгоритма адаптивного усиления (*AdaBoost*), в основе которого лежит идея отбора и комбинирования слабых классификаторов. Тестирование реализации предложенного алгоритма показало приемлемое время обработки изображений в тренировочной выборке. Установлено также, что в реальных условиях требуется обучающая выборка большого объема, а на обучение каскада классификаторов необходимо потратить несколько дней.

В [9] описан метод детектирования лиц на изображениях, в основу которого положены идеи алгоритма *Viola&Jones*. Для повышения скорости и точности

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
						10
Изм.	Лист	№ докум.	Подпись	Дата		

детектирования в предобработке изображения предложено использовать серию фильтров, где каждый следующий фильтр обрабатывает только «перспективные» части изображения, полученные от предыдущих фильтров. При этом на основе использования информации о цвете и форме объекта, достигается отделение регионов, не содержащих лиц. Для детектирования лиц использовались совокупности анизотропных гауссовых примитивов, объединенные в каскадную модель (*AdaBoost*). Приведённые результаты экспериментов показывают, что предложенный гибридный метод превосходит классические по качеству и скорости распознавания, однако, сохраняются недостатки, связанные с трудоемкостью обучения каскада классификаторов.

В [10] представлен алгоритм обнаружения объектов, основанный на обучении свёрточной нейронной сети. Особенностью алгоритма является то, что в единой сети решаются сразу две задачи: выделение прямоугольных блоков, содержащих объекты и определение принадлежности объекта некоторому классу. Для распознавания исходное изображение приводится к размеру 448×448 , делится на ячейки 7×7 , для каждой ячейки формируется метка, характеризующая ее принадлежность объекту некоторого класса.

Достоинством предложенного алгоритма является высокая скорость обработки изображений. К недостаткам следует отнести затраты на обучение сети, снижение точности обнаружения по сравнению с аналогичными алгоритмами, ограничения, связанные с размерами ячеек сетки (возникают трудности при обнаружении мелких объектов и объектов, расположенных близко друг к другу).

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		11

2.5. Выбор подхода к решению задачи обнаружения объектов

Для решения задачи обнаружения объектов была выбрана теория активного восприятия изображения.

Базовым преобразованием ТАВ является U -преобразование, которое реализуется в два этапа [11]. На первом этапе к изображению применяется Q -преобразование, после которого получаем матрицу визуальных масс m размером 4×4 элемента:

$$m[i, j] = \sum_{k=\left(\frac{i-1}{4}\right) \cdot N}^{\frac{i}{4} \cdot N} \sum_{l=\left(\frac{j-1}{4}\right) \cdot M}^{\frac{j}{4} \cdot M} I[k, l], i = \overline{1, 4}, j = \overline{1, 4},$$

где I – изображение размером $N \times M$ отсчётов. Для корректной интерпретации результатов Q -преобразования отсчёты изображения должны принадлежать положительной области значений. С позиций ТАВ Q -преобразование соответствует этапу предварительной обработки изображения.

На втором этапе к результату Q -преобразования применяется множество фильтров $F = \{F_i\}$, $i = \overline{1, 16}$ (см. рис. 1). В результате, формируется вектор спектральных коэффициентов μ :

$$\mu_i = \sum_{k=1}^4 \sum_{l=1}^4 m[k, l] \cdot F_i[k, l]$$

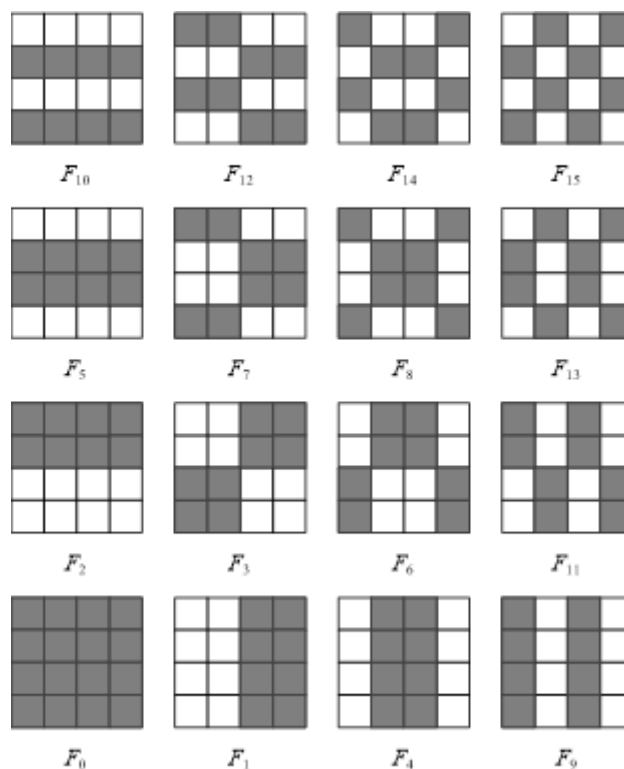


Рис. 1. Фильтры, используемые при вычислении признакового описания

Размер каждого фильтра составляет 4×4 элемента. Элемент фильтра может принимать значения «+1» (тёмные области на рис. 1) и «-1» (светлые области на рис. 1). Конструктивно данные фильтры подобны фильтрам Уолша системы Хармута. Специфика использования данных фильтров заключается в том, что они применяются после реализации Q -преобразования.

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		13

3. Разработка системы на структурном уровне

3.1. Разработка информационной модели системы обнаружения объектов

Этап обучения заключается в выполнении следующих шагов:

1. Формируется «положительные» и «отрицательные» образцы объектов для каждого из обнаруживаемых классов объектов; база данных «положительных» образцов включает изображения обнаруживаемых C классов объектов, каждый образец имеет размер $h_c \times w_c$, где h_c – высота образца, w_c – ширина образца, c – класс образца ($c \in \overline{1, C}$), причём каждый образец содержит изображение детектируемого объекта, база данных «отрицательных» образцов строится аналогично базе для «положительных», каждый «отрицательный» образец не содержит изображение детектируемого объекта;

2. Вычисляется признаковое описание для изображений, находящихся в указанных базах данных; формирование признакового описания изображения I выполняется по следующему алгоритму:

2.1. Изображение I разбивается на L^2 равных частей без перекрытия $\mathbf{P} = \{P_i\}$, $i = \overline{1, L^2}$ размером $(N/L) \times (M/L)$ отсчётов (эквивалентность частей изображения выбрана исходя из того, что изображение обрабатывается в условиях априорной неопределённости), где P_i – i -ая область изображения, L – количество разбиений одной стороны изображения;

2.2. Для каждой части вычисляется признаковое описание на основе U -преобразования

$$D = FC[I],$$

где оператор $FC[\bullet]$ вычисляет признаковое описание изображения и реализуется следующим образом:

$$\forall j = \overline{1, L^2} : \\ D = D \cup U[P_j],$$

где $U[\bullet]$ – оператор вычисления U -преобразования, D – признаковое описание изображения I . Таким образом, число частей, на которое разбивается изображение определяет размерность признакового описания: $1 \times (15 \cdot L^2)$. В результате выполнения данного шага формируется признаковое описание положительных образцов для каждого класса $F_P = \{F_{P,i}\}$ и признаковое описание отрицательных образцов для каждого класса: $F_N = \{F_{N,i}\}$, $i = \overline{1, C}$;

3. С использованием метода опорных векторов на основе вычисленных признаковых описаний формируется модель для детектирования каждого из классов объектов; таким

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		14

образом, каждый классификатор обучается различать два класса: «положительный» образец некоторого класса и «отрицательный» образец для того же класса:

$$\mathbf{S} = \{S_i\}, S_i = SVM_t [F_{P,i}, F_{N,i}], i = \overline{1, C},$$

где $SVM_t[\bullet]$ – оператор определения параметров модели метода опорных векторов, \mathbf{S} – множество моделей, полученных в результате обучения классификатора для каждого из классов объектов;

Этап применения заключается в выполнении следующих шагов:

1. Предварительная обработка изображения I – деление изображения I , на котором выполняется детектирование объектов, на области с шагом s_h по горизонтали и s_v по вертикали, размер области для i -го класса объектов известен заранее и установлен на этапе обучения, выделение областей выполняется на нескольких масштабах; это используется для обнаружения объектов, размер которых меньше размера эталона

$$\mathbf{Q} = \{Q_{i,(x,y),s}\}, i = \overline{1, T},$$

где \mathbf{Q} – множество всех областей, полученных по изображению, $Q_{i,(x,y),s}$ – i -ая область, (x, y) – координата верхнего правого угла области, s – масштаб изображения ($0 < s < 1$), T – количество полученных областей;

2. Для каждой области Q_i формируется признаковое описание:

$$D_{Q_{ixys}} = FC [Q_{i,(x,y),s}], i = \overline{1, T},$$

где $D_{Q_{ixys}}$ – признаковое описание области $Q_{i,(x,y),s}$;

3. При выполнении обнаружения на вход классификатора направляются полученные на предыдущем этапе признаковые описания областей изображения:

$$C_{Q_{ixys}} = SVM_u [D_{Q_{ixys}}, \mathbf{S}],$$

где SVM_u – оператор определения класса по признаковому описанию $D_{Q_{ixys}}$ на основе множества моделей \mathbf{S} , $C_{Q_{ixys}}$ – класс области $Q_{i,(x,y),s}$;

4. Учитывая, что на предыдущем шаге для одного объекта, находящемся на тестовом изображении, генерируется множество близких друг другу вариантов расположения («гипотез»), необходимо выполнить подавление немаксимумов с использованием алгоритма *Soft-NMS*, описанного в [7]; данный метод выбирает область с максимальной оценкой правдоподобия некоторого класса объектов, а все остальные области, перекрывающиеся в определённом процентном соотношении с данной областью, подавляет.

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		15

3.2. Разработка общей структуры системы

Система обнаружения объектов на изображении включает в себя два этапа обработки данных: обучение и применение. На этапе обучения формируется признаковое описание детектируемого объекта и обобщается в одну модель для классификации. На втором этапе происходит процесс обнаружения объекта на изображении методом скользящего окна.

Структурная схема, описывающая обучение и применение представлена на рис. 2.

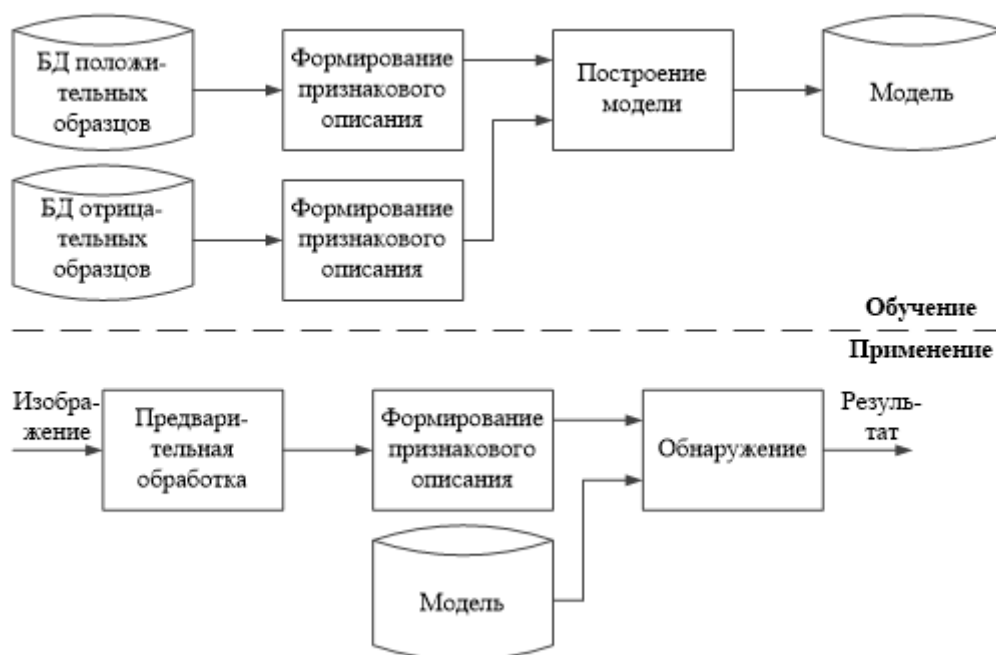


Рис. 2. Расширенная структурная схема

3.3. Подсистема обучения

Для того чтобы приступить к формированию модели классификации требуется составить базу данных изображений. В нашем случае модель будет содержать всего два класса: автомобиль и не автомобиль, соответственно выборка будет включать изображения для обоих классов.

Подсистема обучения включает в себя три этапа: предварительная обработка, сегментация и вычисление признаков.

Общая структура подсистемы представлена на рис. 3.

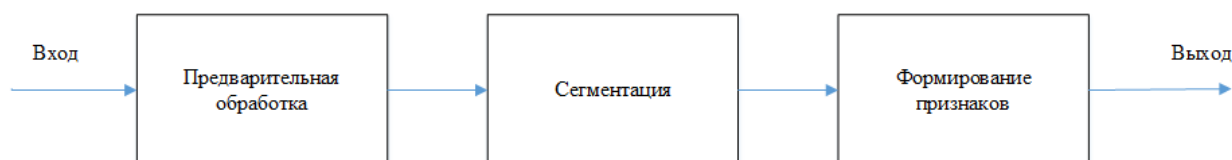


Рис. 3. Структурная схема подсистемы обучения.

В блоке предварительной обработки происходит нормализация изображения, таким образом производится переход от «пиксельного» представления обрабатываемого изображения к числовому.

Далее происходит сегментация изображения, оно делится на n -частей по ширине и высоте, где n число кратное четырем. При выборе n нужно учитывать, что каждая полученная область должна иметь размер не меньше 4×4 . После сегментации над каждым из сегментов производится Q -преобразование.

В блоке формирования признакового описания на каждую из полученных областей накладываются фильтры. В итоге получается, что для каждого сегмента имеется свой набор векторных признаков, и чем больше он будет, тем больше признаков будет сформировано для входного изображения.

3.4. Подсистема применения

Для перехода к этапу применения системы необходимо иметь заранее обученную модель классификатора.

В подсистему передается входное изображение, на котором и будет производиться поиск объектов. Данное изображение подвергается предварительной обработке. В нее входит нормализация всего изображения, построение пирамиды Гаусса, а также сегментация. Под сегментацией понимается деление исходного изображения на области, по которым будет проходить скользящее окно. Каждый сегмент имеет пересечение с соседним. Размер этих областей соответствует размеру изображений из обучающей выборки. Далее к этим областям применяется Q -преобразование. На этом этап предварительной обработки заканчивается.

Так как области имеют пересечения получается, что результатом предварительно обработки является цельный массив чисел. Далее начинается этап формирования признакового описания. К каждой полученной области применяются фильтры, и в результате получаем признаковое описание для каждого сегмента. К результату Q -преобразования применяется множество фильтров $F = \{F_i\}$, $i = \overline{1,16}$. В результате, формируется вектор спектральных коэффициентов μ :

$$\mu_i = \sum_{k=1}^4 \sum_{l=1}^4 m[k,l] \cdot F_i[k,l]$$

Таким образом имеется признаковое описание для целого входного изображения.

В итоге данные действия с входным изображением избавляют нас от пересчета отдельных признаков для каждой области, которая попадает в скользящее окно. Таким образом система не тратит свое время на обработку одного и того же массива данных (см. рис. 4).

Далее по имеющейся модели классификатор принимает решение о принадлежности той или иной области определенному классу.



[15]	[15]	[15]	[15]	[15]	[15]	[15]	[15]
[15]	[15]	[15]	[15]	[15]	[15]	[15]	[15]
[15]	[15]	[15]	[15]	[15]	[15]	[15]	[15]
[15]	[15]	[15]	[15]	[15]	[15]	[15]	[15]
[15]	[15]	[15]	[15]	[15]	[15]	[15]	[15]
[15]	[15]	[15]	[15]	[15]	[15]	[15]	[15]
[15]	[15]	[15]	[15]	[15]	[15]	[15]	[15]
[15]	[15]	[15]	[15]	[15]	[15]	[15]	[15]
				[15]	[15]	[15]	[15]
				[15]	[15]	[15]	[15]
				[15]	[15]	[15]	[15]
				[15]	[15]	[15]	[15]
				[15]	[15]	[15]	[15]
				[15]	[15]	[15]	[15]
				[15]	[15]	[15]	[15]

Рис. 4. Схема предварительной обработки

3.5. Подсистема классификации

Подсистема классификации – сравнивает признаковое описание входного изображения с эталонами из модели. В качестве результата входному изображению присваивается класс из модели, чей признаковый описатель оказался ближе по значению с признаковым описателем входного изображения.



Рис. 5. Структура классификации входного изображения

В качестве классификатора будет использоваться классификатор на основе метода опорных векторов.

Метод опорных векторов (*SVM*) - это набор связанных методов контролируемого обучения, которые анализируют данные и распознают шаблоны, используемые для классификации. Поскольку *SVM* является классификатором, то, учитывая набор примеров обучения, каждый из которых помечен как принадлежащий одной из двух категорий, алгоритм обучения *SVM* создает модель, которая предсказывает, относится ли новый пример к одной категории или другой. Интуитивно модель *SVM* представляет примеры как точки в пространстве, отображаемые таким образом, что примеры отдельных категорий делятся на максимально возможное расстояние. Затем новые примеры отображаются в одно и то же пространство и, как предполагается, относятся к категории, основанной на той стороне разрыва, на которую они попадают.

Цель классификатора заключается в создании модели, способной прогнозировать целевые значения экземпляров данных в наборе тестирования, для которых известны только атрибуты. В общем и целом, задачу классификации можно рассматривать как проблему двух классов, имеющую цель в том, чтобы отделить один класс от другого с помощью функции, индуцированной из доступных примеров. Основная задача здесь - создать классификатор, который будет хорошо обобщать данные.

Более формально, *SVM* строит гиперплоскость или набор гиперплоскостей в высоком или бесконечномерном пространстве, которые могут использоваться для классификации, регрессии или других задач. Интуитивно хорошее разделение достигается гиперплоскостью, которая имеет наибольшее расстояние до ближайших точек данных тренировки любого класса, так как в целом чем больше маржа, тем ниже погрешность обобщения классификатора.

Идеей *SVM* является поиск оптимальной гиперплоскости, то есть такого линейного классификатора, который обеспечит максимальную дистанцию между собой и ближайшими примерами объектов из каждого класса (пример представлен на рисунке ниже):

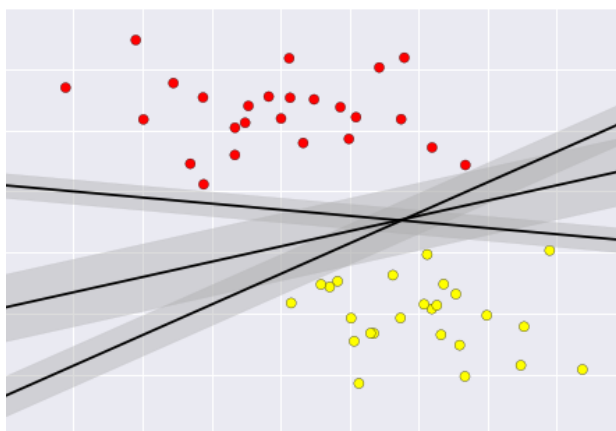


Рис. 6. Пример разделяющих линейных классификаторов

3.6. Разработка алгоритма принятия решения

Для обнаружения объектов на всем изображении будет использован метод скользящего окна. Таким образом на вход классификатора будет поступать признаковое описание только рассматриваемой области. Следовательно, если в результате классификации будет принято решение о том, что эта область принадлежит искомому классу, ее координаты будут сохранены в памяти.

После того как скользящее окно закончит проходить исходное изображение, то данное изображение уменьшается методом пирамиды Гаусса (рис. 7).

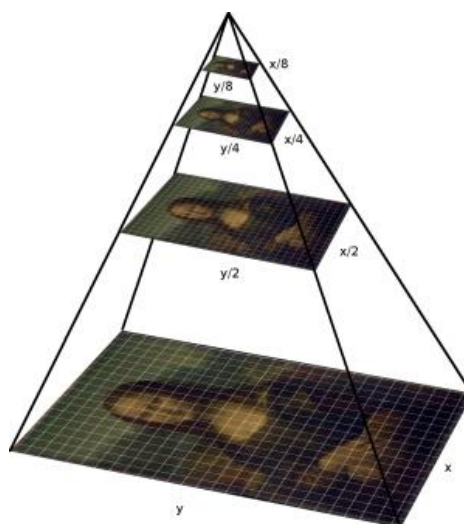


Рис. 7. Пирамида Гаусса

Пока размер изображения больше скользящего окна процесс поиска будет повторяться заново. После того как размер скользящего окна станет больше изображения поиск объектов прекращается. Далее начинается этап анализа найденных областей. Он происходит с применением метода подавления максимумов (*NMS – non-maximum suppression*).

Подавление немаксимумов (*Non-maximum suppression*) широко используется в нескольких ключевых аспектах компьютерного зрения и является неотъемлемой частью многих подходов к обнаружению объектов. Его необходимость проистекает из несовершенства алгоритмов обнаружения локализовать область своего интереса, что приводит к появлению нескольких групп рядом с реальным местоположением. Подходы к обнаружению объектов, основанные на скользящих окнах обычно создают несколько окон с высокими оценками, близкими к правильному местоположению объектов. Этот относительно плотный результат, как правило, не подходит для понимания содержимого изображения. По сути, число оконных гипотез на этом этапе просто не соответствует реальному числу объектов в изображении.

Поэтому целью *NMS* является сохранение только одного окна на группу, соответствующего точному локальному максимуму функции ответа, в идеале одно обнаружение на объект. Наиболее распространенный подход к *NMS* состоит из “жадной” итеративной процедуры, которая называется *Greedy NMS*. Процедура начинается с выбора лучшего окна, которое имеет наивысшую оценку и предполагается, что оно действительно покрывает объект. Затем, окна, которые близки к выбранному окну, подавляются. Из оставшихся окон выбирается область с наилучшей оценкой и процесс повторяется до тех пор, пока не останется смежных областей.

В результате получается исходное изображение с размеченными на нем областями.

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		23

Блок-схема алгоритма представлена на рис. 8.

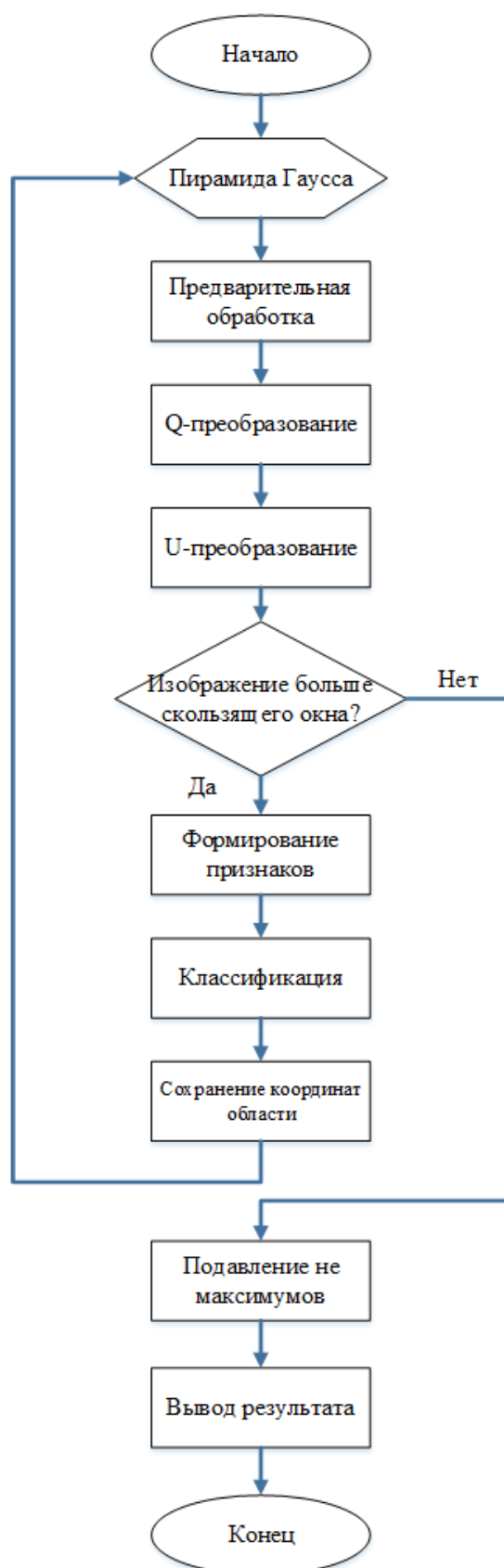


Рис. 8. Блок-схема алгоритма принятия решения

4. Разработка программных средств

Здесь рассмотрены основные функции, которые реализуют функционал системы обнаружения объектов.

Функция *norma()* предназначена для нормализации изображения.

Таблица 1 – Описание функции *norma()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>source</i>	Входной параметр	Входное изображение
2	<i>massive</i>	Выходной параметр	Изображение в виде числового массива

Функция *div()* предназначена для разбиения входного изображения на части по вертикали и горизонтали, а потом формирования векторов.

Таблица 2 – Описание функции *div()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>source</i>	Входной параметр	Входное изображение
2	<i>d</i>	Входной параметр	Количество частей на которое нужно разбить входное изображение
3	<i>res_vec</i>	Выходной параметр	Результирующий вектор

Функция *q_preobr()* производит *Q*-преобразование над входным массивом.

Таблица 3 – Описание функции *q_preobr()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>source</i>	Входной параметр	Входной массив данных
2	<i>result</i>	Выходной параметр	Результирующая преобразованная матрица

Функция *m_create()* применяет фильтр к исходному массиву данных.

Таблица 4 – Описание функции *m_create()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>source</i>	Входной параметр	Входной массив
2	<i>f</i>	Входной параметр	Накладываемый фильтр
3	<i>res</i>	Выходной параметр	Значение после наложения фильтра

Функция *create_features()* формирует признаки из обучающей выборки и записывает их в файлы с расширением *.feat*.

Таблица 5 – Описание функции *create_features()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>path</i>	Входной параметр	Путь до изображений обучающей выборки

Функция *features()* извлекает признаки из полученного массива данных.

Таблица 6 – Описание функции *features()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>prep</i>	Входной параметр	Массив признаков для каждой области исходного изображения
2	<i>x</i>	Входной параметр	
3	<i>y</i>	Входной параметр	
4	<i>min_wdw_sz</i>	Входной параметр	Размер скользящего окна
5	<i>N</i>	Входной параметр	
6	<i>M</i>	Входной параметр	
7	<i>res_vec</i>	Выходной параметр	Полученный вектор признаков

Функция *preproc()* производит предварительную обработку изображения. Формирует массив признаков для всего исходного изображения.

Таблица 7 – Описание функции *preproc()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>gray</i>	Входной параметр	Изображение в градации серого
2	<i>d</i>	Входной параметр	Количество частей на которое нужно разбить входное изображение
3	<i>min_wdw_sz</i>	Входной параметр	Размер скользящего окна
4	<i>norm_mass</i>	Выходной параметр	Нормализованный массив
5	<i>res_vec</i>	Выходной параметр	Результирующий вектор признаков для всех областей изображения
6	<i>N</i>	Выходной параметр	
7	<i>M</i>	Выходной параметр	

Функция *analyze()* анализирует разметку объектов сделанную вручную и результат работы программы.

Таблица 8 – Описание функции *analyze()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>detection</i>	Входной параметр	Обнаруженная область

Функция *overlapping_area()* определяет степень перекрытия двух обнаруженных областей.

Таблица 9 – Описание функции *overlapping_area()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>detection_1</i>	Входной параметр	Первая обнаруженная область
2	<i>detection_2</i>	Входной параметр	Вторая обнаруженная область
3	<i>overlap_area</i>	Выходной параметр	Степень перекрытия

Функция *nms()* предназначена для подавления максимумов результатов детектирования.

Таблица 10 – Описание функции *nms()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>detections</i>	Входной параметр	Массив областей
2	<i>threshold</i>	Входной параметр	Требуемая степень перекрытия
3	<i>new_detections</i>	Выходной параметр	Новый массив областей

Функция *startTest()* главная функция обнаружения. Вызывается из графического интерфейса.

Таблица 11– Описание функции *startTest()*

№	Имя параметра	Тип параметра	Комментарий
1	<i>detections</i>	Входной параметр	Массив областей
2	<i>threshold</i>	Входной параметр	Требуемая степень перекрытия
3	<i>new_detections</i>	Выходной параметр	Новый массив областей

4.1. Разработка системы анализа качества обнаружения

Для анализа качества работы системы требуется определить какое количество объектов было обнаружено, сколько пропущено и сколько ложных срабатываний. Для этого воспользуемся инструментом разметки изображения *VIA*. Данный инструмент позволяет вручную выделять интересующие область и сохранять данные в формате *JSON(JavaScript Object Notation)*. Работа в данном инструменте представлена на рис. 9.

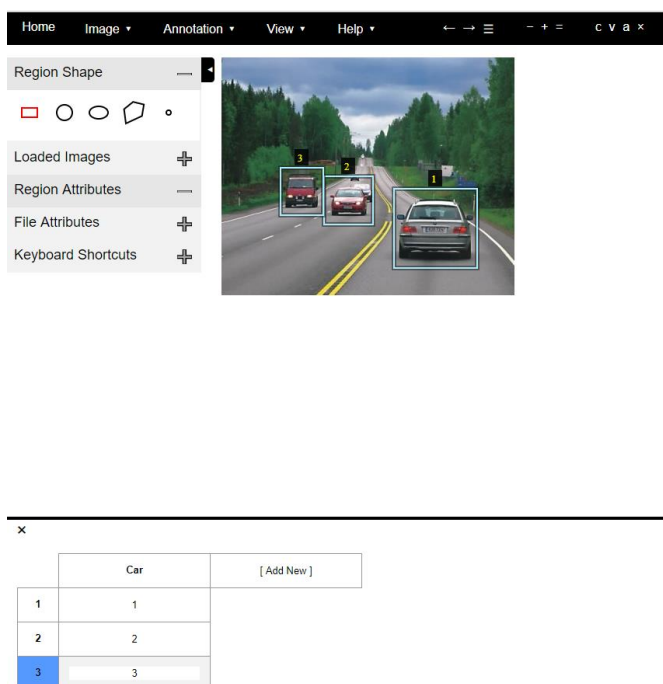


Рис. 9. Создание разметки

В результате выполнения ручной разметки изображения получился файл, который выглядит следующим образом (см. рис. 10).

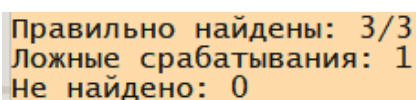
```
{
  "test3.jpg28780": {
    "fileref": "",
    "size": 28780,
    "filename": "test3.jpg",
    "base64_img_data": "",
    "file_attributes": {},
    "regions": {
      "0": {
        "shape_attributes": {
          "name": "rect",
          "x": 216,
          "y": 160,
          "width": 122,
          "height": 119,
          "region_attributes": {}
        },
        "1": {
          "shape_attributes": {
            "name": "rect",
            "x": 70,
            "y": 137,
            "width": 67,
            "height": 67,
            "region_attributes": {}
          },
          "2": {
            "shape_attributes": {
              "name": "rect",
              "x": 134,
              "y": 153,
              "width": 65,
              "height": 59,
              "region_attributes": {}
            }
          }
        }
      }
    }
  }
}
```

Рис. 10. *JSON* файл разметки

После того как была получена данная разметка этот файл необходимо «распарсить», чтобы в дальнейшем использовать содержащиеся в нем данные в системе анализа.

Система анализа будет работать следующим образом: получение информации из файла разметки(координаты областей), оценка степени перекрытия обнаруженных областей с размеченными вручную. Если степень перекрытия больше определенного порога то выделение считается засчитанным. Таким образом все области из файла разметки сравниваются с теми данными, которые в результате работы выдала система.

В результате работы системы анализа качества в консоль будет выведено следующее сообщение (см. рис. 11).



```
Правильно найдены: 3/3
Ложные срабатывания: 1
Не найдено: 0
```

Рис. 11. Сравнение разметки с результатом системы

Также стоит отметить, что при создании разметки выделяемая область должна быть больше объекта.

5. Тестирование системы

5.1. Пользовательский интерфейс

Структура с точки зрения пользователя программы представлена в виде графического интерфейса, из которого производится вызов основных модулей системы обнаружения объектов.

Главное окно пользовательского интерфейса разработанной системы представлено на рис. 12. Здесь располагаются следующие кнопки:

1. *Download model* – загрузить обученную модель
2. *Download image* – загрузить изображение
3. *Download config* – загрузить конфигурационный файл
4. *Testing* – начать тестирование системы
5. *Save model* – файл для сохранения обученной модели
6. *Positive samples* – выбор каталога с положительной выборкой
7. *Negative samples* – выбор каталога с отрицательной выборкой
8. *Training* – начать обучение модели

Поле главного окна пользовательского интерфейса визуально разделено на две части. Левая часть используется непосредственно для тестирования уже составленной модели классификатора, а правая служит для обучения модели основанной на положительной и отрицательной выборках.

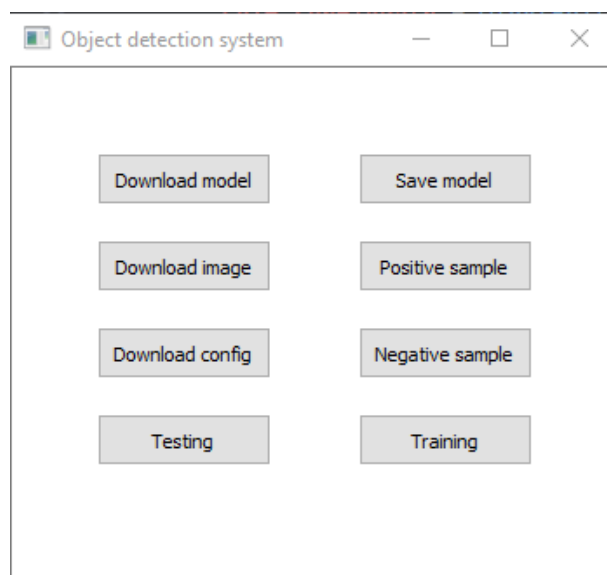


Рис. 12. Пользовательский интерфейс

Тестирование будет заключаться в проверке главной функциональности разработанной системы – формировании модели для классификации и обнаружение объектов на изображении.

Для разработки интерфейса пользователя была использован модуль PyQt5. Для создания графического интерфейса были применены следующие классы и методы:

1. *QPushButton()* – создание кнопки
2. *getSaveFileName()* – вызов диалога сохранения файла
3. *getExistingDirectory()* – вызов диалога выбора каталога
4. *getOpenFileName()* – вызов диалога открытия файла

Использование пользовательского интерфейса

1. Формирование модели классификации:

Для того чтобы обучить новую модель необходимо сначала выбрать два каталога с тренировочными изображениями: положительные и отрицательны, а затем задать путь для сохранения составленной модели. После этих действий по нажатию кнопки Training будет запущен процесс формирования модели.

После сохранения модели в окне пользовательского интерфейса появится надпись:
Classifier saved to C:/Users/Nikolay/Downloads/home.model

Это означает что модель успешно создана и сохранена по указанному пути.

2. Обнаружение объектов на изображении:

Чтобы запустить систему обнаружения требуется для начала выбрать заранее обученную модель на шаге 1. Далее необходимо выбрать исходное изображение, на котором будет производиться поиск объектов, а также конфигурационный файл. По нажатию кнопки Testing будет запущен процесс поиска.

Как только поиск начнется на экране появится входное изображение с проходящим по нему окошком.

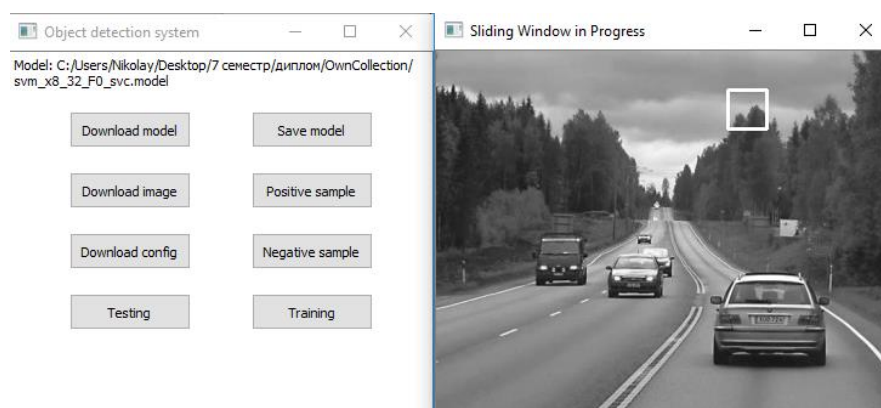


Рис. 13. Начало процесса обнаружения

По окончании поиска появится новое окно с результирующим изображением.

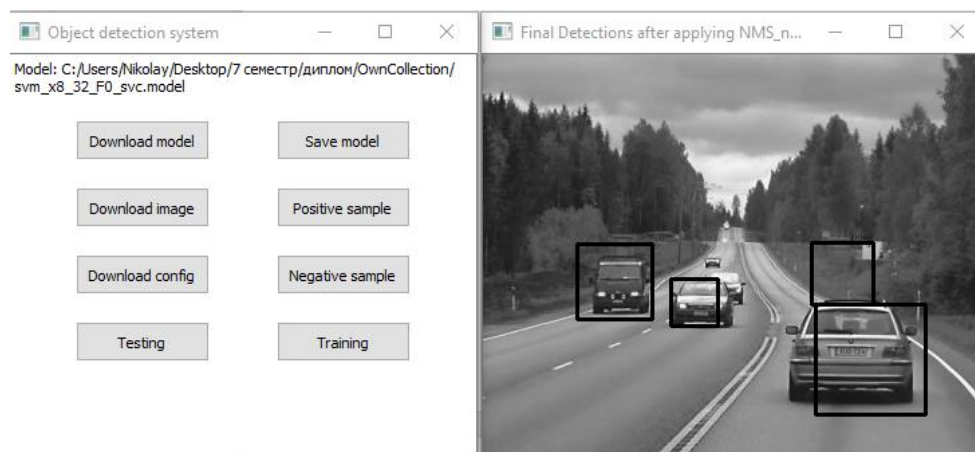


Рис. 14. Результат обнаружения в пользовательском интерфейсе

5.2. Описание набора данных

Для тестирования в качестве объекта обнаружения были выбраны фотографии автомобилей. Обучение модели проводилось на выборке, состоящей из 3500 положительных изображений и 4000 отрицательных, все изображений имеют размер 32х32.

Почти все изображения используемы для тестирования были взяты с ресурса Яндекс.Карты, что показывает их приближенность к реальным условиям.

Для тестирования использовались изображения, размер которых примерно равен 400х250 пикселей, где транспортное средство находится приблизительно в центре изображения.

Ниже приведены тестовые изображения:



Рис. 15. Тестовое изображение 1



Рис. 16. Тестовое изображение 2



Рис. 17. Тестовое изображение 3



Рис. 18. Тестовое изображение 4



Рис. 19. Тестовое изображение 5

Результаты вычислительного эксперимента

Результат обработки тестового изображения 1 показан на рис. 20

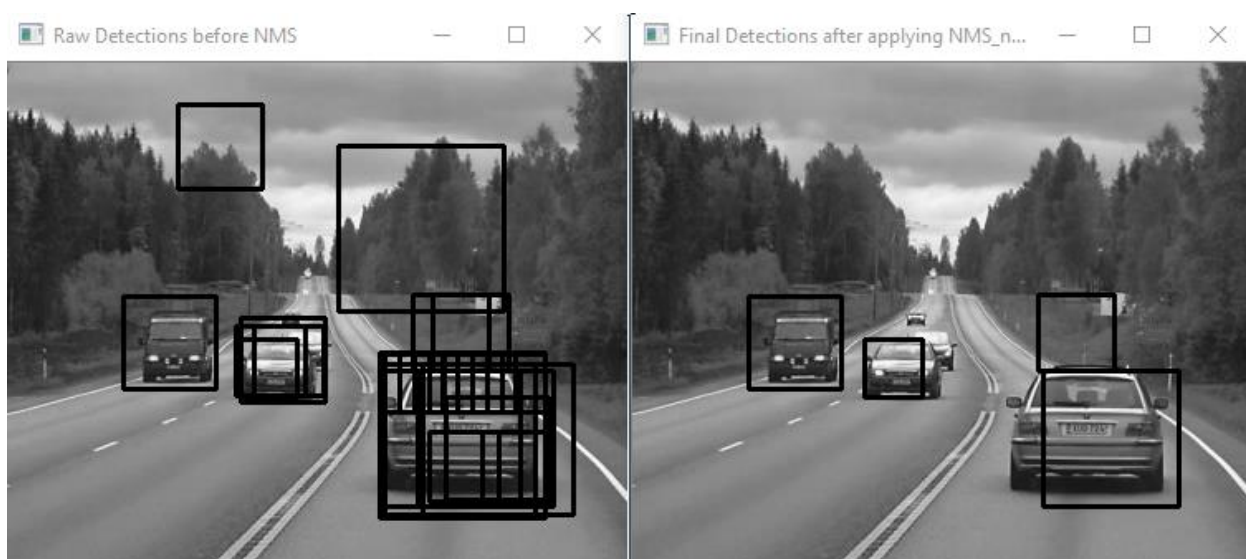


Рис. 20. Результат тестирования изображения 1

Разметка представлена на рис. 21

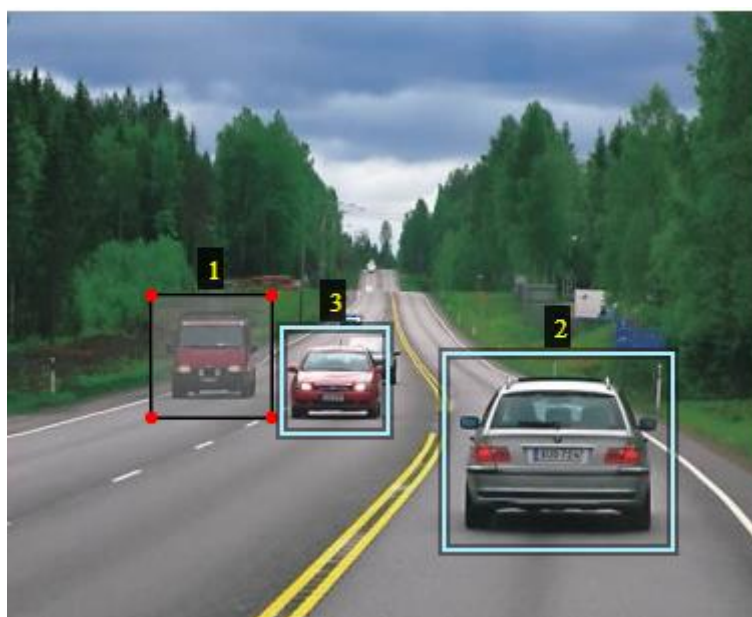


Рис. 21. Разметка изображения 1

Анализ разметки:

Правильно найдены: 3

Не найдены: 0

Ложные срабатывания: 1

					ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)	Лист
Изм.	Лист	№ докум.	Подпись	Дата		36

Результат обработки тестового изображения 1 показан на рис. 22

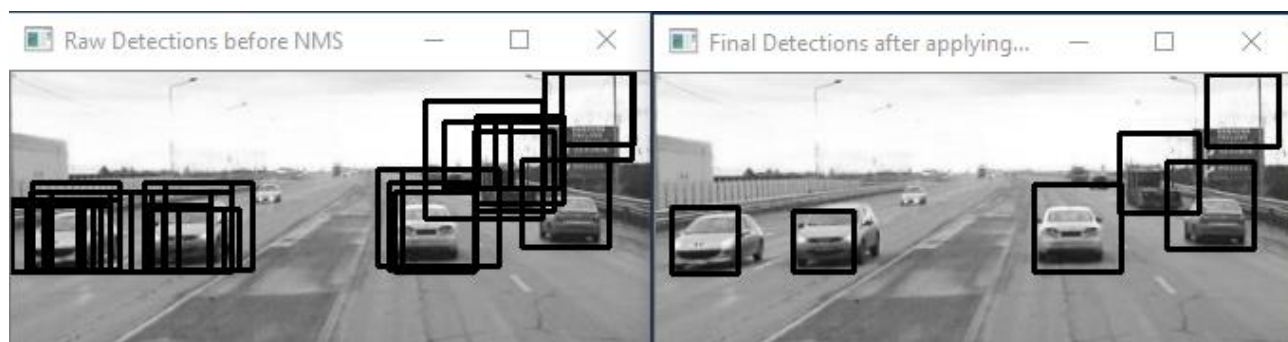


Рис. 22. Результат тестирования изображения 2

Разметка представлена на рис. 23

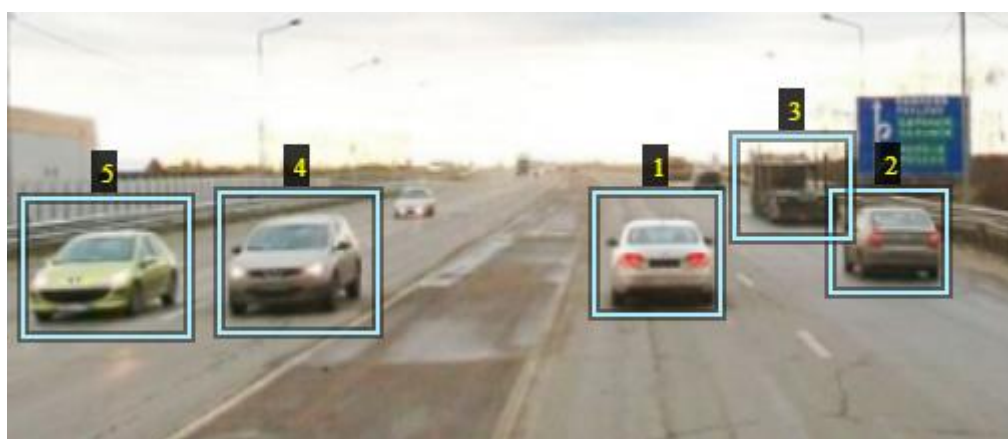


Рис. 23. Разметка изображения 2

Анализ разметки:

Правильно найдены: 5

Не найдены: 0

Ложные срабатывания: 1

Результат обработки тестового изображения 1 показан на рис. 24

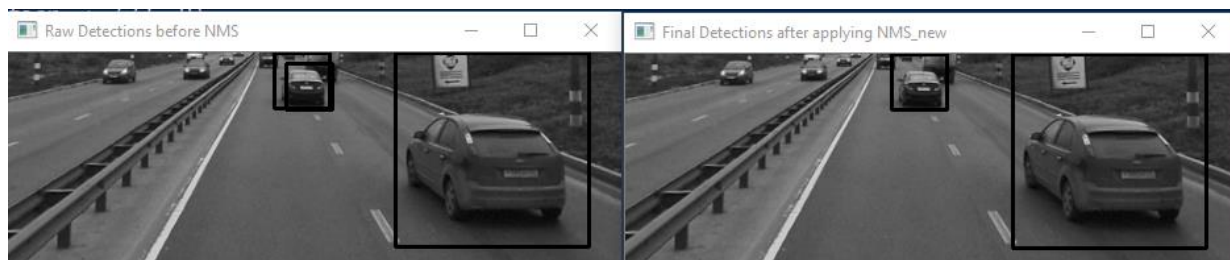


Рис. 24. Результат тестирования изображения 3

Разметка представлена на рис. 25

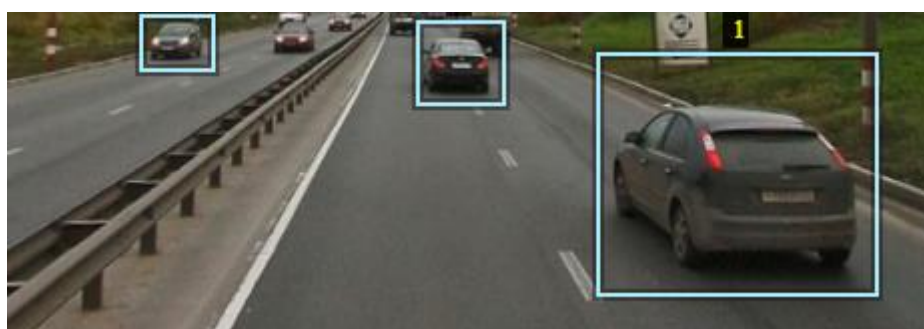


Рис. 25. Разметка изображения 3

Анализ разметки:

Правильно найдены: 2

Не найдены: 1

Ложные срабатывания: 0

Результат обработки тестового изображения 1 показан на рис. 26



Рис. 26. Результат тестирования изображения 4

Разметка представлена на рис. 27



Рис. 27. Разметка изображения 4

Анализ разметки:

Правильно найдены: 2

Не найдены: 0

Ложные срабатывания: 1

Результат обработки тестового изображения 1 показан на рис. 28



Рис. 28. Результат тестирования изображения 5

Разметка представлена на рис. 29

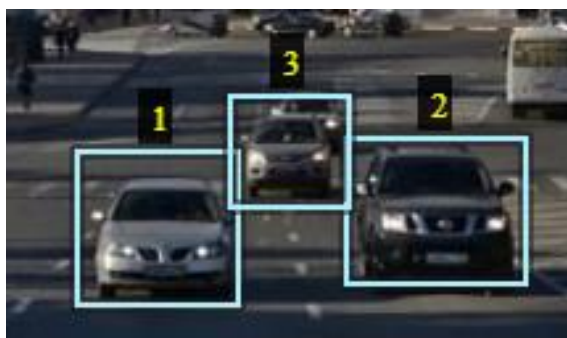


Рис. 29. Разметка изображения 5

Анализ разметки:

Правильно найдены: 2

Не найдены: 1

Ложные срабатывания: 2

Заключение

В процессе выполнения выпускной работы была спроектирована и реализована программная система поиска объекта на изображении. Система предназначена для поиска на изображении местоположения объекта и выделения области, в которой он расположена. В результате тестирования данной системы была доказана её работоспособность и возможность решать поставленную задачу. В дальнейшем планируется добавить возможность работы данной системы с уличными *IP* – камерами.

					<i>ВКР-НГТУ-09.03.03-(14-В-1)-006-2018 (ПЗ)</i>	<i>Лист</i>
<i>Изм.</i>	<i>Лист</i>	<i>№ докум.</i>	<i>Подпись</i>	<i>Дата</i>		440

Список литературы

1. Э. Таненбаум Современные операционные системы //Питер, 2017
2. Б. Страуструп Язык программирования C++ //Addison-Wesley,1985
3. П. Неймейер Программирование на JAVA //Эксмо, 2016
4. Б. Любанович Простой Python //Питер, 2016
5. Д. Адлер R in a Nutshell //O'Reilly, 2009
6. Д.С. Азаренко Детектирование объекта на изображении и определение его смещения на двух различных изображениях // Искусственный интеллект, 2013, №3, С. 90-97
7. А.Н. Алфимцев, И.И. Лычков Метод обнаружения объекта в видеопотоке в реальном времени // Вестник ТГТУ. 2011. Том 17. No 1., С. 44-55
8. Бутенко В. В. Поиск объектов на изображении с использованием алгоритма адаптивного усиления // Молодой ученый. — 2015. — №4. — С. 52-56.
9. Гребнов И. В. Новый метод детектирования человеческих лиц на цифровых изображениях // Вестник ИГЭУ» Вып.4 2008, с. 77-81
10. Joseph Redmon, Santosh Divvala, Ross Girshick, Ali Farhadi; The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 779-788
11. Utrobin V. A. Physical interpretation of the elements of image algebra // Phys. Usp. 47 1017–1032 (2004)