МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт Математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

01.03.02 Прикладная математика и информатика
(код и наименование направления подготовки, специальности)

Системное программирование и компьютерные технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Обучение каскадного классификатора для распознавания автомобилей из кадров видеопотока»

Студент К.А. Ворожейкина
(И.О. Фамилия) (личная подпись)

Руководитель к.тех.н, Т.Г. Султанов
(Ученая степень, звание, И.О. Фамилия)

Консультант К.А. Селиверстова
(Ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

Тема выпускной квалификационной работы: «Обучение каскадного классификатора для распознавания автомобилей из кадров видеопотока».

В данной выпускной работе рассматривается процесс распознавания автомобилей из кадров видеопотока.

В работе представлены обученные каскады и исполняемый файл на языке Python, который принимает в себя один из двух ключей (изображение «-i» или видеопоток «-v»). Данная программа может обрабатывать оба типа данных, указанных выше.

Выпускная квалификационная работа содержит в себе: введение, три главы, заключение, список используемой литературы и используемых источников.

Введение содержит информацию о том, насколько данная тема актуальна на сегодняшний день, а также какие поставлены цели и задачи, для реализации программы по распознаванию автомобилей из кадров видеопотока с использованием каскадных классификаторов.

В первой главе рассматриваются уже существующие и применяющиеся на практике варианты распознавания автомобилей.

Вторая глава имеет описательный характер разработанного и применяемого метода распознавания автомобилей, его характеристики и методику реализации.

В третьей главе идет речь о работоспособности реализованной программы и обученных каскадов, эффективности их работы.

В заключении представлены выводы о проделанной работе.

В работе представлено: 7 формул, 26 рисунков, 5 таблиц и 20 ссылок на использованные источники и 1 приложение. Общий же объем выпускной квалификационной работы составляет 45 страниц.

Abstract

The title of the graduation work is "training a cascade classifier based on the OpenCV library for recognizing cars in photos and videos".

The aim of the work is to implement a program for the recognition of vehicles using a cascade of classifiers and to analyze its effectiveness.

The object of the work is process of recognizing cars in photos or videos.

The subject of the work is the cascade classifier method.

The first chapter tells us about the types of recognition that have already been implemented. We also talk about their advantages and disadvantages, as well as why we can't use implemented programs.

The second chapter tells us about the chosen method and its implementation by training cascading classifiers. There is also a detailed description of all the parameters that were used during the training.

The third chapter about the performance of the implemented program and the effectiveness of the trained cascades. This chapter also provides analysis of different trained cascades.

The recommendation for the result of this work is an implemented program using trained cascades for recognizing automobiles in photos and videos.

Application area - the developed algorithm can be used in solving computer vision problems in the field of security.

The relevance of the work allows to solve problems more effectively in the field of computer vision and in particular the detection and recognition of objects.

Содержание

Введение

Теория распознавания образов — раздел кибернетики, развивающий теоретические основы и методы классификации и идентификации предметов, явлений, процессов, ситуаций и так далее, в целом объектов, которые характеризуются конечным набором некоторых свойств и признаков. Технологии по распознаванию объектов на изображении и тем более на видеопотоке уже давно изобретены и устоялись в повседневной жизни.

Компьютерное зрение проявляется в повседневной жизни довольно часто: будь то поиск преступников по камере наблюдения или угнанный автомобиль. Данное направление крайне перспективно, так как возможность отследить что-либо с помощью компьютерного зрения без использования человеческого ресурса крайне полезна. Именно поэтому и саму цель выпускной квалификационной работы, а именно «Обучение каскадного классификатора для распознавания автомобилей из кадров видеопотока» можно по праву считать актуальной и значимой на сегодняшний день. Целью данной работы является обучение каскадного классификатора и реализация программы для распознавания автомобилей из кадров видеопотока.

Для достижения поставленной цели нужно решить следующие задачи:

1. Проанализировать научную и учебно-методическую литературу, необходимую для понимания процесса по распознаванию объектов.
2. Проанализировать и провести сравнение существующих алгоритмов по распознаванию автомобилей.
3. Выбрать метод реализации распознавания автомобилей.
4. Реализовать распознавание автомобилей из кадров видеопотока.
5. Протестировать реализованный алгоритм.
6. Определить эффективность реализованного алгоритма.
7. Основываясь на полученных данных сделать вывод об эффективности работы реализованных алгоритмов.

1 Теоретическое обоснование задачи распознавания автомобилей из кадров видеопотока

1.1 Описание исследуемой задачи по распознаванию автомобилей

Исходя из ситуации на сегодняшний день, потребность в распознавании объекта, группы объектов на изображении или видеопотоке крайне велика. Не важно, лицо ли это определенного человека из огромной толпы или же государственный номер транспортного средства в потоке трафика. Вся эти данные, полученные в результате работы алгоритмов компьютерного зрения, могут быть обработаны и проанализированы любыми учреждениями, работающими в государственной или же в частной сфере.

Распознавание автомобилей на данный момент активно используется в государственных структурах, например, камеры дорожного движения активно фиксируют нарушения, которые допустил водитель автомобиля, а также с легкостью могут распознавать автомобильные гос. номера.

Можно выделить ряд актуальных задач, где необходимо использовать распознавание автомобилей:

- для реализации поиска автомобилей без государственных номеров (например, найти угнанный автомобиль, примерно зная, откуда он, выехал и куда мог поехать);
- для реализации в системе «Умный город» алгоритм поможет сразу выявлять нарушение (например, если водитель ведет себя неадекватно, то на главный пост охраны сразу поступит отснятый видеоматериал и данные об автомобиле) [11];
- для реализации программы, которая будет считать количество свободных мест на парковочных площадках (например, водитель сможет заранее просматривать количество свободных и занятых мест);

- для реализации программы по сбору статистики об аварийных ситуациях на определенных участках дорог;
- для реализации программ для беспилотных автомобилей [5].

Стоит так же отметить, что задача по распознаванию автомобиля нетривиальна и не имеет единого решения. Поэтому необходимо изучить вопрос по распознаванию объектов и реализовать программу, которая сможет выполнять поиск автомобилей на кадрах видеопотока.

1.2 Обзор реализованных алгоритмов распознавания автомобилей

Исследуя ранее обозначенную проблему по распознаванию автомобилей на кадрах видеопотока, были найдены уже реализованные алгоритмы. Рассмотрим алгоритмы, которые на сегодняшний день способны осуществить распознавание автомобилей.

«Intlab Auto Mmr» – это комплект средств разработки (SDK). Он предназначен для интеграции в сторонние приложения с целью распознавания автомобилей на фото или видео [12].

Этот комплект является частью линейки продуктов от INTLAB и предоставляет возможность детектирования цвета, марки и модели транспортного средства в режиме реального времени. Примеры работы данной программы представлены на рисунке 1.

Рисунок 1 – Пример результата работы «Intlab Auto Mmr»

Однако на сегодняшний день существуют не только SDK или блоки кода по детектированию транспорта, но и облачные решения. Примером можно взять такую API как «Sighthound Cloud».

«Sighthound Cloud API» - это API для обнаружения лиц, просто людей, а также транспортных средств: а именно их марок, моделей и государственных регистрационных номеров [17].

Отличие данной API от предыдущего примера заключается в том, что все вычисления проводятся в облаке компании Sighthound и результатом данной работы является файл формата JSON, который в дальнейшем можно использовать в своих целях. Пример результата данного API можно увидеть на рисунке 2.

Рисунок 2 – Результат работы «Sighthound Cloud API»

Продолжая рассматривать «облачные» решения, можно затронуть одну из разработок Российской компании, именуемой «Яндекс».

«Auto.ru» от Яндекс - в 2016 году Яндекс выпустил обновление для своего дочернего проекта «auto.ru», мобильное приложение которого умеет

распознавать марку и модель авто. Ну а после распознавания, уже благодаря сайту, приложение подскажет стоимость автомобиля и прочую информацию о нем. Пример работы приложения можно увидеть на рисунке 3.

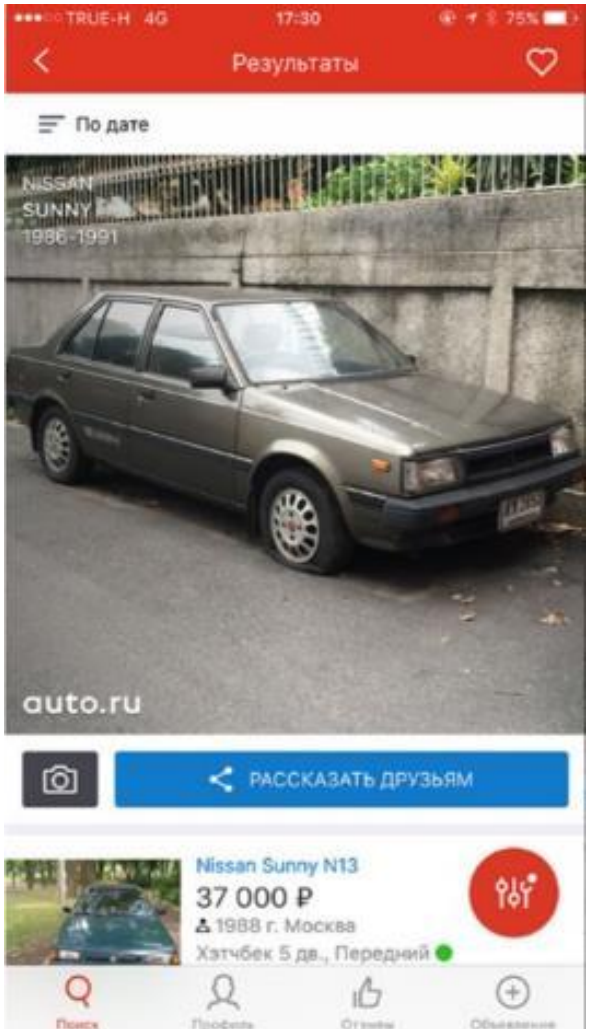

Рисунок 3 – Результат работы приложения «auto.ru»

К сожалению, данное распознавание работает только внутри приложения «auto.ru» и нет никакой возможности воспользоваться данным методом в своих целях.

Существует куда больше библиотек, сервисов, API и не только по распознаванию объектов на изображении или видео. Однако все эти способы и наработки имеют ряд своих минусов. В таблице 1 представлено сравнение ранее описанных сервисов.

Таблица 1 – Сравнительная таблица качеств перечисленных сервисов

Название	Возможность интеграции	Интернет-соединение	Стоимость
IntLab auto MMR	полная интеграция в код	не требуется	разовая оплата продукта в момент интеграции
SighthoundCloud.	полная интеграция в код	требуется постоянно	тарификация по количеству обращений
Auto.ru	отсутствует	требуется постоянно	бесплатное использование

Исходя из представленной информации, можно сделать вывод о том, что оптимального решения нет, так как одни сервисы слишком дорогостоящие, другим нужен постоянный доступ к сети Интернет, а будут ли покрываться потребности купленным продуктом заранее не известно.

В связи с этим было принято решение самостоятельно разобраться в принципах и методах распознавания (детектирования) объекта или группы объектов на изображении или видеопотоке и реализовать собственный подход к распознаванию автомобилей, который можно будет легко интегрировать в иные программные продукты.

При реализации программы собственноручно будет иметься точное понимание того, как работает распознавание, как работать с программой и какие слабые места необходимо будет улучшать в дальнейшем.

1.3 Постановка задачи на реализацию программы для распознавания автомобилей из кадров видеопотока

Требуется обучить каскадный классификатор для распознавания автомобилей из кадров видеопотока, а так же реализовать программу на языке Python, которая будет использовать в работе вышеупомянутый обученный каскад для распознавания и выделения автомобилей.

Реализованная программа должна обеспечивать следующие функциональные возможности:

- распознавание автомобилей на кадрах видеопотока должно проходить в независимости от наличия подключения к сети Интернет;
- обученный каскад должен иметь возможность интегрироваться в любые программные продукты;
- реализованная программа должна работать со статичным кадром и с видеопотоком;
- реализованная программа должна получать на вход файл и проводить самостоятельную обработку файла перед началом распознавания;
- реализованная программа должна иметь возможность распознавать более одного автомобиля в кадре;
- реализованная программа в случае успешного распознавания должна выделять распознанные объекты (автомобили);
- реализованная программа должна иметь возможность экстренного прерывания распознавания кадров видеопотока по нажатию кнопки «q».

Исходя из этого, необходимо учесть данный перечень функциональных возможностей при реализации программы по распознаванию автомобилей из кадров видеопотока.

2 Математическая формулировка модели распознавания автомобилей из кадров видеопотока

2.1 Анализ и выбор вычислительного метода для реализации программы по распознаванию автомобилей из кадров видеопотока

В предыдущей главе были рассмотрены уже реализованные методы, которые так или иначе не удовлетворяют заданным требованиям. Также были поставлены функциональные требования для реализуемой программы по распознаванию автомобилей из кадров видеопотока. Так как необходимо понимать принцип распознавания, выбор пал на свободно распространяемую библиотеку компьютерного зрения OpenCV. Данная библиотека представляет собой набор алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом [5].

Библиотека OpenCV реализована на большинстве основных языков, таких как C/C++, Python, Java, Ruby, Matlab, Lua* и многих других. Лицензия BSD означает, что данная библиотека может свободно использоваться в академических и коммерческих целях [14].

Изначально библиотека разрабатывалась под C++, однако позже была портирована и на другие языки, в том числе и Python, на котором в дальнейшем будет реализовано распознавание автомобильного транспорта. Выбран именно язык Python, так как реализация на данном языке простая и имеет более простую отладку. Так же Python выбран потому, что цель сводится исключительно к распознаванию автомобиля из кадров видеопотока и не требует дополнительных манипуляций с полученными результатами на данном этапе.

Для реализации распознавания рассмотрим метод каскадного обучения, на базе каскадов Хаара.

Данный способ детектирования предметов на изображении был основан на машинном обучении, основные методы и принципы которого были указаны в статье Пола Виолы (Paul Viola) и Майкла Джонса (Michael Jones) [8]. При детальном рассмотрении каскадов Хаара, можно заметить, что обучение протекает медленно, однако при правильном подходе к обучению, обученный каскад будет работать крайне эффективно. Готовый обученный каскад Хаара принимает изображение, после чего с помощью своих алгоритмов определяет, есть ли на изображении искомый объект. Иными словами, каскад занимается классификацией - разделяет входные данные на два типа: присутствует или отсутствует искомый объект.

Правильно обученный каскад крайне эффективно распознает искомый объект и обладает неплохой устойчивостью к небольшим отклонениям. Изначально каскад Хаара рассматривался как метод обнаружения лиц. Само распознавание проходит достаточно хорошо даже с учетом того, что входные данные будут представлены с углом наклона до 30 градусов. При увеличении угла вероятность распознавания резко уменьшается [6],[10]. Данный каскад можно обучить на распознавание и других объектов, как например дорожные знаки или автомобили.

Говоря о распознавании с помощью обученных каскадов, стоит затронуть принцип работы окна сканирования. Задача по распознаванию объекта на кадрах видеопотока в общем виде будет выглядеть следующим образом. Имеется изображение, внутри которого присутствуют искомые объекты. Кадр будет представлен двухмерной матрицей из пикселей с определенным размером (высота × ширина), где каждый отдельно взятый пиксель имеет свое значение: от 0 до 255. По завершении выполнения своей работы, данный алгоритм может определить на изображении черты искомого объекта и соответственно отметить их.

Признаки Хаара позволяют описать определенные объекты на изображении, за счет перепада их цвета вследствие теней или иных причин,

создающих контраст. Например, с помощью признаков Хаара можно четко выделить на изображении перепады яркости между глаз.

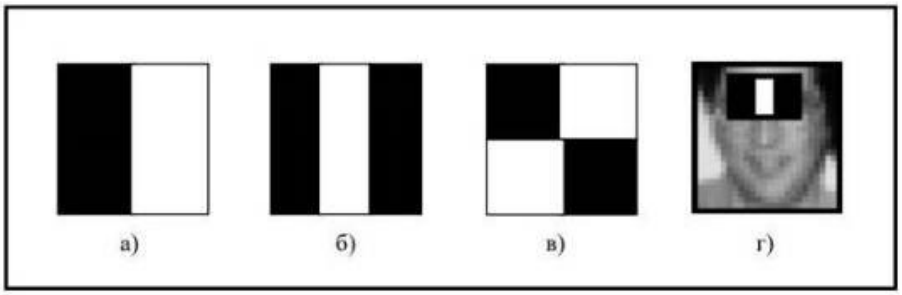

Рисунок 4 – Прямоугольные признаки Хаара, задействованные во время обучения классификатора: а) граничный признак; б) линейный признак; в) диагональный признак; г) пример расположения признака относительно окна сканирования

Рисунок 5 – Признак Хаара на изображении

Рисунок 5 является примером признака Хаара на изображении: белые прямоугольники — первая группа областей, а черный — вторая. Значение признака Хаара — это разность сумм яркостей пикселей первой и второй группы. В математической форме это будет выглядеть так, как представлено на формулах 1-3.

$$a_i = \sum_{i=y_{a_i}}^{y_{a_i}+h_{a_i}-1} \sum_{j=x_{a_i}}^{x_{a_i}+w_{a_i}-1} x_{ij} \quad (1)$$

$$b_i = \sum_{i=y_{b_i}}^{y_{b_i}+h_{b_i}-1} \sum_{j=x_{b_i}}^{x_{b_i}+w_{b_i}-1} x_{ij} \quad (2)$$

$$h = \sum_{i=1}^{N_a} a_i - \sum_{i=1}^{N_b} b_i \quad (3)$$

Где x_{ij} - яркость пикселя с координатами [i, j];

a_i - сумма яркостей пикселей в i-й области первой группы;

b_i - сумма яркостей пикселей в i-й области второй группы;

h - значение признака Хаара для этого изображения;

h_{a_i} - высота i-й области первой группы;

w_{a_i} - ширина i-й области первой группы;

h_{b_i} - высота i-й области второй группы;

h_{b_i} - ширина i-й области второй группы;

N_a - количество областей первой группы;

N_b - количество областей второй группы;

h - значение признака Хаара для взятого изображения.

Вычисление значения признаков Хаара является нахождение разности сумм пикселей на областях изображения внутри белых и черных прямоугольников. Как граничные, так и линейные признаки могут иметь как вертикальную (как можно наблюдать на рисунке 4), так непосредственно и горизонтальную ориентации [20].

Так же библиотека OpenCV может задействовать и дополнительные признаки в некоторых методах, как представлены на рисунке 6 [3].

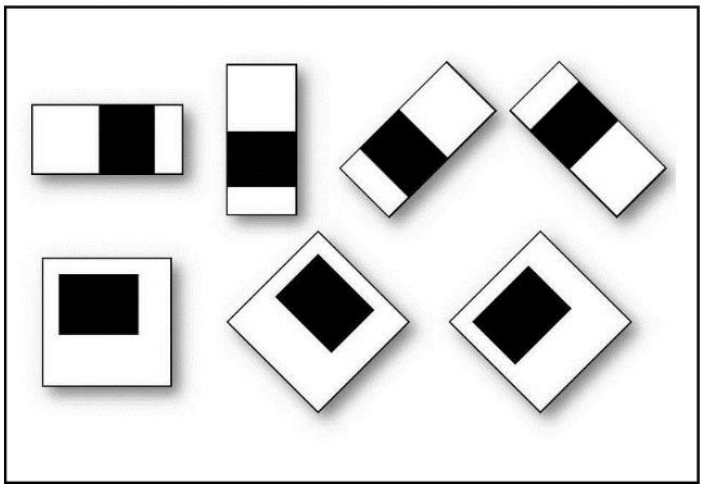

Рисунок 6 – Дополнительные признаки в OpenCV

Для эффективного вычисления признаков Хаара используются интегральные представления изображений.

Сделаем предположение, что есть некое черно-белое изображение $f(y,\ x)$ с четко установленным размером $M{\cdot}N$. Интегральное представление $If(y,x)$ изображения $f(y,\ x)$ примем цифровое изображение размером $(M{+}1){\cdot}(N\ +\ 1)$, значение пикселей которого вычисляется по формуле 4.

$$I_f\ (y,x) = \begin{cases} \sum_{y`<y,x`<x}^{0} f(y`,x`), & \text{при } y{=}0 \text{ или } x{=}0 \\ & \text{при } x{>}0 \text{ и } y{>}0 \end{cases} \quad (4)$$

Благодаря данному рекурсивному методу, всего за один проход можно построить интегральное представление изображения:

Представим значения пикселей интегрального представления изображения первых строки и столбца в виде нуля по формуле 5.

$$I_f(0,x) = 0 \text{ и } I_f(y,0) = 0 \quad (5)$$

Для всех остальных пикселей $y > 0$ и $x > 0$ с помощью рекурсивной функции, значение можно вычислить по формуле 6.

$$I_f(y,x) = I_f(y-1,x) + I_f(y,x-1) - I(y-1,x-1) + f(y-1,x-1) \quad (6)$$

Если брать интегральное представление изображения, то сумму пикселей внутри прямоугольной области можно вычислить с помощью четырех арифметических операций [11].

На рисунке 7 представлено визуальное отображение области, по которой будет после взято интегральное представление [2],[8].

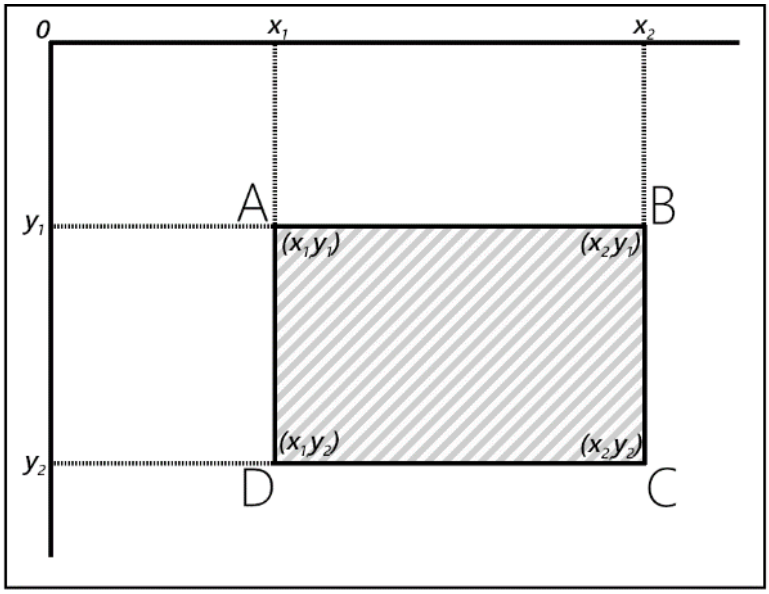

Рисунок 7 – Интегральное представление области изображения

Благородя формуле 7 в независимости от размера зоны прямоугольной формы, формула для вычисления суммы пикселей отдельно взятого изображения внутри прямоугольной области будет состоять всего из трех арифметических операций.

$$\sum_{\substack{y_1 \le y < y_2 \\ x_1 \le x < x_2}} f(y,x) = I_f(y_2,x_2) - I_f(y_1,x_2) - I_f(y_2,x_1) + I_f(y_1,x_1) \quad (7)$$

Следовательно, благодаря использованию интегрального представления изображения, можно достаточно просто вычислить признаки Хаара всего за несколько математических операций, причем не имея зависимости от размера задействованной области.

Это довольно мощное интегральное представление, поскольку оно используется не только в каскадах Хаара, но и в других областях по обработке цифровых изображений или видеопотоком, например, когда нужно провести расчет вейвлет-преобразований или дескрипторов особых точек.

2.2 Обучение каскадного классификатора для реализации программы по распознаванию автомобилей из кадров видеопотока.

Ранее был выбран метод каскадного классификатора Хаара, с помощью которого и будет происходить обучение каскадов и реализация программы по распознавания автомобилей.

Также будет производиться измерение эффективности работы программы, которая будет использовать обученные каскады. Для этого обучение каскадов будет проходить в две стадии.

- на первой стадии будут использоваться выборка из положительных изображений размером 80×60 пикселей с аналогичным размером векторного представления;
- на второй стадии будут использоваться изображения среднего размера, а именно 800×600 пикселей, векторное представление которых будет иметь размер 100×75.

В процессе обучения каскадов будет происходить измерение нагрузки на компьютер.

Обучение каскадов будет производиться на компьютере под операционной системой Linux, с версией библиотеки OpenCV 4.1.2.

Ниже представлены характеристики компьютера, на котором будет производиться обучение:

- процессор Intel(R) Core (TM) i3-5005U CPU @ 2.00GHz (2 ядра, 4 логических);
- оперативная память 12,0 ГБ DDR3 1600MHz;
- жесткий диск ST1000LM024 HN-M101MBB.

Реализацию самого алгоритма по распознавания автомобилей можно разделить на несколько этапов:

- подготовительный этап, в рамках которого необходимо собрать нужное количество положительных и отрицательных снимков. Привести положительные снимки к единому размеру;
- обучение каскадов и измерение затраченных ресурсов в процессе обучения;
- реализация программы для использования обученных каскадов;
- сбор статистики и анализ эффективности обученных каскадов.

На первой стадии было подготовлено 332 положительных и 2000 отрицательных изображений. Для успешного обучения количество отрицательных изображений должно превышать количество положительных изображений минимум в два раза. Также стоит отметить, что размер отрицательных изображений может быть любым, а вот размер положительных изображений строго должен быть одинаковым.

В качестве положительной выборки были взяты фотографии машины с разных ракурсов, данные фотографии кадрированы таким образом, чтобы в кадре находился только автомобиль. В данном случае использовались фотографии с одной машиной, но при обучении можно использовать изображения, на которых изображено больше одного искомого объекта. На рисунке 8 представлены варианты из положительной выборки.

Рисунок 8 – Пример положительных изображений размером 80×60 (до сжатия)

В качестве отрицательной выборки были взяты случайные картинки, на которых отсутствует автомобиль, а также были вырезаны фрагменты из положительных изображений, куда не входит искомый объект (автомобиль). На рисунках 9 и 10 представлен пример отрицательных изображений.

Рисунок 9 – Пример отрицательных изображений

Рисунок 10 – Пример фрагментов, вырезанных с положительного изображения

После подготовки изображений, они были переименованы и разделены по разным папкам:

– good - папка с положительными изображениями размера 80×60;
– bad - папка с отрицательными изображениями разного размера.

Чтобы подготовить негативные изображения необходимо, с помощью команды как показано на рисунке 11, создать список с негативными изображениями.

```
find ./bad/ -name '*' > bad.txt
```

Рисунок 11 – создание списка с негативными изображениями

При создании списка добавляется лишняя запись, которая не содержит в себе имени изображения. На рисунке 12 представлен список отрицательной выборки, который содержит лишнюю запись. Данная запись впоследствии

прерывает обучение на начальной стадии. Следовательно, необходимо убрать лишнюю запись до начала обучения.

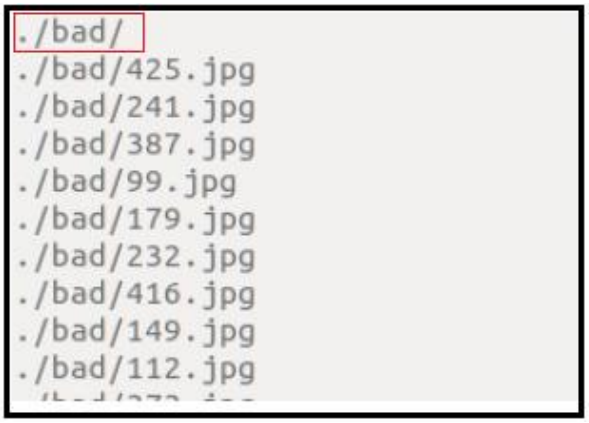

Рисунок 12 – Список отрицательной выборки с лишней записью

Для обработки информации о позитивных изображениях необходимо создать файл с помощью команды, представленной на рисунке 13.

```
find ./good/ -name '*' -exec echo \{\} 1 0 0 80 60 \; > cars.info
```

Рисунок 13 – Создание файла со списком положительной выборки

Где:

- name “*” — это имя файлов (положительных изображений);
- 1 — это количество положительных объектов на изображении (на изображении может присутствовать больше одного положительного объекта, тогда необходимо указать соответствующее количество);
- 0,0,80,60 - координаты прямоугольника на изображении, в котором находится объект.

С помощью команды «cat cars.info» проверяется, как были записаны позитивные изображения. На рисунке 14 показано содержимое файла с информацией о положительной выборке.

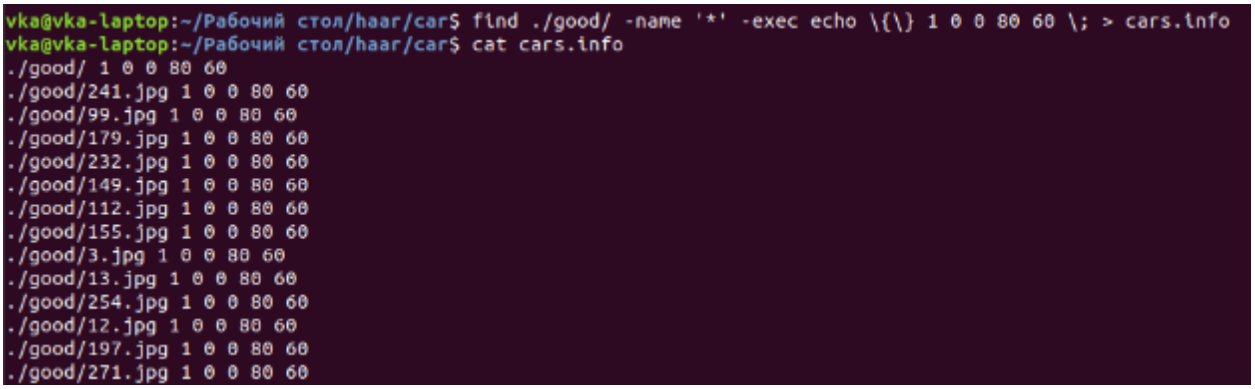

Рисунок 14 – Содержимое файла с информацией о положительной выборке

Как было сказано ранее, из файла необходимо убрать лишнюю запись, из-за которой процесс обучения аварийно прерывается.

После создания файла с положительными изображениями необходимо создать их векторное представление. Для этого используется утилита «opencv_createsamples».

С помощью данной утилиты создается набор положительных образцов в формате, который поддерживается как приложениями, так утилитами «opencv_haartraining» и «opencv_traincascade» [13]. На рисунке 15 показано создание положительных образцов в формате «.vec».

Рисунок 15 – Создание положительных образцов в «.vec» формате

Для проверки созданной коллекции изображений используется команда «opencv_createsamples -vec cars.vec -w 80 -h 60».

В результате отображаются созданные изображения в векторном формате. На рисунке 16 представлен пример изображений в векторном формате.

Рисунок 16 – Векторные изображения

После проведения всех подготовительных этапов начинается процедура обучения каскада. Для этого используется утилита «opencv_traincascade».

«Opencv_traincascade» – это более новая версия в сравнении с «opencv_haartraining», написанная на C++ в соответствии с API OpenCV 2.x [13].

Основное различие между этими двумя утилитами заключается в том, что «opencv_traincascade» поддерживает функции Haar и LBP (Local Binary Patterns) [13].

Входными аргументами утилиты «opencv_traincascade» являются [7],[9],[13],[18]:

– data <cascade_dir_name> - папка, в которую будет помещен обученный каскад;
– vec <vec_file_name> - vec файл позитивных изображений (созданный утилитой opencv_createsamples);
– bg <background_file_name> - файл с описанием изображений, не содержащих искомый объект;
– numPos<number_of_positive_samples> - число позитивных изображений;
– numNeg<number_of_negative_samples> - число негативных изображений;
– numStages <number_of_stages> - число этапов обучения;
– w <sampleWidth>, h <sampleHeight> - ширина и высота изображений для обучения;
– featureType <{HAAR(default),LBP}> - тип каскада (HAAR, LBP);
– minhitrate - коэффициент, определяющий качество обучения, то есть процент правильных обнаружений. (По умолчанию устанавливают значение 0.999, если установлено 0.999, то есть по исходной выборке будет не более, чем 1 - 0.999 =0.1% пропусков целей. Чем выше коэффициент, тем выше уровень ложных тревог);
– maxFalseAlarmRate - уровень ложной тревоги. (По умолчанию используется значение 0.5). В случае, если выборка подобрана правильно (верное соотношение объема выборки к количеству стадий обучения), то уровень требуемой тревоги будет быстро достигнут и обучение будет остановлено;
– mode ALL - использование полного комплекта признаков Хаара. От данного аргумента зависит скорость обучения, а также точность алгоритма. В случае если объект не меняет ориентацию, полный комплект признаков Хаара использовать не нужно [16];

– precalcValBufSize <precalculated_vals_buffer_size_in_Mb>, precalcIdxBufSize <precalculated_idxs_buffer_size_in_Mb> - выделяемая память для обучения.

Для обучения каскада команда будет иметь следующий вид: «opencv_traincascade -data haar_car -vec cars.vec -bg bad.txt -numPos 332 -numNeg 2000 -numStages 5 -w 80 -h 60 –minhitrate 0.999 – maxFalseAlarmRate 0.5 -featureType HAAR -precalcValBufSize 2048 -precalcIdxBufSize 2048 -mode ALL». На рисунке 17 показано отображение команды в терминале.

Рисунок 17 – Команда для запуска процесса обучения

После запуска в терминале распечатываются параметры, и начинается обучение каскада. Каждый этап распечатывает анализ во время обучения. Обучение начинается с нулевой стадии. Каждая строка во время обучения представляет собой объект, который обучается и содержит некоторые выходные данные о его соотношения поставленной цели и уровня ложных срабатываний.

В ходе обучения было зафиксировано время на пройденных стадиях:

– на нулевой стадии обучение заняло 1 час 16 минут;
– на первой стадии обучения заняло 2 часа 17 минут;
– на второй стадии обучение заняло 3 часа 8 минут;

– в начале третьей стадии система оповестила о том, что порог указанного уровня ложных срабатываний достигнут и нет смысла продолжать дальнейшее обучение.

В сумме обучение составило 5 часов 44 минуты. На рисунке 18 и на рисунке 19 представлены стадии обучения и их аргументы.

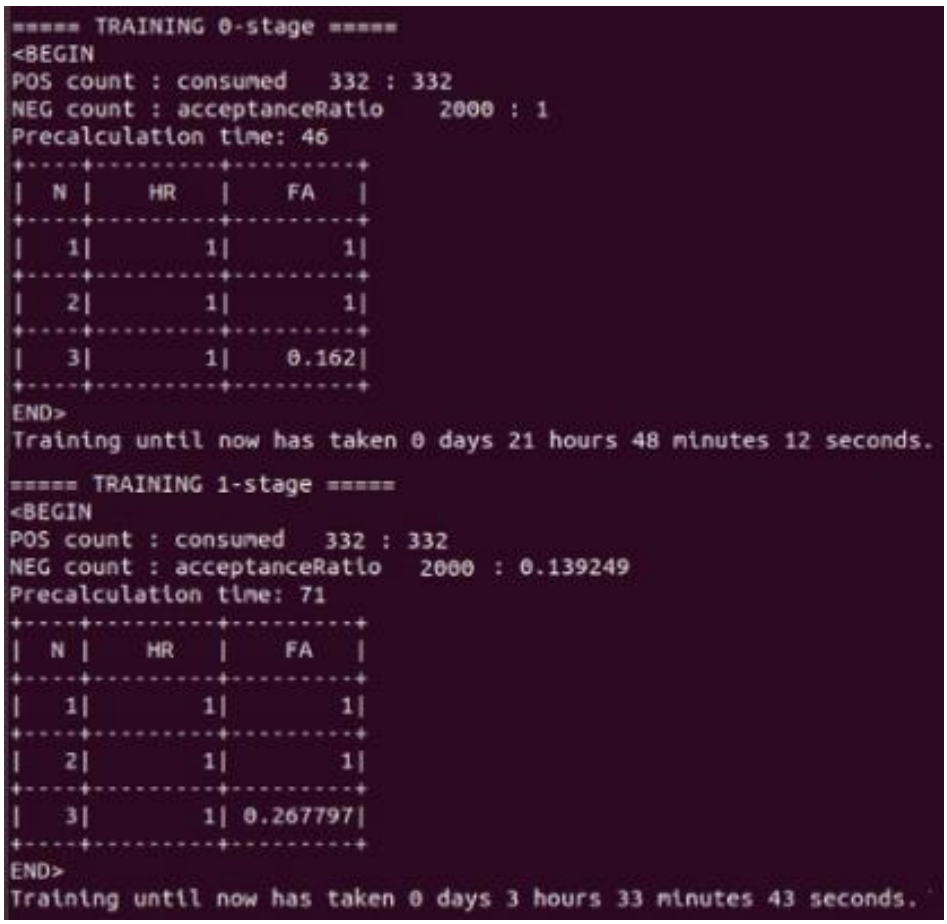

Рисунок 18 – Нулевая и первая стадии обучения

Рисунок 19 – Вторая и третья стадии обучения

После завершения процесса в указанном каталоге создаются файлы с расширением XML. На каждую итерацию создается один файл. В конце обучения создается общий каскад файл с именем «classifier.xml». Этот файл является обученным каскадом, который можно использовать для обнаружения автомобилей с помощью OpenCV.

Стоит отметить, что каскад был обучен на положительных изображениях, размером 80×60 пикселей.

Аналогичным образом можно провести обучение на изображениях размером 800×600 пикселей. Данное обучение займет гораздо больше времени, так как вес изображений, как и сложность векторных представлений увеличивается. Создается векторное представление со следующими значениями: «opencv_createsamples -vec cars.vec -w 100 -h 75».

Начинается процесс обучения с параметрами «opencv_traincascade -data haar2 -vec cars.vec -bg bad.txt -numPos 332 -numNeg 2000 -numStages 5 -w 100 -h 75 –minhitrate 0.999 – maxFalseAlarmRate 0.5 -featureType HAAR -precalcValBufSize 1024 -precalcIdxBufSize 1024 -mode ALL». На рисунке 20 показано отображение команды в терминале.

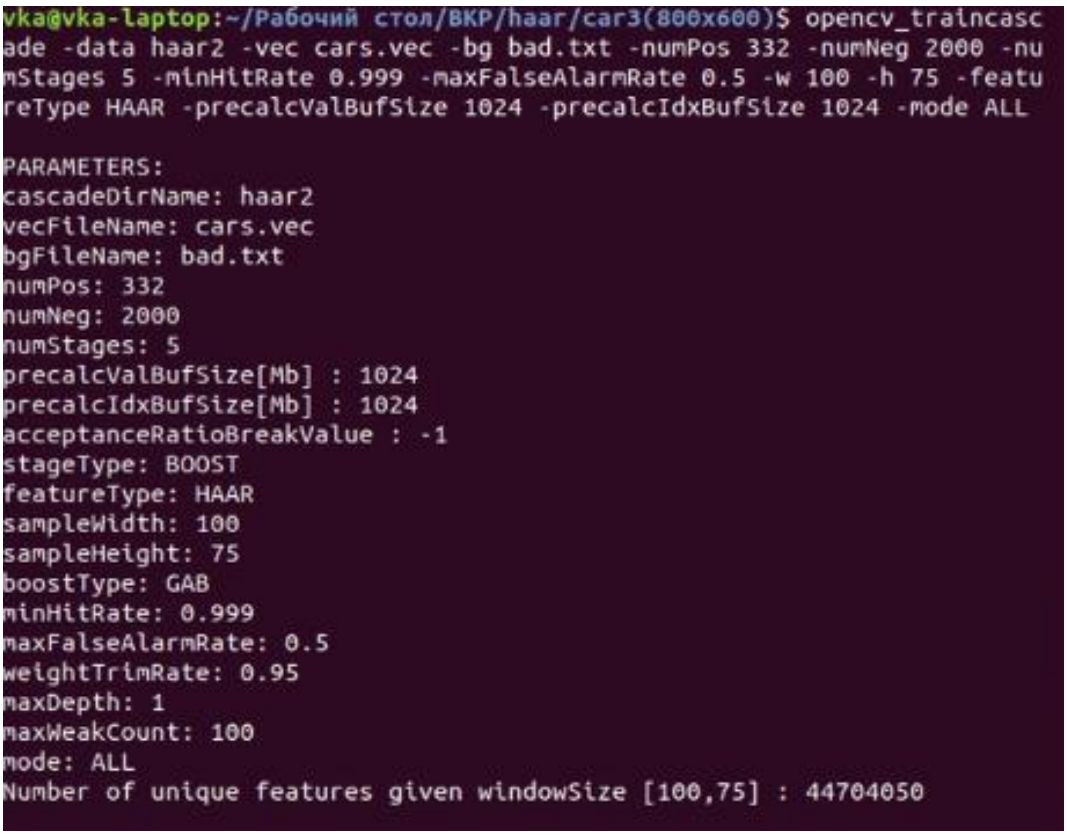

Рисунок 20 – Команда для запуска процесса обучения

Как и ранее было зафиксировано время в процессе обучения на разных стадиях:

- на нулевой стадии обучение заняло 21 час 48 минут;
- на первой стадии обучения заняло 31 час 44 минут;
- на второй стадии обучение заняло 37 часов;
- в начале третьей стадии система оповестила о том, что порог указанного уровня ложных срабатываний, достигнут, и нет смысла продолжать дальнейшее обучение.

В сумме обучение составило 3 дня 18 часов 34 минуты. На рисунке 21 представлены стадии обучения и их аргументы.

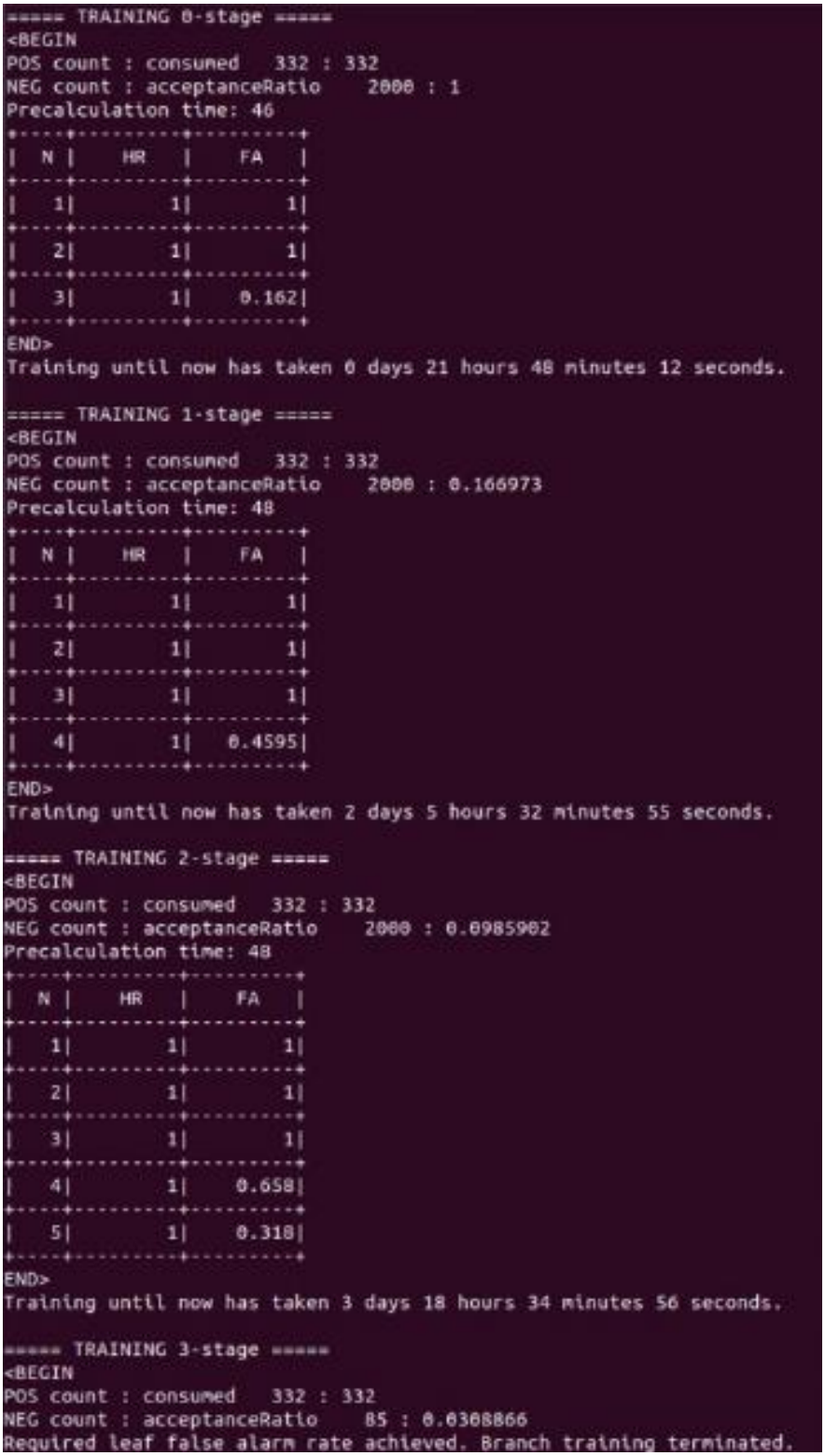

Рисунок 21 – Вторая и третья стадии обучения

После обучения был получен обученный каскад в виде файла с расширением XML. Обученные каскады представлены в приложении А.

Для использования обученных каскадов была реализована программа на языке Python [4],[15],[19].

Используя обученный каскад, программа пытается обнаружить искомый объект, в данном случае автомобиль на отдельно взятом изображении или кадре видеопотока. С помощью входных параметров определяется, что необходимо распознать.

Параметр «-i» служит для распознавания автомобиля на отдельно взятом кадре (изображении) (image), параметр «-v» отвечает за распознавания автомобиля на кадрах видеопотока (video). Полная программная реализация представлена в приложении А.

Результатом работы программы является отдельное окно с изображением или последовательно открывающиеся окна с кадрами видеопотока, на которых в случае успешного распознавания будет выделен, синим прямоугольником искомый объект (автомобиль).

Таким образом, после проведения анализ методологии распознавания объектов на изображении или кадрах видеопотока было проведено обучение нескольких каскадов. Так же была реализована программа на языке Python. Основной функционал реализованной программы будет заключаться в том, что она будет использовать обученные каскады для поиска искомого объекта на изображении или видеопотоке.

3 Проведение тестирования и анализ эффективности реализованной программы по распознаванию автомобилей из кадров видеопотока

3.1 Описание реализованной программы по распознаванию автомобилей на базе обученных каскадов

В результате выполнения выпускной квалификационной работы были изучены методы каскадного обучения, были обучены каскады и реализована программа для распознавания автомобилей на кадрах видеопотока с использованием обученных каскадов.

Реализованная программа имеет следующие возможности:

- работа с фото и видео;
- выделение искомого объекта;
- возможность распознавания более одного объекта.

Перед запуском программы необходимо удостовериться, что обученный каскад, исходный файл, а также материал для распознавания лежат в одной папке. В ином случае при запуске программа выдаст ошибку.

Для запуска программы из терминала необходимо запустить Python файл, передавая ему параметр типа данных и название файла с расширением.

Параметр «-i» запускает распознавание автомобиля по указанному изображению, параметр «-v» запускает распознавание автомобиля по указанному видеопотоку.

После запуска программы на экране появляется изображение или кадр из видеопотока, на котором обученный каскад завершив поиск, в случае успеха выделяет распознанный объект в рамку, в данном случае - автомобиль или группа автомобилей.

Для демонстрации работоспособности рассмотрим работу обученного каскада в программе на рисунке 22.

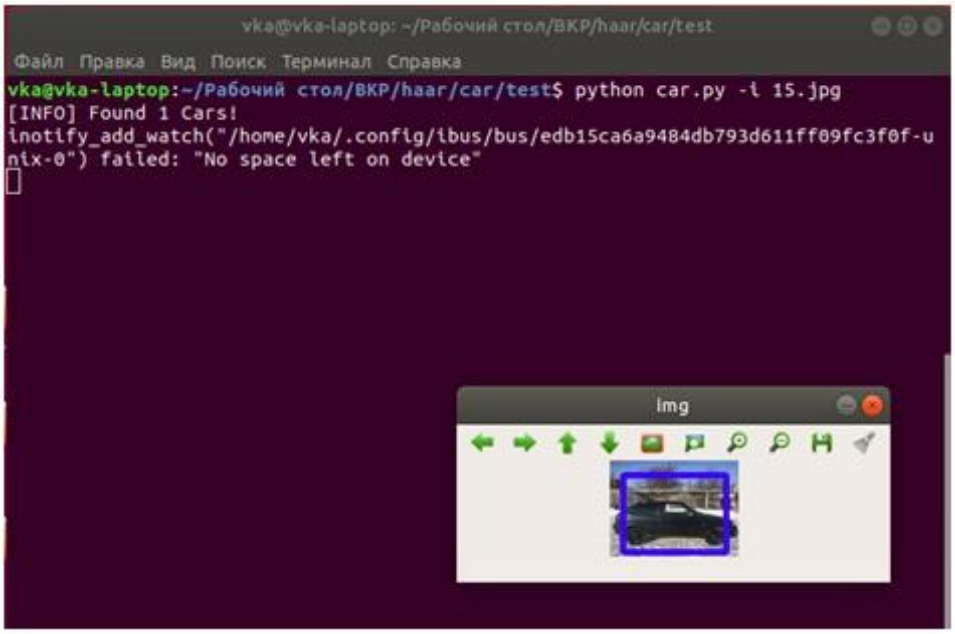

Рисунок 22 – Результат работы программы с изображением

В результате работы программы получаем найденный автомобиль, отмеченный синим прямоугольником.

Для распознавания по видео запустим программу и выбираем видеофайл. На рисунке 23 представлена работа программы с видеопотоком.

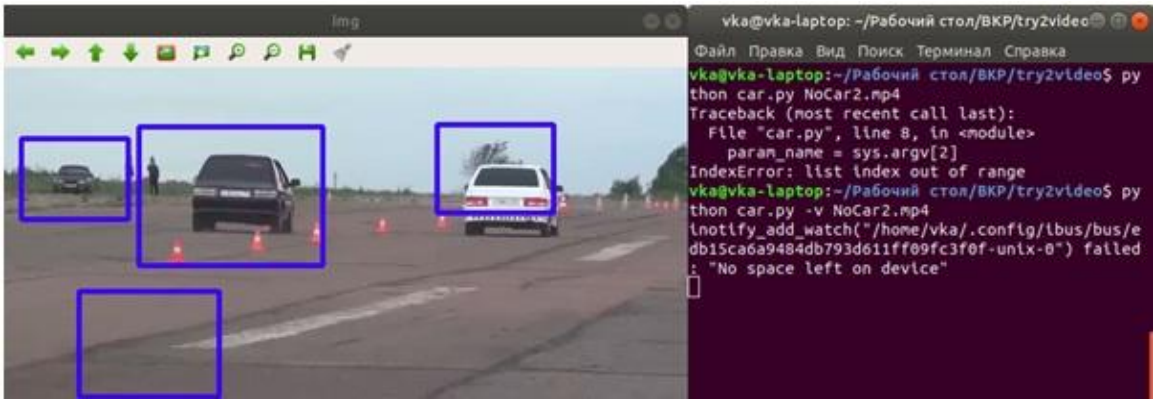

Рисунок 23 – Результат работы программы с видеопоком

В результате работы получаем последовательное отображение кадров видеопотока, на которых распознаются автомобили. Рисунок 23 служит примером того, что обученный каскад может распознавать более одного автомобиля. Но также можно заметить, что у обученного каскада могут быть ложные срабатывания, для проверки качества обученных каскадов необходимо провести тестирование с разными параметрами.

3.2 Тестирование реализованной программы по распознаванию автомобилей с использованием обученных каскадов и анализ эффективности их работы

Для того чтобы понять, насколько эффективно работает обученный каскад в программе, необходимо рассмотреть сразу несколько аспектов. Для начала стоит рассмотреть стадию обучения.

Во время обучения каскадов производилось измерение следующих параметров:

- время обучения;
- температура процессора, а именно температуру каждых из его ядер и общую;
- нагрузка на оперативную память.

При обучении каскада на изображениях размера 80×60 и векторном представлении 80×60 были получены следующие данные:

- общее время обучения составило 5 часов 44 минуты;
- максимальная температура процессора составила 71 градус;
- средняя температура за весь процесс обучения составила 60 градусов;
- максимальная температура ядер составила 69 градусов;
- нагрузка на оперативную память была 8.5Gb/11.6Gb (72%).

Полученные данные можно сравнить с теми данными, которые были получены при обучении каскада на изображениях размера 800×600 и векторном представлении 100×75:

– общее время обучение составило 90 часов 34 минуты;
– максимальная температура процессора составила 73 градуса;
– средняя температура за весь процесс обучения составила 66 градусов;
– максимальная температура ядер составила 72 градуса;
– нагрузка на оперативную память была 10.8Gb/11.6Gb (93.1%).

На рисунке 24. отображено графическое представление полученных данных.

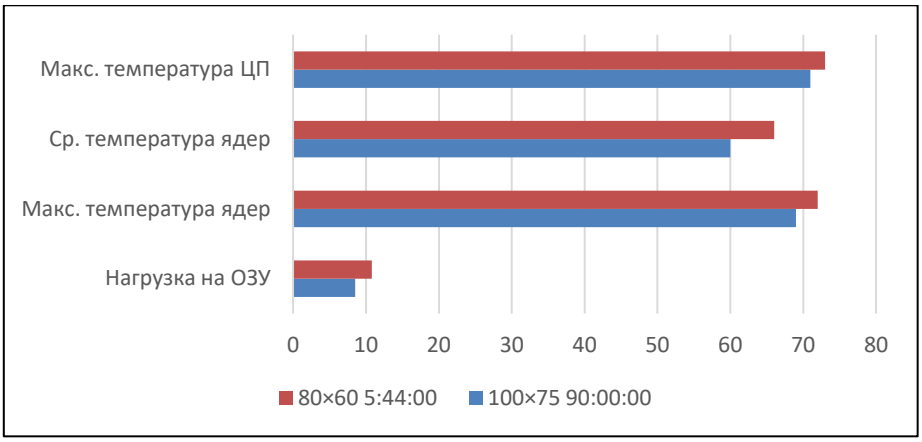

Рисунок 24 – Сравнение нагрузки во время обучения двух каскадов

Исходя из графика, можно сделать вывод о том, что при обучении каскада на изображении размера 80×60 и векторном представлении 80×60. компьютер загружался пусть и не сильно, но заметно меньше.

Исходя их данного показателя сложно сделать вывод о том, что каскад, который был обучен за меньший промежуток времени и с меньшими затратами ресурсов компьютера будет работать лучше или хуже.

Для определения того, какой же каскад эффективнее справляется с обнаружением, был проведен ряд экспериментов. Стоит заметить, что

рекомендуется использовать изображения примерного того же разрешения, что и разрешение, на котором был обучен каскад.

Для проведения тестирования было взято 10 цветных и 10 черно-белых изображений, которые не участвовали в процессе обучения. А именно изображения следующих размеров:

– 60×80 пикселей;
– 100×75 пикселей;
– 110×83 пикселей;
– 120×90 пикселей;
– 130×98 пикселей.

Данный диапазон был выбран не случайно. Каскады были обучены на векторных представлениях 80×60 и 100×75. Именно поэтому для проведения анализа эффективности распознавания берется диапазон, куда точно входят эти векторные представления. Учитывая, что не всегда на изображении находится автомобиль целиком, может находиться только его часть или автомобиль будет достаточно удален на изображении. В таблице 2 представлено количество положительных и ложных срабатываний при поиске автомобилей на цветных изображениях.

Таблица 2 – Количество положительных и ложных срабатываний при распознавании автомобиля на цветных изображениях

Размер изображений	Кол-во цветных изображений	Кол-во положительных срабатываний обученного каскада размером 80×60	Кол-во положительных срабатываний обученного каскада размером 800×600	Кол-во ложных срабатываний обученного каскада размером 80×60	Кол-во ложных срабатываний обученного каскада размером 800×600
80×60	10	0	0	10	10
100×75	10	10	0	0	10

Продолжение таблицы 2

Размер изображений	Кол-во цветных изображений	Кол-во положительных срабатываний обученного каскада размером 80×60	Кол-во положительных срабатываний обученного каскада размером 800×600	Кол-во ложных срабатываний обученного каскада размером 80×60	Кол-во ложных срабатываний обученного каскада размером 800×600
110×83	10	10	3	0	7
120×90	10	9	7	1	3
130×98	10	6	9	4	1

Аналогичным образом проверялась работа каскадов на черно-белых изображениях. Результаты работы распознавания представлены в таблице 3.

Таблица 3 – Количество положительных и ложных срабатываний при распознавании автомобиля на черно-белых изображениях

Размер изображений	Кол-во черно-белых изображений	Кол-во положительных срабатываний обученного каскада размером 80×60	Кол-во положительных срабатываний обученного каскада размером 800×600	Кол-во ложных срабатываний обученного каскада размером 80×60	Кол-во ложных срабатываний обученного каскада размером 800×600
80×60	10	0	0	10	10
100×75	10	10	0	0	10
110×83	10	10	6	0	4
120×90	10	8	8	2	2
130×98	10	6	10	4	0

На рисунке 25 и рисунке 26 графически представлены полученные результаты по распознаванию автомобилей на цветных и черно-белых изображениях.

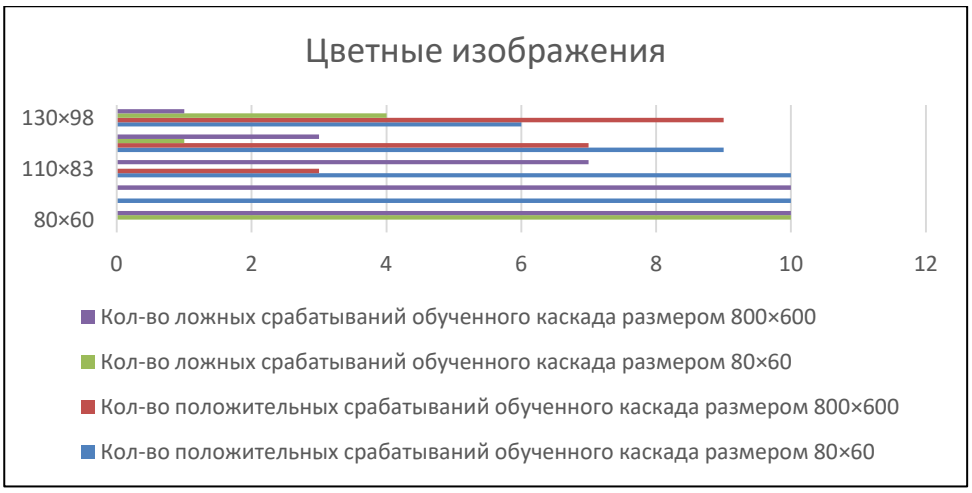

Рисунок 25 – Результаты распознавания цветных изображений

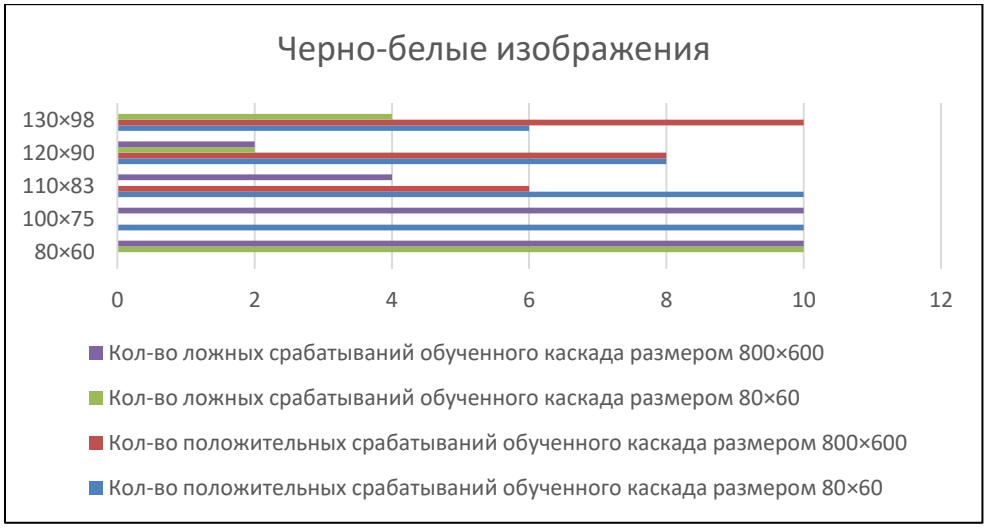

Рисунок 26 – Результаты распознавания черно-белых изображений

Исходя из данных отображенных данных, можно сделать вывод о том, что результаты практически идентичны и нет необходимости обесцвечивать изображения перед началом распознавания. Из ранее полученных результатов можно заметить, что каскады могут быть подвержены ложным срабатываниям. Для этого были взяты изображения, на которых отсутствует искомый объект. В таблице 4 представлен результат работы каскадов с ложными изображениями.

Таблица 4 – Количество ложных срабатываний

Размер изображений	Кол-во изображений, на которых отсутствует автомобиль	Кол-во найденных объектов обученного каскада размером 80×60	Кол-во найденных объектов обученного каскада размером 800×600
80×60	10	0	0
100×75	10	0	0
110×83	10	1	0
120×90	10	1	0
130×98	10	2	0

Исходя из полученных результатов, можно сделать вывод о том, что каскад, обученный на изображениях размера 800×600 менее подвержен ложным срабатываниям, чем каскад размера 80×60.

Для того, чтобы полностью оценить работоспособность каскада, было проведено тестирование распознавания автомобилей на кадрах видеопотока. Для этого использовалось 3 цветных видео следующих размеров:

– 800×600;
– 110×83;
– 800×600, где масштаб автомобиля сжат до размера 110×83.

Стоит отметить, что на видео присутствуют и другие автомобили. Положительным срабатыванием будет считаться верное выделение автомобиля. Ложным срабатыванием будет считаться выделение других объектов на видео. Полученные результаты отображены в таблице 5.

Таблица 5 – Результаты распознавания на кадрах видеопотока.

Размер видео	Кол-во найденных объектов обученного каскада размером 80×60	Кол-во найденных объектов обученного каскада размером 800×600 (векторное представление 100×75)	Кол-во ложных срабатываний обученного каскада размером 80×60	Кол-во ложных срабатываний обученного каскада размером 800×600 (векторное представление 100×75)
800×600	0	0	30	60
110x83	1	1	0	0
800×600, где масштаб автомобиля сжат до размера 110×83	3	5	23	19

В результате всех проведенных сравнений можно сделать вывод, что эффективнее всего работает каскад, которому были предоставлены для обучения изображения 800×600, из которых были созданы векторные представления размером 100×75.

Тем не менее, ложных срабатываний можно было бы получить гораздо меньше, если бы была возможность воспользоваться более мощным компьютером, на котором бы происходило обучение.

Заключение

Во время исследования вопроса по распознаванию автомобильного транспорта в рамках выпускной квалификационной работы были достигнуты следующие результаты:

- были изучены принципы распознавания объектов;
- было обучено два каскада;
- была реализована программа на языке Python, которая может работать с любыми обученными каскадами на поиск чего-либо на кадрах видеопотока;
- были протестированы обученные каскады на различных фото и видео материалах;
- была собрана статистика об эффективности и проведен ее анализ.

Как было сказано ранее, основная проблема всего обучения заключается в том, что при работе над выпускной квалификационной работой использовался компьютер, технических характеристик которого недостаточно для более детального обучения каскадов на большой выборке с ужесточенными параметрами ложного срабатывания и добавлениями дополнительных условий. Если каскад будет обучен на выборке, которая минимум в 10 раз превышает текущую, а обучение будет запущено с более низким коэффициентом ложных срабатываний, то вероятность верного распознавания автомобиля на кадрах видеопотока будет гораздо выше, чем у обученных ранее каскадов.

Таким образом, подводя итоги по выполненным этапам в данной работе, можно сделать вывод о том, что была выполнена поставленная цель, а именно была реализованная программа, использующая обученный каскад для распознавания автомобилей из кадров видеопотока. Был произведен анализ эффективности, после чего были выявлены недостатки и выделен оптимальный способ обучения каскадных классификаторов.

Список используемой литературы и используемых источников

1. Константин Кулаков [Электронный ресурс]. – Режим доступа: https://kostyakulakov.ru/opencv-обучение-каскада-хаара/ – Обучение каскада на автомобильных номерах. 2015.

2. Моделирование и распознавание 2D/3D образов [Электронный ресурс]. – Режим доступа: https://api-2d3d-cad.com/viola-jones-method/ – Выделение объектов по методу Виолы Джонса. 2018.

3. Научное общество GraphiCon [Электронный ресурс]. – Режим доступа: https://www.graphicon.ru/html/2012/conference/RU2%20-%20Vision/gc2012yuzhakov.pdf – Расширенный набор характеристик каскада. 2012.

4. Портал информатики для гиков [Электронный ресурс]. – Режим доступа: http://espressocode.top/python-haar-cascades-for-object-detection/ – Python | каскады Хаара для обнаружения объектов. 2019.

5. Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/company/yandex/blog/437674/ – беспилотные автомобили. 2019.

6. Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/133826/ – Метод Виолы Джонса. 2011.

7. Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/208092/ – Обучение OpenCV каскада Хаара. 2014.

8. Carnegie Mellon University [Электронный ресурс]. – Режим доступа: https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf – Оригинальная статья Паула Виолы и Майкла Джонсона про расширенные Каскады. 2001.

9. CodingRobin [Электронный ресурс]. – Режим доступа: https://coding-robin.de/2013/07/22/train-your-own-opencv-haar-classifier.html – Train Your Own Opencv Haar Classifier. 2013.

10. Computational Vision [Электронный ресурс]. – Режим доступа: http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf – Статья Паула Виолы и Майкла Джонсона о детектировании лиц. 2004.

11. Future2day [Электронный ресурс]. – Режим доступа: https://future2day.ru/umnyj-gorod-texnologii-i-perspektivy-razvitiya/ – Умный город. 2019.

12. IntLab [Электронный ресурс]. – Режим доступа: https://www.intlab.com/products/intlab-auto-mmr – SDK по распознаванию автомобилей. 2020.

13. OpenCV [Электронный ресурс]. – Режим доступа: https://docs.opencv.org/2.4/doc/user_guide/ug_traincascade.html – Cascade Classifier Training (attributes). 2019.

14. OpenCV [Электронный ресурс]. – Режим доступа: https://opencv.org – Официальный сайт библиотеки OpenCV. 2020.

15. Python [Электронный ресурс]. – Режим доступа: https://www.python.org – Официальный сайт языка Python. 2020.

16. Semantic Scholar [Электронный ресурс]. – Режим доступа: https://pdfs.semanticscholar.org/4548/61b54cad3fdc8b255decd2dbe9ad0abf48d6.pdf – Fast and Efficient Rotated Haar-like Features using Rotated Integral Images. 2009.

17. Sighthound [Электронный ресурс]. – Режим доступа: https://www.sighthound.com/products/cloud – Облачная API по распознаванию автомобилей. 2020.

18. TechCave [Электронный ресурс]. – Режим доступа: https://techcave.ru/posts/55-obuchenie-kaskadnogo-klassifikatora-v-opencv-opencv-traincascade-opencv-createsamples.html – Обучение каскадного классификатора в OpenCV. 2015.

19. Towards Data Science [Электронный ресурс]. – Режим доступа: https://towardsdatascience.com/computer-vision-detecting-objects-using-haar-

cascade-classifier-4585472829a9 – Computer Vision — Detecting objects using Haar Cascade Classifier. 2019.

20. Towards Data Since [Электронный ресурс]. – Режим доступа: https://towardsdatascience.com/a-guide-to-face-detection-in-python-3eab0f6b9fc1 – A guide to Face Detection in Python. 2019.

Приложение А

Файловые ресурсы реализованной программы для распознавания автомобилей из кадров видеопотока

Полный код программы на языке Python, а так же обученные каскады в формате XML находятся на приложенном к данной бакалаврской работе компакт-диске. Все файловые ресурсы находятся в папке под названием «Приложение А».

Приложенные файловые ресурсы описаны в таблице А.1.

Таблица А.1 – Описание приложенных файловых ресурсов на компакт-диске

Наименование файла	Описание файла
car.py	Программный код на языке Python, который использует обученные каскады для распознавания автомобилей из кадров видеопотока
cascade1.xml	Каскад, который был обучен на изображениях размером 80×60. Файл имеет расширение XML и содержит в себе параметры, полученные при обучении
cascade2.xml	Каскад, который был обучен на изображениях размером 800×600. Файл имеет расширение XML и содержит в себе параметры, полученные при обучении