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)

03.02.02 Математическое обеспечение и администрирование информационных систем
(код и наименование направления подготовки, специальности)

Мобильные и сетевые технологии
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Распознавание образа студента для автоматического учёта посещаемости».

Студент Р.В. Утарбаев
(И.О. Фамилия) (личная подпись)

Руководитель канд. пед. наук, доцент, О.М. Гущина
(Ученая степень, звание, И.О. Фамилия)

Консультант М.В. Дайнеко
(Ученая степень, звание, И.О. Фамилия)

Тольятти 2021

Аннотация

Тема данной выпускной квалификационной работы «Распознавание образа студента для автоматического учёта посещаемости».

В данной выпускной квалификационной работе рассматривается процесс распознавания не только лиц, но и детектирование определенного образа отдельно взятого студента.

В работе представлены исполняемые файлы для каждой стадии, а также обученные каскады. Программа запускается через терминал, а также не принимает никаких ключей, поскольку работа ведется исключительно с видеопотоком.

Данная выпускная квалификационная работа представляет из себя: введение, три основные главы, заключение, а также список используемой литературы и задействованных источников.

Введение затрагивает актуальность данной темы, а также производит постановку задач для реализации распознавания образа студента для автоматического учёта посещаемости.

Первый раздел содержит описание уже существующих методов распознавания образа человека.

Второй раздел содержит непосредственное описание той модели, которая необходима к реализации.

Третий раздел представляет собой тестирование работоспособности реализованной программы и обученных каскадов, а также их сравнение.

Заключение представляет из себя выводы, сделанные в результате проделанной работы.

В работе использовано 7 формул, 2 таблицы, 30 рисунков и 25 ссылок на внешние ресурсы Общий объем выпускной квалификационной работы составил 52 страницы.

Abstarct

The title of the graduation work is Recognizing a student's image for automatic registration of attendance.

The aim of the work is to implement a programme for detecting students' images by using a cascade classifier and a LBP model, as well as to analyze its effectiveness.

The object of the investigation is the process of detecting students' images in the video stream.

The subject of the research is the cascade classifier method and the development of an LBP model.

The graduation work deals with the process of recognizing not only faces, but also detecting a particular image of an individual student.

The work presents the executables for each stage, as well as the cascades. The programme is run via the terminal, and also does not accept any keys, since the work is carried out exclusively by means of the video stream.

The first chapter provides some information about the existing methods of detecting images of certain people. It also reveals the disadvantages of implemented methods and explains why they are considered to be not suitable.

The second chapter gives some information about the chosen method of detecting students' images and explains how they are implemented by using the cascade classifier and the LPB model. This chapter also describes in details the process of training the cascade classifier and data involved during this training.

The third chapter assesses the effectiveness of the programme implemented. It also gives some information about the state of the system during training, as well as compares the cascades under investigation.

The result of this work is a programme that detects students' images and a window application designed for managing the received data.

The developed algorithm can be used in educational institutions in order to keep track of students' attendance and safety.

Содержание

Введение

Сама по себе теория распознавания образов – это раздел кибернетики. Данный раздел занимается тем, что изучает как теоретические основы, так и методы классификации всевозможных предметов, различных явлений, каких-либо процессов.

Если говорить о распознавании с другой стороны, то по факту – это процесс поиска, выделения, а и далее сегментирования обработанных данных с помощью определенных алгоритмов, учитывая распознанные закономерности.

Более поверхностно, распознавание объектов – типичное идентифицированные характеристик объектов и последующее их классифицирование.

Данная задача довольно востребована при автоматизации в любых сферах деятельности, поскольку сама идея внедрения распознавания объектов помогает сократить на конечных стадиях количество ошибочных детектировании каких-либо объектов или явлений, в отличие от человеческих ресурсов.

И речь идет не только про распознавание образов с фотографий или видеопотока. Само распознавание работает с практически любыми данными. Единственный критерий, который позволяет работать алгоритмам распознавания образов – это наличие характеристик, по которым можно будет классифицировать объекты и объединять их в группы.

Данный вопрос, связанный с распознаванием чего-либо сам по себе довольно трудоемкий и затратный в плане ресурсов процесс. Это связанно с огромным количеством пунктов.

Так или иначе, направление довольно перспективно и работать может в огромном количестве направлений. Поэтому и тема выпускной квалификационной работы является «Распознавание образа студента для автоматического учета посещаемости».

С одной стороны, можно немного усомниться в актуальности данной темы, беря в учет текущую эпидемиологическую обстановку. Однако, не смотря на уже реализованный алгоритм распознавания образов, он далеко не идеален. И в том случае если уже частично реализованный продукт будет усовершенствоваться, то и спустя определенное количество времени эпидемиологическая обстановка изменится.

К тому же, данную тему можно актуализировать и в малом бизнесе. Особенно это касается тех, у которого есть свой офис с относительно небольшим штатом сотрудников и нет возможности покупать уже готовые решения их по распознаванию. Так, с помощью такой технологии можно вести политику безопасности, а именно фиксировать всех, кто появляется на территории офиса или рабочей зоны за условно учётный день, будь то сотрудник или гость.

Цель данной выпускной работы – разработка системы учета посещаемости студентов на основе использования алгоритмов распознавания их образов.

Для достижения поставленной цели необходимо решить определенные задачи:

1. Провести анализ научной, а также учебно-методической литературы.
2. Провести анализ уже существующих алгоритмов по распознаванию образов.
3. Выбрать метод по распознаванию образов студентов и реализовать распознавание образов студентов.
4. Провести тестирование и определить эффективность реализованного алгоритма.
5. Опираясь на полученные данные разработать систему учета посещаемости студентов и сделать об эффективности применения алгоритма распознавания для этой задачи.

После выполнения всех выше поставленных задач, можно будет считать, что цель данной выпускной работы была достигнута.

1 Теоретическое обоснование задачи распознавания образа студентов

1.1 Описание исследуемой задачи по распознаванию образов

Ситуация на сегодняшний день складывается так, что потребность распознавания образа человека может и не так высока, но по-прежнему довольно стабильна.

Так же нет особой разницы, в какой из сфер деятельности будет применять распознавание образов: будь то высшее учебное заведение, которое будет вести учет посещаемости занятий студентами, или же частная компания с небольшим штатом (порядка 20-30 человек), которая сможет фиксировать сотрудник, которые будут посещать офис, а также посторонних лиц, которые так или иначе попадут в поле зрения реализованного алгоритма по распознаванию образов. По сути, конечным пользователем может быть кто угодно, ибо требования для работы данного программного обеспечения достаточно просты, но в то же время стабильно одинаковы.

Уже реализованные программные продукты широко используются не только в высших учебных заведениях и частных компаниях, но также и в военной сфере (для обеспечения безопасности и предотвращения несанкционированного проникновения посторонних лиц на охраняемую территорию), и в принципе в учебных заведениях любого типа для учета посещаемости обучающихся, и так далее. Некоторые коммерческие разработки добились таких высоких показателей, что, например, одна из больших IT-компаний «Apple» использует технологию распознавания лица «Face-ID» и на столько уверена в работоспособности своего алгоритма, что защищает им буквально все – от экрана блокировки, до проведения финансовых операций. Тем не менее, даже несмотря на то, что, в общем и целом, все алгоритмы так или иначе похожи друг на друга, существует

огромная разница в их реализации и применении, поскольку у каждой из них свои требования и специальные возможности.

Поэтому, при подходе к решению данного вопроса необходимо изучить основные методы распознавания образов и лиц и реализовать программу, которая позволит распознавать образы студентов и вести их учет.

1.2 Обзор реализованных алгоритмов распознавания образов и лиц

Во время исследования обозначенной задачи по распознаванию образов студентов, были найдены уже реализованные алгоритмы распознавания. Рассмотрим поближе алгоритмы, которые на сегодняшний день так или иначе способны распознавать образы и/или лица людей.

«Sighthound Cloud API» [5] — это облачное API для детектирования образов, лиц, цвета кожи, эмоции и многого другого. Данное API предоставляется компанией «Sighthound», а все запросы, которые будут поступать из API будут попадать на сервер и уже там обрабатываться. Результат работы данного продукта представлены на рисунке 1.

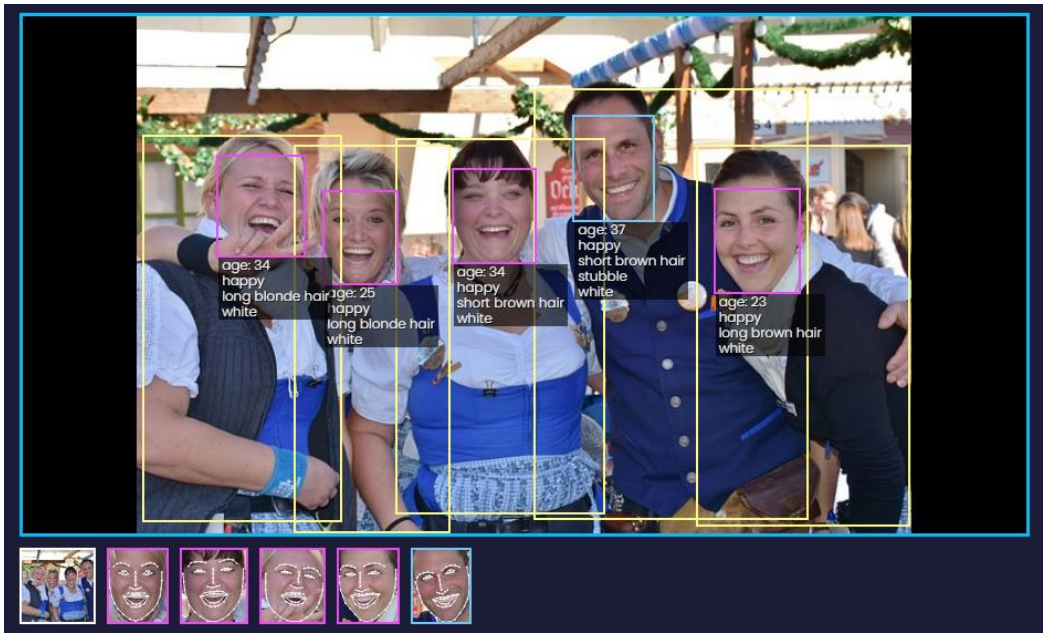


Рисунок 1 – Результат работы «Sighthound Cloud API»

«Betaface» — это коммерческий продукт с профессиональной направленностью. Используется для поиска лиц на фото и видео, но, к сожалению, не доступен для частного использования или в малом бизнесе [1].

Проблема заключается в том, что разработчики данного продукта работают только с огромными компаниями и мировыми брендами. Тем не менее, есть демо стенд, на котором обычные пользователи смогут воспользоваться данным продуктом в ознакомительных целях на базе известных людей.

Однако это лишь малый функционал, ибо все возможности доступны только после приобретения компанией по индивидуально озвученной цене. Предполагается, что это сделано в целях безопасности, чтобы риск кражи интеллектуального имущества был сведен к минимуму. Пример результатов работы «BetaFace» можно увидеть на рисунке 2.

Рисунок 2 – Результат работы «BetaFace»

Если рассматривать отечественных производителей, то Российская компания «Яндекс» внедрила в свой продукт под названием «Алиса» такую возможность, как поиск людей на фото [12].

Отличительная особенность данного решения является тот факт, что это уже готовое кроссплатформенное решение по детектированию образов людей.

Данный продукт работает практически на всех популярных платформах и под разными операционными системами.

Одним из недостатков является полная замкнутость такой системы. Проблема в том, что данный алгоритм по поиску лица на изображении с последующей обработкой его работает исключительно в экосистеме Яндекса и не имеет возможностей для интеграции в сторонние продукты, в отличие от предоставленных ранее алгоритмов.

Результат работы распознавания можно наблюдать на рисунке 3.

Рисунок 3 – Результат работы «Алисы» от «Яндекс»

Выше представленные решения по распознаванию образов и лиц являются не единственными в своем направлении. Однако каждый из методов распознавания имеет не только плюсы, но и минусы. Таблица 1 содержит сравнительную сводку по каждому из вышеперечисленному способу детектирования.

Таблица 1 – Сравнительная таблица качеств представленных методов

Название	Интеграция	Достоинства	Недостатки	Стоимость
Sighthound Cloud.	полная интеграция в код	Отсутствие вычислительной нагрузки	Постоянное наличие Интернет-соединения	тарификация по количеству обращений
BetaFace	полная интеграция в код	Высокое качество обработки; работа с материалами низкого качества	Сотрудничество только с крупными компаниями	тарификация по количеству обращений / разовая оплата
Алиса	отсутствует	Неограниченное использование	Порог ложных срабатываний выше платных конкурентов	бесплатное использование

Исходя из данных, предоставленных выше, можно сделать вывод, что оптимальное решение под поставленную задачу отсутствует. Это связно с тем, что каждый из представленных методов имеет свои определенные недостатки, которые будут критичны в решении поставленной задачи. Одни методы слишком дорогостоящи и нет представления об окупаемости программного продукта. Другие методы требуют постоянного доступа в сеть, что в свою очередь ставит зависимость перед стабильным интернет-соединением, и какие последствия могут понести простые перебои с электропитанием или временным обрывом сети. Третьи же довольно удобны и бесплатны, но, к сожалению, не имеют возможности какой-либо интеграции, что, по сути, приравнивает их к абсолютно бесполезным.

1.3 Постановка задачи на реализацию программы для распознавания образов студентов

Для выполнения поставленной задачи требуется обучить каскадный классификатор для распознавания образа лица, после нужно будет обучить каскад определенного лица, а после реализовать программу на языке Python, основной задачей которого будет с помощью вышеупомянутых обученного каскада и каскадного классификатора распознавать образы студентов, которые уже были внесены в каскад.

В результате, реализованная программа должна иметь ряд функциональных возможностей, а именно:

- реализованная программа должна распознавать образ определенного студента с кадров любого видеопотока;
- реализованная программа должна иметь возможность интеграции обученного каскада в любой программный продукт;
- реализованная программа в случае успешного определения образа студента должна выделять его на видеопотоке;
- реализованная программа должна иметь возможность распознавать более одного студента в кадре;
- реализованная программа должна иметь функциональную возможность экстренного завершения распознавания по нажатию «q».
- реализованная программа должна иметь возможность вести учет о распознанных студентах посессионно.

Исходя из вышеперечисленных требований к программе, будет необходимо учесть их во время разработки программного продукта.

В данном разделе была поставлена задача на реализацию программы по распознаванию образа студента для автоматического учёта посещаемости. В последующих разделах будут описаны этапы реализации программы, тестирование и анализ, а также результаты ее работы.

2 Математическая формулировка модели по распознаванию образов студентов

2.1 Анализ и выбор вычислительного метода для реализации программы по распознаванию образов студентов

В предыдущем разделе были рассмотрены уже реализованные способы распознавания лиц и образов, которые, к сожалению, так или иначе не удовлетворяют заданным требованиям.

Так же, в предыдущем разделе были обозначены основные требования к функциональным возможностям для реализуемой программы по распознаванию.

Для того, чтобы понять основной принцип работы распознавания чего-либо в принципе, выбор пал на библиотеку компьютерного зрения со свободным распространением именуемой «OpenCV». Данная библиотека представляет из себя набор специальных алгоритмов для реализации компьютерного зрения, а также обработки изображений и видеопотока и численных алгоритмов для общего назначения с открытым кодом [3].

Реализация данной библиотеки довольно обширна. Есть возможность использовать «OpenCV» на основных языках программирования, а именно: «Java», «C/C++», «MatLab», «Ruby», «Lua*» и многих других.

Библиотека «OpenCV» носит лицензию BSD – Berkeley Software Distribution. Данный тип лицензии означает, что это программное обеспечение (в данном случае набор библиотек) распространяется в виде исходного кода и был создан в целях обмена опытом в учебных заведениях и за их пределами, а также в коммерческих целях.

Изначальный язык программирования, под который и разрабатывалась данная библиотека, это «C++». Однако, спустя время, в виду своей высокой функциональности и производительности, она была портирована и на другие языки, указанные выше. Одним из этих языков как раз и был язык

программирования Python, на котором и будет реализован алгоритм по распознаванию образов студентов [4].

Данный язык программирования был выбран не случайно. В отличие от других языков программирования, библиотека «OpenCV» на языке «Python» имеет одну из простейших отладок и требует гораздо меньше кодового окружения, в сравнении с другими языками программирования. Также, язык «Python» был выбран именно потому, что для реализации поставленной задачи не требуется особых дополнительных манипуляций с полученным результатом на данном этапе.

В целях реализации распознавания, будет рассмотрен метод каскадного обучения, который базируется на каскадах Хаара. Этот метод распознавания объектов на изображении, в основе которого было заложено машинное обучение, будет использоваться далее, поскольку принципы и основные методы которого уже были заложены в статье Майкла Джонса (Michael Jones) и Пола Виолы (Paul Viola) [2],[15].

Стоит заметить, что особенностью каскадов Хаара является тот факт, что сам процесс обучения каскада крайне медленный, однако при правильном подходе данный метод может показывать высокие результаты в распознавании объектов.

Обученный каскад Хаара на входе получает изображение, в процессе обработки применят свои алгоритмы и предоставляет результат. Под результатом понимается информация о том, есть ли на исходном изображении искомый объект. Другими словами, можно сказать, что, по сути, каскад производит классификацию – делит исходную информацию на два типа: а именно сообщает о наличии или отсутствии искомого объекта на исходном изображении.

Каскад Хаара, который изначально был обучен верно, в последствии может крайне эффективно распознавать искомый объект. Так же такой каскад обладает довольно неплохой устойчивостью к небольшим искажениям.

Данный каскад подходит лучше всего для работы с лицами, поскольку даже на углах наклона примерно в 30 градусов детектирование происходит крайне успешно [10], [14].

Однако, при увеличении угла выше 30 градусов, эффективность резко падает. Однако, несмотря на то что данный алгоритм крайне хорошо работает с лицами, его так же можно обучить и на работу с другими объектами. В пример можно привести уже реализованные программы, которые с успехом распознают человеческую походку, предметы интерьера и экстерьера, дорожные знаки, любого вида транспорт и многое другое.

Если затрагивать вопрос о том, как работают уже обученные каскады, то нельзя не упомянуть принцип работы окна сканирования. Задача по детектированию искомых объектов на предоставленном изображении будет выглядеть следующим образом. В наличие есть изображение. Также на этом изображению присутствуют объекты. Один из этих объектов, находящихся на изображении, нам необходимо детектировать.

Исходное изображение можно представить в виде двухмерной матрицы определённого размера (ширина × высота), состоящей из пикселей. Каждый пиксель, отдельно взятый из такой матрицы, будет обладать определенным значением в диапазоне от 0 до 255. В результате выполнения алгоритма, программа сможет определить черты искомого объекта на исходном изображении. А при необходимости выделить искомый объект необходимым образом.

Работа признаков Хаара основывается на перепадах цвета на изображение в следствие теней или иных источников, создающих контраст. Именно эти перепады позволяют определить искомый объект на изображении.

Так, с помощью признаков Хаара, а в частности перепадов яркости, можно определить переносицу между глаз на обычной фотографии лица человека. Пример работы такого признака можно увидеть на рисунке 4.

Рисунок 5 представляет из себя пример того, как будет выглядеть признак Хаара на изображении, где два прямоугольника белого цвета – это первая группа областей, а прямоугольник черного цвета – это вторая группа.

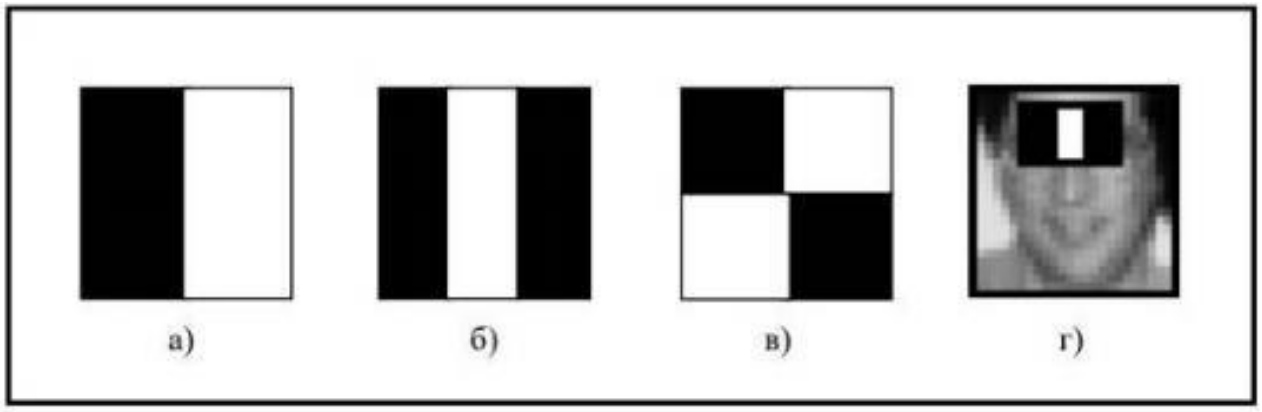


Рисунок 4 – Признаки Хаара прямоугольной формы, задействованные во обучении каскадного классификатора: а) граничный; б) линейный; в) диагональный; г) пример признака и его положения в окне сканирования

Рисунок 5 – Признак Хаара на исходном изображении

Численное значение признака Хаара – это математическая разность сумм численной яркости из обеих групп. Если записывать вычисление признака в виде математических формул, то их представление можно увидеть на формулах ниже, где a_i – это сумма яркости пикселов в i-й области первой группы (формула 1), b_i – это сумма яркости пикселов в i-й области

второй группы (формула 2) и h – это значение признака Хаара для взятого изображения.

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

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

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

где x_{ij} – яркость пикселя с координатами [i, j];

h_{ai} – высота i-й области первой группы;

w_{ai} – ширина i-й области первой группы;

h_{bi} – высота i-й области второй группы;

h_{bi} – ширина i-й области второй группы;

N_a – количество областей первой группы;

N_b – количество областей второй группы;

Нахождение разности сумм пикселей в областях изображения принадлежащим белым и чёрным прямоугольникам – это и есть вычисление значения признаков Хаара.

Что граничные, что линейный признаки – все они имеют либо вертикальную ориентацию, либо горизонтальную ориентацию (рисунок 4) [25].

В некоторых случаях, для того, чтобы можно было добиться определенного результата, библиотека «OpenCV» может задействовать и дополнительные признаки при работе определенных методов. Такие признаками можно увидеть на рисунке 6 [8].

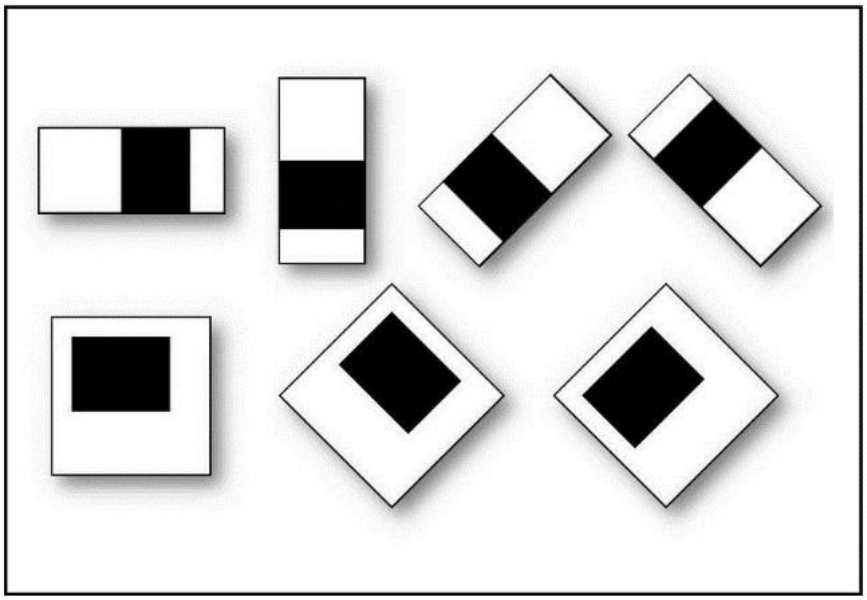


Рисунок 6 – Дополненные признаки Хаара в библиотеке «OpenCV»

Для того, чтобы эффективно вычислять признаки Хаара, алгоритм использует так называемые интегральные представления растровых изображений.

Допустим, что существует некое изображение черно белого цвета *f(y, x)* и установленным конкретно размером M × N. Тогда интегральным представлением изображения будет I *f(y, x)*. Далее принимаем растровое изображение с размерами (M + 1) × (N + 1). В таком случае численное значение пикселов вычисляются по формуле 4.

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

где I *f(y, x)* – интегральное представление изображения;

x – координата по оси X;

y – координата по оси Y.

При использовании такого рекурсивного метода, благодаря всего одному проходу можно выстроить интегральное представление растрового изображения.

Далее необходимо представить величины пикселей интегрального представления растрового изображения для первого столбца и первой строки по формуле 5 в виде нуля.

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

где $I_f(0,x)$ – величина пикселей интегрального представления;

x – координата по оси X;

y – координата по оси Y.

Остальные пиксели, значения которых $y > 0$ и $x > 0$ можно просчитать по формуле 6.

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

где $I_f(y,x)$ – величина пикселей интегрального представления;

x – координата по оси X;

y – координата по оси Y.

При работе с интегральным представлением изображения, благодаря использованию всего 4 арифметических операции можно полностью посчитать сумму всех пикселей в прямоугольной области.

Рисунок 7 демонстрирует графическое отображение области, из которой будет браться интегрально представление [7] [16].

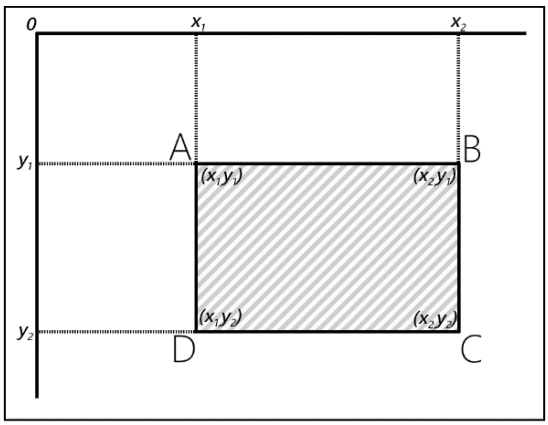


Рисунок 7 – Область изображения, представленная интегрально

Используя формулу 7, можно рассчитать сумму пикселей отдельно взятого изображения.

Для того, чтобы рассчитать сумму пикселей, находящихся внутри прямоугольной области, достаточно будет применить всего 3 арифметических операции. Так же, стоит заметить, что формула 7 не закреплена за каким-либо определенным размером изображения. В данном случае она применима к любым размерам, то есть масштабируема.

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

где $f(y,x)$ – прямоугольная область;

x_1, x_2 – начальная и конечная координата по оси X;

y_1, y_2 – начальная и конечная координата по оси Y.

Таким образом, используя интегральное представление растрового изображения, вычислить признаки Хаара используя всего несколько простых арифметических операций довольно просто.

При всем этом, как было сказано ранее, количество формул, необходимых для вычисления интегрального представления неизменно, поскольку используемые формулы масштабируемы в плане размера изображения, на котором они будут применяться.

2.2 Подготовка данных для обучения каскадных классификаторов

Для успешного обучения каскадного классификатора потребуется обширная база как с положительной, так и отрицательной выборкой. Если говорить про положительную выборку, то там относительно все просто. А именно необходимо огромное количество лиц, которые будут предоставляться каскаду как то, что необходимо искать и то, что по факту является лицом.

Однако, по мимо положительно выборки нам так же потребуется и отрицательная, которая представляет из себя набор различных изображений, содержащих в себе что угодно, но только не лица.

Если искать такие огромные выборки в сети Интернет, то в большинстве случаев будут попадаться либо небольшие объемы изображений, либо изображения будут объединены тематически, что тоже не особо хорошо, либо изображения с ватер-марками, которые в будущем так же негативно повлияют на процесс обучения каскадов.

Для того, чтобы понять, что именно нужно для обучения, необходимо выдвинуть определенные требования к изображениям, а именно:

- изображения разного размера, не превышающих одного мегабайта;
- изображения, не содержащие в себе лиц;
- изображения без ватер-марки;
- изображения абсолютно разных тематик.

Для того, чтобы получить большое количество изображений, которые удовлетворяли бы поставленным требованиям, необходим ресурс с открытыми изображениями разной тематики.

Одним из таких облачных ресурсов по хранения скриншотов со свободным доступом, является сайт LightShot.

Данный ресурс характерен тем, что все изображения хранятся в свободном доступе.

Для реализации поставленной задачи был разработан программный продукт на языке «C#». Данный программный продукт представляет из себя оконное приложения для операционной системы «Windows», работающее на платформе .Net Framework 4.7. На стадии разработки данный программный продукт получил название «PhotoFounder».

Поскольку на данном ресурсе каждая ссыкла на изображение представлена в виде шести символьного адреса, который идет после имени домена и его домена, то диапазон изображений необходимо указывать числом в тридцати семеричной системе счисления, а именно числа от 0 до 9 и 26 букв английского алфавита.

Перед запуском модуля загрузки, необходимо проверить поля с начальным и конечным адресом на заполненность. Если проверка была пройдена, то инициализируется модуль загрузки. Саму же проверку и инициализацию в случае ее успешного прохождения можно увидеть на рисунке 8.

```
private void buttonDownload_Click(object sender, EventArgs e)
{
    if ((textBoxTo.Text != "") && (textBoxFrom.Text != ""))
    {
        var fl = new FormDownload(textBoxFrom.Text, textBoxTo.Text);
        fl.Show();
    }
}
```

Рисунок 8 – Проверка полей на заполненность

Стоит отметить, что в адресации изображений используются только строчные буквы английского алфавита. Поэтому, для того чтобы программа отрабатывала правильно после указания диапазона, начальные и конечные значения сначала проверяются на корректность указания порядка (от меньшего значения, до большего).

После оба числа переводятся из тридцати семеричной системы в десятичную и запускается цикл, который будет работать от начального числа до конечного, увеличивая его на один и переводя число из десятичной в тридцати семеричную для получения ссылки на изображение. Функции переводов можно увидеть на рисунке 9 и на рисунке 10.

```
public static long From36To10(string thrsix)                                   //перевод из 36СС в 10СС
{
    long dec = 0;
    long j = 5;
    for (int i = 0; i < 6; i++)
    {
        dec += (long)Array.IndexOf(alphabetForAdess, thrsix[i].ToString()) * Pow(36L, j);

        j--;
    }
    return dec;
}
ссылка: 1
```

Рисунок 9 – Функция перевода в десятичную систему счисления

```
public static String From10To36(long dec)                                      //перевод из 10СС в 36СС
{
    String[] thrsix = new string[6];
    for (int j = 0; j < 6; j++) thrsix[j] = "0";
    long i = 5;
    while (true)
    {
        if (dec >= 36)
        {
            long ostatok = (dec - ((dec / 36) * 36));
            thrsix[i] = alphabetForAdess[ostatok];
            i--;
            dec = dec / 36;
        }
        else
        {
            thrsix[i] = alphabetForAdess[dec];
            break;
        }
    }
    return thrsix[0] + thrsix[1] + thrsix[2] + thrsix[3] + thrsix[4] + thrsix[5];
}
ссылка: 1
```

Рисунок 10 – Функция перевода из десятичной системы счисления

После того, как получается цельная ссылка путем соединения адреса ресурса с его доменом и 6 символов тридцати семеричного числа,

полученная страница начинает загружаться и запускается таймер. После загрузки производится поиск на наличие одного из шести ресурсов по хранению изображений.

Если ни одного из ресурсов не было найдено, или же таймер превысил 20 секунд (таймером отлавливаются «битые» ссылки, которые никогда не загрузятся), то считается, что загрузка изображения была неуспешной и берется следующая ссылка на следующее изображение.

Если же один из ресурсов был успешно обнаружен, то происходит загрузка данного изображения. Сохраненное изображение получает имя, которое совпадает с уникальным адресом на ресурсе и получает разрешение «.PNG». Так сложилось, что все изображения, что хранятся на этих ресурсах имеют такое расширение(.png) и поэтому нет необходимости в конвертировании их в нужный формат.

Во время работы модуля программы по загрузке фотографий из диапазона отображается информатор в виде прогресс-бара и нескольких текстовых полей. Они информируют о том, сколько было совершено попыток загрузить изображение, сколько из низ было успешно, а также какое количество изображений осталось загрузить до выполнения поставленной задачи.

Стоит обратить внимание, что все файлы сохраняются в папке, которая автоматически создается в случае своего отсутствия рядом с исполняемым файлом «PhotoFounder» ’а.

В итоге, данный программный продукт может работать стабильно в автономном режиме. Замеры нагрузки на компьютер не производились, поскольку даже на момент отладки в пиковых нагрузках, загруженность центрального процессора составляла не более десяти процентов.

2.3 Обучение каскадного классификатора в целях реализации программы по распознаванию образов студентов.

Реализация программы по распознаванию образов студентов, как и было сказано ранее, будет проходить благодаря обучению каскадов Хаара и последующему их использованию.

Также по итогу будет произведен замер производительности обученного каскада.

Сама выборка представляет из себя 2000 положительных изображений и порядка 69000 отрицательных изображений;

Процесс обучения будет сопровождаться постоянным замером нагрузки на компьютер.

Каскад будет обучаться на стационарном компьютере под OS Linux. Версия библиотеки, используемой для обучения - «OpenCV» 4.1.2.

Далее представлены характеристики компьютера, участвующего в обучении:

- CPU Intel(R) Pentium(R) CPU G3460 @ 3.50GHz;
- RAM 12 Гб DDR3 1333MHz;
- SSD накопитель KINGSTON SA400S37240G.

Для большего удобства стоит разделить реализацию алгоритма для детектирования студентов на несколько отдельных этапов:

- этап первый подготовительный: в рамках данного этапа необходимо привести все положительные изображения к необходимому виду и размеру;
- этап второй обучение: в рамках данного этапа будут обучен каскад, который будет использоваться в детектировании образов студентов;
- этап третий реализация: в рамках данного этапа будет реализован программный продукт, который сможет, опираясь на предоставленный каскад и дополнительные необходимые данные, реализовать распознавание образов студентов;
- этап четвертый сбор статистики: на данном этапе будет собрана все статистика и проанализирована эффективность обученных каскадов.

В рамках первого этапа было подготовленно две выборки. Выборка отрицательных изображений составляла порядка 69000 изображений. Положительная выборка содержала в себе 2000 изображений.

В процентном соотношении, положительная выборка составляла 1,38 процента от отрицательной. Необходимо заметить, что в отличии от отрицательной выборки, где изображения имеют случайный размер как по вертикали, так и по горизонтали, положительная выборка должна быть одного размера.

Положительная выборка представляет собой набор лиц, взятых с открытого источника. Исключительной особенностью данных лиц является тот факт, что по сути данных людей никогда не существовало.

Данные лица были сгенерированы с помощью генеративно-состязательных сетей. Генеративно-состязательная нейронная сеть – это когда одна нейронная сеть генерирует изображение, а вторая проверяет, выглядит ли это изображение реалистично. На рисунке 11 можно увидеть пример генерации лиц.

Рисунок 11 – Пример лиц, сгенерированных нейронными сетями

В качестве отрицательно выборки были взяты изображения, процесс получения которых был описан в предыдущей главе. Как было сказано ранее, данные изображения представляют собой случайные картинки различного содержания.

Данные изображения имеют абсолютно разный размер. Отрицательная выборка была отфильтрована вручную и не содержит лиц в любых проекциях. Это сделано для того, чтобы во время обучения у каскадного классификатора не возникало никаких противоречий, когда отрицательная и положительная выборка содержит относительно одинаковые изображения. Пример подготовленной выборки можно увидеть на рисунке 12.

Рисунок 12 – Пример изображений, загруженных с помощью «PhotoFounder»

После того, как изображения были готовы, они были отформатированы и разложены по отдельным папкам. Исходя из количества выборок, папок тоже всего две. В папке good хранятся изображения лиц. В папке bad хранится гораздо больше изображений, представляющих отрицательную выборку.

В целях подготовки отрицательной выборки, с помощью команды как на рисунке 13 можно создать список изображений из отрицательной выборки.

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

Рисунок 13 – Генерация списка с именами отрицательной выборки

Минусом данной команды, является тот факт, что при генерации списка, в начало списка попадает запись, которая не несет в себе имени изображения. Данную запись можно увидеть на рисунке 14. Ее необходимо убрать. В случае, если первая запись не будет удалена, то при последующих операциях, на одном из этапов будет получена ошибка, которая не будет указывать на то, что существует такая «ошибочная» запись.

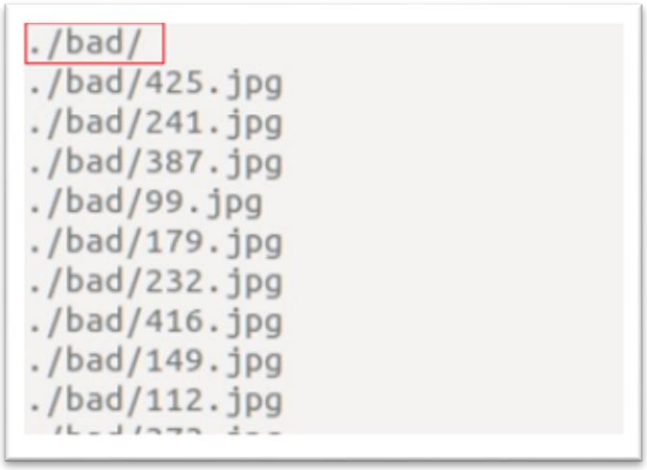


Рисунок 14 – Список отрицательной выборки с лишней записью

Аналогично команду как на рисунке 13 необходимо применить и для положительной выборки. Так же, как и у отрицательной выборки, необходимо удалить первую запись во избежание возникновения ошибок во время обучения каскада.

Еще одно отличие при записи информации об изображениях с положительной выборки является тот факт, что помимо самого названия изображения, необходимо указать, где именно на изображении находится искомый объект.

Как выглядит такая команда, можно увидеть на рисунке 15.

```
find ./good/ -name '*' -exec echo \{\} 1 0 0 600 600 \; > face.info
```

Рисунок 15 – Генерация файла списка с именами изображений положительной выборки

Здесь:

– name “*” — это название файла из положительной выборки, а цифра «1» — это количество объектов, считающихся положительными на изображении. Как было сказано ранее, количество положительных объектов на положительном изображении может быть отличным от единицы.

– «0 0 600 600» – прямоугольные координаты зоны нахождения положительного объекта.

Для того, чтобы посмотреть содержимое файла с именами положительных изображений и объектов на нем, который был только что сгенерирован, можно применить команду «cat face.info».

Рисунок 16 содержит в себе результат выполнения вышеуказанной команды.

Так же, как и в предыдущем случае, в целях избегания ошибок на стадии обучения необходимо удалить первую запись, которая не содержит в себе информации о положительном изображении.

Далее, необходимо создать векторное представление каждого из изображений. Для его создания потребуется команда, которая встроена в библиотеку «OpenCV». Команда «opencv_createsamples» запустит утилиту, которая создаст необходимые векторные представления каждого изображения. По сути, эта утилита генерирует набор положительных образов в том формате, который поддерживается не только операционной системой, но и понятном для библиотеки «OpenCV». Именно с таким типом изображений смогут работать утилиты «opencv_haartraining» и «opencv_traincascade» [17], [21].

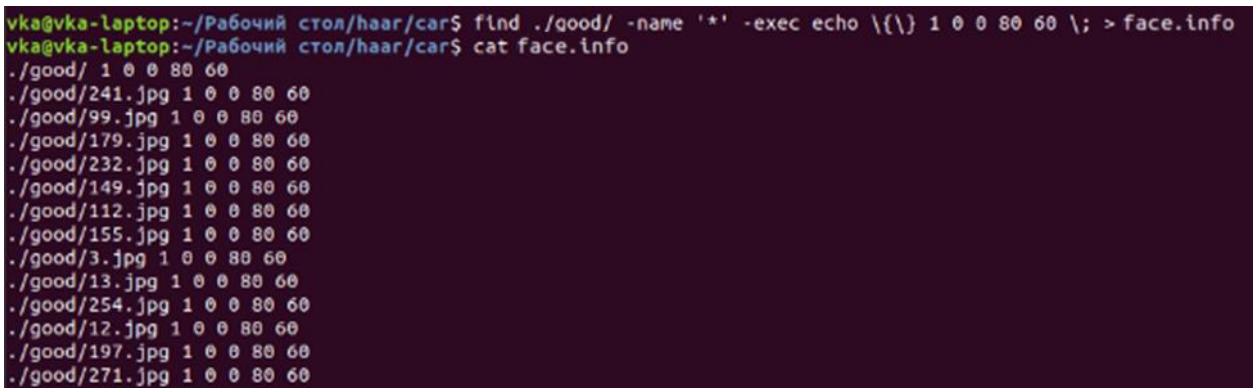


Рисунок 16 – Результат выполнения команды «cat face.info»

Создание образцов положительных изображений в формате «.vec» можно наблюдать на рисунке 17.

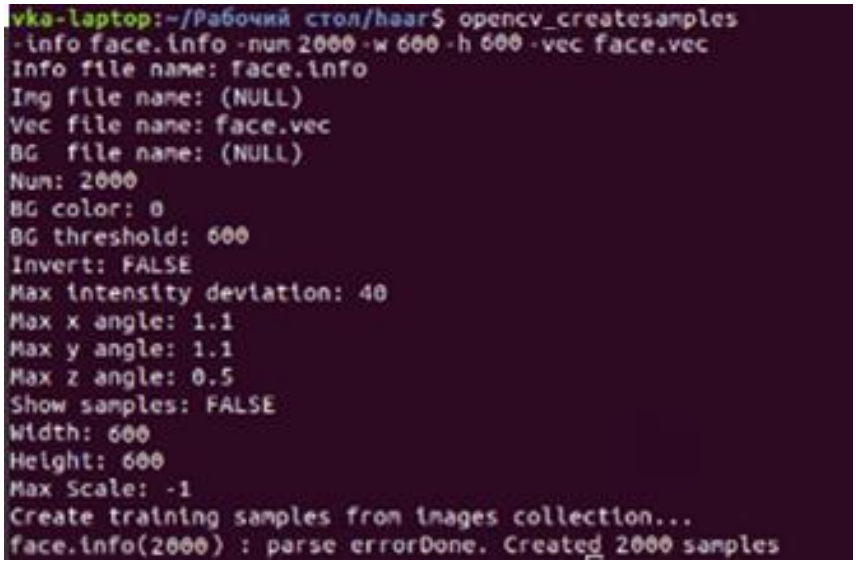


Рисунок 17 – Генерация образов положительной выборки в формате «.vec»

После генерации векторных представлений из изображений положительной выборки, с помощью команды «opencv_createsamples -vec face.vec -w 600 -h 600» можно проверить корректность их генерации.

На рисунке 18 как раз представлены примеры изображений, прошедших конвертирование в векторный формат.

После того, как все изображения были подготовлены нужным образом, можно приступить к обучению каскада. Для того, чтобы каскад обучить,

необходимо использовать встроенную в библиотеку «OpenCV» утилиту «opencv_traincascade» [18]. Данная утилита является улучшенным вариантом в сравнении с утилитой «opencv_haartraining», которая использовалась в релизных сборках API «OpenCV» до версии 2.x на языке C++ [21].

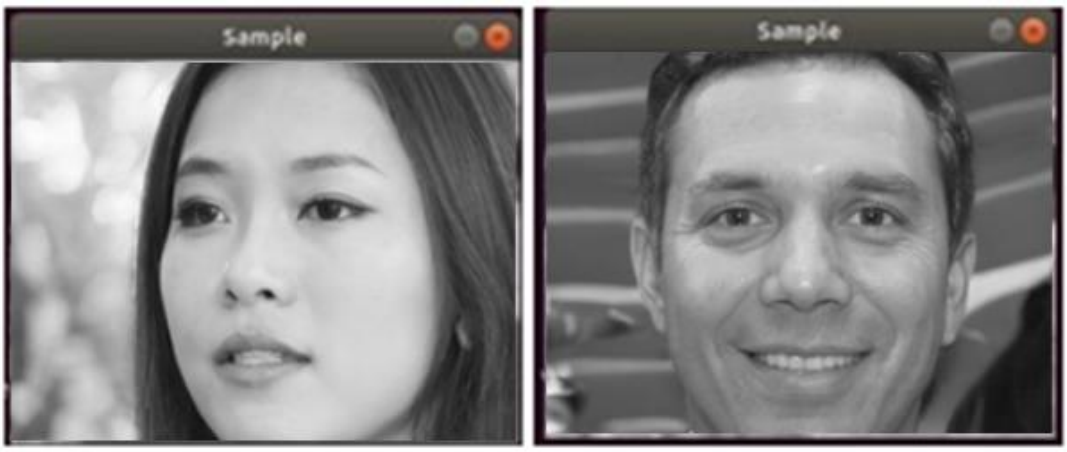


Рисунок 18 – Векторное представление изображений из положительной выборки

Более новая версия «opencv_traincascade» была использована именно потому, что она позволяет использовать функции LBP (Local Binary Patterns) метода [19],[21],[22].

Именно PLB метод позволит нам не просто детектировать лицо студента, а именно распознавать кто именно этот студент и уже на основе полученных данных производить дальнейшие манипуляции.

Далее указан перечень аргументов, которые используются в утилите «opencv_traincascade» [6],[11],[13],[21]:

– data <cascade_dir_name> - имя директории, в которую будет сохранен обученный каскад;

– vec <vec_file_name> - полный адрес файла относительно исполняемого файла формата «.vec», в котором содержится положительная выборка (полученная благодаря использованию утилиты «opencv_createsamples»);

– bg <background_file_name> - полный адрес файла относительно исполняемого файла, который содержит название файлов отрицательной выборки;

– numPos<number_of_positive_samples> - количество изображений положительной выборки;

– numNeg<number_of_negative_samples> - количество изображений отрицательной выборки;

– numStages <number_of_stages> - количество этапов обучения каскада;

– w <sampleWidth>, h <sampleHeight> - параметры ширины и высоты изображений из положительной выборки для обучения;

– featureType <{HAAR (default), LBP}> - тип обучаемого каскада (HAAR, LBP);

– minhitrate – значение коэффициента, определяющего качество обучения. А именно процент верных обнаружений. (По умолчанию значение равно 0.999. В случае, если «minhitrate» задан 0.999, то в исходной выборке пропуск целей будет не более, чем 1 из 999 что равно 0.01%. Чем выше коэффициент «minhitrate», тем чаще будут срабатывать «ложные тревоги»);

– maxFalseAlarmRate - значение коэффициента тревоги. (Значение, заданное по умолчанию - 0.5). В том случае, если выборки подобраны верно (оптимальное соотношение объема выборки и количества стадий обучения), то уровень допустимой тревоги будет быстрее получен, а сам процесс обучения будет завершен;

– mode ALL – необходимость использования полного перечня признаков Хаара из библиотеки «OpenCV». Данный параметр прямо влияет на то, как долго будет проходить обучение, а также какая точность в результате данного обучения будет достигнута. Если заранее известно, что искомый объект не будет менять ориентацию, то необходимость в данном параметре отсутствует, поскольку применение полного комплекта признаков будет необоснованным [20],[23];

– precalcValBufSize <Значение в Mb>, precalcIdxBufSize < Значение в Mb> - память, которая будет выделена на обучение.

Команда на обучение каскада будет выглядеть следующим образом: «opencv_traincascade -data haar_face -vec face.vec -bg bad.txt -numPos 2000 -numNeg 69000 -numStages 20 -w 200 -h 200 –minhitrate 0.999 –maxFalseAlarmRate 0.5 -featureType HAAR -precalcValBufSize 2048 -precalcIdxBufSize 2048 -mode ALL».

На рисунке 19 демонстрируется начало обучения, запущенное командой выше.

После того, как команда была передана в терминал, началось обучение каскада. На каждом этапе утилита «opencv_traincascade» выводит в консоль информацию о текущем состоянии обучения.

Само обучение начинает отсчет с нулевой стадии. Каждый блок, выводимый в консоль во время обучения, представляет собой информацию об объекте, а также определенные выходные данные о том, какое соотношение поставленной цели к уровню ложных срабатываний было достигнуто.

По ходу обучения шла фиксация времени уже пройденных стадий:

– нулевая стадия обучения заняла 16 часов 27 минут;
– первая стадия обучения заняла 17 часов 58 минут;
– вторая стадия обучения заняла 18 часов 13 минут;
– третья стадия обучения заняла 19 часов 27 минут;
– четвертая стадия обучения заняла 19 часов 30 минут;
– пятая стадия обучения заняла 19 часов 39 минут;
– шестая стадия обучения заняла 19 часов 42 минут.

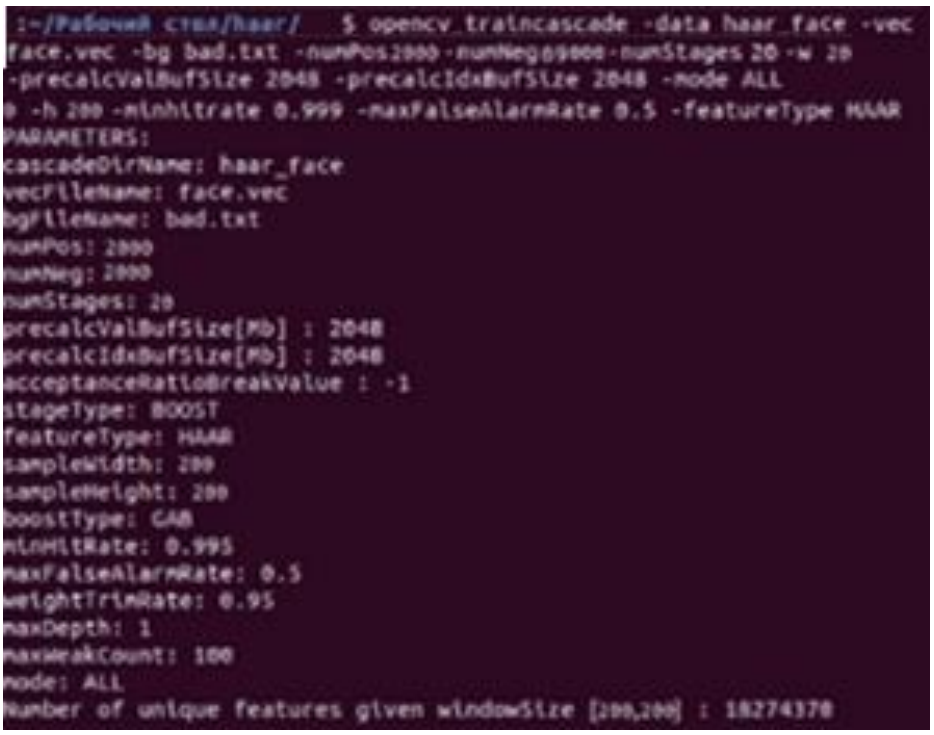


Рисунок 19 – Начало процесса обучения каскада

В начале восьмой стадии, утилита прекратила свое выполнение с комментарием, что указанный порог уровня ложных срабатывания в процессе обучения был достигнут и дальнейшее обучение не имеет смысла.

Суммарное время обучения составило 129 часов 56 минут. На рисунке 20 представлена последняя стадия обучения.

Также рисунок 20 содержит информацию о том, что в процессе обучения был достигнут и дальнейшее обучение не имеет смысла. Результатом работы команды является файл каскада в формате XML [4],[9],[24].

Так же, стоит отметить, что данный результат не является идеальным, поскольку не были пройдены все стадии обучения (произошло это исходя из того фактора, что не удалось изначально представить к обучению идеально подобранное соотношение положительной и отрицательной выборок).

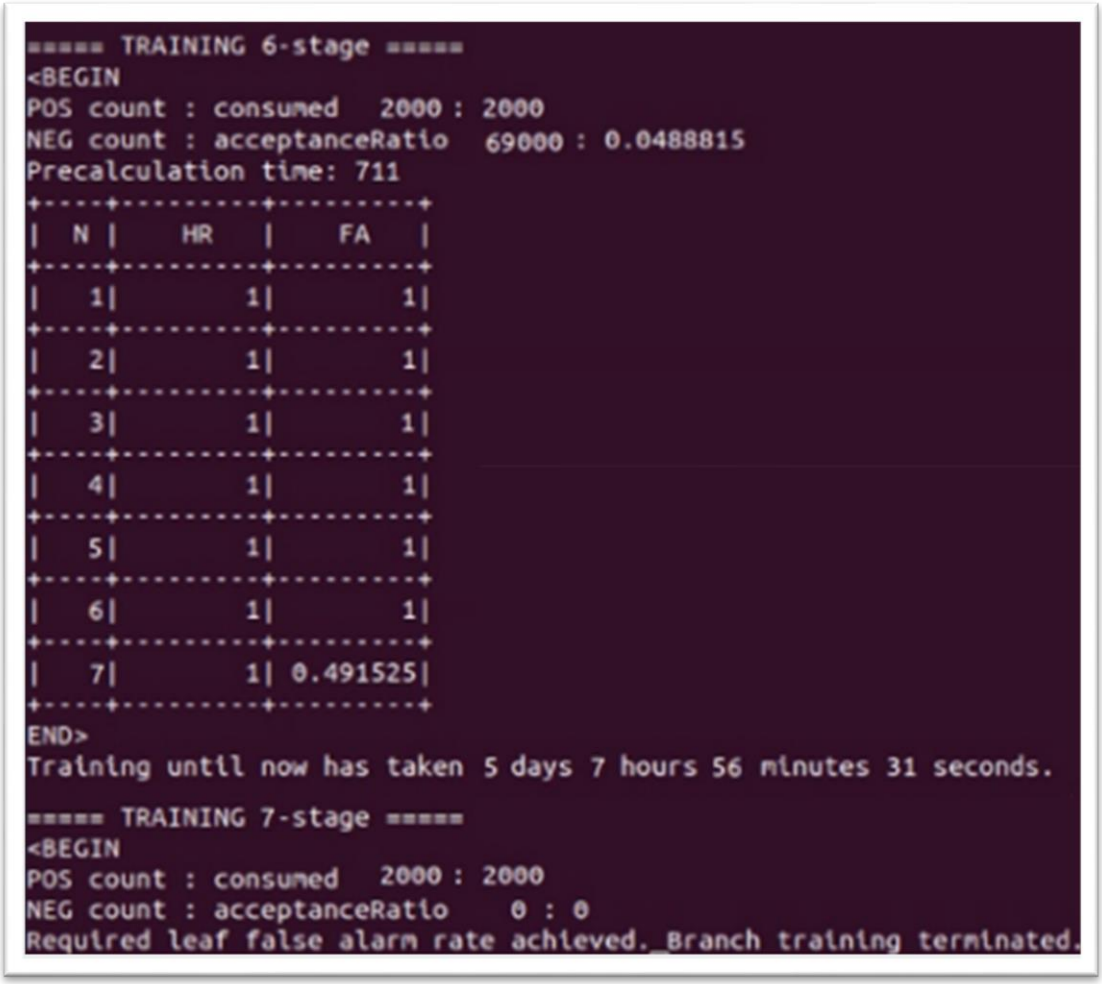


Рисунок 20 – Последняя и завершение обучения

Тем не менее, в целях проведения сравнительного анализа был обучен еще один каскад. Для его обучения использовалась та же выборка как отрицательных, так и положительных изображений.

Единственным отличием был только то, что для обучения второго каскада выборки были «урезаны» в четыре раза, а именно 500 положительных изображений и 17250 отрицательных изображений.

Основной целью данного каскада было получение информации о том, что есть ли действительно такая необходимость использовать столь большие выборки во время обучения каскада и получается ли от этого бóльшая эффективность. Время, затраченное на обучение второго каскада, составило

примерно треть от того времени, за которое обучился до необходимого уровня срабатывания ложных тревог первый каскад.

Исходя из работы, проделанной в данном разделе, было обучено два каскада. Первый обучался на всей выборке, что была подготовлена, а суммарное время обучения заняло порядка 130 часов.

Так же был обучен еще один каскад, который задействовал меньшую часть выборки, что была подготовлена. Данный каскад обучался порядка 45 часов. Второй каскад был обучен в целях проведения сравнительного анализа и получения сводной информации о том, что присутствует ли действительно такая необходимость, как использование большой выборки во время обучения каскадного классификатора для получения более высоких показателей эффективности.

В следующем разделе будет проведено тестирование обученных каскадов. Так же будет проведен анализ эффективности обоих каскадов в целях получения информации о том, целесообразно ли использовать большие выборки во время обучения.

3 Тестирование и анализ эффективности реализованной программы по распознаванию образов студентов

3.1 Описание реализованной программы по распознаванию образов студентов

Результатом выполнения данной выпускной квалификационной работы были изучены основные методы каскадного обучения.

Также была реализована программа для распознавания образов студентов и ведения их учета.

Основной функционал реализованной программы предоставляется следующие возможности:

- работа с любым подключаемым видеопотоком;
- детектирование лиц;
- распознавание образа определенного студента;
- возможность детектирования сразу нескольких лиц и/или образов студентов.

Для запуска программы необходимо запустить командой в терминале файл, содержащий код на языке Python.

Данный файл запускается без передачи каких-либо ключей. Команду запуска можно увидеть на рисунке 21.

Так же на рисунке 21 можно увидеть информацию в текстовом представлении о том, что образы каких студентов были распознаны во время работы программы.

Что касается последнего пункта из списка функциональных возможностей, а именно то, что реализованная программа должна иметь возможность вести учет о распознанных студентах посессионно, а значит это тоже необходимо реализовать.

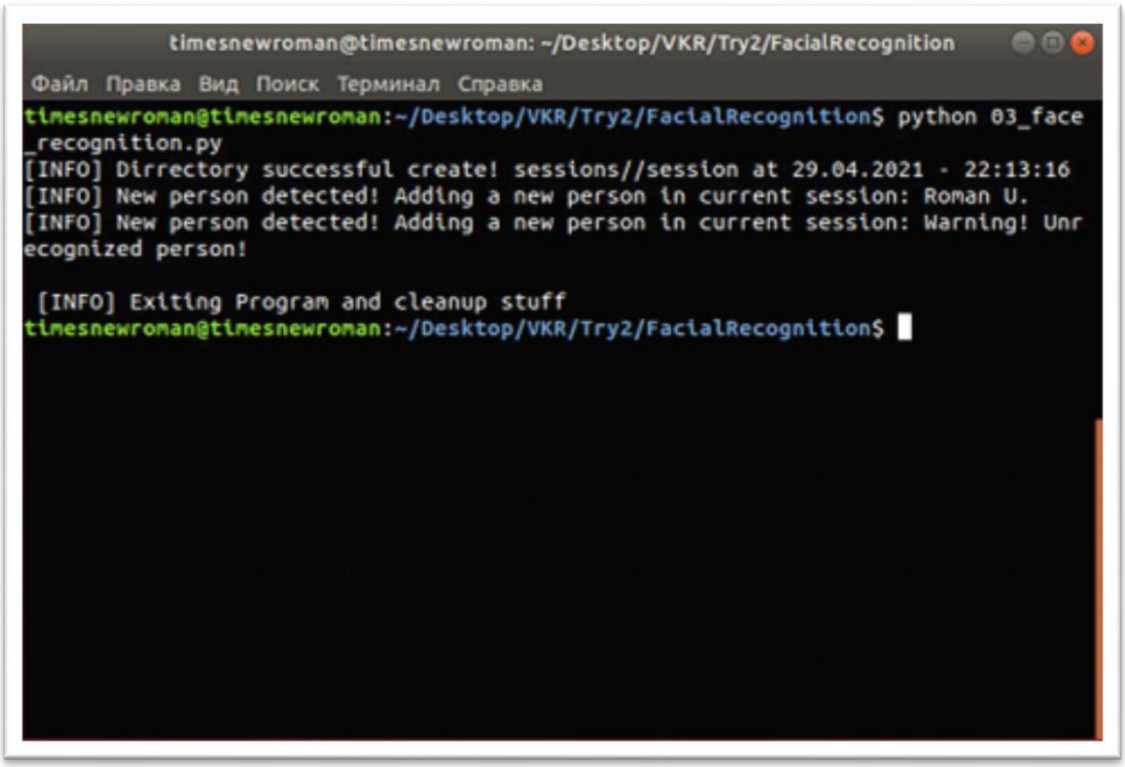


Рисунок 21 – Запуск программы, ее отработка и завершение

Сама задача по ведению учета может быть реализована крайне разнообразно. Все зависит от того, что требуется в конечном итоге. Сессионные данные можно выгружать куда-либо на облако или их необходимо куда-либо отправлять. Для реализации данной функциональной возможности был выбран менее сложный путь, а именно создание локальной директории, в которую будет вестись учет.

Каждая сессия представляет из себя разово запущенную программу. Имя директории генерируется из текущей даты и времени, это можно увидеть на рисунке 22.

После того, как очередная сессия была запущена, программа сообщает о том, что директория была или не была успешно создана.

Комбинация дата-время для имени директории хороший вариант нэйминга, поскольку данное имя не будет повторяться и как следствие – не появится проблем с сессиями с повторяющимися названиями.

```
path = 'sessions//session at ' + time.strftime("%d.%m.%Y - %H:%M:%S")
try:
    os.makedirs(path)
except OSError:
    print ('[INFO] Dirrectory does not create! %s' % path)
else:
    print ('[INFO] Dirrectory successful create! %s' % path)
```

Рисунок 22 – Генерация имени директории для создаваемой сессии

Содержимое папки после отработки программы можно увидеть на рисунке 23.

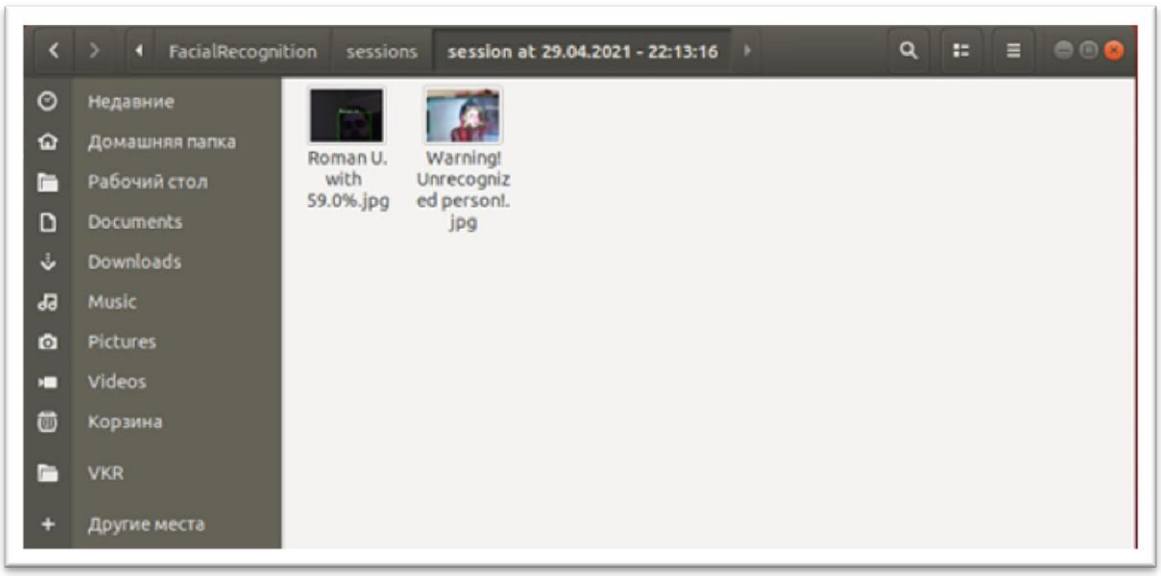


Рисунок 23 – Созданная сессионная директория после отработки программы

Каждый раз, когда программа будет детектировать образ определенного студента, для фиксации его посещения будет делаться снимок с подписанной на ней информацией – имя студента и вероятность того, что это именно он.

Каждое изображение будет подписано соответствующим ему именем или будет указано, что личность не распознана. Сами же образы будут выделены на изображении зеленым цветом и соответственно подписаны. Пример можно увидеть на рисунке 24.

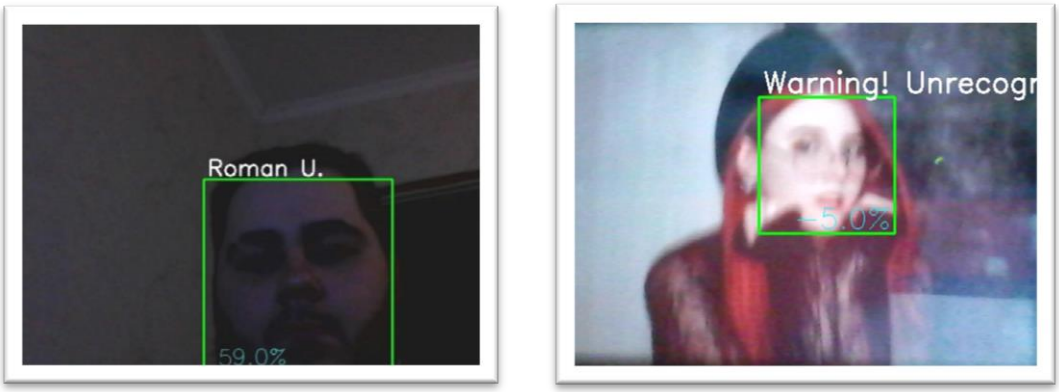


Рисунок 24 – Содержимое папки сессии

Поскольку программа на данной стадии не будет содержать графического интерфейса для взаимодействия с программой, для вызова функции завершения используется клавиша «ESC». Она прерывает детектирование образов, а также вызывает функции, которые будут очищать память, задействованною программой. Так же, исходя из логики работы с программой, теоретически, в дальнейшем она работать будет в 2 этапа.

Первый этап заключается в отработке обученного каскада с помощью программы на языке программирования Python. Данные с видеопотоков будут собираться в одном месте дальнейшей работы над ними.

Второй этап заключается в работе с этими данными, полученными после того, как каскадный классификатор и LBP модели были обучены и были детектированы образы студентов с видеопотока (будь то камеры, расположенные над входами в аудитории или видеопоток с виртуальных комнат). Поскольку в университетах подавляющее большинство персональных компьютеров и ноутбуков работают на платформе «Windows»,

было принято решение реализовать программу для удобной работы с данными.

Для того, чтобы можно было работать с полученными данными с первого этапа, необходимо иметь программу, которая будет отвечать следующим функциональным возможностям, а именно:

- реализованная программа должна работать на ОС «Windows» версиях 7 и выше;
- реализованная программа должна иметь графический и интуитивно понятный интерфейс;
- реализованная программа должна иметь возможность использовать данные как локальные, так и подгружать данные с облачных хранилищ;
- реализованная программа должна представлять данные как по одному студенту по диапазону дат, так и по всей группе;
- реализованная программа должна затрачивать на работу малое количество ресурсов, во избежание проблем с выполнением на слабых устройствах.

Для реализации всех требуемых функциональных возможностей, которые были заявлены выше, было решено реализовать оконное приложение на языке «C#». Блок схема, содержащая алгоритм работы оконного приложения, представлена на рисунке 25.

Данное приложение позволяет из выгружать из XML группы, даты, а также имена студентов, над которыми велось «наблюдение». Далее программа позволяет выбрать день и сразу же в этот момент начинает проверять, был ли в этот день данный студент там, где его должна была детектировать камера. Так как сессии с программы по детектированию образов студентов записаны в формате «session at dd.mm.yyyy hh_mm_ss», то для более удобного отображения оконная программа будет делить время на первую-девятую пару (временные интервалы задаются в config файле), а так же прямым временем в том случае, если студент был детектирован каким либо образом вне учебное время.

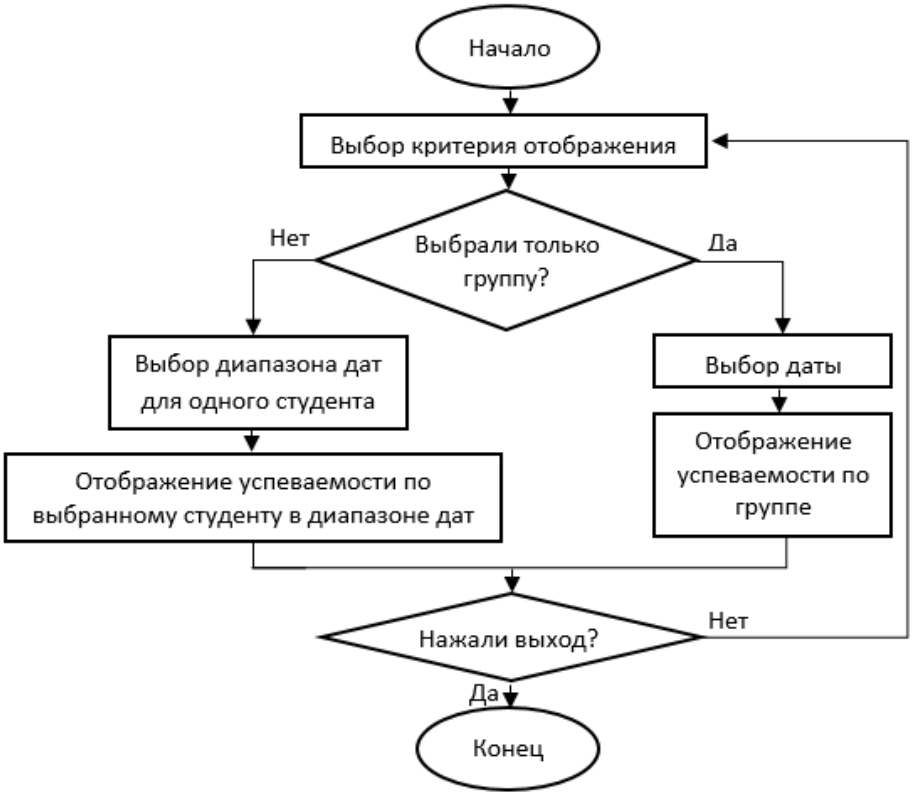
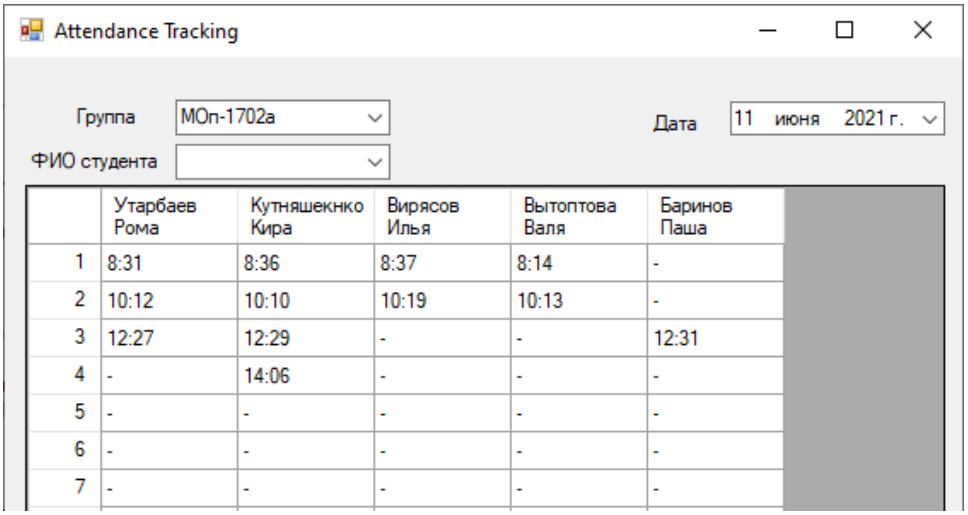


Рисунок 25 – Блок схема оконного приложения

Результат работы программы при просмотре по группам можно увидеть на рисунке 26.



	Утарбаев Рома	Кутняшекнко Кира	Вирясов Илья	Вытоптова Валя	Баринов Паша
1	8:31	8:36	8:37	8:14	-
2	10:12	10:10	10:19	10:13	-
3	12:27	12:29	-	-	12:31
4	-	14:06	-	-	-
5	-	-	-	-	-
6	-	-	-	-	-
7	-	-	-	-	-

Рисунок 26 – Результат работы программы по просмотру посещаемости группы в определенный день

Так же для более детального просмотра информации если выбрать ФИО студента, то откроется второй дополнительный «Date Picker», в котором можно будет выбрать вторую дату для создания интервала, на котором будет выведена детально информация по студенту по дням, какие пары он посещал. Результат работы модно увидеть на рисунке 27.

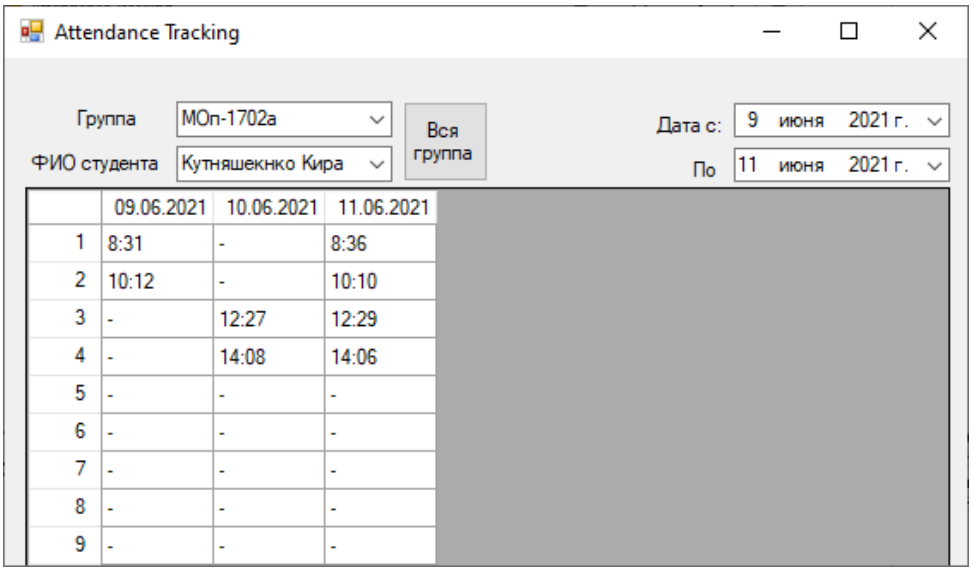


Рисунок 27 – Результат работы программы по просмотру посещаемости студента в диапазоне дат

Данные обновляются при изменении любых параметров. Так же, по нажатию кнопки «Вся группа», программа переходит в режим просмотра посещаемости всей группы. Также, после перехода в режим просмотра посещаемости всей группы, кнопка «Вся группа» в логических соображениях скрывается, как и второй селектор даты.

Текущие возможности программы покрывают функциональные требования, однако, в перспективе на будущее можно так же будет реализовать выбор отображения информации о посещаемости по предмету, по преподавателю, а также строить тепловую карту посещаемости студентов.

Так же по двойному нажатию на время, когда студент присутствовал на занятиях, откроется изображение, на котором он был детектирован. Данный

функционал будет крайне полезен в том случае, если будут какие-либо спорные моменты и необходимо будет просмотреть определенные данные за определенный день и время.

Если смотреть по нагрузке, то в среднем это 20Mb оперативной памяти и нагрузка CPU порядка 10% в моменты нагрузки. Эти данные можно наблюдать на рисунке 28.

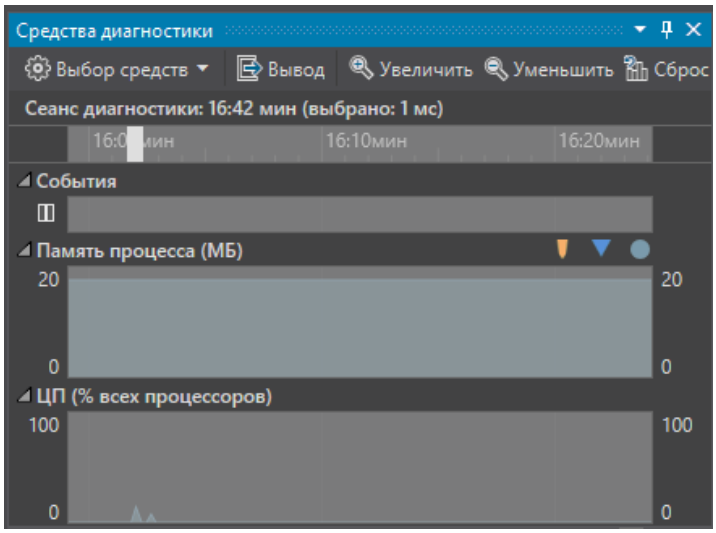


Рисунок 28 – Нагрузка на систему программой во время работы

Это означает, что реализованная программа соответствует требованиям последнего пункта ранее заявленных функциональных требований.

3.2 Тестирование реализованной программы по распознаванию образов студентов

В целях получения объективной оценки качества распознавания образов студентов обученными каскадами, необходимо учитывать сразу несколько параметров. Для начала рассмотрим стадию, на которой обучались сравниваемые каскады.

Во время того, как обучались каскады, система записывала определённые показатели:

- температура процессора;
- температура отдельно взятых ядер;
- нагрузка на оперативную память (RAM);
- полное время обучения.

Во время обучения первого каскада на полных выборках, что были подготовлены, были зафиксированы следующие показатели:

- полное время обучения 5 дней, 7 часов и 56 минут;
- максимальная температура ядер составила 69 градусов;
- максимальная температура процессора составила 70 градусов;
- средняя температура процессора составила 64 градуса;
- максимальная загрузка оперативной памяти (RAM) составила 11.4/11.6Gb (98.28%).

Во время обучения второго каскада на четвертой части подготовленной выборки, статистические данные немного отличаются:

- полное время обучения 2 дня, 1 час и 13 минут;
- максимальная температура ядер составила 66 градусов;
- максимальная температура процессора составила 67 градусов;
- средняя температура процессора составила 59 градусов;
- максимальная загрузка оперативной памяти (RAM) составила 8.3/11.6Gb (71.55%).

На рисунке 29 приведено графическое представление сравнения показателей, представленных выше.

Исходя из содержимого графика на рисунке 28 можно сделать вывод только о том, что нагрузка на систему во время обучения рознится не сильно. Несмотря на то, что разница во времени обучения существенна, сделать вывод о том, на сколько каскады работают эффективно нельзя. Именно для этого нужно провести еще ряд тестирований. А именно на одинаковых фрагментах видео пытаться успешно детектировать образы.

Для проведения данного тестирования был немного изменен исходный код программы на языке Python, после чего программа считывала данные не

с видеопотока веб-камеры, а непосредственно с видео, которые были записаны заранее.

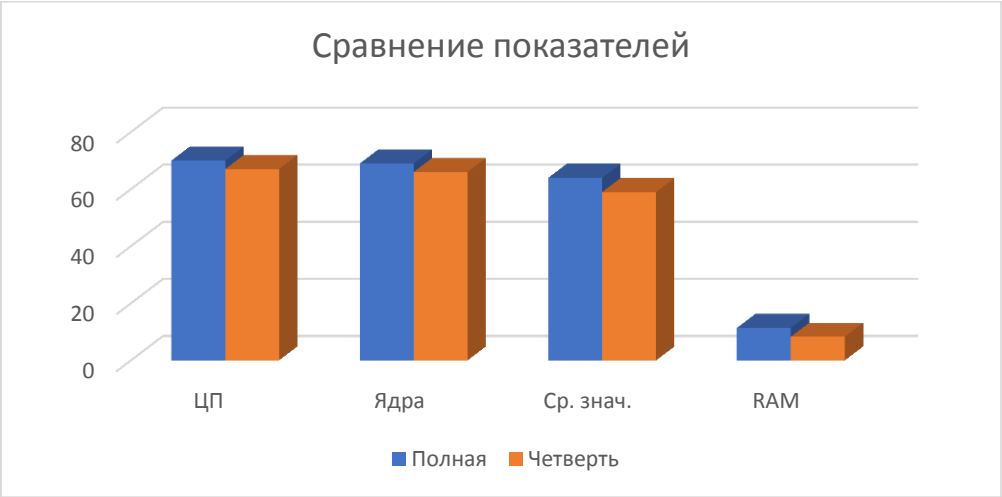


Рисунок 29 – Сравнение показателей во время обучения

Заранее записанные видео обеспечит идентичные исходные данные при работе с разными каскадами.

Сами видео были записаны в трех экземплярах на разной дальности. После каждая запись прошла обработку эффектом «рыбий глаз» с малым коэффициентом искажения.

Данное искажение было сделано по причине того, что не все камеры, которые используются имеют «идеальную» линзу, в связи с чем одни и те же студенты на 2 разных камерах в один и тот же момент могут быть распознаны неверно.

После того, когда было подготовленно девять записей (три записи с тремя разными уровнями искажения) и оба каскада поочередно были задействованы в детектировании образов студентов, была составлена сводная таблица из полученных данных.

Исходя из того, что записей было несколько, было учтено среднее значение по трем записям с разным искажением. Содержимое можно увидеть в таблице 2.

Таблица 2 – Сравнение каскадов по количеству верных и ложных срабатываний

Каскад	Кол-во студентов, участвующих в детектировании	Кол-во раз появлений студентов в кадре в ходе видео	Кол-во успешных срабатываний	Кол-во образов, которые были детектированы неверно	Кол-во образов, распознанных как «неопознанный»
Полный	12	24	19	2	3
Четверть	12	24	12	7	5

Для большей наглядности можно посмотреть на рисунок 30. На нем приведена сравнительная статистика результатов работы двух каскадов.

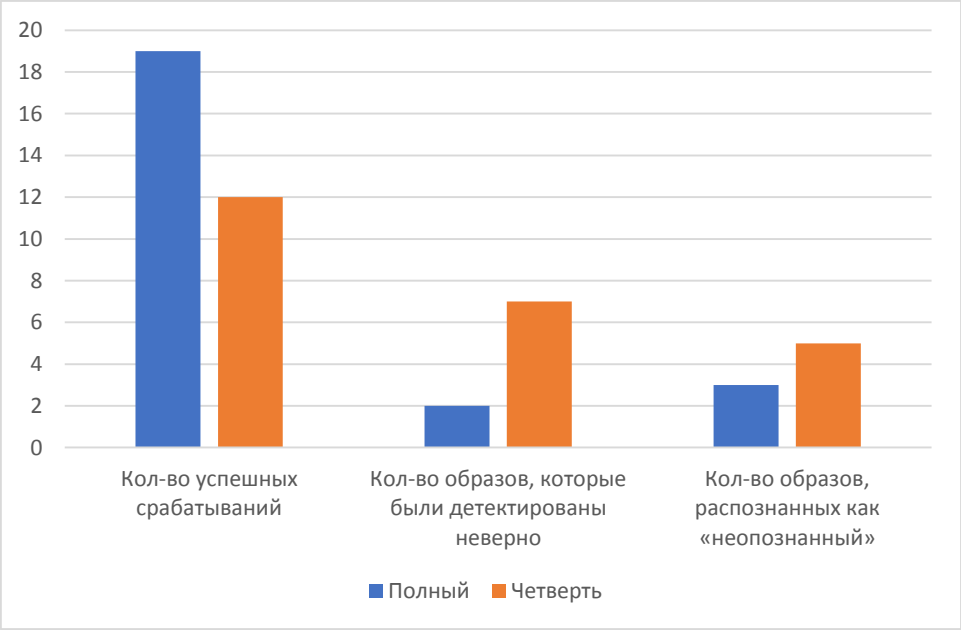


Рисунок 30 – Сравнительная статистика двух каскадов

На рисунке 29 представлены три пары столбцов. Первая пара столбцов показывает, на сколько хорошо справились с распознаванием образов студентов каскады.

Остальные пары столбцов наоборот показывают, на сколько плохо они отработали и где распознали студента неверно или вообще не увидели в найденных студентах тех студентов, которых необходимо было искать.

Так же во время работы второго каскада, который был обучен на меньшей выборке, в отличие от первого каскада были замечены абсолютно ложные срабатывания, когда там, где нет лиц в принципе, программа детектировала при помощи каскада и LBP модели неопознанные образы.

Так же хотелось бы заметить, что можно было бы сказать, что на качество распознавания и количество верных срабатываний может прямо влиять то, как обучена LBP модель распознавания определённого образа (какое количество исходников задано для создания модели необходимой при детектировании). Однако, при тестировании обученных каскадов использовались идентичные LBP модели и на конечный результат это не повлияло. Стоит отметить, что записи, на которых было сильно заметно искажение из-за эффекта «рыбьего глаза» каскад все равно пытался детектировать образ студента и делал это довольно успешно.

Связано это с тем, что при обучении каскадов были задействованы все признаки Хаара, а при обучении LBP модели использовалось порядка 60 изображений. Все это дало возможность иметь устойчивость к небольшим искажениям (именно искажениям исходного видеопотока, а не боковым поворотам самих студентов). Исходя из представленных графиков можно заметить, что разница между каскадом, который был обучен на двух тысячах положительных изображений и на каскаде, который представлял всего лишь четверть от первого довольно заметна и очевидна.

Каскад, который во время обучения задействовал гораздо большую выборку по изображениям показал достойные результаты. Тем не менее, чтобы добиться гораздо более высоких результатов, необходимо еще больше как положительных, так и отрицательных изображений, поскольку это на прямую влияет на то, как успешно будет происходить детектирование лиц и последующее распознавание образов.

Заключение

В процессе исследования поставленного вопроса по распознаванию образов студентов в рамках данной выпускной квалификационной работы были достигнуты определенные результаты.

После того, как была четко поставлена задача, было обучено два каскадных классификатора на выборках, разного объема. Обучение каскадных классификаторов задействовало изображения, полученные с помощью специально реализованной под эту цель программой, которая и подготовила большую часть изображений. Далее были обучены LBP модели с помощью лиц необходимых людей, использование которых позволило на распознанном лице детектировать образ определенного студента.

Реализованная система была протестирована на девяти видеофрагментах, состоящих из трех уникальных и еще шести производных, которые были получены с помощью модифицирования тех трех исходных. Результаты тестирования были записаны и проанализированы, была получена сводная статистка о качестве их работы.

Последним этапом была реализация программы ведению автоматического учета посещаемости. Она позволила автоматизировать просмотр посещаемости как группы в целом, так и в разрезе по датам по каждому отдельно взятому студенту.

Реализованная программа получила довольно простой и одновременно интуитивно понятный интерфейс. Тем не менее, на данном этапе в реализованной программе недостаточно полей для более удобного представления информации в графическом виде для педагогов. Для повышения удобства необходимо больше общей информации о том, как программу хотели бы видеть сами педагоги.

Исходя из вышесказанного, можно сделать вывод о том, что цели, поставленные в данной выпускной квалификационной работе, были достигнуты.

Список используемой литературы

1. 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.
2. Научное общество GraphiCon [Электронный ресурс]. – Режим доступа: https://www.graphicon.ru/html/2012/conference/RU2%20-%20Vision/gc2012yuzhakov.pdf – Расширенный набор характеристик каскада. 2012.
3. Портал информатики для гиков [Электронный ресурс]. – Режим доступа: http://espressocode.top/python-haar-cascades-for-object-detection/ – Python | каскады Хаара для обнаружения объектов. 2019.
4. Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/133826/ – Метод Виолы Джонса. 2011.
5. Хабр [Электронный ресурс]. – Режим доступа: https://habr.com/ru/post/208092/ – Обучение OpenCV каскада Хаара. 2014.
6. Яндекс Алиса [Электронный ресурс]. - Режим доступа: https://yandex.ru/alice/to_know_alice - Официальный сайт Яндекс Алисы
7. Betaface API version 2.0 [Электронный ресурс]. - Режим доступа: https://www.betafaceapi.com/wpa/index.php/documentation - Официальный сайт BetaFace Api
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. Da'san M, Alqudah A, Debeir O (2015) [Электронный ресурс]. – Режим доступа: https://www.researchgate.net/publication/308735816_Face_detection_using_Viola_and_Jones_method_and_neural_networks - Face detection using Viola and Jones method and neural networks.

12. Face Detection Based on Viola-Jones Algorithm Applying Composite Features [Электронный ресурс]. – Режим доступа: https://ieeexplore.ieee.org/abstract/document/8806572 - Face detection algorithm 2016

13. Face Detection Techniques [Электронный ресурс]. - Режим доступа: https://www.researchgate.net/publication/326667118_Face_Detection_Techniques_A_Review#pfd - Face Detection Techniques

14. Face Detection Using OpenCV and Haar Cascades Classifiers [Электронный ресурс]. - Режим доступа: https://www.researchgate.net/publication/343127730_Face_Detection_Using_OpenCV_and_Haar_Cascades_Classifiers - Face detection by OpenCV

15. Local Binary Patterns and Its Application to Facial Image Analysis: A Survey [Электронный ресурс]. - Режим доступа: https://hal.archives-ouvertes.fr/hal-01354386/document - About LBP model

16. Multi-view Face Detection and Recognition using Haar-like Features [Электронный ресурс]. - Режим доступа: https://www.researchgate.net/publication/238560015_Multi-view_Face_Detection_and_Recognition_using_Haar-like_Features - Haar-like Features

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

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

19. Performances of the LBP Based Algorithm over CNN Models for Detecting Crops and Weeds with Similar Morphologies [Электронный ресурс]. - Режим доступа: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7218891/ - LBP model perfomance

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

21. 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.

22. Sighthound Sentinel [Электронный ресурс]. - Режим доступа: https://www.sighthound.com/products/sighthound-sentinel - Официальный сайт Sighthound Cloud Api

23. 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.

24. 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.

25. 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.