Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ Информатика и системы управления

КАФЕДРА Теоретическая информатика и компьютерные технологии

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ ПО КУРСУ АЛГОРИТМЫ КОМПЬЮТЕРНОЙ ГРАФИКИ НА ТЕМУ:

Алгоритмы анализа закрытия глаз при получении изображения через камеру с инфракрасной подсветкой

Студент _____ _____

подпись, дата *фамилия, и.о.*

Научный руководитель _____ _____

подпись, дата *фамилия, и.о.*

2022

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

На первый взгляд ценность алгоритмов моргания глаз может быть недооценена, так как способы их применения не очевидны. На самом деле существует много областей, в которых эти алгоритмы используются. Например, наиболее распространенная из них — айтрекинг(окулография). Айтрекинг — методика распознавания движений глаз(в том числе моргание). Окулография часто используется в медицине для диагностики и лечения аутизма, болезни Паркинсона и других заболеваний. Немаловажную роль эта методика играет в обеспечении безопасности на дорогах, так как все больше и больше компаний, занимающихся перевозками, внедряют системы для контроля за состоянием водителя. Также, айтрекинг стал популярен в области общения с обездвиженными людьми, существует устройство, которое реагирует на движение глаз больного, и на каждый сигнал глаз больного устройство выполняет определенную команду. Например, если пациент моргнул два раза левым глазом, программа включит телевизор или позовет медсестру. Именно с помощью такой системы ученый Стивен Хокинг успешно взаимодействовал с окружающим миром несмотря на то, что он не мог двигаться и говорить. Поэтому на самом деле данные алгоритмы являются очень актуальными в наше время, так как позволяют решить ряд проблем в разных областях.

Целью данной курсовой работы является изучение алгоритмов распознавания моргания или закрытия глаз, также, будут рассмотрены отличия изображений, снятых на камеру с инфракрасной подсветкой, от обычных изображений.

Задачу распознавания моргания или закрытия глаз можно разделить на некоторые подпункты:

- локализация лица человека и выделение этой области
- локализация глаз человека в найденной области
- определение закрыты или открыты глаза методом яркого зрачка

В данной курсовой работе мы будем рассматривать возможные решения каждой из этих трех подзадач.

1. Отличия ИК-изображений от обычных

Начальным этапом работы подобных алгоритмов является получение изображения, при этом для того, чтобы результат получился максимально точным, изображения должны быть относительно независимы от условий освещения и должны облегчать задачу распознавания глаз. С этой целью и были выбраны ИК-изображения, так как они сводят к минимуму влияние изменений окружающего освещения. Также, ИК-подсветка создает эффект яркого зрачка, который составляет основу нашей системы обнаружения и отслеживания[1, с. 65]. Яркий зрачок получается, если глаза освещаются ИК-осветителем, излучающим свет вдоль оптической оси камеры. В ИК-диапазоне сетчатка отражает почти весь ИК-свет, полученный по пути обратно в камеру, и на изображении создается эффект яркого зрачка. При освещении вне оптической оси камеры зрачки кажутся темными, поскольку отраженный от сетчатки свет не попадает в объектив камеры. Пример эффекта яркого/темного зрачка можно увидеть на рис. 1. Этот эффект зрачка виден в очках и без них, с контактными линзами и даже в некоторой степени работает с солнцезащитными очками.

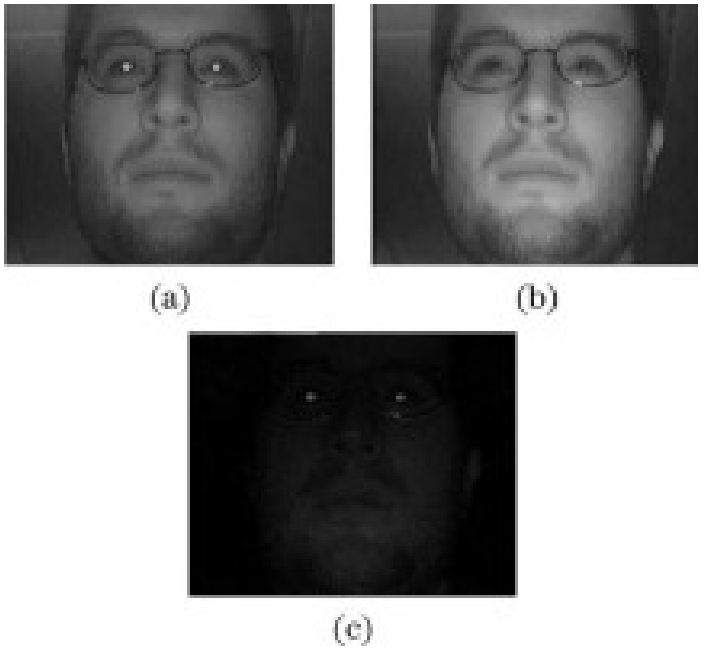

Рис.1. Пример эффект яркого и темного зрачка.

Немаловажным фактором при выборе освещения для применения алгоритмов распознавания моргания глаз в области контроля за состоянием водителя является то, что ближняя ИК-подсветка не улавливается водителем и не мешает ему контролировать ситуация на дороге.

2. Обзор предметной области

Первым этапом работы алгоритмов моргания или закрытия глаз является локализация области лица, для данной задачи существует несколько алгоритмов:

- метод контурного анализа
- метод Виолы-Джонса

2.1 Метод контурного анализа

Контурный анализ является совокупностью методов выделения, описания и преобразования контуров изображения. Контур целиком определяет форму изображения и содержит всю необходимую информацию для распознавания изображений по форме[2, с.2]. Алгоритм такого метода:

- Перевод изображений в градации серого
- Поиск границ оператором Собеля
- Нахождение контуров
- Аппроксимация кривых

Перевод изображений в градации серого заключается в том, что для конкретных изображений все значения пикселей переводятся в одномерную величину, показывающую прозрачность каждого пикселя. Данная величина может лежать только в интервале от 0 до 255, где значение ноль представляет черный цвет, а 255 представляет белый цвет. Для данного преобразования используется следующая формула(1)[3, с. 170]:

$$f(x,y)=0.299\,R(x,y)+0.587\,G(x,y)+0.114\,B(x,y) \qquad (1)$$

Рассмотрим подробнее оператор Собеля:

Результатом работы этого оператора является нулевой вектор в точке с постоянной яркостью или вектор, пересекающий границу областей с различной яркостью и

направленный в сторону увеличения яркости, в точке, находящейся на границе таких областей [3, с. 7-8]. Алгоритм его работы заключается в формулах(2,3), приведенных ниже, где A — исходное изображение.

$$\mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \tag{2}$$

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}} \tag{3}$$

2.2 Метод Виолы-Джонса:

Метод Виолы-Джонса — это алгоритм, позволяющий распознавать объекты на изображениях в реальном времени. Данный алгоритм основан на следующих этапах:

- перевод изображения в интегральное представление
- признаки Хаара

2.2.1 Интегральное представление изображения

Изображение в интегральном представлении — это матрица размером с исходное изображение, где в каждом ее элементе хранится сумма интенсивности пикселей, находящихся левее и выше данного элемента[4, с. 49]. Элементы такой матрицы рассчитываются по следующей формуле(4):

$$F(x, y) = \sum_{i=0,\, j=0}^{i \le x,\, j \le y} I(i, j) \ , \tag{4}$$

где F(x, y) — интегральное изображение, I(i, j) — исходное изображение. Пример

расчета интегрального представления изображения приведен на рисунке 2.

0	1	1	1
1	2	2	3
1	2	1	1
1	3	1	0

Исходное изображение $I(i, j)$

0	1	2	3
1	4	7	11
2	7	11	16
3	11	16	21

Интегральное изображение $L(x, y)$

Рис.2. Пример интегрального изображения

Основным преимуществом интегрального представления изображений является возможность очень быстро вычислить сумму пикселей произвольного прямоугольника, такой расчет происходит за линейное время, пропорциональное числу пикселей исходного изображения.

2.2.2 Признаки Хаара

Поиск нужного объекта происходит с помощью признаков Хаара, в методе Виолы-Джонса они организованы в каскадный классификатор, который применяется к каждому положению скользящего окна, которое двигается по исходному изображению с шагом 1 пиксель с изменением масштаба при каждом проходе [5, с. 1]. В случае нахождения признаков, подходящих под условие задачи, данное окно фиксируется. В зависимости от типа задачи поиска, используются разные признаки Хаара, в стандартном методе Виолы-Джонса используются прямоугольные признаки, называющиеся примитивами Хаара, они приведены на рисунке 3.

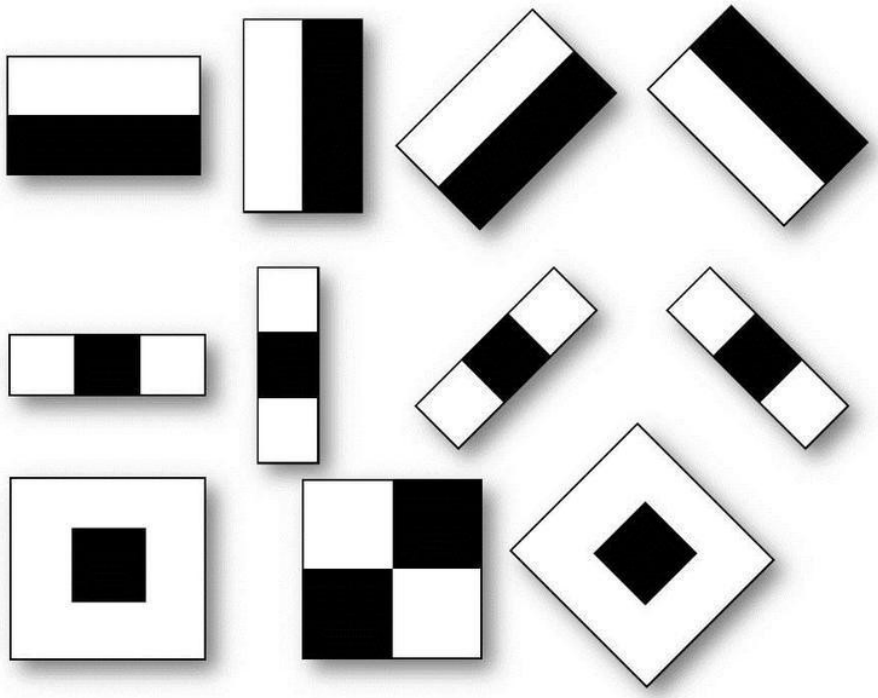

Рисунок 3 - Примитивы Хаара

Классификаторы могут быть обучены на разные области, такие как: части тела, автомобильные номера или другие определенные предметы. Для обучения классификатора понадобятся позитивные и негативные изображения, где позитивные изображения — изображения, содержащие нужный объект, негативные — изображение, не содержащие данный объект. При этом желательно, чтобы пропорции оставались неизменны, то есть масштаб объекта, который необходимо обнаружить, на позитивных изображениях был приблизительно такой же, как на изображениях, на которых этот классификатор будет в дальнейшем применяться.

Задача построения единственного классификатора с высоким процентом обнаружения нужного объекта является довольно сложной, а иногда нерешаемой, поэтому чаще используют каскадную структуру классификаторов(пример такого классификатора приведен на рисунке 4), состоящую из более слабых классификаторов[6, с. 37]. Под слабым классификатором подразумевается классификатор, который получает правильный ответ с той же вероятностью, что и случайное угадывание. Однако, множество таких классификаторов в совокупности

дадут довольно сильный классификатор. Таким образом, на каждом этапе принимается решение принадлежит ли данный объект интересующему множеству, если объект в данное множество не входит, проверка дальше не продолжается, и процесс классификации останавливается, так как в итоговое интересующее нас множество данный объект точно не входит. В ином случае, проверка продолжается, пока не дойдет до последнего этапа, и классификатор не обнаружит, что данный объект принадлежит нужному множеству.

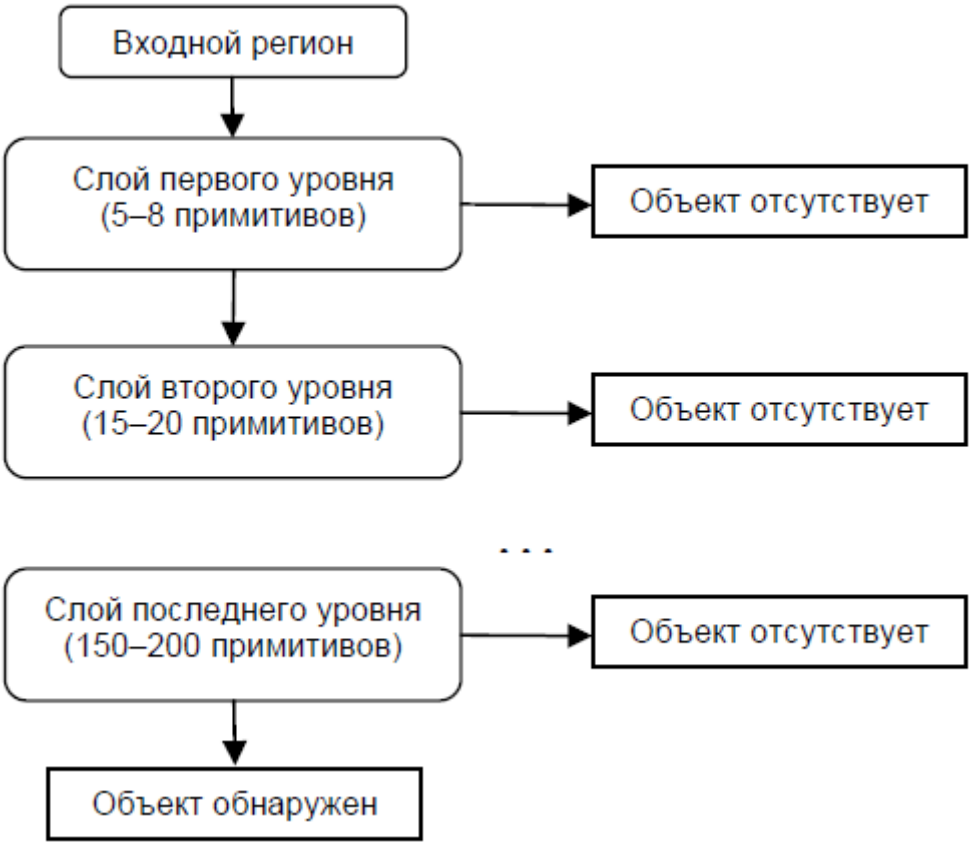

Рис.4. Пример каскадного классификатора

2.3 Эффект яркого зрачка

Эффект яркого зрачка на изображении достигается при помощи использования инфракрасной подсветки. В центре объективов подобных камер расположены точечные светодиоды, излучающие слабый инфракрасный свет, при попадании его на глаза появляются такие эффекты как[7, с. 36]:

- роговичный блик
- эффект яркого зрачка

Пример таких явлений приведен на рисунке 5.

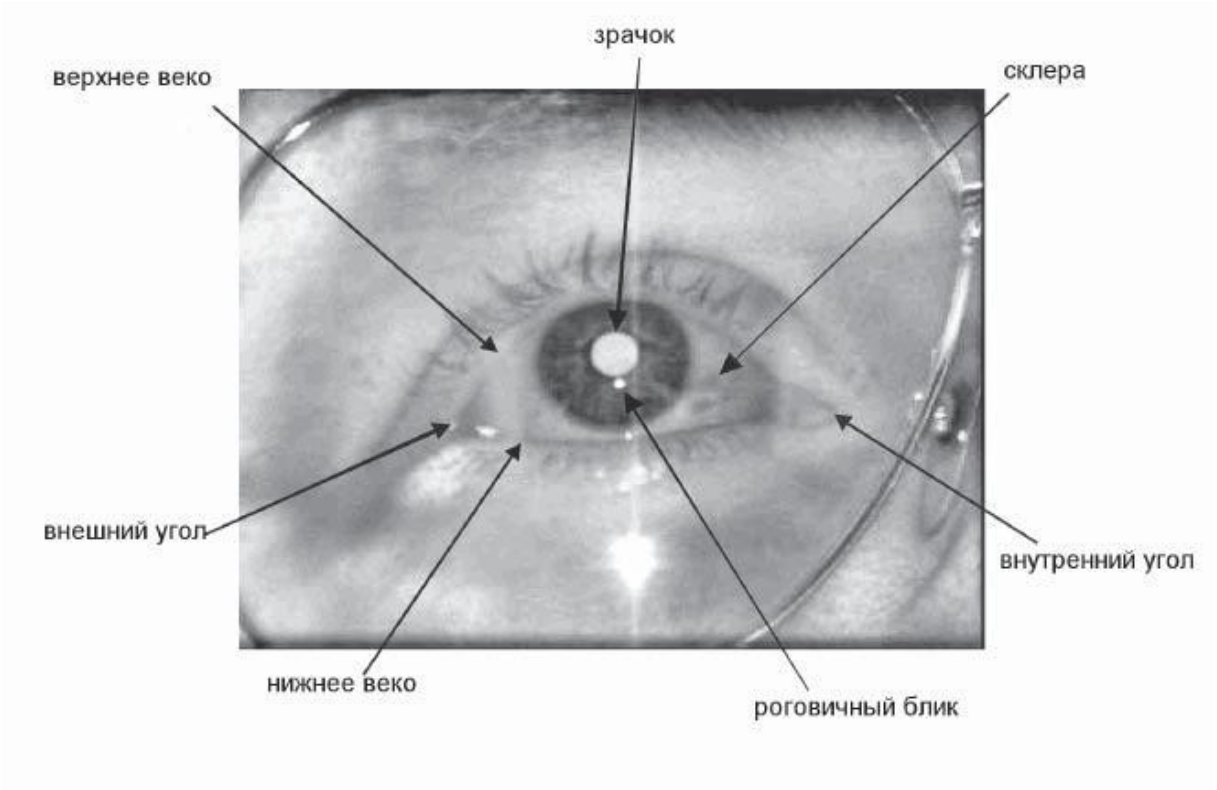

Рис.5. Эффект яркого зрачка

3. Реализация алгоритмов

Так как изображения представляются в виде матрицы, то для реализации описанных алгоритмов была выбрана библиотека Python NumPy. Её основными преимуществами являются более компактное хранение массивов, возможность выполнения операций над матрицами и оптимизация этих операций[8].

Для получения и сохранения изображения была использована библиотека OpenCV, с помощью нее при получении изображения можно сразу перевести его в оттенки серого при помощи функции cv2.imread(filename[, flags]) и параметра(flags) IMREAD_GRAYSCALE[9].

Применив эту же библиотеку можно использовать метод Виолы-Джонса, так как она содержит функции, позволяющие использовать классификатор Хаара. В рамках конкретно данного алгоритма нужно поэтапно выделить область лица человека на изображении, а затем в готовой области найти область обоих глаз. Для этого используется функция cv2.CascadeClassifier(flags), где в параметре flags указывается путь к файлу со значениями распознавания определенного объекта. Для распознавания лица используется файл «haarcascade_frontalface_default.xml», а для распознавания глаз — файл "haarcascade_eye.xml".

Далее используется функция detectMultiScale() для нахождения областей локализации нужного объекта. Данная функция имеет несколько параметров:

1. frame — исходное изображение
2. scaleFactor — параметр, который определяет величину, на которую будет увеличиваться размер скользящего окна распознавания на каждой последующей итерации. Для распознавания лица было выбрано значение равное 1.2, то есть окно на каждой итерации будет увеличиваться на 20%, для поиска лиц такого значения будет достаточно, при этом брать всегда наименьшее возможное значение не целесообразно, так как чем больше

величина этого параметра, тем быстрее будет работать данный алгоритм. Для распознавания глаз была взята величина равная 1.1, так как область глаз в несколько раз меньше области лица.

3. minNeighbors — параметр, который определяет сколько похожих прямоугольников должно находиться рядом с прямоугольником-кандидатом, чтобы он был выбран. Данный параметр определяет качество обнаружения: чем выше величина, тем точнее будет работа алгоритма, и тем меньше будет ложных распознаваний объекта. Однако, завышать этот параметр не рекомендуется, так как есть риск, что программа пропустит нужный объект, посчитав его ложным. Оптимальными значениями данного параметра являются 3-6. В рамках данной работы и для обнаружения лица человека, и для обнаружения глаз был выбран параметр равный 5.

4. minSize — параметр, указывающий минимальный размер рамки

Функция detectMultiscale() возвращает список с параметрами (x, y, w, h), по которым в дальнейшем строится прямоугольник, описывающий область, содержащую нужный объект. Прямоугольник строится с помощью библиотеки OpenCV и функции cv2.rectangle(), принимающую на вход параметр frame(исходное изображение), координаты точек верхнего левого угла прямоугольника и нижнего правого угла, а также, цвет граней в формате RGB.

Следующим этапом в работе данного алгоритма после обнаружения областей лица и глаз является определение закрыты глаза на изображении или открыты. Основой идеи для данного процесса является то, что при получении изображения с помощью камеры с инфракрасной подсветкой появляется эффект яркого зрачка, то есть в области глаз находится яркое пятно белого цвета[10]. Так как изначально исходное изображение было переведено в оттенки серого, то в найденной области будет выделяться только блик от зрачка, и он будет иметь самую высокую интенсивность, то есть при получении значения параметра оттенков серого пикселя, принадлежащего области зрачка, будет выведен результат 255, либо близкий к нему.

Исходя из этого, если в области глаз найдется хотя бы один пиксель, интенсивность которого будет равна 255, можно сделать вывод, что глаз открыт, если пикселя с таким значением нет, значит глаз закрыт. Для корректного отображения результата при получении значения яркости пикселя была использована библиотека Python Image Library (PIL)[11].

4. Тестирование

Итак, после реализации алгоритмов необходимо проверить их на работоспособность и точность. Проверка будет проводиться поэтапно:

-проверка работы алгоритма распознавания лиц на изображении на точность

-проверка работы алгоритма распознавания глаз на изображении на точность

-проверка работоспособности алгоритма распознавания открытых или закрытых глаз на уже выделенной ранее области

Тестирование будет проводиться сразу на двух изображениях: на одном из них будет фотография человека с открытыми глазами, на другом — с закрытыми.

4.1 Распознавание лиц

Далее на рисунках 6 и 7 будут приведены изображения человека, полученные через камеру с инфракрасной подсветкой, а на рисунках 8 и 9 будут приведены изображения, полученные после работы алгоритма по распознаванию лиц.

Рис.6. Исходное изображение с закрытыми глазами

Рис.7. Исходное изображение с открытыми глазами

Рис.8 Изображение с закрытыми глазами после работы алгоритма

Рис.9. Изображение с открытыми глазами после работы алгоритма

Из вышеуказанных рисунков видно, что алгоритм распознавания лица работает достаточно точно как для изображений с открытыми глазами, так и с открытыми: нет ложных распознаваний, но при этом нужная область найдена безошибочно. Из этого можно сделать вывод, что параметры функции, осуществляющей данный алгоритмы были подобраны корректно.

4.2 Распознавание глаз

Теперь на этих же исходных изображениях, приведенных на рисунках 6 и 7, будет проведена проверка алгоритма по локализации области глаз. Однако, стоит отметить, что для экономии времени работы программы, область глаз ищется в уже выделенном прямоугольнике, в котором было распознано лицо, поэтому результатом работы данного этапа алогоритма будет изображение с выделенными участками лица и глаз. Пример работы алгоритма на данной стадии приведен на рисунках 10 и 11.

Рис.10. Изображение с закрытыми глазами после работы алгоритма

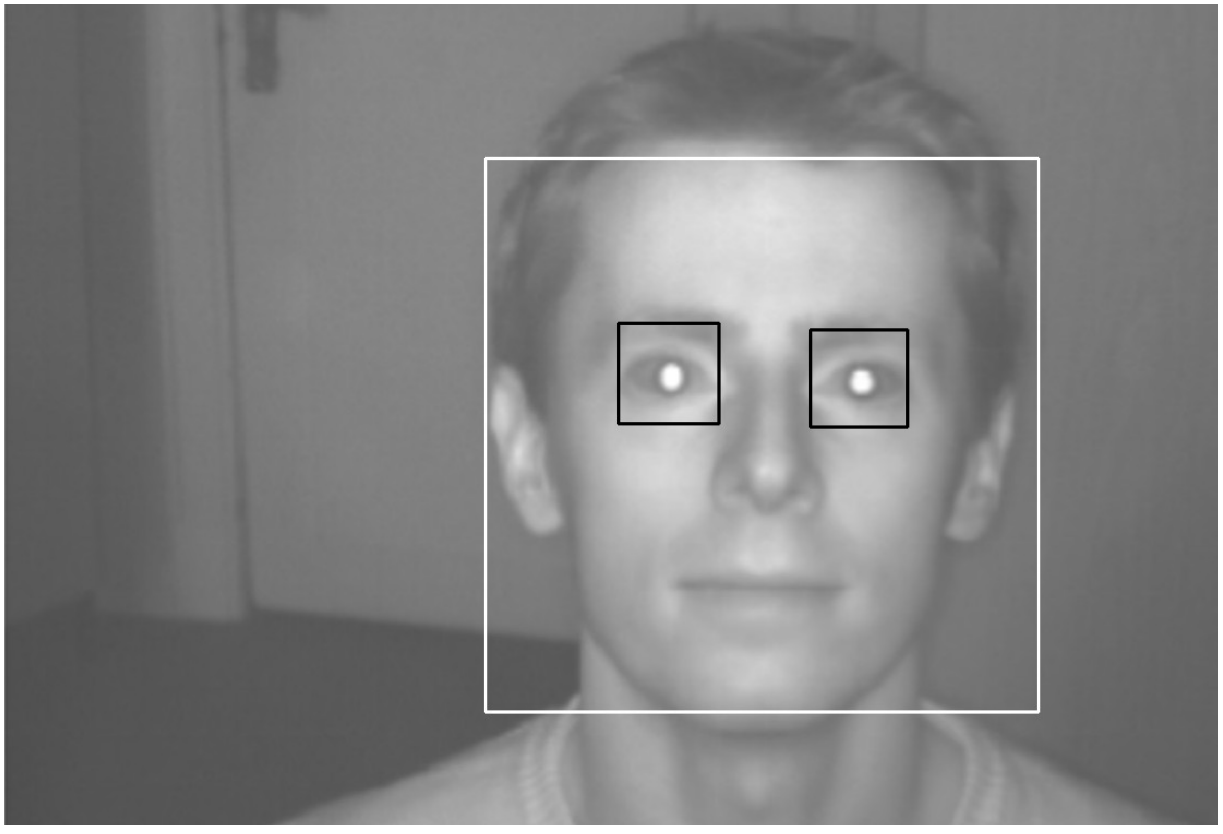

Рис.11. Изображение с открытыми глазами после работы алгоритма

Из приведенных примеров работы программы видно, что распознавание областей глаз производится так же корректно.

4.3 Определение открытых или закрытых глаз

На данном этапе алгоритм программы определяет закрыты или открыты глаза в найденной области. Стоит обратить внимание, что левый и правый глаз в области лица ищутся отдельно, соответственно, и определение в каком состоянии находятся глаза: открыты или закрыты - тоже происходит отдельно для каждого из них. То есть вывод программы для каждого глаза будет отдельный. Вывод программы для изображений, приведенных на рисунках 6 и 7, изображен на рисунках 12 и 13.

```
pygame 2.1.2 (SDL 2.0.16, Python 3.6.9)
Hello from the pygame community. https://www.pygame.org/contribute.html
left eye closed
right eye closed
```

Рис. 12. Результат работы алгоритма для рисунка 6

```
pygame 2.1.2 (SDL 2.0.16, Python 3.6.9)
Hello from the pygame community. https://www.pygame.org/contribute.html
left eyes open
right eyes open

Process finished with exit code 0
```

Рис.13. Результат работы алгоритма для рисунка 7

ЗАКЛЮЧЕНИЕ

В рамках выполнения данной курсовой работы были изучены некоторые из методов распознавания определенных объектов на изображении, наиболее подробно был рассмотрен метод Виолы-Джонса, на базе которого в дальнейшем и был основан алгоритм получившейся программы. Также, были подробно разобраны библиотеки и функции, использующиеся в программе: задачи, решающиеся с помощью приведенных методов, значения и параметров и вывод.

В ходе тестирования алгоритма были получены верные результаты, из чего следует сделать вывод, что все алгоритмы работают корректно, и программа была написана верно.

Одним из возможных недостатков получившейся программы может быть долгое время выполнения при получении на вход изображения большого размера и неустойчивая работа алгоритма при получении на вход изображения в очень плохом качестве. Для решения первой проблемы могут применяться алгоритмы сжатия изображения, но этот процесс должен производиться либо без потери качества изображения, либо с незначительной потерей, которая будет незаметна для алгоритма распознавания, иначе такие действия могут привести ко второму недостатку программы. Для решения второй проблемы могут применяться алгоритмы увеличения качества изображения.

СПИСОК ИСПОЛЬЗУЕМОЙ ЛИТЕРАТУРЫ

1. Bergasa L.M. Real-Time System for Monitoring Driver Vigilance/ Bergasa L.M., Nuevo J., Sotelo M. A., Barea R., Lopez M.E. // IEEE Transactions On Intelligent Transportation Systems. 2006. Vol. 7. No. 1.

2. Серикова А.С. Сегментация и распознавание автомобильных регистрационных номеров// Томский политехнический университет. 2016. С. 2.

3. Лапин М.В. Методы обработки изображения для выделения заранее определенных меток и маркеров// Труды молодых ученых Алтайского государственного университета. 2019. № 16. С. 170-173.

4. Эрман, Е. А. Метод обнаружения лиц на изображении с использованием комбинации метода Виолы - Джонса и алгоритмов определения цвета кожи / Е. А. Эрман, Мамдух Мохаммед Гомаа Мохаммед // Вестник Астраханского государственного технического университета. Серия: Управление, вычислительная техника и информатика. 2015. № 1. С. 49-55.

5. Сыздыкова Г.Ж. Детектирование автомобильных номеров// Томский политехнический университет. 2016. С. 2.

6. Малышева, С. С. Модифицированный алгоритм локализации элементов лица на основе метода Виолы-Джонса / С. С. Малышева // Искусственный интеллект и принятие решений. 2015. № 1. С. 35-44.

7. Барабанщиков, В. А. Организация движений глаз при восприятии изображений лица / В. А. Барабанщиков, К. И. Ананьева, В. Н. Харитонов // Экспериментальная психология. – 2009. – Т. 2. – № 2. – С. 31-60.

8. Документация библиотеки NumPy. [Электронный ресурс]. - URL: https://numpy.org/doc/stable/reference/

9. Документация OpenCV по фильтрам изображений. [Электронный ресурс].- URL: https://docs.opencv.org/4.x/d4/d13/tutorial_py_filtering.html

10. Грушко Ю.В. Аппаратно-программный комплекс аугментативной системы коммуникации на основе технологии Eyetracking // Вест. КРАУНЦ. Физ.-мат. науки.

2019. №2.
10. Документация библиотеки PIL. [Электронный ресурс].- URL: https://pillow.readthedocs.io/en/stable/reference/Image.html

Приложение А. Листинги функций

В листингах 1-3 приведен исходный код функций, отвечающих за выполнение алгоритмов распознавания.

Листинг 1 — реализация алгоритма распознавания лиц на изображении

```
face_detector =cv2.CascadeClassifier(cv2.data.haarcascades +
"haarcascade_frontalface_default.xml")
frame = cv2.imread('image9.jpg', cv2.IMREAD_GRAYSCALE)
faces = face_detector.detectMultiScale(
    frame,
    scaleFactor=1.2,
    minNeighbors=5,
    minSize=(50, 50),
    flags=cv2.CASCADE_SCALE_IMAGE
)
    for (x, y, w, h) in faces:
        cv2.rectangle(frame, (x, y), (x+w, y+h), (255, 255, 0), 2)
```

Листинг 2 — реализация алгоритма распознавания левого глаза

```
eyes_detector = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_eye.xml")
img_face_left = frame[y:y+h, x:x+int(w/2)]
open_eyes_glasses_left = eyes_detector.detectMultiScale(
  img_face_left,
  scaleFactor=1.1,
  minNeighbors=5,
  minSize=(30, 30),
  flags=cv2.CASCADE_SCALE_IMAGE
  )
  for (ex, ey, ew, eh) in open_eyes_glasses_left:
```

```
11      cv2.rectangle(frame, (x+ex, y+ey), (x+ex + ew, y+ey + eh), (0, 255, 0), 2)
```

Листинг 3 — реализация алгоритма распознавания правого глаза

```
1 eyes_detector = cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_eye.xml")
2 img_face_right = frame[y:y+h, x+int(w/2):x+w]
3 open_eyes_glasses_right = eyes_detector.detectMultiScale(
4    img_face_right,
5    scaleFactor=1.1,
6    minNeighbors=5,
7    minSize=(30, 30),
8    flags=cv2.CASCADE_SCALE_IMAGE
9    )
10   for (ex, ey, ew, eh) in open_eyes_glasses_right:
11          cv2.rectangle(frame, (x+int(w/2)+ex, y+ey), (x+int(w/2)+ex + ew, y+ey + eh), (0,
255, 0), 2)
```

Листинги 5-6 демонстрируют исходный код программы, который определяет открыты или закрыты глаза

Листинг 5 — реализация алгоритма определения состояния левого глаза

```
1 for (ex, ey, ew, eh) in open_eyes_glasses_left:
2    cv2.rectangle(frame, (x+ex, y+ey), (x+ex + ew, y+ey + eh), (0, 255, 0), 2)
3    for i in range (x+ex, x+ex+ew):
4        for j in range (y+ey, y+ey+eh):
5            cpixel = pixels[i,j]
6            if cpixel >= 250:
7                res_left = 1
8                break
9  if res_left == 1:
10     print("left eye open")
```

```
11 else:
12     print("left eye closed")
```

Листинг 6 — реализация алгоритма определения состояния правого глаза

```
1 for (ex, ey, ew, eh) in open_eyes_glasses_right:
2    cv2.rectangle(frame, (x+int(w/2)+ex, y+ey), (x+int(w/2)+ex + ew, y+ey + eh), (0, 255, 0), 2)
3    for i in range (x+int(w/2)+ex, x+int(w/2)+ex+ew):
4        for j in range (y+ey, y+ey+eh):
5            cpixel = pixels[i,j]
6            if cpixel >= 250:
7                res_right = 1
8                break
9  if res_right == 1:
10     print("right eye open")
11  else:
12      print("right eye closed")
```

Приложение В. Результаты тестирования

На рисунках 14- 19 приведены дополнительные примеры работы программы.

Рис. 14.Исходное изображение

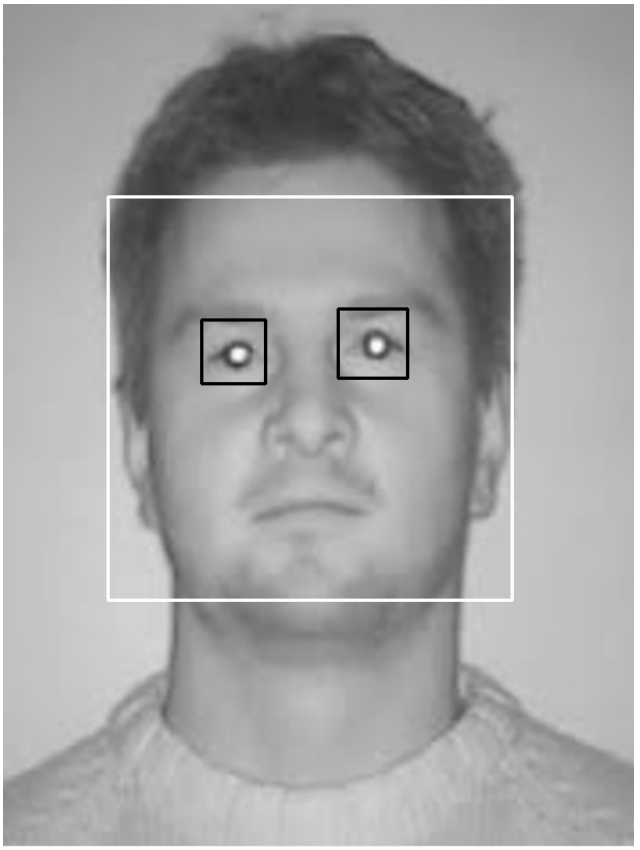

Рис.15. Изображение после работы алгоритма

```
pygame 2.1.2 (SDL 2.0.16, Python 3.6.9)
Hello from the pygame community. https://www.pygame.org/contribute.html
left eye open
right eye open

Process finished with exit code 0
```

Рис.16. Вывод программы

Рис.17. Исходное изображение

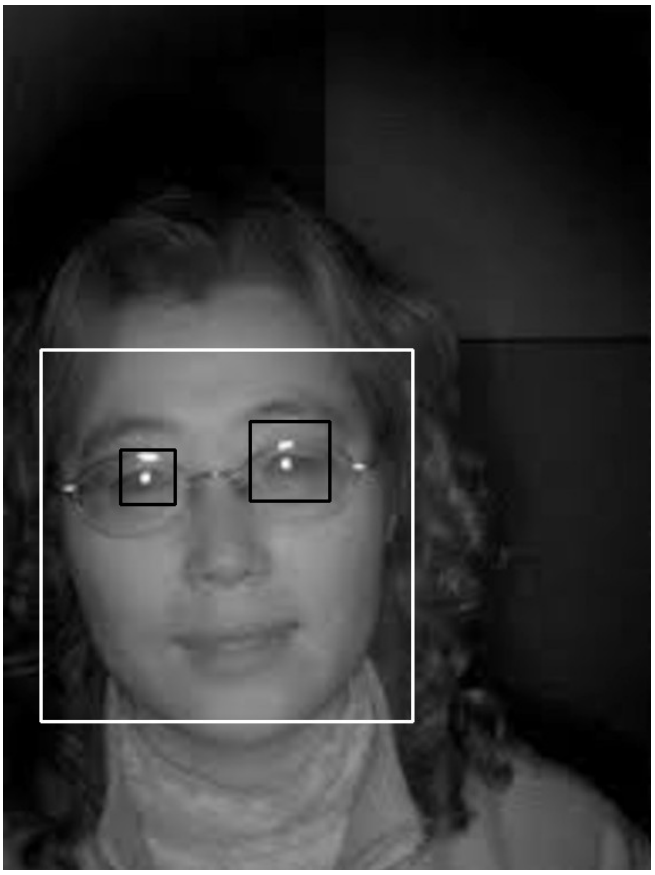

Рис.18. Изображение после работы алгоритма

```
pygame 2.1.2 (SDL 2.0.16, Python 3.6.9)
Hello from the pygame community. https://www.pygame.org/contribute.html
left eye open
right eye open

Process finished with exit code 0
```

Рис.19. Вывод программы