**Министерство образования и науки Российской Федерации**

Федеральное государственное бюджетное образовательное учреждение высшего образования

**«ТОМСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ СИСТЕМ УПРАВЛЕНИЯ И РАДИОЭЛЕКТРОНИКИ» (ТУСУР)**

Кафедра компьютерных систем в управлении и проектировании (КСУП)

ПРОГРАММНЫЙ ПРОДУКТ ПО РАСПОЗНАВАНИЮ ГЛАЗ ЧЕЛОВЕКА НА ФОТО И В ВИДЕОПОТОКЕ НА ОСНОВЕ БИБЛИОТЕКИ OPENCV С ИСПОЛЬЗОВАНИЕМ КАСКАДОВ ХААРА

Курсовая проект по предмету «Интеллектуальные системы»

|  |  |
| --- | --- |
|  | Студент группы 586-M1  Гаан С.Е.  « » 2018  К.т.н., доцент каф. КСУП  \_\_\_\_\_\_\_\_\_\_\_\_\_ Калентьев А. А.  « » 2018 |

Томск 2018

Оглавление

[1 Машинное зрение 3](#_Toc517039588)

[2 Обзор библиотеки компьютерного зрения OpenCV 4](#_Toc517039589)

[3 Генетические алгоритмы распознавания объектов. 6](#_Toc517039590)

[3.1 Метод Виолы-Джонса. 6](#_Toc517039591)

[3.2 Что такое признаки Хаара. 8](#_Toc517039592)

[3.3 Интегральное представление изображения. 9](#_Toc517039593)

[4 Обучение классификатора Виолы-Джонса 10](#_Toc517039594)

[4.1 Алгоритм обучение каскада Хаара 12](#_Toc517039595)

[4.1.1 Подготовительный этап 12](#_Toc517039596)

[4.1.2 Обработка «Хороших» изображений 13](#_Toc517039597)

[4.1.3 Создание основного каскада 15](#_Toc517039598)

[4.2 Тестирование каскада Хаара. 17](#_Toc517039599)

[5 Проект программы 18](#_Toc517039600)

[5.1 Диаграмма классов 18](#_Toc517039601)

[5.2 Макеты пользовательского интерфейса 18](#_Toc517039603)

[5.3 Обнаружение глаз человека при работе с видеопотоком 20](#_Toc517039604)

[5.4 Обнаружение глаз человека при работе с фотографиями 22](#_Toc517039605)

[6 Работа с программой CutIimageArea 23](#_Toc517039606)

[6.1 Формирование файла Good.dat 23](#_Toc517039607)

[6.2Приведение всех положительных изображений к единому виду 23](#_Toc517039608)

[6.3 Форма для обучения каскада Хаара. 25](#_Toc517039609)

[Заключение 28](#_Toc517039610)

[Список использованной литературы 29](#_Toc517039611)

# 1 Машинное зрение

**Машинное зрение** — это применение компьютерного зрения для промышленности и производства. В то время как компьютерное зрение — это общий набор методов, позволяющих компьютерам видеть. [1] Областью интереса машинного зрения, как инженерного направления, являются цифровые устройства ввода-вывода и компьютерные сети, предназначенные для контроля производственного оборудования, такие как роботы-манипуляторы или аппараты для извлечения бракованной продукции. Машинное зрение является подразделом инженерии, связанным с вычислительной техникой, оптикой, машиностроением и промышленной автоматизацией. Одно из наиболее распространенных приложений машинного зрения — инспекция промышленных товаров, таких как полупроводниковые чипы, автомобили, продукты питания и лекарственные препараты. Люди, работавшие на сборочных линиях, осматривали части продукции, делая выводы о качестве исполнения. Системы машинного зрения для этих целей используют цифровые и интеллектуальные камеры, а также программное обеспечение, обрабатывающее изображение для выполнения аналогичных проверок. [2]

# 2 Обзор библиотеки компьютерного зрения OpenCV

(Open Source Computer Vision Library) — это библиотека компьютерного зрения, которая поставляется с открытым исходным программным кодом. Спектр возможностей данной библиотеки очень широк. В ней собрано большое количество алгоритмов для использования технологий компьютерного зрения. После подключения данной библиотеки к своему проекту пользователь получает доступ к более чем 500 функций, предназначенных для решения разнообразных задач. Помимо алгоритмов для работы с технологиями компьютерного зрения, данная библиотека применяется и для обработки изображений, содержит большое число численных алгоритмов и многое другое. Данная библиотека реализована на языках программирования C/C++, однако поставляется также и для других языков (таких как Python, Java, Ruby, Matlab, Lua и т.д). Библиотека может использоваться на различных операционных системах, к числу которых относятся Linux, Mac OS X, iOS, Android и др. Целью разработки данной библиотеки является повышение эффективности вычислений в приложениях реального времени. Язык C, на котором была написана библиотека, является оптимизированным. Библиотека OpenCV способна использовать многоядерные процессоры. Главной целью библиотеки OpenCV является предоставление легкого в использовании интерфейса, который поможет облегчить использование технологий компьютерного зрения в довольно сложных приложениях. Функции, которые поддерживает библиотека, охватывают разнообразные сферы компьютерного зрения, от медицины, безопасности и до стереозрения и робототехники. Все это благодаря тому, что компьютерное зрение и машинное обучение — два неразрывно связанных понятия. Кроме того, библиотека OpenCV содержит библиотеку MLL (с английского Machine Learning Library). Даная библиотека является библиотекой общего назначения и ориентирована на распознавание статических образов и технологию кластеризации. Данная библиотека является очень эффективной для решения задач компьютерного зрения, которое как раз и является основой OpenCV.

Библиотека OpenCV имеет лицензию, которая была разработана таким образом, что у разработчика появляется возможность создавать коммерческий продукт, при этом использовать функциональные возможности библиотеки либо частично, либо полностью. Основным условием использования OpenCV является тот факт, что разработчик обязан делать свой проект с открытым исходным кодом, а также делать доступными для всех разработанные им улучшения для библиотеки. На сегодняшний день существует сообщество, участники которого делятся опытом и обсуждают различные темы и проблемы, связанные с OpenCV.

Библиотека OpenCV направлена на обеспечение работы инструментов для устранения различных проблем, которые могут возникнуть в компьютерном зрении. Функционала библиотеки будет достаточно для решения довольно сложных задач в области компьютерного зрения. Однако есть такие задачи, для решения которых функционала библиотеки недостаточно. В связи с этим, можно на основе компонентов библиотеки создать собственное решение задачи почти любой сложности. Существуют нескольких проверенных и довольно надежных методов, которые используют компоненты библиотеки. После создания проекта можно найти некоторые слабые места и исправить путем дописывания собственного кода. Полученное решение можно использовать в качестве эталонного и применять в дальнейшем. Таким образом, любое слабое место можно исправить, используя встроенное решение. [3]

# 3 Генетические алгоритмы распознавания объектов.

# 3.1 Метод Виолы-Джонса.

В настоящее время метод Виолы–Джонса является самым популярным методом для поиска области с нужным объектом на изображении в силу своей высокой скорости и эффективности. Пол Виола и Майкл Джонс разработали и представили этот метод в 2001. В основе метода Виолы–Джонса по поиску нужного объекта лежат идеи: интегральное представление изображения по признакам Хаара, метод построения классификатора на основе алгоритма адаптивного бустинга, и метод комбинирования классификаторов в каскадную структуру. Эти идеи позволяют осуществлять поиск лица в режиме реального времени.

Признак Хаара состоит из смежных прямоугольных областей. Эти области позиционируются на изображении, далее происходит суммирование интенсивности пикселей в областях, затем между суммами вычисляется разность. Значение полученной разности и является значением определенного признака, определенного размера, определенным образом расположенного на изображении. На рисунке. 3.1 представлены граничные, центральные и линейные признаки Хаара. рисунке. 3.2 демонстрирует пример использования признаков Хаара. На пример, для всех изображений, область в районе глаз темнее, чем область в районе щек. Следовательно, общим признаком Хаара для лиц является 2 смежных прямоугольных региона, лежащих на глазах и щеках.

На этапе обнаружения в методе Виолы—Джонса используется окно, определенного размера, которое движется по изображению. Признак Хаара рассчитывается для каждой области изображения, над которой проходит окно. Наличие или отсутствие предмета в окне определяется разницей между значением признака и обучаемым порогом. Для описания и поиска нужного объекта с достаточной точностью необходимо большее число признаков. Поэтому в методе Виолы—Джонса признаки Хаара организованы в каскадный классификатор. [5]

Преимущество использования признаков Хаара является наибольшая, по сравнению с остальными признаками, скорость и эффективность. При использовании интегрального представления изображения, признаки Хаара могут вычисляться за постоянное время.

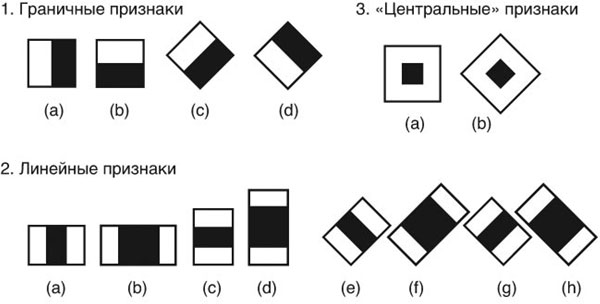


Рисунок 3.1 – Признаки Хаара.

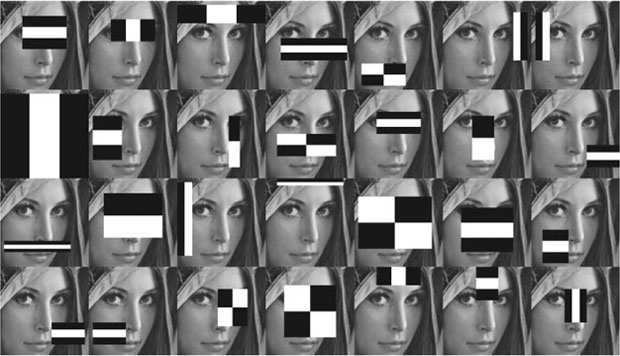


Рисунок 3.2 – Примеры использования признаков Хаара для распознавания лиц людей.

Методу Виолы–Джонса присуща высокая вероятность точного обнаружения нужного объекта, даже при наблюдении объекта под небольшим углом, примерно до 30°. Но при большем угле наклона вероятность обнаружения объекта резко падает.

# 3.2 Что такое признаки Хаара.

Признак - это отображение f ∶ X → D, где D - множество допустимых значений признака. Признаки делятся на следующие типы в зависимости от множества D:

1. Бинарный признак, D = {0,1};
2. Номинальный признак, D - конечное множество;
3. Порядковый признак, D - конечное упорядоченное множество;
4. Количественный признак, D - множество действительных чисел.

На рисунке 3.3 приведено графическое представление нескольких признаков Хаара.

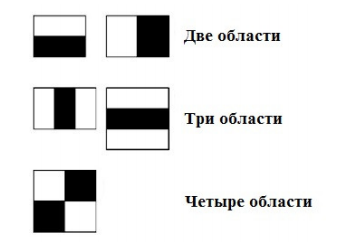


Рисунок 3.3 – Примеры признаков Хаара.

Для окна фиксированного размера признак Хаара представляет собой множество прямоугольных областей белого и черного цветов. Значение признака для исследуемой области изображения вычисляется как: **F = W – B**, где W- это сумма значений пикселей, находящихся в белых областях признака, а B- сумма значений пикселей, находящихся в черных областях. Понятно, что при вычислении значения признака Хаара, напрямую суммируя значения всех необходимых пикселей исследуемой области, будет проделываться n \* m операций, где n и m - это ширина и высота исследуемой области соответственно. Однако данный процесс можно значительно ускорить, если использовать так называемое интегральное представление изображения.

# 3.3 Интегральное представление изображения.

Интегральное представление изображения является матрицей, совпадающей по размерам с исходным изображением. Значение каждой ячейки этой матрицы является суммой интенсивностей всех пикселей, находящихся левее и выше данной ячейки. Элементы матрицы вычисляются по следующей формуле (3.1):

(3.1)

где I[y, x] представляет собой сумму пикселей в прямоугольнике от (0, 0) до (y, x);

Im[i, j] - яркость пикселя исходного изображения.

Расчет интегральной матрицы производится за один проход и занимает линейное время относительно числа пикселей в изображении.

Построив интегральную матрицу можно очень быстро вычислять сумму пикселей произвольного прямоугольника произвольной площади.

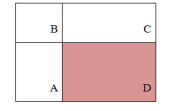


Рисунок. 3.4 Прямоугольник с областями A, B, C и D

Пусть в прямоугольнике ABCD на рис.2 есть интересующий нас объект D. Из рисунка очевидно, что сумму пикселей в прямоугольнике D можно выразить посредством суммы и разности смежных прямоугольников.

Таким образом, значение любого признака Хаара можно вычислить посредством всего нескольких операций, количество которых зависит только от количества прямоугольников в признаке и не зависит от площади исследуемой области.

# 4 Обучение классификатора Виолы-Джонса

Классификатор формируется на примитивах Хаара путём расчёта значений признаков. Для обучения на вход классификатора сначала подаётся набор «Хороших» изображений с предварительно выделенной областью на изображении и набором «Плохих» изображений, на которых отсутствует объект поиска, далее происходит перебор примитивов и расчёт значения признака. Вычисленные значения сохраняются в файле в формате xml.

Классификатор строится на основе алгоритма бустинга (от англ.**boost**–улучшение, усиление) для выбора наиболее подходящих признаков для искомого объекта на данной части изображения. В общем случае бустинг — это комплекс методов, способствующих повышению точности аналитических моделей. Эффективная модель, допускающая мало ошибок классификации, называется «сильной». «Слабая» же, напротив, не позволяет надежно разделять классы или давать точные предсказания, делает большое количество ошибок. Поэтому бустинг означает «усиление» «слабых» моделей и является процедурой последовательного построения композиции алгоритмов машинного обучения, когда каждый следующий алгоритм стремится компенсировать недостатки композиции всех предыдущих алгоритмов.

В результате работы алгоритма бустинга на каждой итерации формируется простой классификатор вида (3.2):

 (3.2)

где – направление знака неравенства,

 – значение порога,

**–** вычисленное значение признака,

z – окно изображения размером 24×24 пикселов.

Полученный классификатор имеет минимальную ошибку по отношению к текущим значениям весов, задействованным в процедуре обучения для определения ошибки.

Для поиска объекта на цифровом изображении используется обученный классификатор, представленный в формате xml. Классификатор формируется на примитивах Хаара.

Структура классификатора:

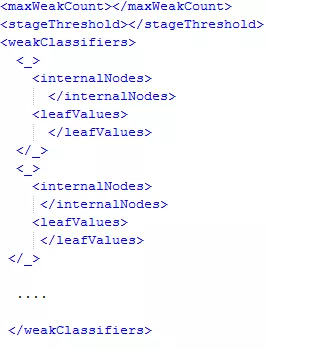


Рисунок. 4.1 – Структура классификатора.

где **maxWeakCount** – количество «слабых» классификаторов;

**stageThereshold** – максимальный порог яркости;

**weakClassifiers** – набор «слабых» классификаторов, на основе которых выносится решение о том, находится объект на изображении или нет;

**internalNodes** и **leafValues** – параметры конкретного «слабого» классификатора.

Первые два значения в internalNodes не используются, третье — номер признака в общей таблице признаков (она располагается в XML-файле под тегом features), четвертое — пороговое значение «слабого» классификатора. Если значение признака Хаара меньше порога «слабого» классификатора, выбирается первое значение leafValues, если больше — второе.

На основе этого базиса строится каскад классификаторов, принимающих решение о том, распознан объект на изображении или нет. Наличие или отсутствие предмета в окне определяется разницей между значением признака и порогом, полученным в результате обучения. [7]

# 4.1 Алгоритм обучение каскада Хаара

# 4.1.1 Подготовительный этап

Утилита для обучения каскада Хаара изначально встроена в пакет OpenCV. Для обучения необходимо собрать коллекцию различных изображений, которые понадобятся в дальнейшем:

1. Реальные фотографии объекта. Чем более похожа выборка будет на то, что нужно будет распознавать, тем лучше будут результаты.
2. Выборка отрицательных фотографий, на которых нет объекта распознавания. Фотографии должны быть сделаны в той же среде, где будет распознавание.

Для обучения нужно иметь 2 папки с примерами:

* «Good» — папка с позитивными изображениями;
* «Bad» — с отрицательными.

Стоит учесть, что OpenCV отказывается работать с точками, пробелами и специальными символами. Нельзя использовать данные символы в названиях примеров. Идеальные названия: «0.bmp», «1. bmp» и т.д.

Для каждой папки с примерами необходимо иметь файл описания, в котором описаны используемые изображения. Стандартно их называют «Good.dat» и «Bad.dat». Файлы должны располагаться на одном уровне с папками.

Файлы описания для отрицательных и положительных объектов имеют разную структуру. Для файла отрицательных примеров это список относительных путей к изображениям:



Рисунок. 4.2 – Пример файла с описанием «Плохих» изображений.

Для файлов с положительными примерами запись сложнее. Кроме пути должно быть указанно положение рассматриваемого объекта и его размер. Необходимо помнить, что один кадр — один объект.



Рисунок. 4.3 – Пример файла с описанием «Хороших» изображений.

«Good \0.bmp» — адрес объекта относительно файла описания, «1» — количество положительных объектов на изображении, «0 0 414 148» — координаты прямоугольника на изображении, в котором находится объект. Если объектов несколько, то запись приобретает вид: «Good \0.bmp 2 100 200 50 50 300 300 25 25».

Удобнее всего, когда каждый объект представляет собой отдельный кадр, при этом координаты объекта равны размеру кадра.

# 4.1.2 Обработка «Хороших» изображений

Для начала необходимо открыть командную строку и перейти в директорию с файлом описания «Good.dat».

Все «Хорошие» изображения необходимо привести к единому формату. Делается это с помощью программы «**opencv\_createsamples»**, которую нужно запустить через консоль:



Рисунок. 4.4 – Команда для создания пачки положительных изображений.

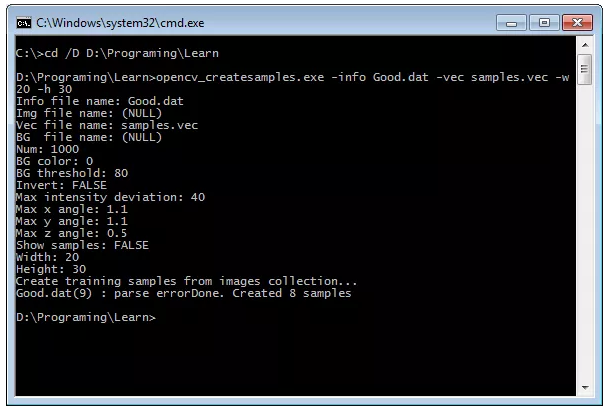


Рисунок. 4.5 – Ввод команда в консоль.

**Good.dat** – файл описания положительных изображений. Указывается либо полный адрес, либо относительно программы opencv\_createsamples.exe).

**-vec samples.vec** – файл, в который будет сохранена приведённая к общему формату база положительных изображений. Адрес должен быть указан относительно программы opencv\_createsamples.exe (допустим полный путь в системе).

**-w** 20 **-h** 30 — размер шаблона. Он должен приблизительно отражать пропорции искомого объекта. Идеально ставить его таким, чтобы человек сам мог отличить изображённый объект, но не больше того. Чем больше шаблон, тем дольше обучение.

Результатом работы программы является файл samples.vec, в котором будут расположены все положительные изображения в формате, близком к bmp и с размером w\*h. Если всё прошло успешно, должен появиться файл samples.vec.

# 4.1.3 Создание основного каскада

Для подсчёта итогового каскада используется программа «**opencv\_traincascade.exe**», расположенная в той же папке, что и «opencv\_createsamples.exe». Минус этой программы в ее долгой работе.

Команда для создания каскада:

«opencv\_traincascade.exe -data dataplates -vec samples.vec -bg Bad.dat -numStages 20 -minhitrate 0.99 -maxFalseAlarmRate 0.4 -numPos 800 -numNeg 1000 -w 20 -h 30 -mode ALL -precalcValBufSize 512 -precalcIdxBufSize 512»

1. **dataplates.xml** (имя выходного обученного классификатора);
2. **samples.vec** (файл с позитивными изображениями, созданный с помощью модуля opencv\_createsamples.exe);
3. **Bad.dat** (файл со списком негативных изображений);
4. **numPos** 800 (количество позитивных изображений, умноженное на определенный коэффициент меньше единицы (1000 \* 0,8));
5. **numNeg** 1000 (количество негативных изображений);
6. **numStages** 20 (максимальное количество уровней в обученном каскадном классификаторе);
7. **precalcValBufSize** 512 (размер буфера значений предварительно вычисленных признаков в мегабайтах);
8. **precalcIdxBufSize** 512 (размер буфера индексов предварительно вычисленных признаков в мегабайтах);
9. **w** 20 **h** 30 (ширина и высота изображений обучающей выборки);
10. **minHitRate** 0.99 (минимальная доля правильно классифицированных позитивных объектов обучающей выборки для каждого уровня каскада);
11. **maxFalseAlarmRate** 0.4 (максимально доля ложно-позитивных срабатываний для негативных объектов обучающей выборки для каждого уровня каскада).

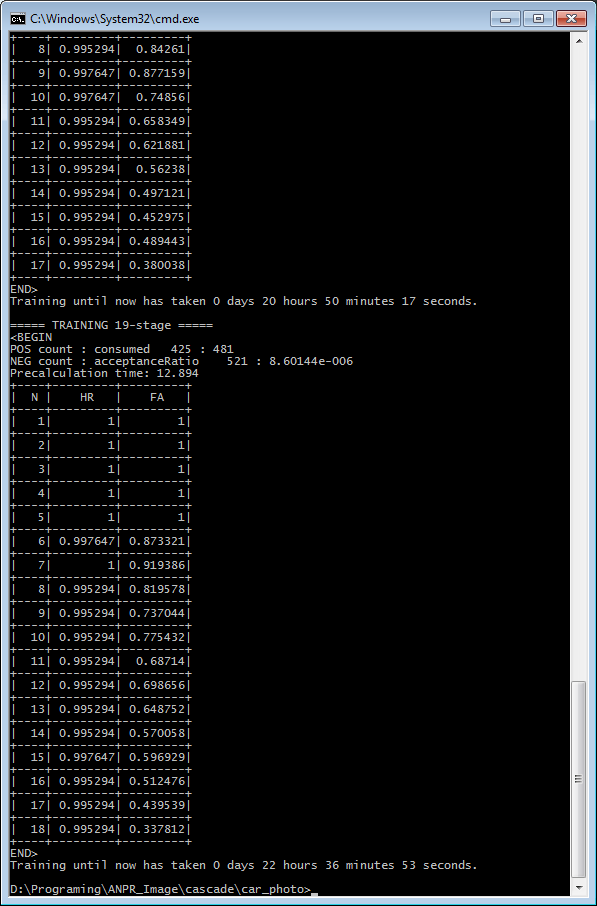


Рисунок. 4.6 – Обучение каскада Хаара.

После того, как алгоритм закончит работу, должна появиться папка «**haarcascade**» и файл «**cascade.xml**» в этой папке, это и будет созданный каскад. [8]

# 4.2 Тестирование каскада Хаара.

Для тестирование обученного каскада Хаара будет разработано приложение на С# использую библиотеку OpenCV. Распознание объектов в видео потоке производилась с помощью метода «DetectMultiScale». Данные метод имеет такие входные параметры как:

* Image<Gray, byte> Image – Иходное изображение;
* Double ScaleFactor – Коэффициент увеличения изображения;
* Int MinNeighbors – Группировка предварительно обнаруженных событий;
* Size MinSize – Минимальный размер объекта;
* Size MaxSize – Максимальный размер объекта.

И метода «CascadeClassifier», который предварительно был инициализирован готовым каскадом Хаара.

# 5 Проект программы

Для формального описания системы был выбран унифицированный язык моделирования (UML). На основе UML построена диаграмма классов. Рисунок 5.1

Достаточно важным моментом при проектировании программы, является внешний вид программы. Внешний вид программы должен максимально отражать функциональные особенности программы, чтобы пользователь за минимальное время мог освоить программу и воспользоваться ей.

# 5.1 Диаграмма классов

# 

Рисунок. 5.1 – Диаграмма классов.

# 5.2 Макеты пользовательского интерфейса

Программа состоит из диалогового окна, на котором находится кнопка “Запустить” для запуска программы, и поля PictureBox для показа обработанного изображения, полученного с веб-камеры. Рисунок 5.2

Так же программа имеет ряд настроек:

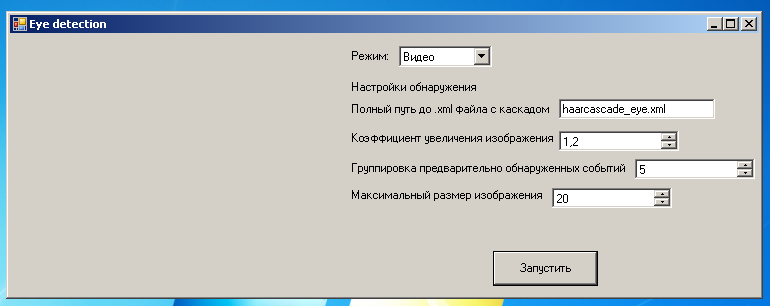
1. Режим работы:
2. “Видео” – используется для распознавания глаз человека из видеопотока;
3. “Фото” – используется для распознавания глаз человека с загруженного изображения;
4. Полный путь до .xml файла с каскадом. Значение по умолчанию “haarcascade\_eye.xml”. Используется для указания пути до необходимого файла с каскадом. Возможно указать полный путь вида “C:\Users\Sergey\Desktop \trunk\DefinitioEyes\bin\Debug\ haarcascade\_eye.xml”. Однако если файл каскада находится в каталоге с запускаемым файлом можно указывать только имя необходимого каскада “haarcascade\_eye.xml”.
5. Коэффициент увеличения изображения. Показывает во сколько раз необходимо увеличивать исходное изображение перед применением каскада.
6. Группировка предварительно обнаруженных событий. Показывает скольким критериям должен соответствовать участок изображения, чтобы обнаружить на нем необходимый объект. Чем меньше данный параметр, тем больше будет появляется ложных обнаружений.
7. Максимальный размер изображения. Указывает максимальный размер объекта, который необходимо искать. 

Рисунок. 5.2 – Диалоговое окно при старте.

# 5.3 Обнаружение глаз человека при работе с видеопотоком

Так как для обучения каскада Хаара было взято всего 1000 положительных и 1000 негативных изображений, то каскад не всегда корректно справляется с поставленной задачей. При хорошем освещении и качественной веб камере обнаружение человеческих глаз происходит примерно в 90-95% случаях. Рисунок 5.3-5.4

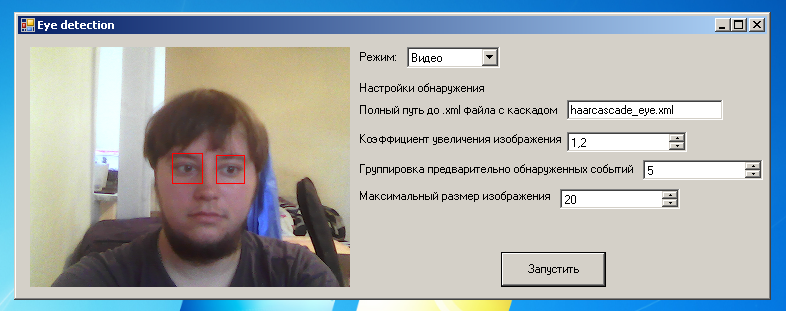


Рисунок. 5.3 – Обнаружение человеческих глаз при идеальных условиях.

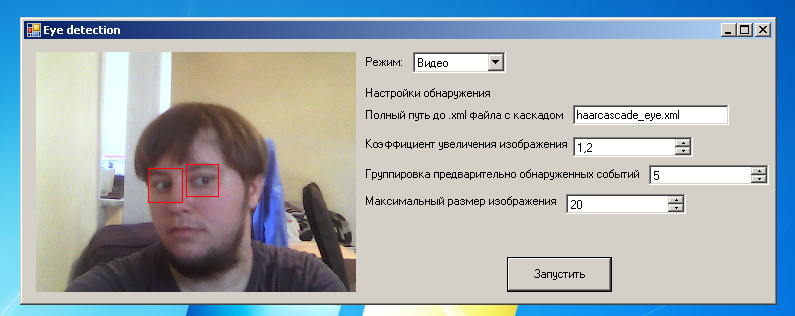


Рисунок. 5.4 – Обнаружение человеческих глаз при повороте головы.

Однако если уменьшить освещение или увеличивать поворот головы, то качество распознавания глаз резко падает. И программа не может обнаружить человеческие глаза. Рисунок 5.4-5.5

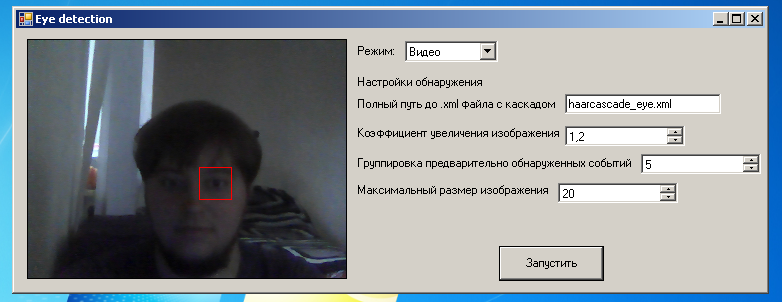


Рисунок. 5.5 – Обнаружение человеческих глаз при плохом освещении.

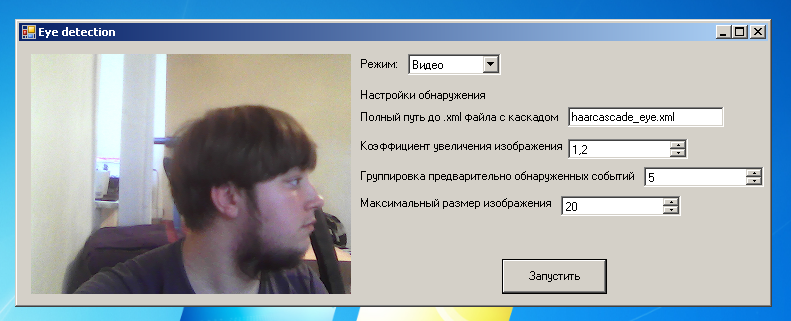


Рисунок. 5.6 – Обнаружение человеческих глаз при большом повороте головы.

Кроме того, программа была протестирована на предмет распознавания глаз представителей кошачьего семейства. Рисунок. 5.7

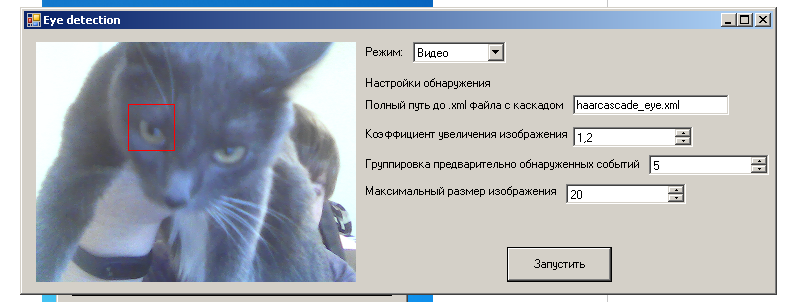


Рисунок. 5.7 – Распознавания глаз кота.

Программа плохо распознает кошачьи глаза. Скорее всего это связанно с разным строением глаз человека и животного, и разного преобладающего цвета в них.

# 5.4 Обнаружение глаз человека при работе с фотографиями

Так же программа балы протестирована на распознавания человеческих глаз с различных фотографий. Рисунок 5.8-5.9

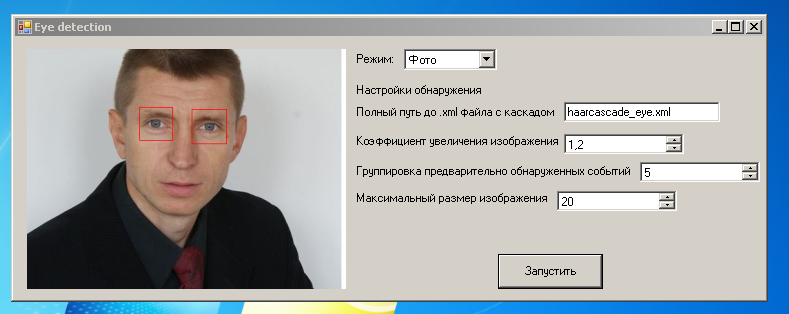


Рисунок. 5.8 – Обнаружение человеческих глаз по фотографии.

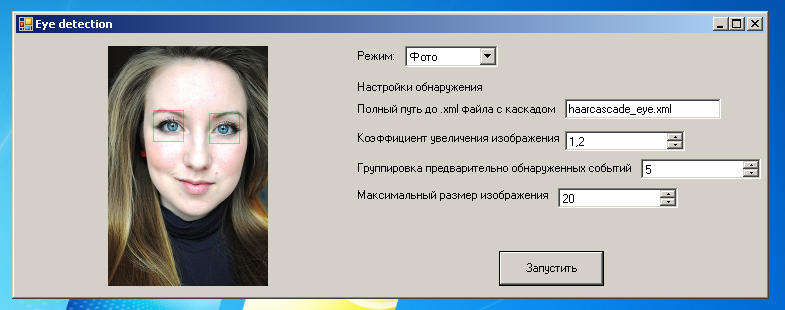


Рисунок. 5.9 – Обнаружение человеческих глаз по фотографии.

# 6 Работа с программой CutIimageArea

# 6.1 Формирование файла Good.dat

Для формирования файла Good.dat с помощь программы CutIimageArea необходимо выполнить следующие действия:

1. Нажать на кнопку “Загрузка изображений” и выбрать папку, в которой находятся необходимые изображения для обработки. В программе уже имеются стандартные изображения для распознавание человеческих глаз. Они находятся в “CutIimageArea\Data\Face”.
2. После загрузки изображений в выбранной папке программа создаст папку “Good”, куда будет помещать вырезанные фрагменты из загруженных фотографий, и создаст файл “Good.dat”, в который будет записывать все необходимые данные.
3. Далее необходимо выбрать нужную область на изображении с помощью левой кнопки мыши и нажать на кнопку “Сохранить выбранную область”. На одном изображении можно сохранять сколько угодно таких областей, но по одной области за одно сохранение.
4. Когда работа с текущим изображением завершена, нужно нажать на кнопку “Следующее изображение” и перейти к следующему загруженному изображению.
5. С ним также необходимо выполнить действия, описанные в третьем пункте.
6. После завершения обработки всех загруженных изображений можно перейти к формированию каскада.

# 6.2Приведение всех положительных изображений к единому виду

Форма для приведения всех положительных изображений к единому виду представлена на изображении Рисунке 6.1.

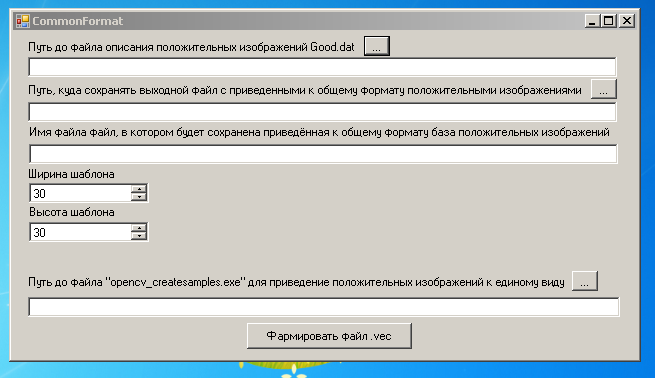


Рисунок. 6.1 –Приведения всех положительных изображений к единому виду.

Для того, чтобы успешно сформировать файл “Good.vec”, необходимо заполнить следующие поля:

1. Путь до файла описания “Good.dat”. В этом поле необходимо указать полный путь до файла “Good.dat”. В адресе файла не должно находиться русских символов. Пример пути: “ CutIimageArea\Data\Face\Good”.
2. Путь, куда необходимо сохранить сформированный файл “.vec”. При указании пути в нем так же не должно находиться русских символов. Пример пути: “C:\Users\Sergey\Desktop”.
3. Имя, которое будет присвоено файлу “.vec”. Пример: “Good”.
4. Ширина и высота зависит от пропорций объекта, для которого формировался файл “Good.dat”.
5. Путь до файла “opencv\_createsamples.exe”. Данная программа будет осуществлять преобразование файла из “Good.dat” в “Good.vec”. В проект уже добавлена данная программа. Поэтому необходимо указать ее в данном поле. Пример пути: “CutIimageArea\Data\OpenCV\opencv\_createsamples.exe”.
6. После этого необходимо нажать на кнопку “Формировать файл .vec” и программа “opencv\_createsamples.exe” сформирует данный файл и сохранит его по пути, указанному в пункте 2, с именем, указанном в пункте 3.

# 6.3 Форма для обучения каскада Хаара.

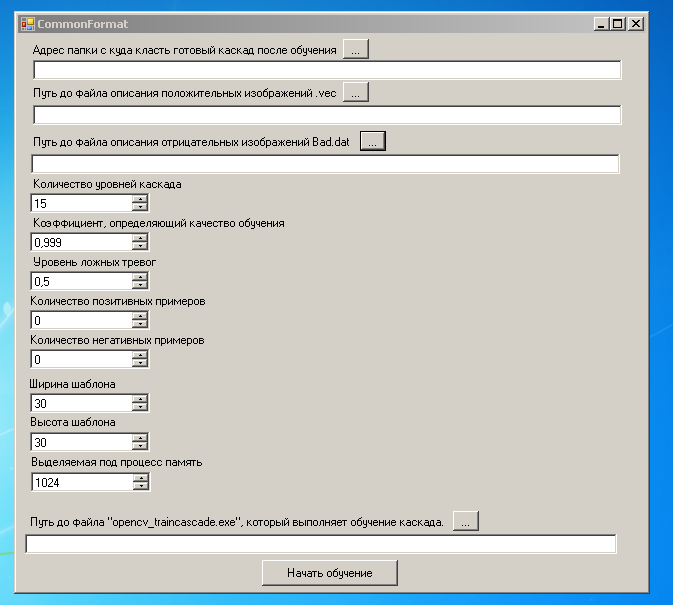
Форма для обучения каскада Хаара представлена на изображении Рисунке 6.2. 

Рисунок. 6.2 –Форма для обучения каскада Хаара.

Для обучения каскада Хаара необходимо заполнить следующие поля данной формы.

1. Адрес папки, куда сохранять обученный каскад. В данном адресе не должно содержаться русских символов. Пример адреса: “C:\Users\Sergey\Desktop”
2. Путь до файла описания положительных изображений “.vec”. В адресе пути не должно находиться русских символов. Данный файл формировался в пункте 6.2 с помощью модуля opencv\_createsamples.exe.
3. Путь до файла со списком негативных изображений. В адресе пути так же не должно находиться русских символов. Пример данного файла находится в проекте “CutIimageArea\Data\Bad”.
4. Количество уровней каскада, которые программа будет обучать. Чем больше уровней, тем точнее распознавание и дольше обучение. Допустимое количество от 16 до 25.
5. Коэффициент, определяющий качество обучения. По сути, это процент “правильных” обнаружений. Если установлено .999, то по исходной выборке будет не более 1- 0.999 =0.1% пропусков целей. Чем выше коэффициент, тем выше уровень ложных тревог. В самом простом варианте, если выборка хорошая, можно ставить 0.99-0.999. Если выборка плохая (объектов мало, они смешиваются с фоном) — то следует опускать коэффициент.
6. Уровень ложных тревог – максимальная доля ложно-позитивных срабатываний для негативных объектов обучающей выборки для каждого уровня каскада.
7. Количество позитивных изображений, умноженное на определенный коэффициент меньше единицы (1000 \* 0,8);
8. Количество негативных изображений;
9. Ширина и высота изображений обучающей выборки;
10. Размер буфера значений и размер буфера индексов, предварительно вычисленных признаков, в мегабайтах.
11. Путь до файла “opencv\_traincascade.exe”. В проект уже добавлена данная программа. Поэтому необходимо указать ее в данном поле. Пример пути: “CutIimageArea\Data\OpenCV\ opencv\_traincascade.exe”.
12. После заполнения всех полей необходимо нажать на кнопку “Начать обучение”. Запустится программа “opencv\_traincascade.exe”, по завершению обучения данная программа сформирует “.xml” файл с каскадом по пути указанном в пункте 1. Если в папке, указанной в пункте 1 уже имеются файлы с прошлых обучений, то обучение каскада не будет запущенно.

# Заключение

В ходе проделанной работы была подробно изучена тема «Компьютерное зрение». Была проанализирована библиотека компьютерного зрения OpenCV. Было проведённое обучение алгоритма на основе примитивов Хаара. А также была написана программа для распознавания человеческих глаз с фотографий и видеопотока. Она была протестирована с использованием веб-камеры и загрузке различных фотографий на человеке при разных условиях и животном, в данном случае, коте.

# Список использованной литературы

1. Branislav Kisacanin. Embedded Computer Vision [Текст] / Branislav Kisacanin. – Springer, Embedded, 2008. – 284 с.
2. Машинное (техническое, компьютерное) зрение: // материалы сайта [Электронный ресурс]. – URL: http://robodem.ru/machinevision (дата обращения 01.11.2016)
3. OpenCV шаг за шагом. // материалы сайта [Электронный ресурс]. – URL: http://robocraft.ru/page/opencv/ (дата обращения 14.11.2016)
4. Генетический алгоритм. // материалы сайта [Электронный ресурс]. – URL:http://www.machinelearning.ru/wiki/index.php?title=Генетический\_алгоритм (дата обращения 24.11.2016)
5. Метод Виолы-Джонса. // материалы сайта [Электронный ресурс]. – URL: http://www.intuit.ru/studies/courses/10619/1103/lecture/18229?page=1 (дата обращения 20.11.2016)
6. OpenCV шаг за шагом. Интегральное изображение. // материалы сайта [Электронный ресурс]. – URL: http://robocraft.ru/blog/computervision/536.html (дата обращения 20.12.2016)
7. Обучение OpenCV каскада Хаара. // материалы сайта [Электронный ресурс]. – URL: https://habrahabr.ru/post/208092/ (дата обращения 20.12.2016)
8. Обучение каскадного классификатора в OpenCV. // материалы сайта [Электронный ресурс]. – URL: http://techcave.ru/posts/55-obuchenie-kaskadnogo-klassifikatora-v-opencv-opencv-traincascade-opencv-createsamples.html (дата обращения 25.12.2016)