НАУЧНО-ОБРАЗОВАТЕЛЬНОЕ СОРЕВНОВАНИЕ «ШАГ В БУДУЩЕЕ, МОСКВА»

1255

регистрационный номер

ИУ

название факультета

«Системы обработки информации и управления»

название кафедры

«Обнаружение лица на изображении методом Виолы-Джонса»

название работы

Автор:

Барышников Михаил Игоревич

фамилия, имя, отчество

Школа № 1580 при МГТУ имени Н.Э.Баумана

Класс 11 «С»

наименование учебного заведения, класс

Научный руководитель: –––

Москва - 2019

Оглавление

АННОТАЦИЯ *3*

ВВЕДЕНИЕ *4*

1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ *5*

1.1Описание алгоритма **5**

1.2 Схема распознавания **5**

1.3 Признаки класса **6**

1.4 Схема обучения **9**

1.5 Интегральное представление изображений **10**

1.6 Обучение **12**

2. ПРАКТИЧЕСКАЯ ЧАСТЬ *14*

2.1 Формирование набора изображений для обучения каскада классификаторов **15**

2.2 Тренировка каскада классификатора **16**

2.3 Проверка точности работы классификатора **18**

3. Выводы *19*

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ *20*

ПРИЛОЖЕНИЕ 1 ..... *21*

ПРИЛОЖЕНИЕ 2 ..... *23*

ПРИЛОЖЕНИЕ 3 ..... *24*

АННОТАЦИЯ

Данная научная работа посвящена созданию и реализации на языке C++ алгоритма Виолы-Джонса, позволяющего находить на изображении лицо человека. Распознавание лиц является одной из самых изученных задач в таких областях, как цифровая обработка изображений, компьютерное зрение, биометрия, организация видеоконференций и создание интеллектуальных систем безопасности и контроля доступа. Алгоритм Виолы-Джонса это метод, основанный на преобразовании изображения в интегральную форму, что позволяет быстро и с постоянной скоростью вычислять необходимые объекты. Целью данной работы является написание программы, способной с минимально возможным количеством ложных срабатываний определять координаты лица человека на изображении. Используются признаки, описанные Виолой и Джонсом, их сравнение с признаками Хаара, приводятся достоинства и недостатки каждого из них. При угле наклона больше 30° вероятность обнаружения лица резко падает. Указанная особенность метода не позволяет в стандартной реализации детектировать лицо человека, повернутое под произвольным углом, что в значительной мере затрудняет или делает невозможным использование алгоритма в современных производственных системах с учетом их растущих потребностей. Описываются основные принципы построения алгоритма, такие как преобразование изображений в интегральную форму, методы каскадирования, позволяющие значительно увеличить скорость работы алгоритма и применение методов машинного обучения, необходимых для улучшения работы классификатора. Результаты, полученные в ходе эксперимента, были проанализированы, получен средний процент ошибки классификатора и зависимость времени работы алгоритма от размера изображения. Рассматриваются дальнейшие перспективы развития данной работы.

Ключевые слова:
Распознавание лиц, Метод Виолы-Джонса, интегральное представление, машинное обучение.

ВВЕДЕНИЕ

Задача идентификации и распознавания лиц – это одна из первых практических задач, которая стимулировала становление и развитие теории распознавания и идентификации объектов. Интерес к процедурам, лежащим в основе процесса узнавания и распознавания лиц, всегда был значительным, особенно в связи с возрастающими практическими потребностями: охранные системы, верификация, криминалистическая экспертиза, телеконференции и т.д. Несмотря очевидность того факта, что человек хорошо идентифицирует лица людей, совсем не очевидно, как научить ЭВМ проводить эту процедуру, в том числе как декодировать и хранить цифровые изображения лиц. Еще менее ясными являются оценки схожести лиц, включая их комплексную обработку. Именно поэтому целью моей научной работы является разработка и написание алгоритма, позволяющего обнаружать лица на изображении. За основу программы будет взят метод, предложенный Виолой и Джонсоном в 2001 году.

1.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1Описание алгоритма

В 2001 году Виола и Джон предложили алгоритм для распознавания лиц, который стал прорывом в области распознавания лиц. Метод использует технологию скользящего окна. То есть рамка, размером, меньшим, чем исходное изображение, двигается с некоторым шагом по изображению, и с помощью каскада слабых классификаторов определяет, есть ли в рассматриваемом окне лицо. Метод скользящего окна эффективно используется в различных задачах компьютерного зрения и распознавания объектов.

Метод состоит из 2-х под алгоритмов: алгоритм обучения и алгоритм распознавания. На практике скорость работы алгоритма обучения не важна. Крайне важна скорость работы алгоритма распознавания. По введенной ранее классификации можно отнести к структурным, статистическим и нейронным методам.

Метод имеет следующие преимущества:

- возможно обнаружение более одного лица на изображении;
- использование простых классификаторов показывает хорошую скорость и позволяет использовать этот метод в видеопотоке.

Однако метод сложно обучаем, так как для обучения требуется большое количество тестовых данных и предполагает большое время обучения, которое измеряется днями.

Изначально алгоритм был предложен для распознавания только лиц, но его можно использовать для распознавания других объектов. Одним из вкладов Виолы и Джонса было применение таблицы сумм, которую они назвали интегральным изображением, детальное описание которого будет дано ниже.

1.2 Схема распознавания

Алгоритм распознавания по методу Виолы-Джонса основан на "суммировании" интенсивности(яркости) пикселов находящимися либо под «белой» либо под «чёрной» частью примитива Хаара (с определенными весовыми коэффициентами для «белого» и «чёрного» слоёв соответственно) под скользящим [по растру] окном. Распознавание в этом методе осуществляется по "прецедентам". С помощью "обучающей выборки" строится набор "сильных классификаторов", каждый из которых для квадратного окна говорит: "предположительно, в окне - лицо", или -

"определенно, не лицо". Таким образом, для того, чтобы алгоритм признал картинку в окне за лицо, необходимо, чтобы все "сильные классификаторы" (stages) ответили: "да, лицо предположительно есть". Если хотя бы один из них отверг окно (сказал, что "лица определенно нет"), то алгоритм сразу же отвергает данное окно, другие "сильные классификаторы" не использует, и переходит к следующему окну.

1.3 Признаки класса

Признаки Хаара — признаки цифрового изображения, используемые в распознавании образов. Своим название они обязаны интуитивным сходством свейвлетами Хаара. Признаки Хаара использовались в первом детекторе лиц, работающем в реальном времени.

Исторически сложилось так, что алгоритмы, работающие только с интенсивностью изображения (например значение RGB в каждом пикселе), имеют большую вычислительную сложность. В работе Папагеоргиу, была рассмотрена работа с множеством признаков, основанных на вейвлетах Хаара. Виола и Джонс адаптировали идею использования вейвлетов Хаара и разработали то, что было названо признаками Хаара. Признак Хаара состоит из смежных прямоугольных областей. Они позиционируются на изображении, далее суммируются интенсивности пикселей в областях, после чего вычисляется разность между суммами. Эта разность и будет значением определенного признака, определенного размера, определенным образом с позиционированного на изображении.

Для примера рассмотрим базу данных с человеческими лицами. Общим для всех изображений является то, что 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 окно установленного размера движется по изображению, и для каждой области изображения, над которой проходит окно, рассчитывается признак Хаара. Наличие или отсутствие предмета в окне определяется разницей между значением признака и обучаемым порогом. Поскольку признаки Хаара мало подходят для обучения или классификации (качество немного выше чем у случайной нормально распределенной величины), для описания объекта с достаточной точностью необходимо большее число признаков. Поэтому в методе Виолы—Джонса признаки Хаара организованы в каскадный классификатор.

Ключевой особенностью признаков Хаара является наибольшая, по сравнению с остальными признаками, скорость. При использовании интегрального представления изображения, признаки Хаара могут вычисляться за постоянное время (примерно 60 процессорных инструкций на признак из двух областей).

В стандартном методе Виолы – Джонса используются прямоугольные признаки, изображенные на рисунке 1, они называются примитивами Хаара.

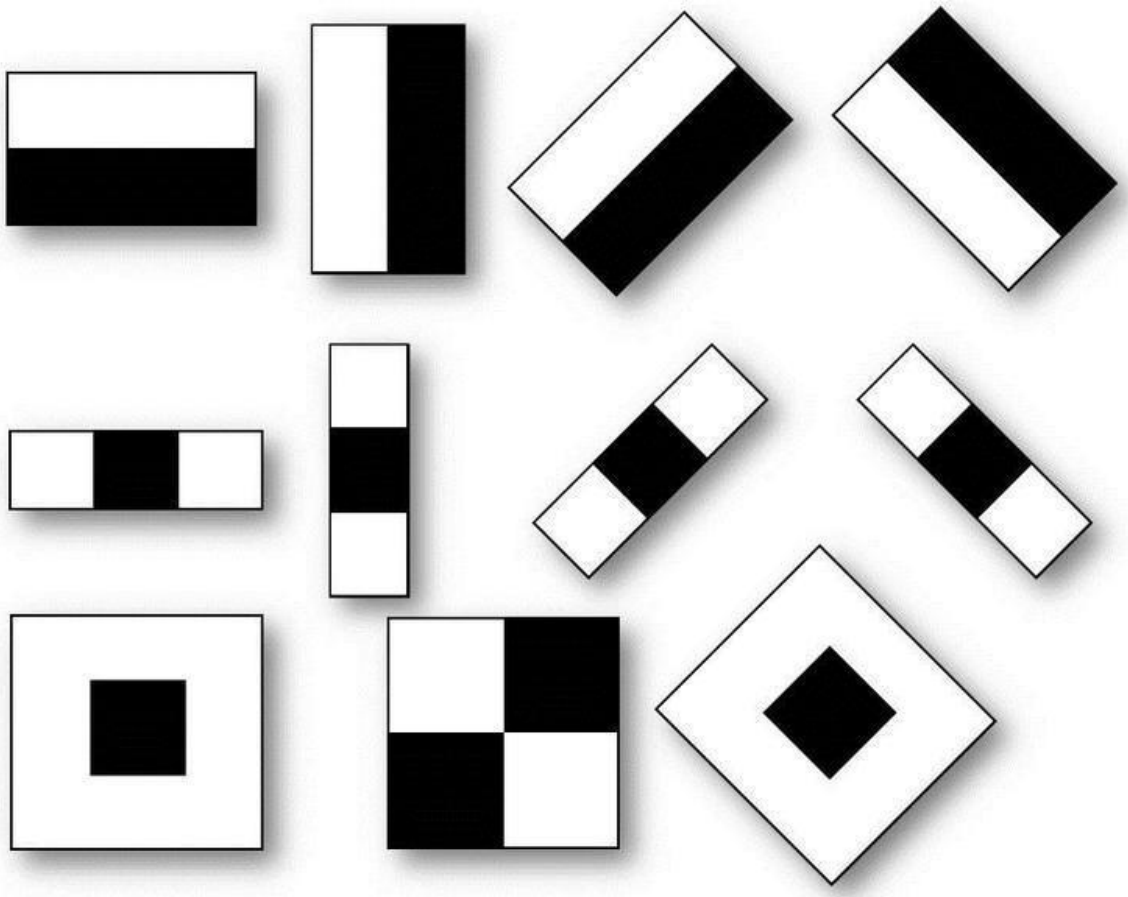

Рис. 1 – Признаки Хаара.

В расширенном методе Виолы – Джонса, использующемся в библиотеке OpenCV используются дополнительные признаки, показанные на рисунке 2:

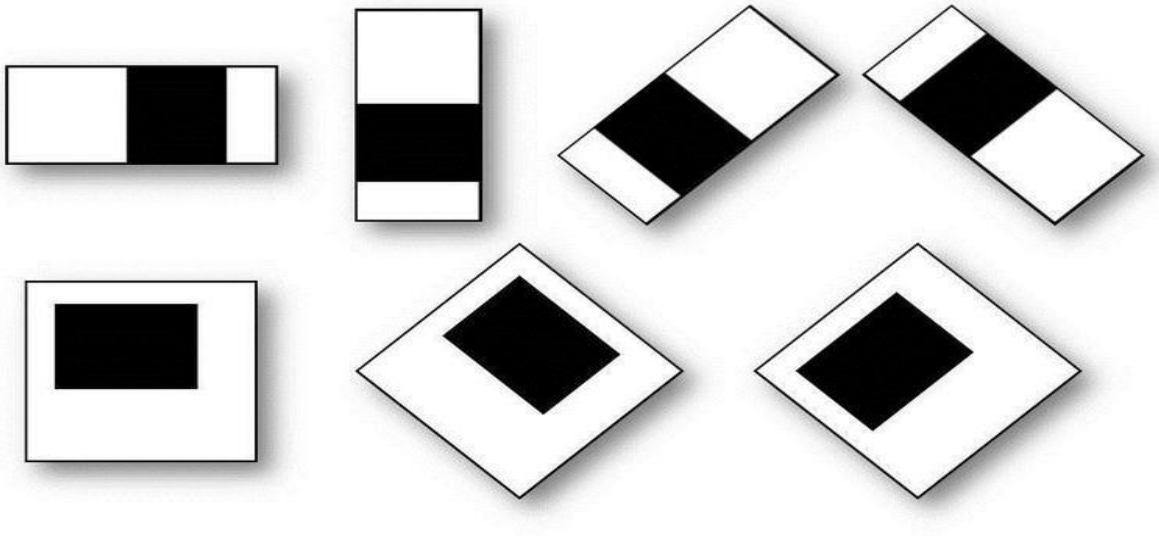

Рис. 2 – Дополнительные признаки Хаара

Вычисляемым значением такого признака будет:

$$F = X\text{-}Y$$

где *X*– сумма значений яркостей точек закрываемых светлой частью признака,

Y– сумма значений яркостей точек закрываемых темной частью признака.

Для их вычисления используется понятие интегрального изображения, рассмотренное выше. Признаки Хаара дают точечное значение перепада яркости по оси *X* и *Y* соответственно.

Алгоритм сканирования окна с признаками выглядит так:

есть исследуемое изображение, выбрано окно сканирования, выбраны используемые признаки;

далее окно сканирования начинает последовательно двигаться по изображению с шагом в 1 ячейку окна (допустим, размер самого окна есть 24*24 ячейки);

при сканировании изображения в каждом окне вычисляется приблизительно 200 000 вариантов расположения признаков, за счет изменения масштаба признаков и их положения в окне сканирования;

сканирование производится последовательно для различных масштабов;

масштабируется не само изображение, а сканирующее окно (изменяется размер ячейки);

все найденные признаки попадают к классификатору, который «выносит вердикт».

В процессе поиска вычислять все признаки на маломощных настольных ПК просто нереально. Следовательно, классификатор должен реагировать только на определенное, нужное подмножество всех признаков. Совершенно логично, что надо обучить классификатор нахождению лиц по данному определенному подмножеству. Это можно сделать, обучая вычислительную машину автоматически.

Обучение машины — это процесс получения модулем новых знаний.

Данный процесс входит в концепцию и технологию под названием Data mining (извлечение информации и интеллектуальный анализ данных), куда входят помимо Машинного обучения такие дисциплины, как Теория баз данных, Искусственный интеллект, Алгоритмизация, Распознавание образов и прочие. Машинное обучение в методе Виолы-Джонса решает такую задачу как классификация.

1.4 Схема обучения

Обобщенная схема алгоритма обучения выглядит следующим образом. Имеется тестовая выборка изображений. Размер тестовой выборки около 10 000 изображений. На рисунке показан пример обучающих изображений лиц. Алгоритм обучения работает с изображениями в оттенках серого.

Обобщённая схема алгоритма обучения показана на рисунке 3.

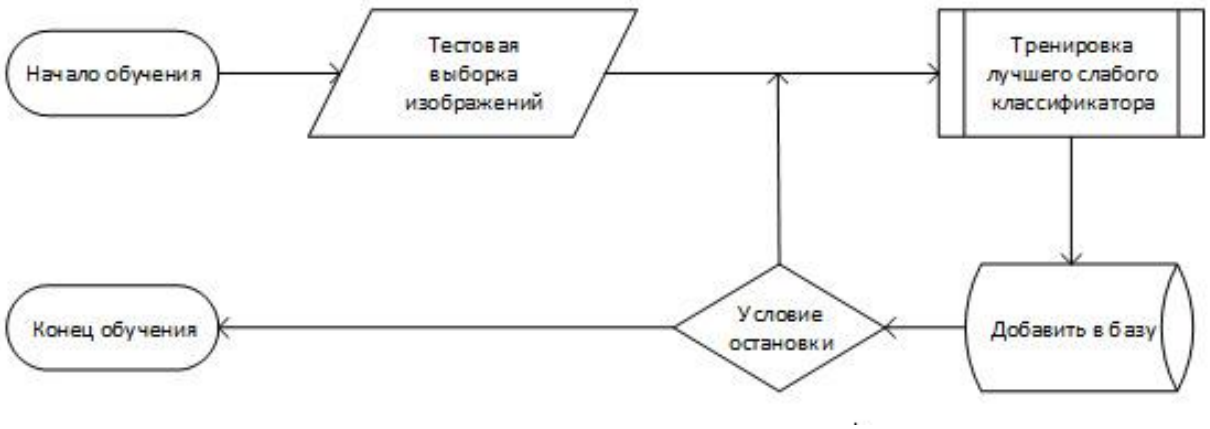

Рис. 3 Обобщенная схема алгоритма

Обучение алгоритма Виола-Джонса – это обучение алгоритма с учителем. Для него возможно такая проблема как переобучение. Показано, что AdaBoost может использоваться для различных проблем, в том числе к теории игр, прогнозировании. В данной работе условие остановки является достижение заранее заданного количества слабых классификаторов в базе.

1.5 Интегральное представление изображений

Для того, чтобы производить какие-либо действия с данными, используется интегральное представление изображений в методе Виолы-Джонса. Одной из полезнейших особенностей интегрального представления является возможность очень быстро вычислить сумму пикселей произвольного прямоугольника (или любой другой фигуры, которую можно аппроксимировать несколькими прямоугольниками).

Интегральное представление изображения – это матрица, совпадающая по размерам с исходным изображением. В каждом элементе ее хранится сумма интенсивностей всех пикселей, находящихся левее и выше данного элемента. Элементы матрицы рассчитываются по формуле (1.1):

$$L(x,y)=\sum_{i=0,\,j=0}^{i\le x,\,j\le y} ¿I(i,j) \qquad (1.1)$$

где *I(i,j)* — яркость пикселя исходного изображения.

Каждый элемент матрицы *L(x,xy)* представляет собой сумму пикселей в прямоугольнике от *(0,0)* до *(x,y)*, т.е. значение каждого пикселя *(x,y)* равно сумме значений всех пикселей левее и выше данного пикселя *(x,y)*. Расчет матрицы занимает линейное время, пропорциональное числу пикселей в изображении, поэтому интегральное изображение просчитывается за один проход.

Расчет матрицы возможен по формуле (1.2):

$$L(x,y) = I(x,y) – L(x-1,y-1) + L(x,y-1) + L(x-1,y) \qquad (1.2)$$

По такой интегральной матрице можно очень быстро вычислить сумму пикселей произвольного прямоугольника (см. рисунок 1), произвольной площади. Пусть в прямоугольнике ABCD есть интересующий нас объект D:

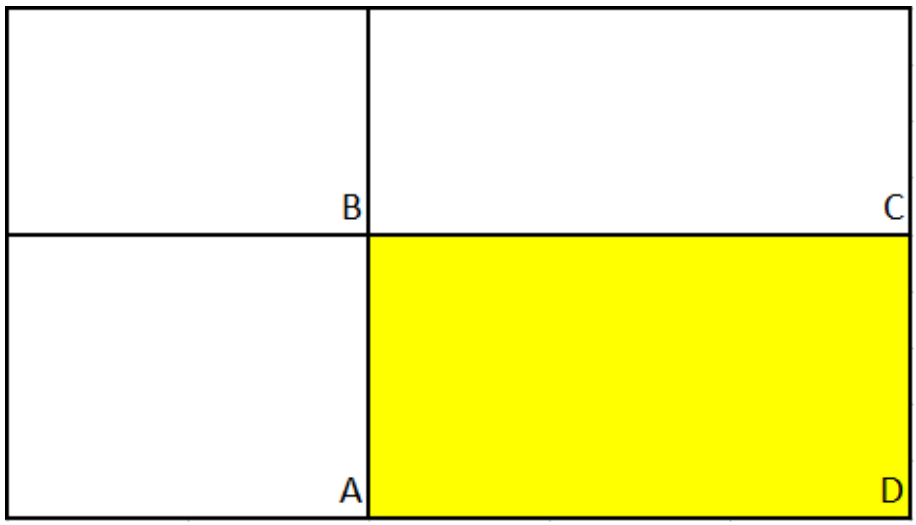

Рис. 4 – Прямоугольная область изображения

Из рисунка 1 понятно, что сумму внутри прямоугольника можно выразить через суммы и разности смежных прямоугольников по формуле (1.4):

$$S(ABCD) = L(A) + L(C) — L(B) — L(D) \quad (1.4)$$

Примерный просчет показан на рисунке 5:

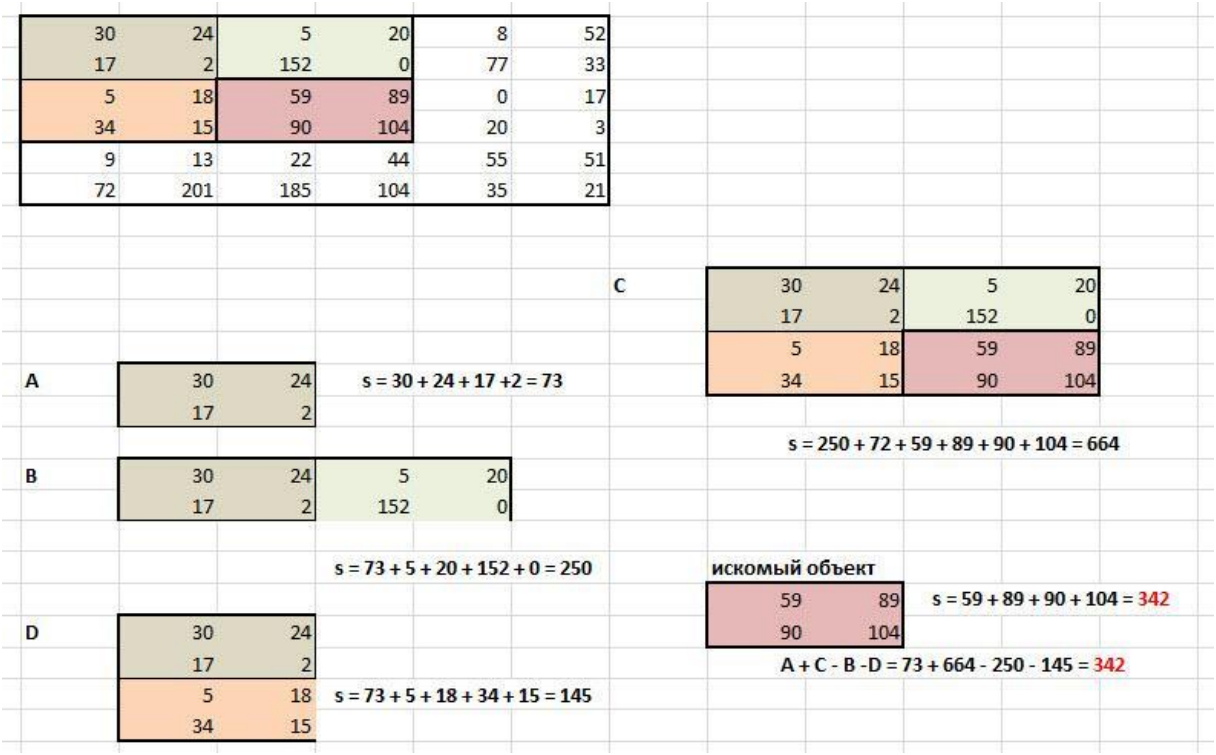

Рис. 5 – Примерный просчет пикселей произвольного прямоугольника

Признак — отображение f: X => Df, где Df — множество допустимых значений признака. Если заданы признаки f1,…,fn, то вектор признаков x = (f1(x),…,fn(x)), называется признаковым описанием объекта x∈X.

Признаковые описания допустимо отождествлять с самими объектами. При этом множество X = Df1* …* Dfn называют признаковым пространством. Признаки делятся на следующие типы в зависимости от множества Df:
− бинарный признак, Df = {0,1};
− номинальный признак: Df — конечное множество;
− порядковый признак: Df — конечное упорядоченное множество;
− количественный признак: Df — множество действительных чисел.
Естественно, бывают прикладные задачи с разнотипными признаками, для их решения подходят далеко не все методы.

1.6 Обучение

Работа с каскадным классификатором включает два этапа: обучение и детектирование. Детектирование вы можете посмотреть, к примеру, на обнаружении лиц. А здесь описано как обучать классификатор.
В OpenCV есть два приложения для тренировки каскадов: opencv_haartraining и opencv_traincascade (новая версия написанная на C++). Что касается достоверности распознавания, то все зависит от обучающей выборки. Можно обучить LBP так, что оно будет приближаться к Хаара особенностям.
opencv_traincascade и opencv_haartraining хранят обученные классификаторы в различных форматах. Новый интерфейс обнаружения (CascadeClassifier класса в objdetect модуле) поддерживают оба формата. opencv_traincascade может сохранять обучение каскада в старом формате. Но opencv_traincascade и opencv_haartraining не может загрузить классификатор в другом формате для дальнейшего обучения после перерыва.
opencv_traincascade приложение может использовать TBB для многопоточности. Чтобы использовать его в режиме многоядерных процессоров OpenCV должно быть откомпилировано с TBB.
Другие утилиты, используемые для обучения:
1. opencv_createsamples используется для подготовки учебного набора положительных и тестовых образцов в формате, который поддерживается как opencv_haartraining так и opencv_traincascade приложениями. На выходе получается файл с расширением *.vec, это двоичный формат, который содержит изображение.
2. opencv_performance может быть использовано для оценки качества классификаторов, но только для обученных opencv_haartraining. Она использует коллекцию размеченных изображений, запускает классификатор и сообщает так называемую производительность, т.е. количество найденных объектов, количество пропущенных объектов, количество ложных срабатываний и другую информацию.
Для обучения необходимо собрать образы. Есть два типа образов: негативные и позитивные. Негативные образы соответствуют отсутствию объекта на изображении. Положительные образы соответствуют

изображением с обнаруженными объектами. Набор негативных образов должен быть подготовлен вручную, в то время как множество положительных образов создается с помощью утилиты opencv_createsamples.

Негативные образы должны быть взяты из произвольного изображения. Эти изображения не должны содержать обнаруженных объектов. Негативные образы перечислены в специальном файле. Это текстовый файл, в котором каждая строка содержит имя файла изображения (относительно каталога файла) с негативным образом. Этот файл должен быть создан вручную. Обратите внимание, что негативные примеры называют также фоновыми пробами или образцами фона изображения. Описанные изображения могут быть разных размеров. Но каждое изображение должно быть больше, чем размер окна обучения.

Позитивные образы создаются утилитой opencv_createsamples. Они могут быть созданы из одного изображения с объектом или из коллекции ранее размеченных изображений.

Позитивные примеры также могут быть получены из коллекции ранее размеченных изображений. Эта коллекция, которая описывается в текстовом файле, похожа на файл описания фона. Каждая строка этого файла соответствует изображению. Первым элементом линии является имя файла. За ней следуют количество экземпляров объекта. Следующие числа являются координатами объектов ограничивающих прямоугольников (x, y, ширина, высота).
Экземпляры объектов взяты из изображений. Затем они изменяются до размера целевых образов и сохраняются в выходной vec-файл. Никаких искажений не применяется, поэтому влияющие аргументы это: -w,-h,-show и -Num.
opencv_createsamples может быть использована для изучения образов, хранящихся в файле позитивных образов. Для того, чтобы сделать это только -vec,-w и -h параметры должны быть указаны.
Следующим шагом является подготовка классификатора. Как уже упоминалось выше opencv_traincascade или opencv_haartraining могут быть использованы для обучения каскада классификаторов, но только opencv_traincascade будет описан в дальнейшем.

После opencv_traincascade приложение завершит свою работу, обученный каскад будет сохранен в cascade.xml файл в папке, которая была принята в качестве данных параметров.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

Практическая реализация данного алгоритма велась на языке C++ в среде Microsoft Visual Studio. База данных лиц была взята из библиотеки MNIST. Написанный алгоритм выглядит следующим образом:

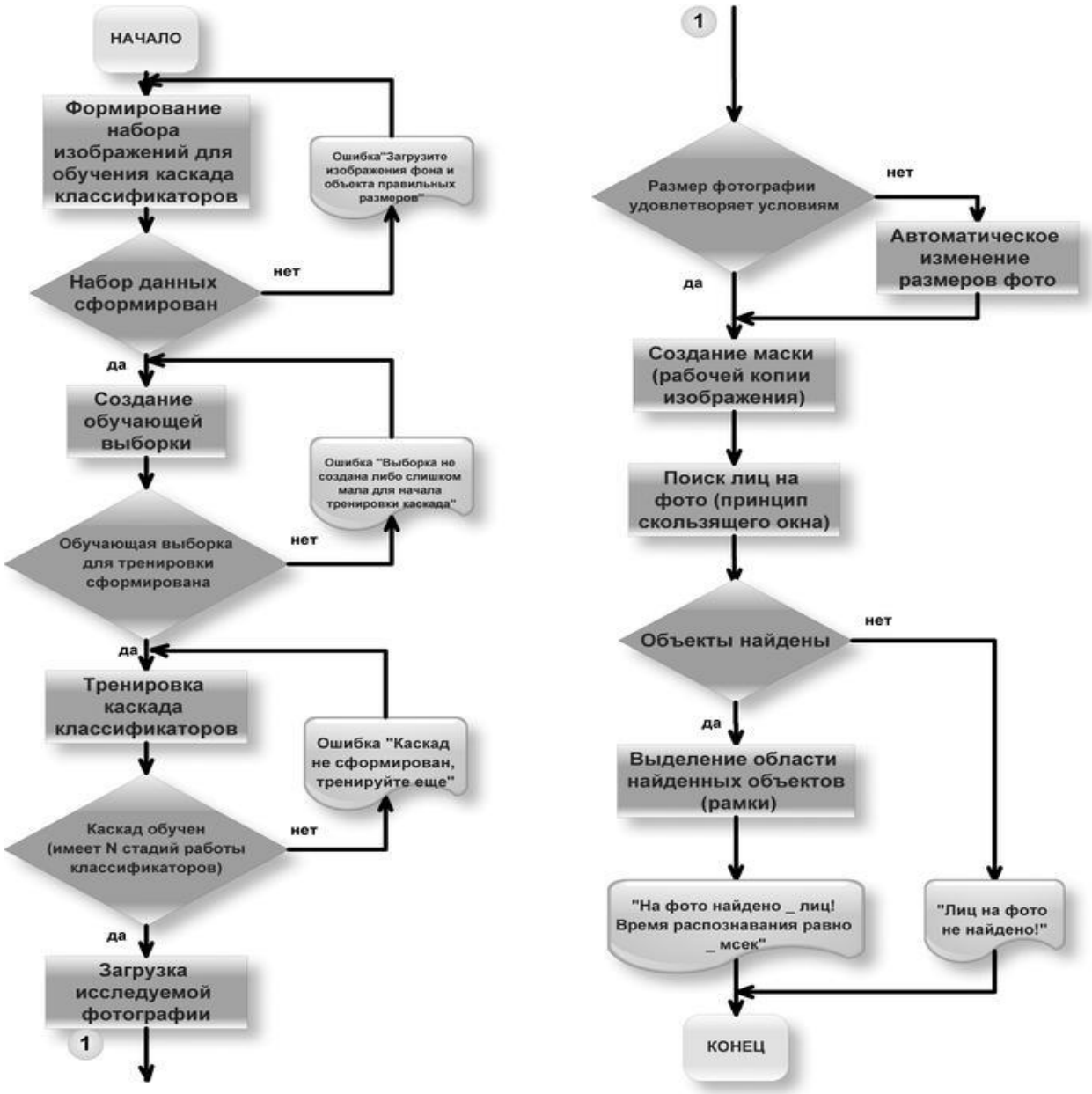

Рис. 6 Алгоритм работы программы

2.1 Формирование набора изображений для обучения каскада классификаторов

Для обучения каскада классификаторов методом обучения с учителем необходимо создать и сформировать некий набор изображений, содержащий в себе как «позитивные» примеры, то есть изображения лиц, так и «негативный», то есть набор случайных объектов. Именно его качество и размер в первую очередь влияет на точность классификатора. Найти уже готовую базу лиц с размеченными в нужном формате данными мне не удалось, поэтому я принял решение создать данный набор самостоятельно.

Я взял архив изображений людей примерно одного качества на одном из файлообменников. В архиве есть фотографии людей разных возрастов, полов и национальностей, что как нельзя лучше подходит для создания обучающей выборки. Затем передо мной встала задача вырезать со всех фотографий лица людей, назвать файлы именами «№.jpg» и создать текстовый файл, хранящий информацию о количестве лиц на фото, координате левого верхнего угла прямоугольника, описанного вокруг лица и его размере. Данный текстовый файл я назвал «Good.dat». Также, я создал файл «Bad.dat», в котором находились изображения любых случайных фрагментов фото.

Теперь необходимо привести все изображения к одному формату, соответствующему пропорциям сканирующего шаблона. В моем случае наилучшим оказалось соотношение ширины к высоте окна 11:20. Сделать это получилось с помощью запуска через консоль утилиты **opencv_createsamples.exe**. Команда, поданная на вход программы выглядит следующим образом:

opencv_createsamples.exe -info Good.dat -vec samples.vec -w 11 -h 20

Good.dat – файл описания положительных изображений.
-vec samples.vec – файл, в который будет сохранена приведённая к общему формату база положительных изображений. Адрес должен быть указан относительно программы opencv_createsamples.exe
-w 11 -h 20 — размер шаблона.
В результате работы программы получим файл samples.vec, содержащий в себе весь набор положительных изображений, приведенных к указанным пропорциям.

2.2 Тренировка каскада классификатора

Для подсчёта итогового каскада будем использовать opencv_traincascade.exe, лежащую в той же папке, что и opencv_createsamples.exe. Созданием структуры обучения, количеством эпох, требуемой точностью и остальными параметрами будем управлять с помощью командной строки.

opencv_traincascade.exe -data haarcascade -vec samples.vec -bg Bad.dat -numStages 16 -minhitrate 0.990 -maxFalseAlarmRate 0.4 -numPos 331 -numNeg 514 -w 11 -h 20 -mode ALL -precalcValBufSize 256 -precalcIdxBufSize 256

Здесь:

-data haarcascade — адрес папки, в которую ведется запись результатов.

-vec samples.vec — адрес посчитанного в прошлом пункте файла с положительными примерами.

-bg Bad.dat — адрес файла-описания отрицательных примеров.

-numStages — количество уровней каскада, которые программа будет обучать. Оптимальное значения подбираем опытным путем.

-minhitrate — коэффициент, определяющий качество обучения.

-maxFalseAlarmRate — уровень ложной тревоги.

-numPos 331 – количество позитивных примеров.

-numNeg 500 — количество негативных примеров.

w 11 -h 20 — размер примитива.

-mode ALL — использовать или нет полный комплект Хаар-признаков. От этого зависит скорость работы и точность алгоритма.

-precalcValBufSize 256 -precalcIdxBufSize 256 — выделяемая под процесс память.

Экспериментальным путем я выяснил, что оптимальными для данной задачи являются следующие параметры:

-numStages — 16

-minhitrate — 0.99

-maxFalseAlarmRate — 0.4

-numPos 331 – количество позитивных примеров.

-numNeg 500 — количество негативных примеров.

w 11 -h 20 — размер примитива.

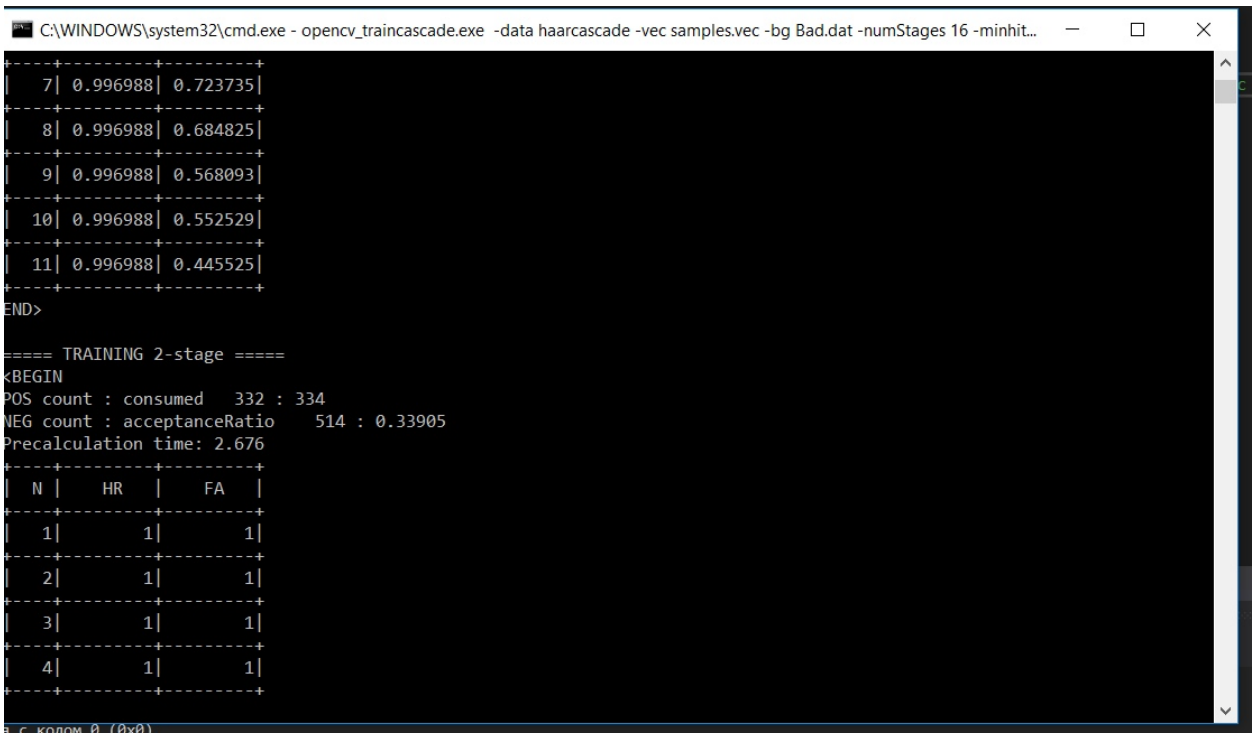

Рис. 7 Пример работы программы

2.3 Проверка точности работы классификатора

Мною было спроектировано и обучено порядка 10 различных конфигураций каскада Хаара. Результат работы наилучшего из них Вы можете видеть на рис.

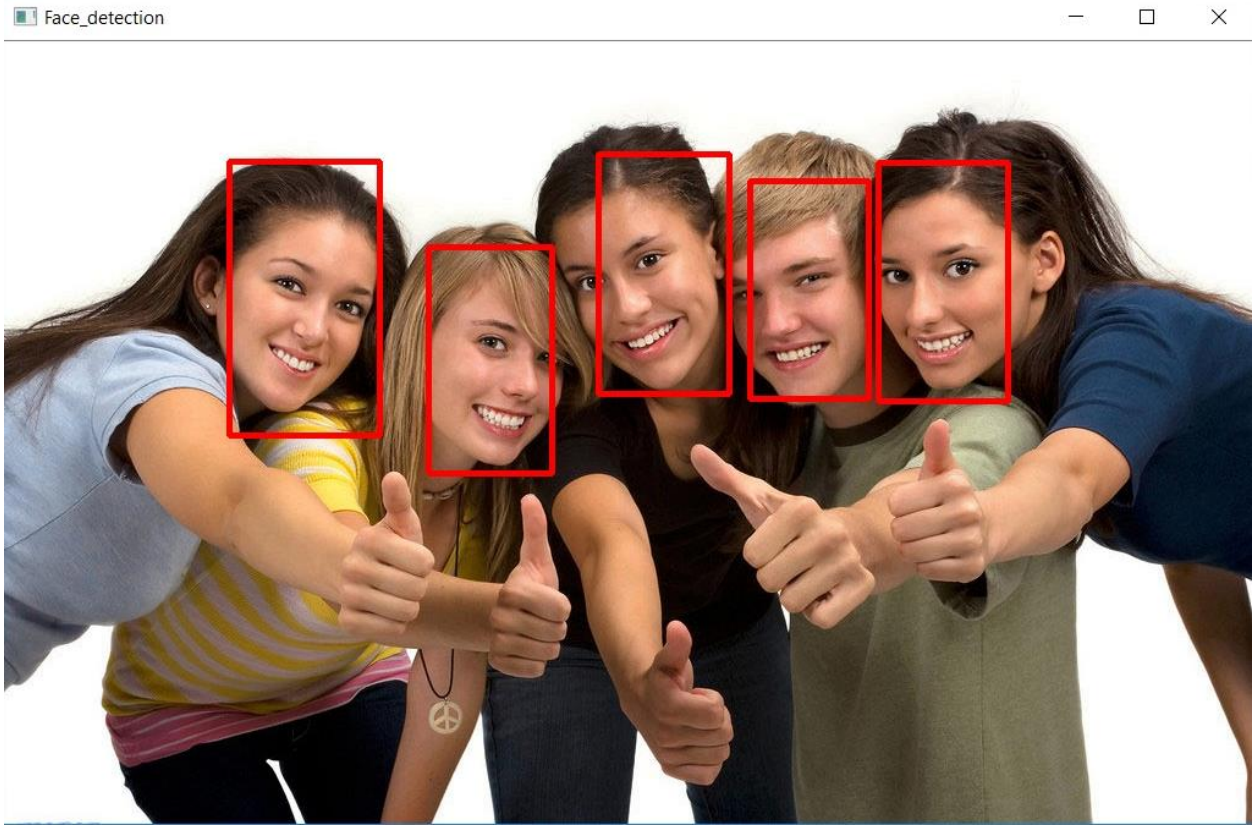

Рис. 7 Результат работы программы

3. Выводы

Результатом моей научной работы стала программа, позволяющая находить лица как на изображении, так и на видео, в условиях его потокового исследования (время работы программы около 70 миллисекунд, что с легкостью позволяет анализировать видеоряд для поиска лиц и их дальнейшей обработки). Я смог создать необходимую для обучения базу данных, грамотно их разметить и подать на вход каскадов Хаара с целью его дальнейшего обучения. Также мною написана программа для анализа потокового видеоряда с фронтальной камеры ноутбука или любой другой камеры. Дальнейшее направление моей работы – обучение каскадов Хаара, для идентификации национальности, пола и примерного возраста.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. P. Viola, M. Jones Rapid Object Detection using a Boosted Cascade of Simple Features, 2001
2. Линда Шапиро, Джордж Стокман, Компьютерное зрение, Бином. Лаборатория знаний, 2006 г. 752 стр.
3. http://www.cs.tau.ac.il/~wolf/ytfaces/ 07.04.2014
4. R Grosse, Micah K. Johnson, Edward H. Adelson William T. Freeman Ground truth dataset and baseline evaluations for intrinsic image algorithms, 2009
5. Lotfi A. Zadeh, Fuzzy Logic, Neural Networks, and Soft Computing, 1994
6. H. Bäcklund, A. Hedblom, N. Neijman A Density-Based Spatial Clustering of Application with Noise, 2011

ПРИЛОЖЕНИЕ 1

```
#include "main_functions.h"

void detectAndDraw(Mat& img, CascadeClassifier& cascade, double scale, bool tryflip)
{
	double t = 0;
	vector<Rect> faces, faces2;
	Mat gray, smallImg;

	cvtColor(img, gray, COLOR_BGR2GRAY);
	double fx = 1 / scale;

	resize(gray, smallImg, Size(), fx, fx, INTER_LINEAR_EXACT);
	equalizeHist(smallImg, smallImg);
	t = (double)getTickCount();
	cascade.detectMultiScale(smallImg, faces,
		1.1, 2, 0
		//|CASCADE_FIND_BIGGEST_OBJECT
		//|CASCADE_DO_ROUGH_SEARCH
		| CASCADE_SCALE_IMAGE,
		Size(30, 30));
	if (tryflip)
	{
		flip(smallImg, smallImg, 1);
		cascade.detectMultiScale(smallImg, faces2,
			1.1, 2, 0
			//|CASCADE_FIND_BIGGEST_OBJECT
			//|CASCADE_DO_ROUGH_SEARCH
			| CASCADE_SCALE_IMAGE,
			Size(30, 30));
		for (vector<Rect>::const_iterator r = faces2.begin(); r != faces2.end(); ++r)
		{
			faces.push_back(Rect(smallImg.cols - r->x - r->width, r->y, r->width, r-
>height));
		}
	}


	t = (double)getTickCount() - t;
	printf("detection time = %g ms\n", t * 1000 / getTickFrequency());
	for (size_t i = 0; i < faces.size(); i++)
	{
		Rect r = faces[i];
		Mat smallImgROI;
		vector<Rect> nestedObjects;
		Point center, pt1, pt2;
		Scalar color = Scalar(0, 0, 255);
		int radius;
		double aspect_ratio = (double)r.width / r.height;
		if (0.75 < aspect_ratio && aspect_ratio < 1.3)
		{
```

```
                        center.x = cvRound((r.x + r.width*0.5)*scale);
                        center.y = cvRound((r.y + r.height*0.5)*scale);
                        radius = cvRound((r.width + r.height)*0.25*scale);
                        pt1.x = center.x - radius;
                        pt1.y = center.y - radius;
                        pt2.x = center.x + radius;
                        pt2.y = center.y + radius;
                        rectangle(img, pt1, pt2, color, 3, 8, 0);
                        //circle(img, center, radius, color, 3, 8, 0);
                }
                else
                        rectangle(img, Point(cvRound(r.x*scale), cvRound(r.y*scale)),
                                Point(cvRound((r.x + r.width - 1)*scale), cvRound((r.y + r.height -
1)*scale)),
                                color, 3, 8, 0);
        }
        imshow("Face_detection", img);
}
```

ПРИЛОЖЕНИЕ 2

```
#ifndef main_functions_H
#define main_functions_H

#include <iostream>
#include <string>
#include <iomanip>
#include <sstream>
#include <opencv2/opencv.hpp>
#include <opencv2/core.hpp>
#include <opencv2/imgcodecs.hpp>
#include <opencv2/highgui.hpp>
#include <opencv2/calib3d.hpp>
#include <opencv2/imgproc.hpp>
#include <opencv2/objdetect.hpp>
#include <opencv2/videoio.hpp>
#include <Windows.h>

using namespace cv;
using namespace std;

void detectAndDraw(Mat& img, CascadeClassifier& cascade, double scale, bool tryflip);



#endif
```

ПРИЛОЖЕНИЕ 3

```
#include "main_functions.h"


string cascadeName;
const int work = 0; // work = 0 - Режим обучения, work = 0 - Режим работы со статичным фото, work = 2 - Режим работы с видео с камеры


int main(int argc, const char** argv)
{
	Mat image;
	bool tryflip;
	CascadeClassifier cascade;
	double scale;
	cv::CommandLineParser parser(argc, argv,
		"{cascade|data/haarcascades/cascade_webcam.xml|}"
		"{scale|1|}{try-flip||}{@filename||}"
	);
	cascadeName = parser.get<string>("cascade");
	scale = parser.get<double>("scale");
	if (scale < 1)
		scale = 1;
	tryflip = parser.has("try-flip");
	// Проверка на ошибки
	if (!parser.check())
	{
		parser.printErrors();
		return 0;
	}
	if (!cascade.load(samples::findFile(cascadeName)))
	{
		cerr << "ERROR: Could not load classifier cascade" << endl;
		return -1;
	}
	// Режим работы
	switch (work)
	{
	case 0:
	{
		String path = "C:\\WINDOWS\\system32\\cmd.exe";
		String cmd = "open";
		String comand1 = "/K cd /D D:\\learning && opencv_createsamples.exe -info Good.dat -vec samples.vec -w 11 -h 20";
		String comand2 = " && opencv_traincascade.exe -data haarcascade -vec samples.vec -bg Bad.dat -numStages 16 -minhitrate 0.990 -maxFalseAlarmRate 0.5 -numPos 332 -numNeg 514 -w 11 -h 20 -mode ALL -precalcValBufSize 256 -precalcIdxBufSize 256";
		String comand = comand1 + comand2;
		ShellExecute(0, cmd.c_str(), path.c_str(), comand.c_str(), NULL, SW_MINIMIZE);
		break;
```

```
        }
        case 1:
        {
                image = imread(samples::findFile("D:\group3.jpg"), IMREAD_COLOR); //
Адрес к любому изображению допустимых форматов(не работает с длинными адресами)
                if (image.empty())
                {
                        cout << "Couldn't read ___.jpg" << endl;
                        return 1;
                }
                detectAndDraw(image, cascade, scale, tryflip);
                break;
        }
        case 2:
        {
                VideoCapture capture;
                string inputName;
                Mat frame;
                inputName = parser.get<string>("@filename");
                if (inputName.empty() || (isdigit(inputName[0]) && inputName.size() == 1))
                {
                        int camera = inputName.empty() ? 0 : inputName[0] - '0';
                        if (!capture.open(camera))
                        {
                                cout << "Capture from camera #" << camera << " didn't work" <<
endl;
                                return 1;
                        }
                }
                if (capture.isOpened())
                {
                        for (;;)
                        {
                                capture >> frame;
                                if (frame.empty())
                                        break;
                                Mat frame1 = frame.clone();
                                detectAndDraw(frame1, cascade, scale, tryflip);
                                char c = (char)waitKey(10);
                                if (c == 27 || c == 'q' || c == 'Q')
                                        break;
                        }
                }
                break;
        }
        }
        waitKey(0);
        return 0;
}
```