ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Актуальность. Учет посетителей в общественном месте, компании или организации, выявление преступников, находящихся в розыске, контроль доступа к объекту на предприятии. С данными проблемами с легкостью справляется человек. Однако если количество посетителей слишком велико, то выявить преступника, который находится в базе данных, а уж тем более посчитать количество людей становится непосильной задачей, даже для нескольких наблюдателей. Существуют различные биометрические системы, которые основываются на уникальных биологических признаках человека, которые трудно подделать и которые однозначно определяют конкретного человека. Например, отпечатки пальцев, сетчатка глаза, форма лица. Однако сбор отпечатков пальцев или изображения сетчатки при учете посетителей не рационально, поскольку это занимает очень много времени. Наиболее подходящим для данного случая является детектирование и распознавание лиц. С этой задачей легко может справиться хорошо обученная машина. В ее задачи входит получение изображения, детектирование лиц и их идентификация. На сегодняшний день автоматизация различных процессов, таких как, идентификация человека по биометрическим признакам, в которых человек не принимает активного участия, а лишь только реагирует на оповещение от системы, стремительно развивается. Все это делает идентификацию человека, достаточно актуальной проблемой, особенно при ограниченных ресурсах. Однако автоматизация данного процесса, в частности распознавания лиц, невозможна без детектирования на изображении. Все это делает важной задачу разработки алгоритмов, устойчивых к различным особенностям строения

человека и различным факторам, влияющих на вероятность ошибки обнаружения, и которые имеют достаточно большую скорость обнаружения при ограниченных ресурсах. Таким образом автоматизация идентификации лиц на изображении невозможна без хорошего алгоритма детектирования лиц, что делает анализ и разработку последнего достаточно актуальной задачей.

Цель работы. Целью данной работы является анализ алгоритма детектирования лиц Виолы-Джонса. Анализ алгоритма с использованием пакета прикладных программ для решения задач технических вычислений Matlab, а также модификация алгоритма и сравнение ошибок первого и второго рода.

Глоссарий

Neural network (англ.) – Нейронная сеть

Multilayer Perceptrons (англ.) – Многослойные перцептрон

ИНС – Искусственная нейронная сеть

PCA (Princiapl Component Analysis) (англ.) – Метод главных компонент

Factor Analysis (англ.) – Факторный анализ

Linear Discriminant Analysis (англ.) – Линейный дискриминантный анализ

SVM (Support Vector Machines) (англ.) – Метод опорных векторов

Naive Bayes classifier (англ.) – Наивный байесовский классификатор

Hidden Markov model (англ.) – Скрытые Марковские модели

Distribution-based method (англ.) – Метод распределения

SNoW (Sparse network of winnows) (англ.) – Разреженная сеть окон

Active Appearance Models (англ.) – Активные модели внешности

AdaBoost (англ.) – Адаптивное ускорение

Haar feature (англ.) – Признак Хаара

Integral Image (англ.) - Интегральное представление изображения

Decision stump (англ.) – Дерево принятия решений

1. АНАЛИЗ МЕТОДОВ И АЛГОРИТМОВ ОБНАРУЖЕНИЯ ЛИЦ

Прежде чем выбрать алгоритм детектирования лиц, который я буду исследовать, необходимо проанализировать уже существующие алгоритмы. В данной главе описаны основные задачи и области применения детектирования лиц в современных системах, а также их сравнительный анализ.

1.1. Описание задачи обнаружения лиц в видеопотоке

Задача обнаружения лиц достаточно распространена в различных автоматизированных системах, где человек практически не принимает участия, а только реагирует на события от системы. Например, когда система пытается выявить преступника, находящегося в базе данных системы, из множества людей. Задача обнаружения лица на изображении часто является первым шагом в процессе решения задачи более высокого уровня — распознавания лица, деталей лица или его мимики. [1]

1.1.1. Область применения

Детектирование лиц зачастую используется в биометрии, в основном как часть системы распознавания лиц, видеонаблюдении, человеко-машинном интерфейсе и управлении базами данных изображений. Многие цифровые камеры используют детектирование лиц для автофокусировки.

Данные системы имеют очень широкую область применения:

- Системы наблюдения, устанавливаемые в общественных местах: в метро, на вокзалах, в аэропортах. Список идентификации в таком случае может

включать людей, находящихся в розыске. Тогда система распознавания осуществляет мониторинг лиц, появляющихся в области видимости камер наблюдения.

- Системы безопасности финансовых учреждений. Банковский сектор регулярно подвергается атакам мошенников, использующих поддельные удостоверения личности для получения денежных займов. Реакция системы идентификации по списку делает возможным принятие превентивных мер по отношению к потенциальным нарушителям.
- Автоматизированные системы обработки и управления содержимого сайтов и социальных сетей. Оперируя биометрическими шаблонами, можно установить связь между изображением лица на фотографии и сетевым профилем соответствующей ему личности. С другой стороны, одним из пунктов пользовательского соглашения с социальными сетями является предоставление пользователем корректной личной информации. Это правило нарушается, когда становится невозможно установить внешний вид пользователя из-за отсутствия изображения лица на его профильной фотографии или из-за наличия посторонних лиц на предоставленной фотографии. Наконец, цифровое изображение человека является объектом гражданского права, в связи с чем возникает проблема отслеживания неправомерного использования фотографий, чаще всего – публичных личностей.
- Поисковые системы используют биометрическую информацию для индексации массивов изображений с целью улучшения точности ответов на запросы пользователей. [2]

1.1.2. Решение задач поиска и распознавания лиц в современных системах видеонаблюдения

Задача автоматического распознавания лиц с целью установления личности имеет большое количество приложений в различных областях. Повышенный интерес к данной технологии вызван проблемами общественной безопасности, потребностью в удаленной аутентификации, развитием человеко-машинных интерфейсов. Что важно, во многих случаях для достижения приемлемого качества распознавания лиц не требуется дорогостоящее специфическое оборудование: источниками образцов могут служить фотографии или видеозаписи, сделанные непрофессиональной камерой. Благодаря многочисленным социальным и файлообменным сетям, изображение лица является одним из наиболее распространенных и доступных биометрических параметров человека. Этот факт породил новый вид задач, связанных с поиском информации в глобальной сети Интернет на основе биометрических данных. [2]

В своей бакалаврской работе я занимаюсь анализом алгоритма Виолы-Джонса, поскольку детектирование лиц является базовым этапом биометрической идентификации в автоматизированных системах.

1.2. Ограничения на систему поиска лиц

Детектирование лица на изображении является достаточно сложной задачей, поскольку человек постоянно находится в движении, положение его лица, а также угол наклона меняется, также стоит учитывать принадлежность человека к определенной расе. Например, негроидная раса имеет богатую на меланин кожу, что делает затруднительным использование признаков Хаара в алгоритме детектирования Виолы-Джонса. Растительность на лице и различные аксессуары и эмоции увеличивают вероятность ошибки обнаружения. [3]

1.3. Анализ существующих алгоритмов детектирования лиц

За все время существования проблемы детектирования лиц было разработано и исследовано множество алгоритмов, позволяющих выделить изображение лица на фотографии или в видеопотоке. Упрощенным вариантом задачи обнаружения лица является его локализация. В этом случае системе заранее известно, что на изображении находится одно лицо. [6] Каждый из алгоритмов имеет свои преимущества и недостатки.

Существующие методы обнаружения лиц можно разбить на четыре категории:

- Эмпирический метод;
- Метод характерных инвариантных признаков;
- Распознавание с помощью шаблонов, заданных разработчиком;
- Метод обнаружения по внешним признакам, обучающиеся системы; [4]

1.3.1. Эмпирический подход «базирующийся на знаниях сверху-вниз»

Данный подход предполагает создание алгоритма, реализующего набор правил, которым должен отвечать фрагмент изображения, для того чтобы быть признанным человеческим лицом. Этот набор правил является попыткой формализовать эмпирические знания о том, как именно выглядит лицо на изображениях и чем руководствуется человек при принятии решения: лицо он видит или нет. Самые простые правила:

- центральная часть лица имеет однородную яркость и цвет;
- разница в яркости между центральной частью и верхней частью лица значительна;
- лицо содержит в себе два симметрично расположенных глаза, нос и рот, резко отличающиеся по яркости относительно остальной части лица.

Метод сильного уменьшения изображения. Для сглаживания помех, а также для уменьшения вычислительных операций предварительно подвергает изображение сильному уменьшению (**Рис.** ***1***). На таком изображении проще выявить зону равномерного распределения яркости, отвечающую за зону нахождения лица, а затем проверить наличие резко отличающихся по яркости областей внутри: именно такие области можно с разной долей вероятности отнести к «лицу». [4]

Рис. 1 Пример сжатия изображения

Метод построения гистограмм. Для определения областей изображения с «лицом» строят вертикальную и горизонтальную гистограммы (**Рис. *2***). В областях, где наиболее вероятно нахождение лица кандидатах происходит его поиск. Данный подход использовался во время развития компьютерного зрения, когда процессоры были еще достаточно «слабыми». Рассмотренные выше методы имеют неплохие показатели по выявлению лица при однородном фоне, а также легко реализуемы. Однако эти методы абсолютно непригодны для обработки изображений, содержащих большое количество лиц. [7]

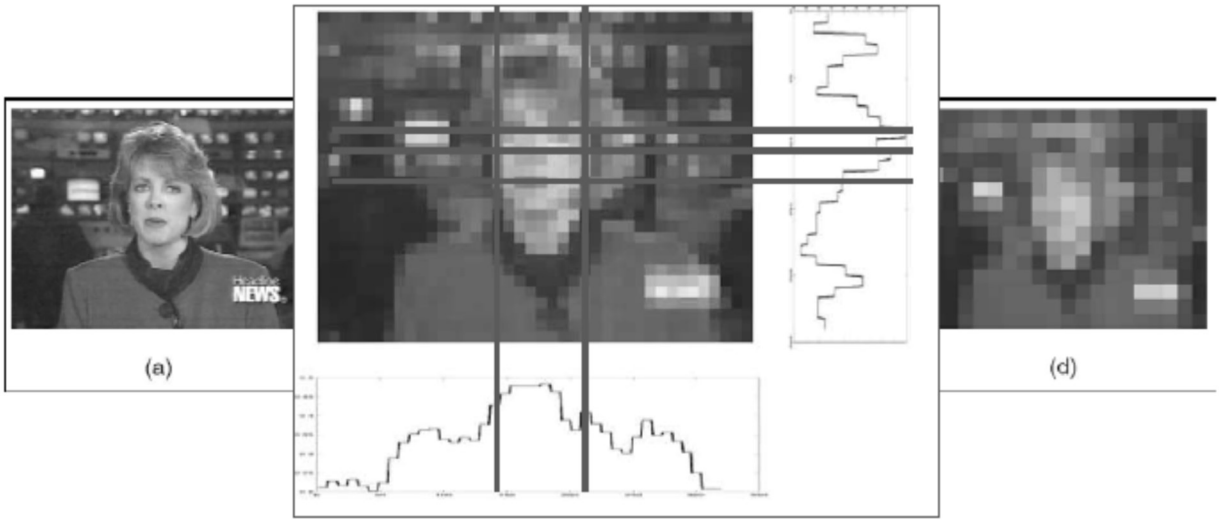


Рис. 2 Поиск лица по гистограмме

1.3.2. Методы характерных инвариантных признаков, базирующиеся на знаниях снизу-вверх

Данный метод использует инвариантные свойства лиц, которые постоянны независимо от условий съемки, таких как ориентация лица в пространстве, освещенность.

Алгоритм работы данных методов распознавания может быть описан следующим образом:

- Детектирование на изображении явных признаков лица: глаз, носа, рта;
- Обнаружение: границы лица, форма, яркость, текстура, цвет;
- Объединение всех найденных инвариантных признаков и их верификация; [8]

1.3.3. Распознавание с помощью шаблонов, заданных разработчиком

Шаблоны задают некий стандартный образ изображения лица, например, путем описания свойств отдельных областей лица и их возможного взаимного расположения. Обнаружение лица с помощью шаблона заключается в проверке каждой из областей изображения на соответствие заданному шаблону. [11]

Особенности подхода:

- два вида шаблонов (деформируемые и недеформируемые)
- шаблоны заранее запрограммированы, необучаемы
- используется корреляция для нахождения лица на изображении

Метод детектирования лица при помощи трехмерных форм. Метод предполагает использование шаблона в виде пар отношений яркостей в двух областях. Для определения местоположения лица необходимо пройти всё изображение на сравнение с заданным шаблоном. Причём делать это необходимо с различным масштабом (**Рис. *3***). [4]

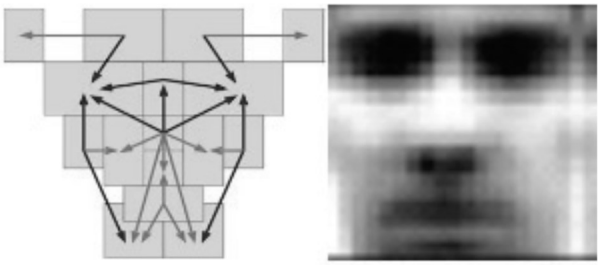


Рис. 3 Метод детектирования лица при помощи трехмерных форм

Модели распределения опорных точек. Эти модели являются статистическими моделями, которые представляют объекты, форма которых может измениться. Их полезная особенность метода — способность выделить форму переменных объектов в пределах учебного набора с небольшим количеством параметров формы. Эта компактная и точная параметризация может использоваться для разработки эффективных систем классификации. К достоинствам распознавания с помощью шаблонов можно отнести относительную простоту реализации и неплохие результаты на изображениях с не очень сложным задним фоном. А главным недостатком является необходимость калибровки шаблона вблизи с изображением лица. [12]

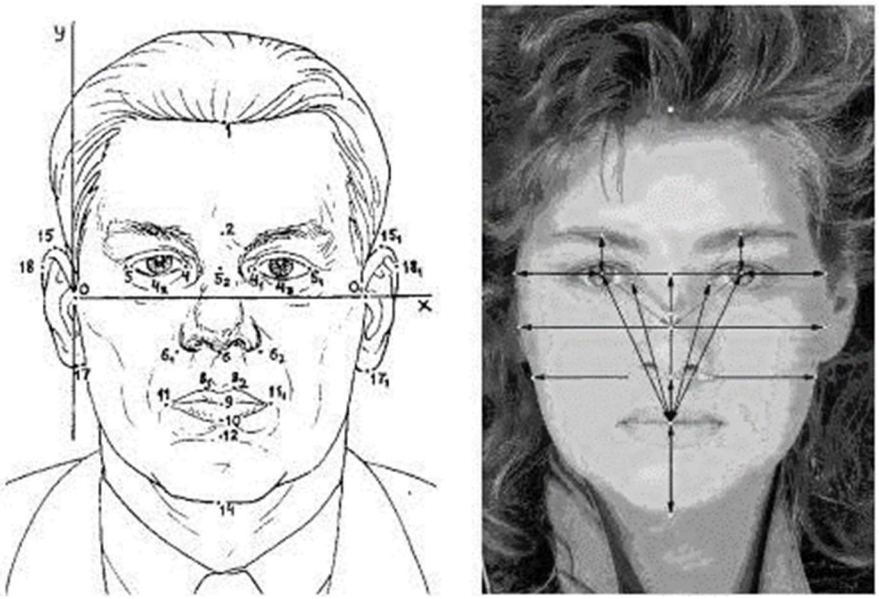


Рис. 4 Характерные точки лица

1.3.4. Методы обнаружения лица по внешним признакам

Изображению ставится в соответствие некоторым образом вычисленный вектор признаков, который используется для классификации изображений на два класса — лицо/не лицо. Обычно поиск лиц на изображениях с помощью методов, основанных на построении математической модели изображения лица, заключается в полном переборе всех прямоугольных фрагментов изображения всевозможных размеров и проведения проверки каждого из фрагментов на наличие лица. Поскольку схема полного перебора обладает такими безусловными недостатками, как избыточность и большая вычислительная сложность, авторами применяются различные методы сокращения количества рассматриваемых фрагментов. [13]

Основные принципы методов:

- Схоластика: каждый сканируется окном и представляется векторами ценности
- Блочная структура: Изображение разбивается на пересекающиеся или непересекающиеся участки различных масштабов и производится оценка с помощью алгоритмов оценки весов векторов

Для обучения алгоритмов требуется библиотека вручную подготовленных изображений лиц и «не лиц», любых других изображений.

Стоит отметить что важнейшей задачей является выделить сильные классификаторы. Именно они будут иметь наивысший приоритет для проверки найденных признаков в изображении. Количество же более слабых классификаторов стоит уменьшать за счёт похожести друг на друга, а также

удалении классификаторов, возникших за счёт шумовых выбросов.

Основные методики выполнения этих задач:

- Искусственные нейронные сети (Neural network: Multilayer Perceptrons);
- Метод главных компонент (Principapl Component Analysis (PCA));
- Факторного анализа (Factor Analysis);
- Линейный дискриминантный анализ (Linear Discriminant Analysis);
- Метод опорных векторов (Support Vector Machines (SVM));
- Наивный байесовский классификатор (Naive Bayes classifier);
- Скрытые Марковские модели (Hidden Markov model);
- Метод распределения (Distribution-based method);
- Разреженная сеть окон (Sparse network of winnows (SNoW));
- Активные модели (Active Appearance Models);
- Адаптированное улучшение и основанный на нём Метод Виолы-Джонса

На сегодняшний день метод искусственных нейронных сетей является наиболее распространенным способом решения задач распознавания лиц на изображении. Искусственная нейронная сеть (ИНС) — это математическая модель, представляющая собой систему соединённых и взаимодействующих между собой нейронов. Нейронные сети не программируются в привычном смысле этого слова, они обучаются. [4]

Самым перспективным в плане высокой производительности и низкой частоты ложных срабатываний и большим процентом верных обнаружений лиц выглядит метод Виолы-Джонса. Основные принципы, на которых основан метод, таковы:

- используются изображения в интегральном представлении, что позволяет вычислять быстро необходимые объекты;
- используются признаки Хаара, с помощью которых происходит поиск нужного объекта;
- используется бустинг для выбора наиболее подходящих признаков для искомого объекта на данной части изображения;
- все признаки поступают на вход классификатора, который даёт результат «верно» либо «ложь»;
- используются каскады признаков для быстрого отбрасывания окон, где не найдено лицо.

Обучение классификаторов идет очень медленно, но результаты поиска лица очень быстры. Алгоритм хорошо работает и распознает черты лица под небольшим углом, примерно до 30 градусов. При угле наклона больше 30 градусов процент обнаружений резко падает. [4]

Таким образом, проанализировав основные методы обнаружения лица на изображении, я пришел к выводу, что одним из самых высокопроизводительных и распространенных на сегодняшний день является алгоритм Виолы-Джонса, исследованием которого я буду заниматься.

1.4. Постановка задачи бакалаврской работы

Проведя анализ известных алгоритмов детектирования лиц мной был выбран алгоритм Виолы-Джонса, поскольку он является наиболее часто используемым на сегодняшний день, имеет достаточно высокую скорость обнаружения лиц, а также относительно прост в реализации.

Основной целью бакалаврской работы является исследование алгоритма детектирования лиц, Виолы-Джонса, модификация алгоритма, для выявления наиболее точного алгоритма классификации, обнаружение ошибок первого и второго рода, тестирование алгоритма на тестовой выборке изображений.

Таким образом можно выделить основные пункты:

- Исследование и анализ работы алгоритма детектирования Виолы-Джонса
- Модификация алгоритма с использованием отличных от базового, алгоритмов классификации. Сравнение результатов.
- Определение вероятностей ошибок первого и второго рода.

1.5. Техническое задание

1.5.1. Введение

Основной целью бакалаврской работы является исследование алгоритма детектирования лиц, Виолы-Джонса, модификация алгоритма, для выявления наиболее точного алгоритма классификации, обнаружение ошибок первого и второго рода, тестирование алгоритма на тестовой выборке изображений.

1.5.2. Основания для разработки

Задание на бакалаврскую работу

1.5.3. Назначение разработки

Программное обеспечение, позволяющее детектировать лицо на изображении и видеопотоке, а также записывать полученное изображение в базу данных для дальнейшей обработки с использованием библиотеки OpenCV.

1.5.4. Требования к программе

- Удобный интерфейс взаимодействия с пользователем
- Высокая производительность

1.5.4.1. Требования к функциональным характеристикам

- Детектирование лиц в видеопотоке
- Запись обнаруженных лиц в базу данных

1.5.4.2. Требования к надежности

Для устойчивого функционирования программного обеспечения необходимо:

- Выполнение резервного копирования базы данных;
- Наличие бесперебойных блоков питания персональных компьютеров;

1.5.4.3. Условия эксплуатации

- Диапазон рабочих температур: 0…+30º С
- Относительная влажность при температуре 25º С - 40-60%
- Только сотрудники могут являться пользователями персонального компьютера.

1.5.4.4. Требования к составу и параметрам технических средств

Технические характеристики:

- Персональный компьютер: процессор Intel или совместимый с тактовой частотой от 2.6 ГГц и выше, 64 битная ОС, 4 Гб оперативной памяти и выше.

1.5.4.5. Требования к информационной и программной совместимости

Операционная система персонального компьютера – Windows XP и выше, либо дистрибутив UNIX-подобной операционной системы.

Требования к защите информации:

- Разграничение доступа к данным в соответствии с должностными обязанностями;
- Класс защищенности системы – не ниже 1Б согласно руководящему документу «Автоматизированные системы. Защита от несанкционированного доступа к информации. Классификация автоматизированных систем и требования по защите информации.»
- Оформить инструкции для администратора по обеспечению безопасности и поддержанию функционирования программного обеспечения

1.5.5. Требования к программной документации

В ходе разработки должна быть подготовлена следующая документация:

- описание программы;
- программа и методика испытаний;
- описание применения;
- графический материал;
- структура программы;
- общая схема алгоритма

2. АНАЛИЗ И ИССЛЕДОВАНИЕ АЛГОРИТМА ВИОЛЫ-ДЖОНСА

2.1. Этапы реализации алгоритма

Алгоритм Виолы-Джонса позволяет обнаруживать объекты на изображении в реальном времени. Для большего понимания метода используется разбиение его на отдельные пункты:

- Признаки Хаара, с помощью которых происходит поиск нужного объекта
- Интегральное представление изображения для быстрого вычисления признаков Хаара
- Адаптивное ускорение (AdaBoost), позволяющее отсеять признаки, которые несут в себе наименьшее количество информации, а также получить сильные классификаторы из множества слабых.

Таким образом реализация алгоритма детектирования сводится к реализации отдельных его методов, которые, в свою очередь, можно реализовать в любом порядке, а затем объединить их в один алгоритм.

2.1.1. Признаки Хаара

Признаки Хаара - признаки цифрового изображения, используемые в распознавании образов (***Рис. 5***). Они использовались в первом детекторе лиц, работающем в реальном времени. Простейший прямоугольный признак Хаара можно определить, как разность сумм пикселей двух смежный областей внутри прямоугольника, который может занимать различные положения и масштабы на изображении. Существуют также различные вариации данных признаков, например, наклоненные на 45 градусов, для увеличения размерности пространства признаков.

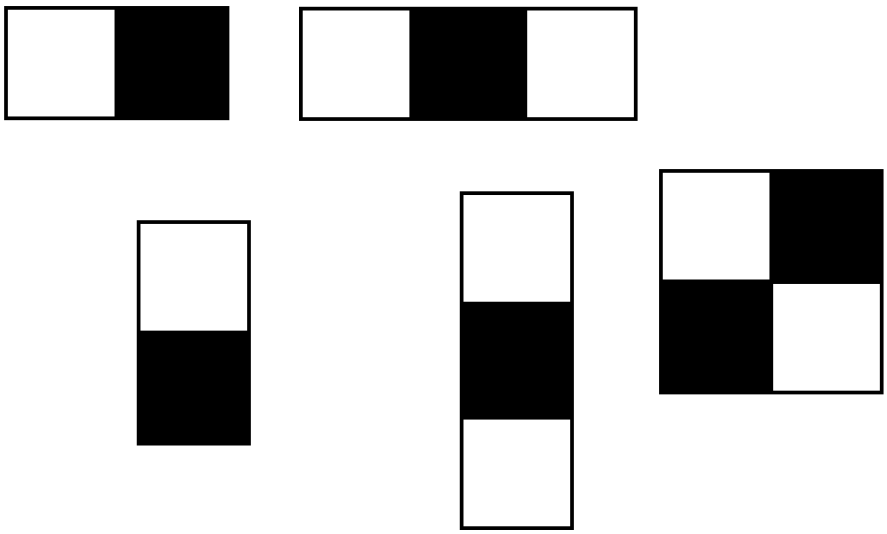


Рис. 5 Пример признаков Хаара

В алгоритме Виолы-Джонса используются признаки Хаара для вычисления разности суммы пикселей между черной областью и белой на изображении. Полученное скалярное значение – характеризует конкретный признак. В данной работе использовались признаки, указанные выше (**Рис. *5***).

Для того чтобы компенсировать влияние различных условий освещенности, изображение должно быть нормализовано (**Рис. *6***), т.е. иметь нулевое математическое ожидание и 1 дисперсию, также важным критерием вычисления признаков является единичное соотношение сторон изображения и его представление в сером формате.

$X = X - \bar{X} -$ получение нулевого математического ожидания

$\bar{X} = \frac{1}{n}\sum_{i=1}^{n} X_i -$ математическое ожидание

$X -$ матрица изображения

$n -$ размерность матрицы изображения

$X = \frac{X}{\sigma} -$ получение единичной дисперсии

$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(X - \bar{X})^2} -$ среднеквадратичное отклонение

Рис. 6 Ненормализованное изображение (слева) и нормализованное (справа)

Сложность вычисления. Для того, чтобы вычислить один признак необходимо вычесть сумму пикселей черной области из белой. При этом необходимо учитывать, что размер и положение каждого отдельного признака будет различным. Так для изображения размерностью 24x24 в общей сумме будет 162336 признаков. Это огромное значение, поскольку для каждого из 5 шаблонов необходимо произвести итераций намного больше, чем количество признаков. Для этого в алгоритме Виолы-Джонса используется интегральное представление изображения.

2.1.2. Интегральное представление изображения

Поскольку для вычисления одного признака необходимо каждый раз вычислять сумму каждой из областей, то для упрощения вычислений используется интегральное представление изображения (**Рис.** *7*). Смысл заключается в следующем:

- Есть изображение размерностью n
- Для координаты (i, j) вычисляется сумма левого и верхнего пикселя, и вычитается значение верхнего левого пикселя
- Итерация повторяется до правой нижней координаты включительно

Формально данный алгоритм можно записать как:

$$I(X) = X[i; j] + X[i - 1;\ j] + X[i;\ j - 1] - X[i - 1;\ j - 1]$$

$$I(X) - \text{интегральное представление изображения } X$$

Рис. 7 Интегральное преобразование для лица

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

1	2	3	4	5
2	4	6	8	10
3	6	9	12	15
4	8	12	16	20
5	10	15	20	25

Рис. 8 Пример интегрального преобразования

Теперь, имея интегральное представление изображения можно за более короткий промежуток времени вычислить площадь каждой из половины прямоугольников. Для этого обозначим вершины прямоугольника, который хотим вычислить. Например, вычислим площадь закрашенного прямоугольника (**Рис. *9***), используя уравнение: $S = D + A - C - B \rightarrow S = 25 + 4 - 10 - 10 = 9$, что соответствует количеству единиц соответствующей области на **Рис. *8***.

1	2	3	4	5
2	4 (A)	6	8	10 (C)
3	6	9	12	15
4	8	12	16	20
5	10 (B)	15	20	25 (D)

Рис. 9 Площадь закрашенного прямоугольника

Так для вычисления всех признаков методам полного перебора для изображения размерностью 24x24 потребовалось ≈ 35 с., в то время как вычисление для интегрального изображения потребовало ≈ 0.6 с.

Таким образом получили улучшение в $\frac{35}{0.6} \approx 58.3$ раз.

2.1.3. Адаптивное ускорение

Получив все 162336 признаков необходимо отсеять наиболее неинформативные (**Рис.** ***11***).

Рис. 10 Информативные признаки

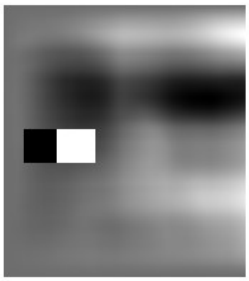


Рис. 11 Неинформативный признак

Поскольку области глаз, носа и рта темнее, то их легко можно выделить с помощью признаков, поэтому они являются информативными (**Рис.** ***10***).

Краткое описание алгоритма. Ускорение (усиление) - это подход в машинном обучении, который основывается на создании высокоточного правила предсказания, путем комбинирования относительно слабых и неточных правил. Алгоритм AdaBoost, предложенный Йоавом Фройндом (Yoav Freund) и Робертом Шапиром (Robert Schapire) был первым алгоритмом усиления, который до сих пор используется и изучается в различных областях. [14] [15]

Подробное описание алгоритма. Данный алгоритм является бинарным, поскольку он может разделять множество только на два класса. Существует также мульти классовый алгоритм, однако в бакалаврской работе он не рассматривается. [16]

Основной задачей адаптивного ускорения является построение сильного классификатора на основе слабых. Слабым классификатором, в данной работе, является признак Хаара. Сильный классификатор – это множество слабых классификаторов, которые наилучшим образом определяют лицо на изображении.

На вход алгоритма подается обучающая выборка размером m, где каждому признаку Хаара x_i соответствует его классификация y_i. Так для 1000 изображений вычисляется один из 162336 признаков Хаара и ему в соответствие ставится +1 или -1 в зависимости от принадлежности его к лицу или не лицу соответственно.

На каждом шаге t=1..T происходит вычисление D_t, для каждого из m примеров, и полученный слабо обученный классификатор, цель которого найти наименьшую ошибку ε_t для D_t.

Последним шагом является комбинирование слабых классификаторов в один сильный F(x). [17]

Особенностью применения алгоритма обучения в алгоритме Виолы-Джонса является отбор сильных классификаторов. Эта цель достигается путем оценивания вероятности правильно классифицированных изображений. Например, вероятность правильной классификации лиц из выборки не должна

быть ниже 95%.

Псевдокод AdaBoost.

Вход: $(x_1, y_1), \dots, (x_n, y_n)$

Инициализация: $D_1(y_i = 1) = \frac{1}{k}; D_1(y_i = -1) = \frac{1}{r}; k + r = n$

$For\ t = 1, \dots, T$:

- Вычисление слабого классификатора h_t
- Вычисление $\varepsilon_t = \sum_{i=1}^{n} D_t(i)[y_i \neq h_t]$
- Если $\varepsilon_t = 0$, то остановиться. Вернуть h_t
- Вычислить $\alpha_t = \frac{1}{2} ln\left(\frac{1-\varepsilon_t}{\varepsilon_t}\right)$
- Обновить веса:

$$D_{t+1} = \frac{D_t}{2}\left(\frac{1}{\varepsilon_t}[y_i \neq h_t] + \frac{1}{1-\varepsilon_t}[y_i = h_t]\right)$$

Финальный классификатор: $H(x) = sign(\sum_{t=1}^{T} \alpha_t\, h_t(x))$

T – количество этапов обучения классификатора.

Таким образом, имея сильный классификатор можно с большой вероятностью определить лицо на изображении, не вычисляя все признаки, что позволяет сэкономить процессорное время и увеличить производительность.***

Разделительный порог. Одной из главных задач алгоритма AdaBoost является нахождение порога, который наилучшим образом разделяет один класс данных от другого. По входным выборкам, относящимся к различным классам можно построить плотности распределения, которые будет иметь свои показатели центра распределения. Приведу пример для выборки, имеющей 30 элементов. Первая ее половина состоит из 5 случайных элементов в диапазоне от 0 до 1, распределенных по нормальному закону распределения и относящихся к одной группе, классифицированной как 1 (красный цвет). Вторая половина также состоит из 5 случайных элементов, но сдвинутых на 0.1, т.е. внесен некоторый шум, который может перемешать незначительную часть элементов между собой и имеет классификацию -1 (синий цвет) (**Рис. *12***).

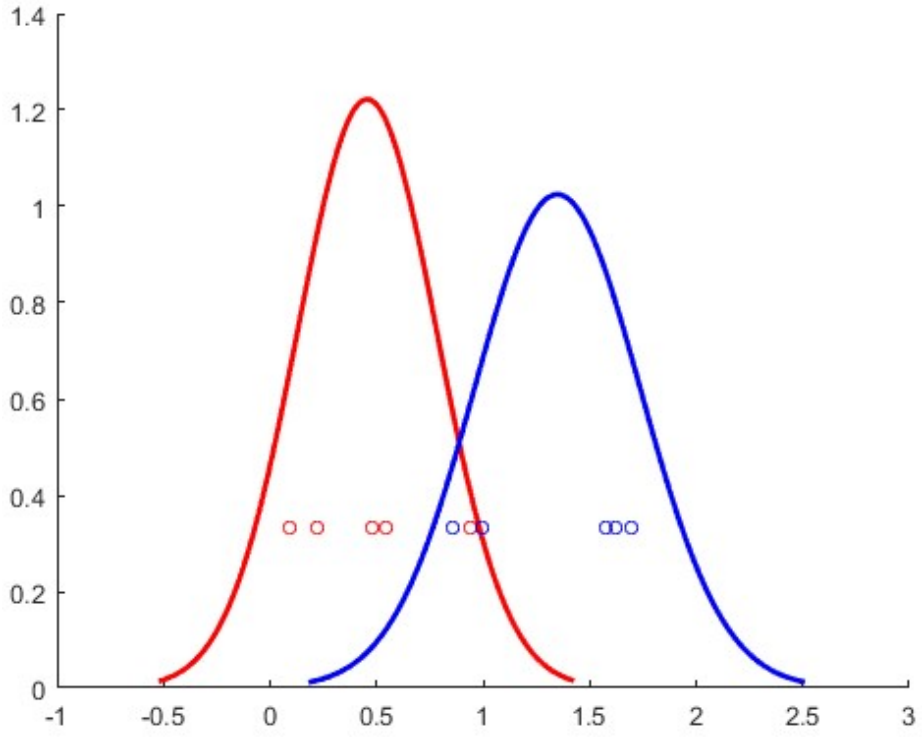
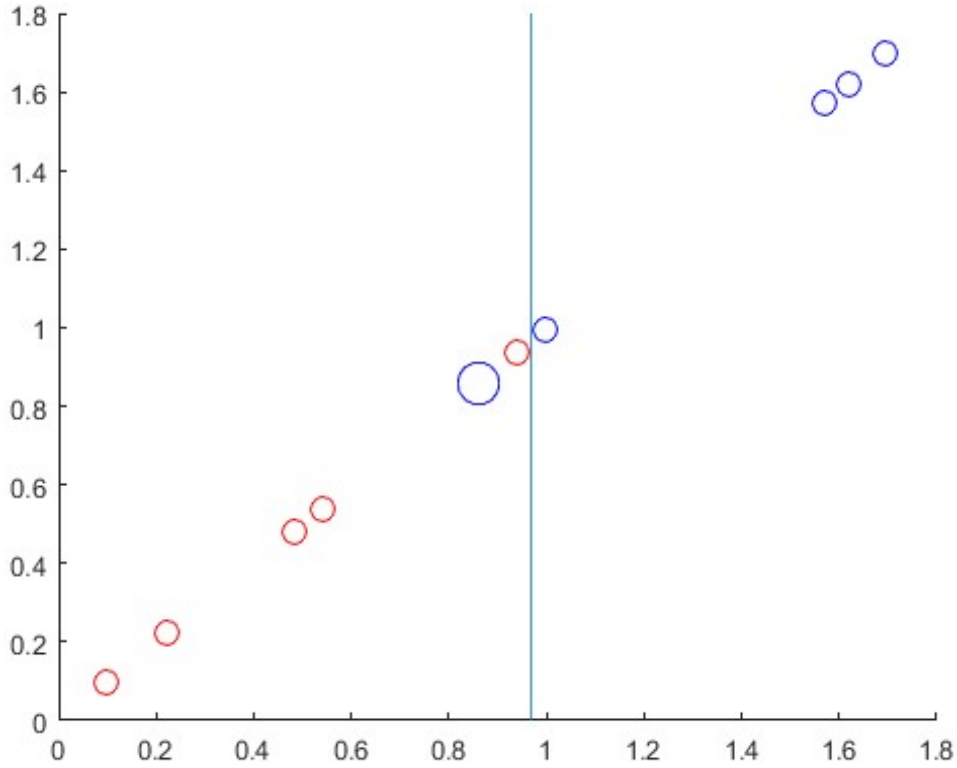


Рис. 12 Плотности распределения данных и их значения

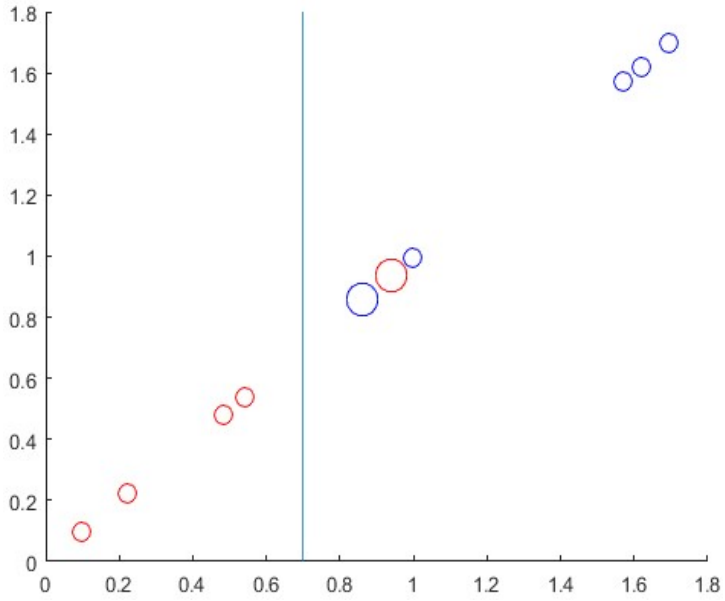
Таким образом на каждой итерации алгоритма AdaBoos находится оптимальный порог, который наилучшим образом разделяет данные на два класса, после чего веса неправильно классифицированных элементов увеличиваются.

- Шаг 1:



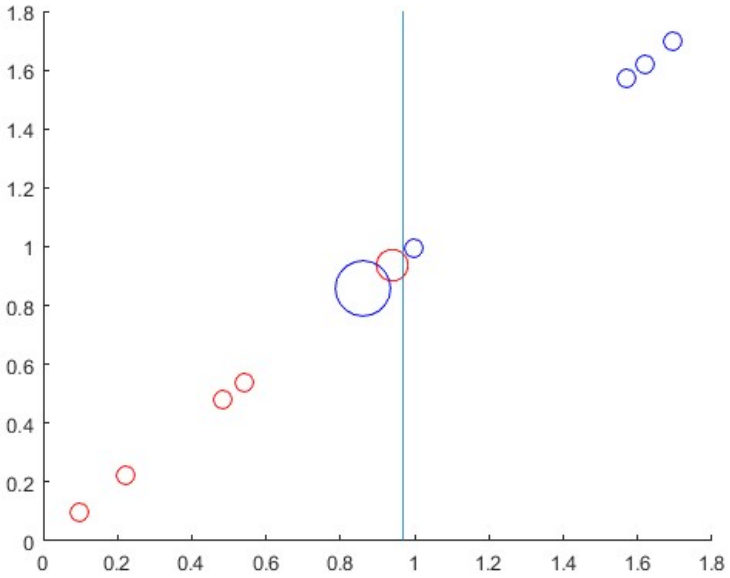
Неправильно классифицирован объект синего класса.

- Шаг 2:



Неправильно классифицирован объект красного класса.

- Шаг 3:



Неправильно классифицирован объект синего класса.

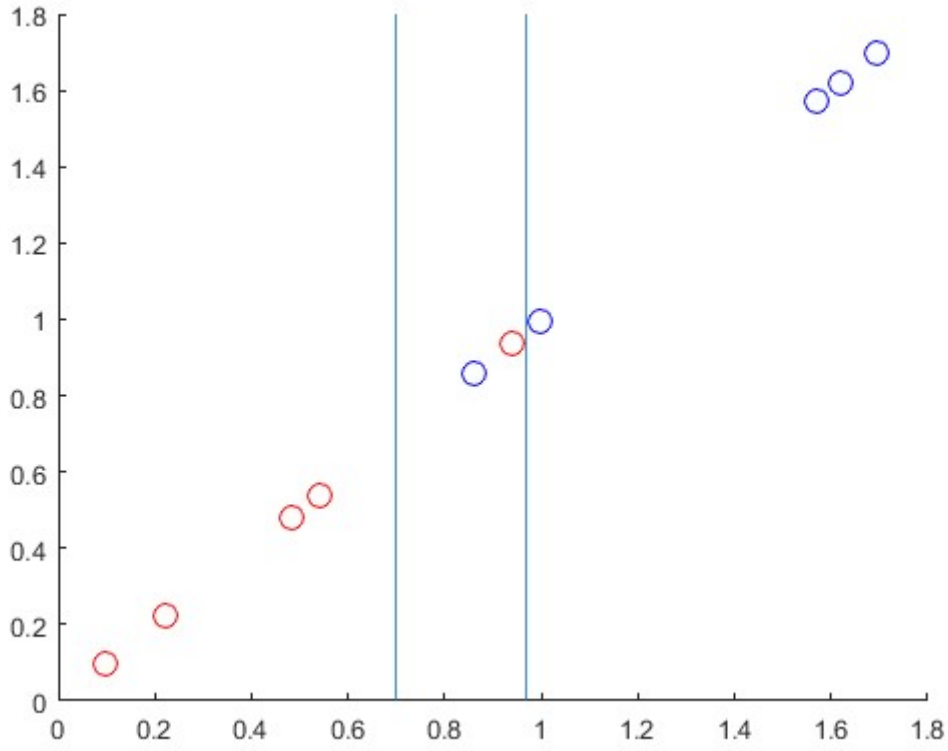


Рис. 13 Сильный классификатор

Поскольку слабый классификатор определяет какие веса будут перевзвешиваться, то произведение таких слабых классификаторов определяют сильный(**Рис.** ***13***).

В данном контексте алгоритм AdaBoost имеет две роли:

- Построение сильного классификатора на основе слабых
- Выделение наилучших признаков, которые с вероятностью не меньше 0.95 правильно классифицируют лицо на изображении.

Данный процесс называю также построением дерева принятия решений, который относится к задаче дискриминантного анализа.

Пример разделения данных:

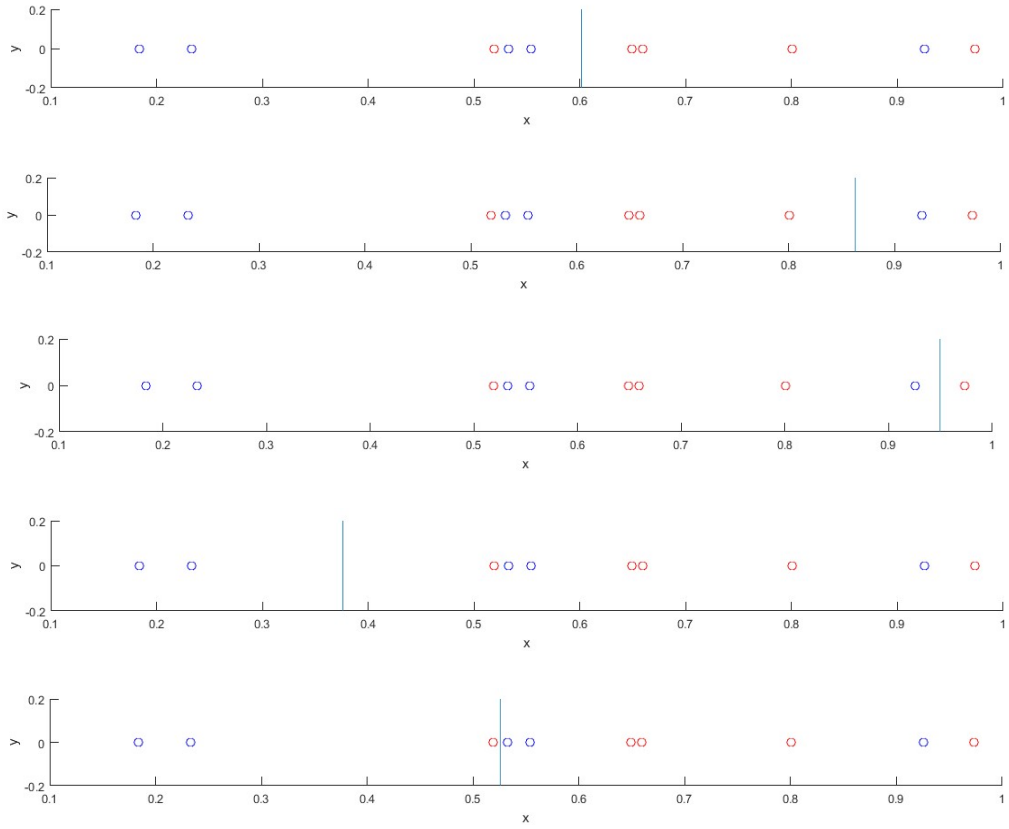


Рис. 14 Пример порогового разделения классов

Таким образом, на каждой итерации формируется слабый классификатор со своим пороговым значением и знаком паритета.

На основе разделения данных приведенных (**Рис.** ***14***) выше можно построить дерево(**Рис.** ***15***):

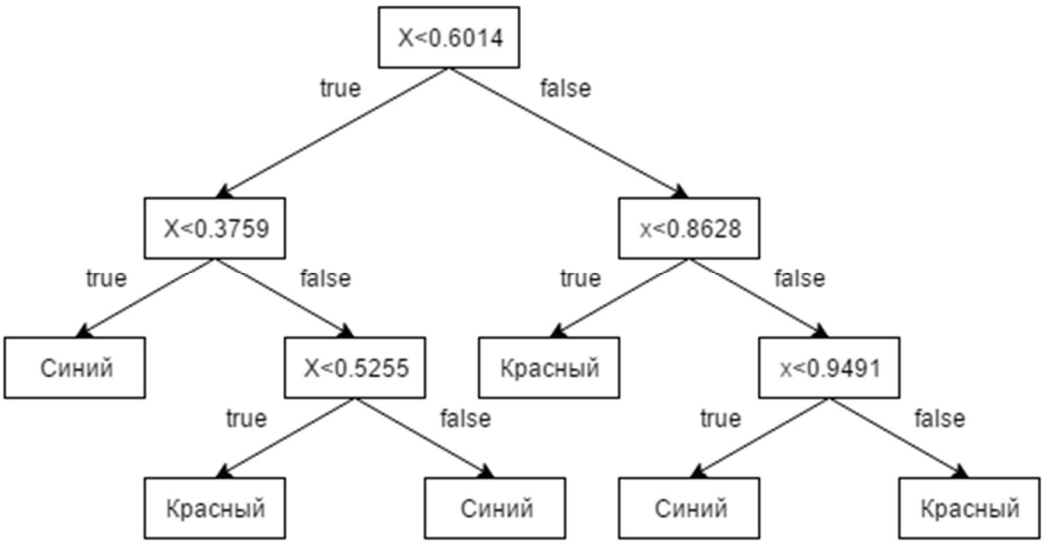


Рис. 15 Пример дерева

Структура дерева представляет собой «листья» и «ветки». На ребрах дерева решения записаны атрибуты, от которых зависит целевая функция, в «листьях» записаны значения целевой функции, а в остальных узлах — атрибуты, по которым различаются случаи. Чтобы классифицировать новый случай, надо спуститься по дереву до листа и выдать соответствующее значение. Подобные деревья решений широко используются в интеллектуальном анализе данных. Цель состоит в том, чтобы создать модель, которая предсказывает значение целевой переменной на основе нескольких переменных на входе.

2.2. Выводы

В данной главе был проанализирован и исследован каждый этап реализации алгоритма Виолы-Джонса. Руководствуясь полученными знаниями можно реализовать каждый этап, получить необходимые оценки вероятностей ошибок первого и второго рода, модифицировать алгоритм различными видами алгоритмов классификации.

3. РЕАЛИЗАЦИЯ АЛГОРИТМА ВИОЛЫ-ДЖОНСА

В данной главе объясняется выбор языка программирования, среды разработки, библиотек и пакетов прикладных программ для разработки и исследования алгоритма. Также описывается каждый этап разработки.

3.1. Выбор средств разработки и исследования

Для того, чтобы исследовать алгоритм Виолы-Джонса – необходимо выбрать подходящую среду разработки. Мной был выбран пакет прикладных программ для решения задач технических вычислений «MATLAB», поскольку он обладает необходимыми средствами для решения конкретных инженерных задач.

3.1.1. MATLAB

Описание. MATLAB — пакет прикладных программ для решения задач технических вычислений и одноимённый язык программирования, используемый в этом пакете. Пакет используют более миллиона инженерных и научных работников, он работает на большинстве современных операционных систем, включая Linux, Mac OS и Microsoft Windows.

Применение. MATLAB обладает достаточно большим спектром методов, используемых почти во всех областях математики:

- Матрицы и линейная алгебра.
- Многочлены и интерполяция.
- Математическая статистика и анализ данных.
- Обработка данных.
- Дифференциальные уравнения
- Разреженные матрицы.
- Целочисленная арифметика.

Таким образом MATLAB предоставляет удобные средства для разработки алгоритмов, включая высокоуровневые с использованием концепций объектно-ориентированного программирования. В нём имеются все необходимые средства интегрированной среды разработки, например, отладчик. [21]

3.2. Реализация этапов алгоритма в пакете MATLAB

Для получения необходимых оценок вероятностей ошибок первого и второго рода были реализованы некоторые методы алгоритма с использованием инструментов пакета MATLAB, такие как вычисление признаков Хаара, представление изображения в интегральной форме, построение сильного классификатора на основе слабых и отсеивание неинформативных признаков из всего множества признаков.

3.2.1. Представление изображения в интегральной форме

Для того, чтобы быстро вычислить признаки Хаара необходимо представить изображение в интегральной форме

3.2.1.1. Общая схема алгоритма

Данная схема алгоритма (**Рис.** ***16***) позволяет представить как структурно будет выглядеть функция, реализующая преобразование.

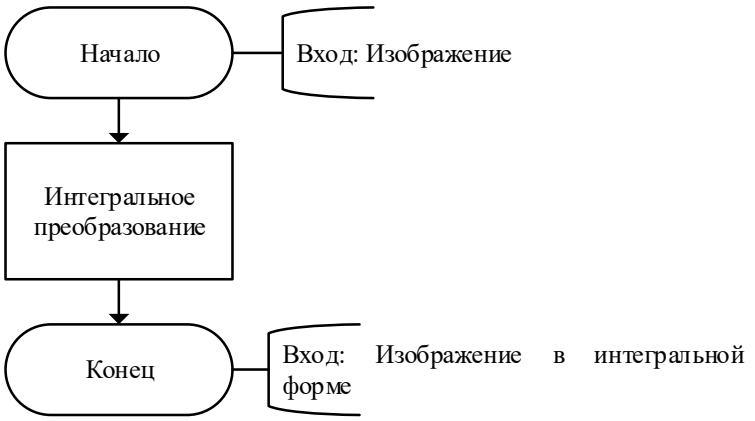


Рис. 16 Общая схема алгоритма интегрального преобразования

3.2.1.2. Схема алгоритма интегрального преобразования

В результате выполнения алгоритма (**Рис.** ***17***) изображение преобразуется в интегральную форму.

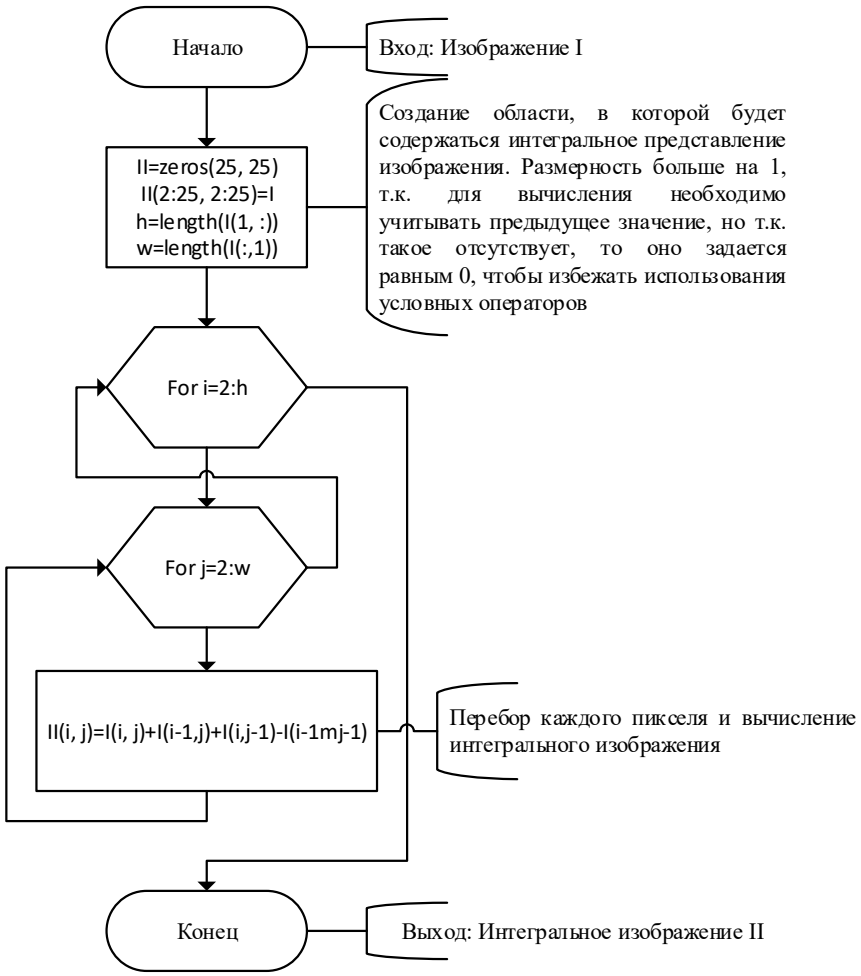


Рис. 17 Подробная схема алгоритма интегрального преобразования

Таким образом, с помощью структурных схем можно легко реализовать алгоритм преобразования изображения в интегральную форму, для которой вычисление признаков Хаара будет намного быстрее.

3.2.2. Вычисление признаков Хаара

Вычисление признаков Хаара является одним из основных алгоритмов, без которого нет возможности проводить дальнейшие этапы алгоритма Виолы-Джонса. Для этого можно воспользоваться ранее вычисленным интегральным изображением для более быстрого вычисления.

3.2.2.1. Общая схема алгоритма

В результате работы алгоритма(**Рис. *18***) будет получена структура(**Рис. *19***), которая содержит все 162336 признаков, которые разделены на группы, в соответствии с их типом(**Рис. *5***).

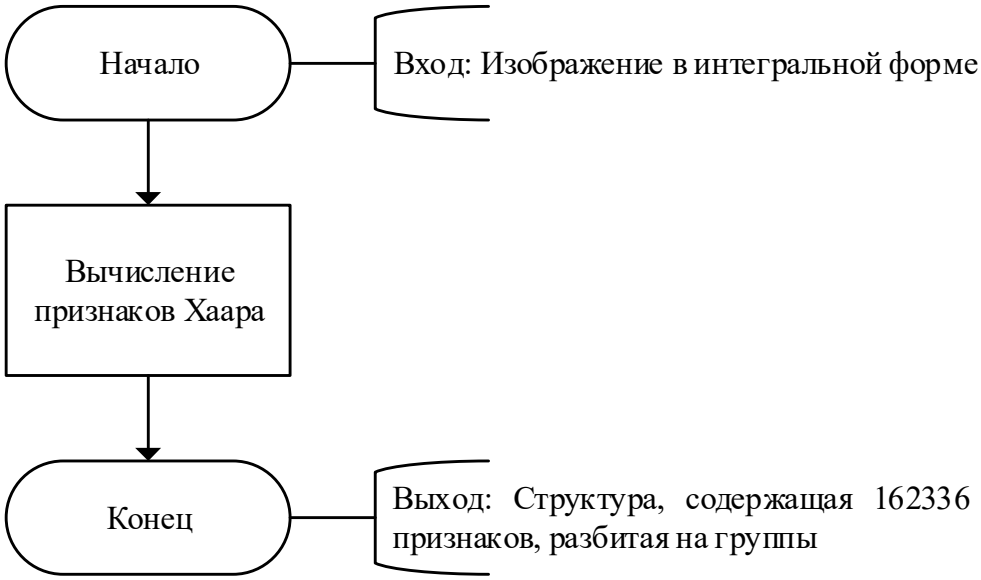


Рис. 18 Общая схема алгоритма вычисления признаков Хаара

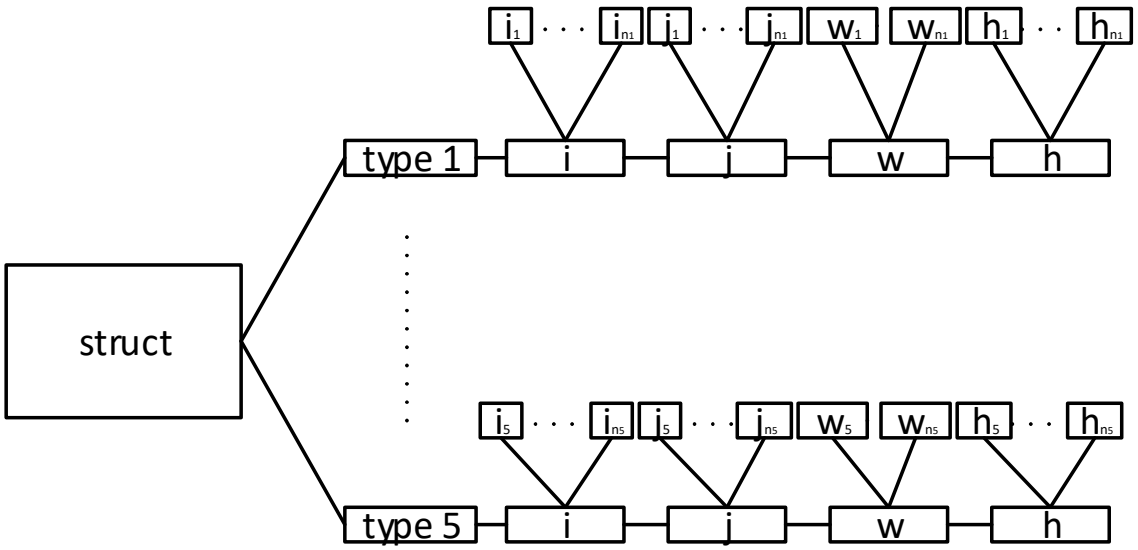


Рис. 19 Структура выходных данных алгоритма

Элементы структуры:

- $type_i$ – тип признака Хаара
- i – координата x признака на изображении
- j – координата y признака на изображении
- w – ширина признака в пикселях
- h – высота признака в пикселях

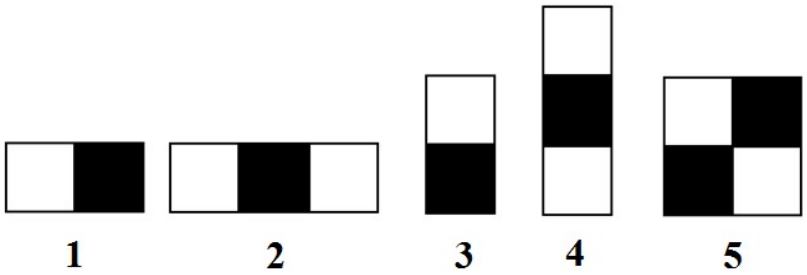


Рис. 20 Признаки Хаара

3.2.2.2. Схема алгоритма вычисления признаков Хаара

Каждый признак вычисляется «грубым» перебором, при этом меняется его координаты и размер. В зависимости от типа признака, их количество определяется следующими значениями: size(f_i)={43200, 27600, 43200, 27600, 20736}, где size(f_i) – размер каждого из признаков(**Рис. *20***).

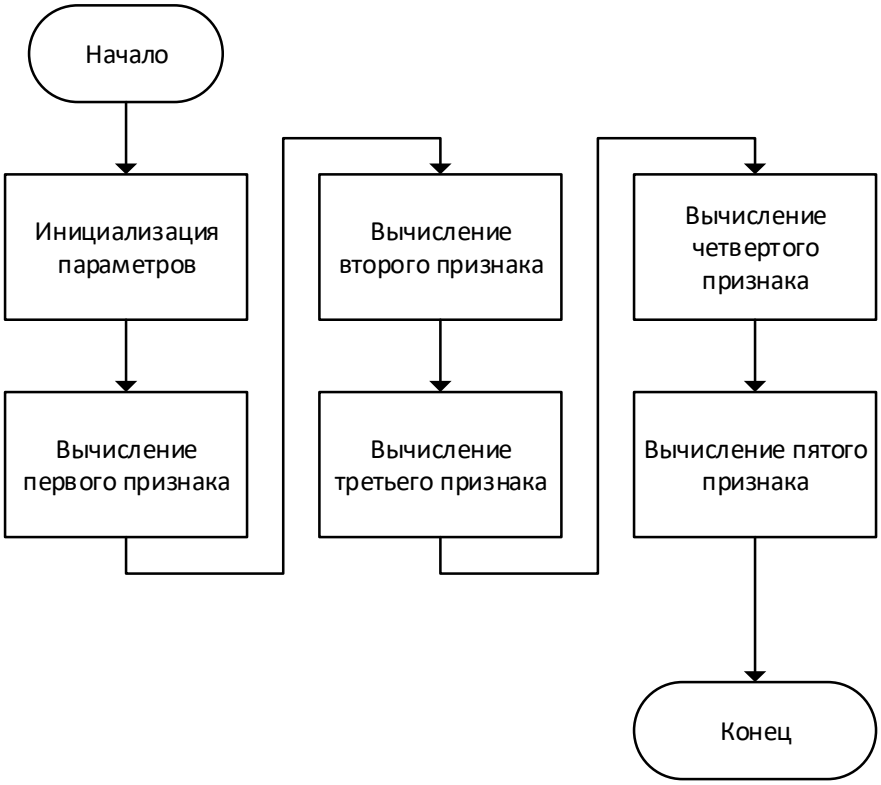


Рис. 21 Схема вычисления признаков Хаара

Вычисление каждого признака индивидуально и зависит только от его формы. Например, первый признак имеет удвоенную ширину, поскольку вертикальная линия разделяет признак на две области.

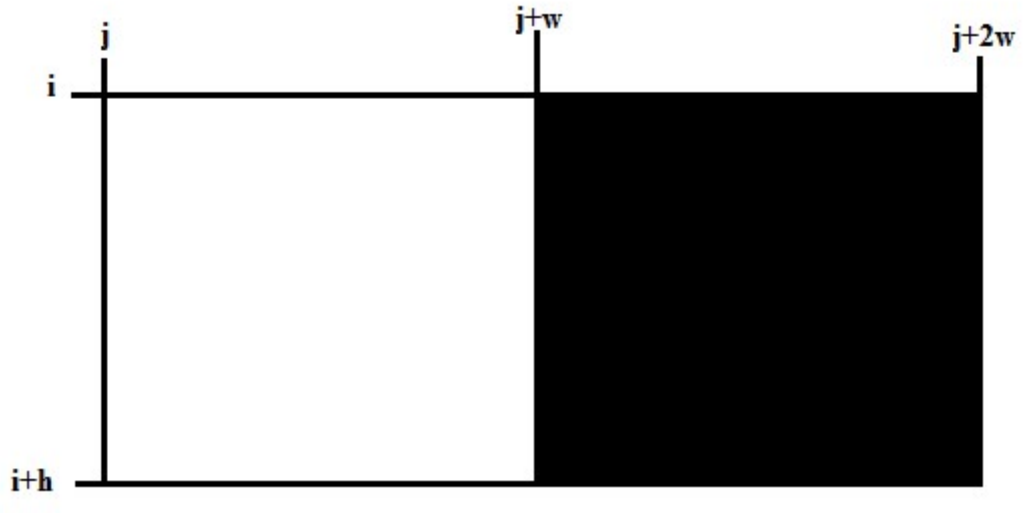


Рис. 22 Наглядный пример принципа вычисления признака

Количество численных значений для каждого признака достаточно велико, поскольку является комбинацией всех возможных значений i, j, w, h в пределах размера изображения.

На вход алгоритма (**Рис.** ***23***) подается изображение в интегральном виде. В процессе работы выполняется четыре цикла, которые перебирают всевозможные варианты признаков и вычисляют их значение. В данном алгоритме:

- S1, S2 – сумма пикселей левой полуплоскость и правой полуплоскости соответственно
- a – разница между полуплоскостями S1 и S2

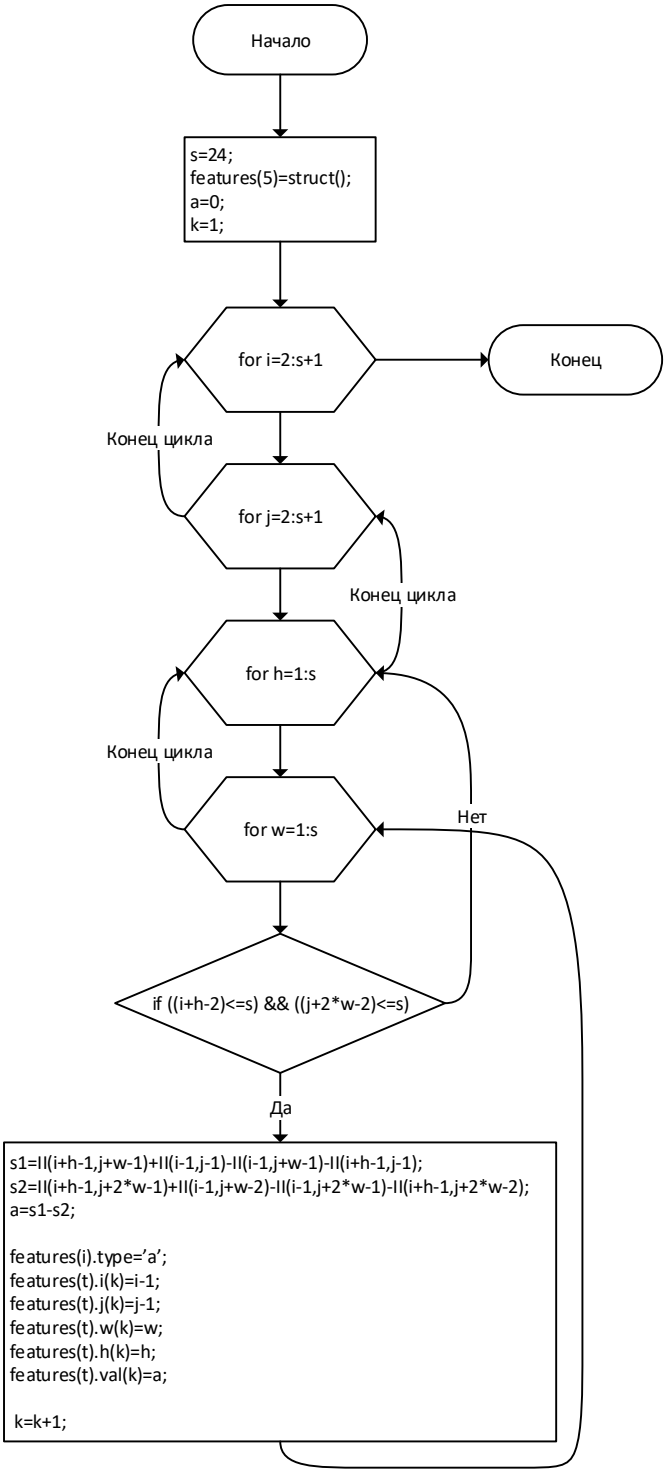


Рис. 23 Алгоритм вычисление одного из признаков

3.2.3. AdaBoost

Для того чтобы обучить сильный классификатор необходимо выполнить алгоритм AdaBoost, на вход которого подается вектор, содержащий значение одного из 162336 признаков для выборки тренировочных изображений.

3.2.3.1. Общая схема алгоритма обучения

На вход алгоритма подается вектор признаков и количество итераций для обучения. Результатом работы является сильный классификатор, который наилучшим образом классифицирует входной вектор, т.е. количество неправильно классифицированных элементов очень мало. Для этого необходимо выполнить достаточное количество итераций.

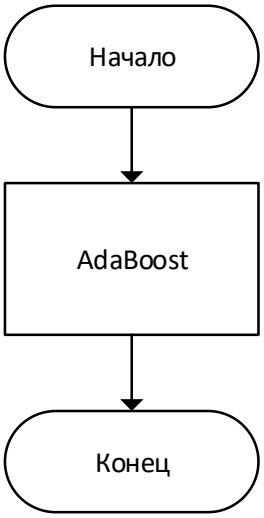


Рис. 24 Общая схема обучения

3.2.3.2. Схема алгоритма AdaBoost

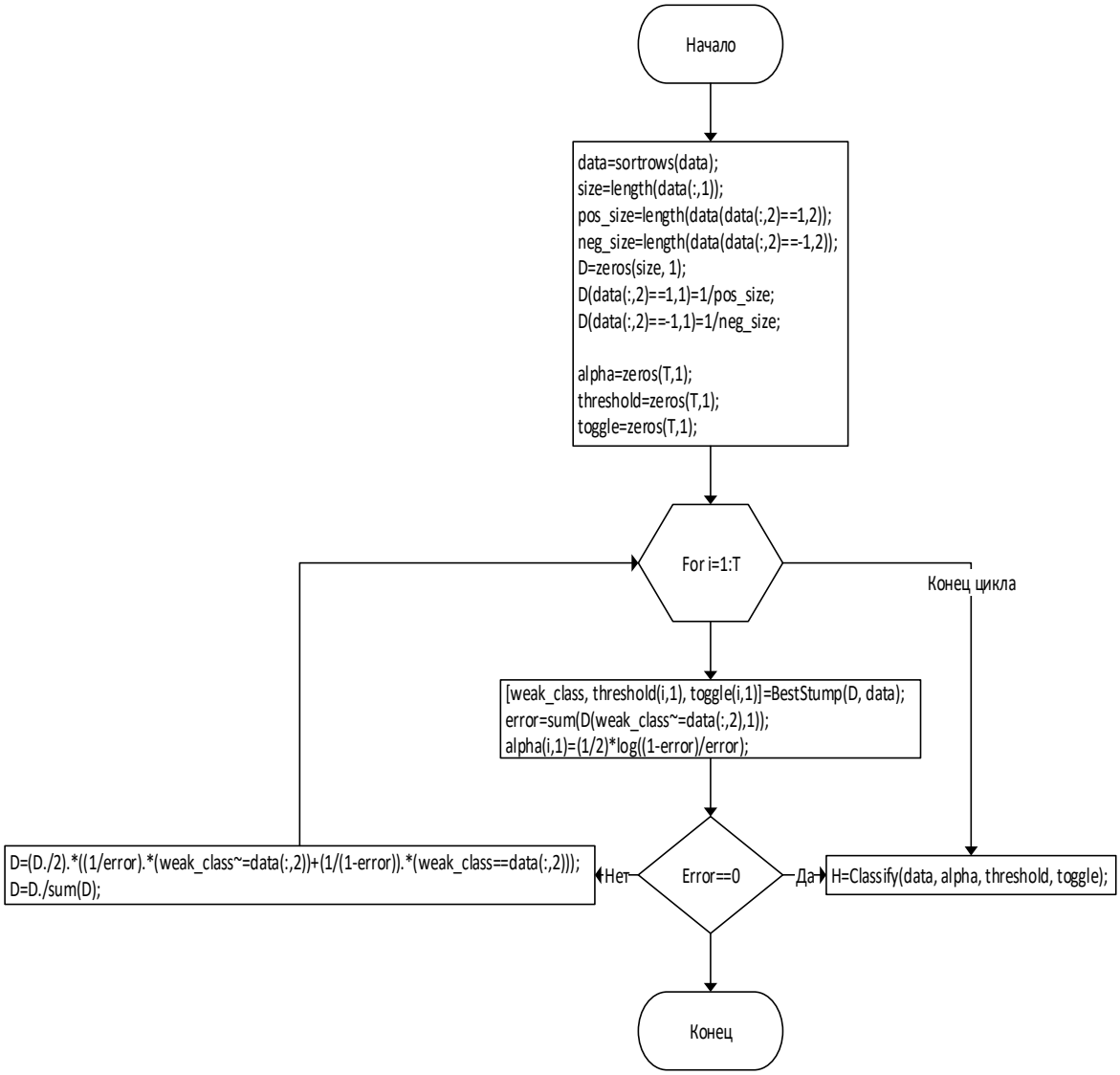


Рис. 25 Схема алгоритма AdaBoost

На этапе выполнения цикла вычисляется слабый классификатор (weak_class), его пороговое значение (threshold), параметр (toggle), отвечающий за знак неравенства, относительно которого происходит поиск признаков, значения которых больше порогового, с помощью функции BestStump, на вход

которой подаются вектор признаков (data) и вероятности возникновения каждого признака(D). Также вычисляются ошибка классификации (error) как сумма вероятностей неправильно классифицированных элементов и весовые коэффициенты (alpha), на основе ошибок классификации. В случае, если все объекты классифицированы правильно, то алгоритм завершается и возвращается сильный классификатор, вычисленный с помощью функции Classify, на вход которой подается вектор весовых коэффициентов, пороговых значений и знаков паритета, вычисленных на каждой итерации.

3.2.3.3. Схема алгоритма BestStump

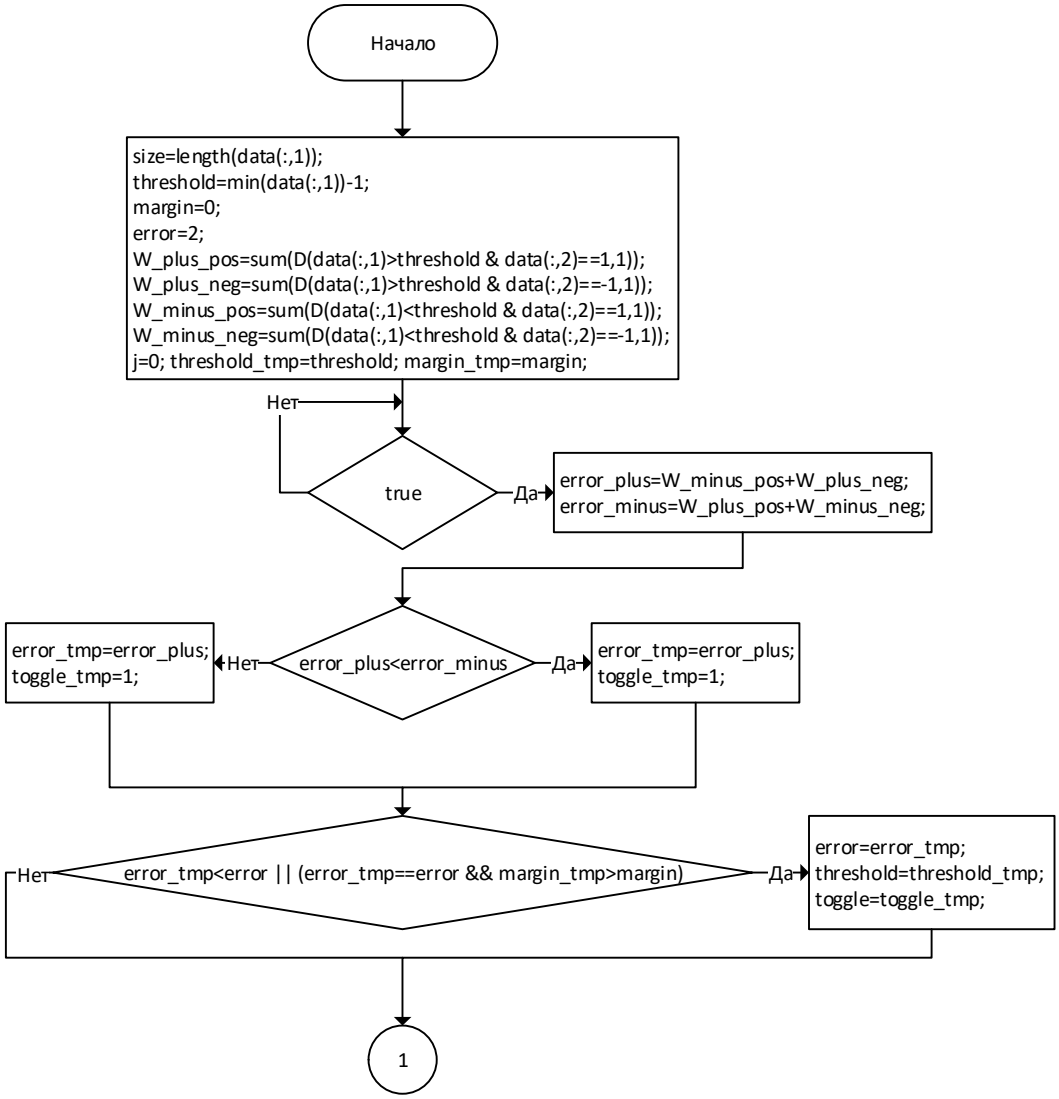


Рис. 26 *(а)*

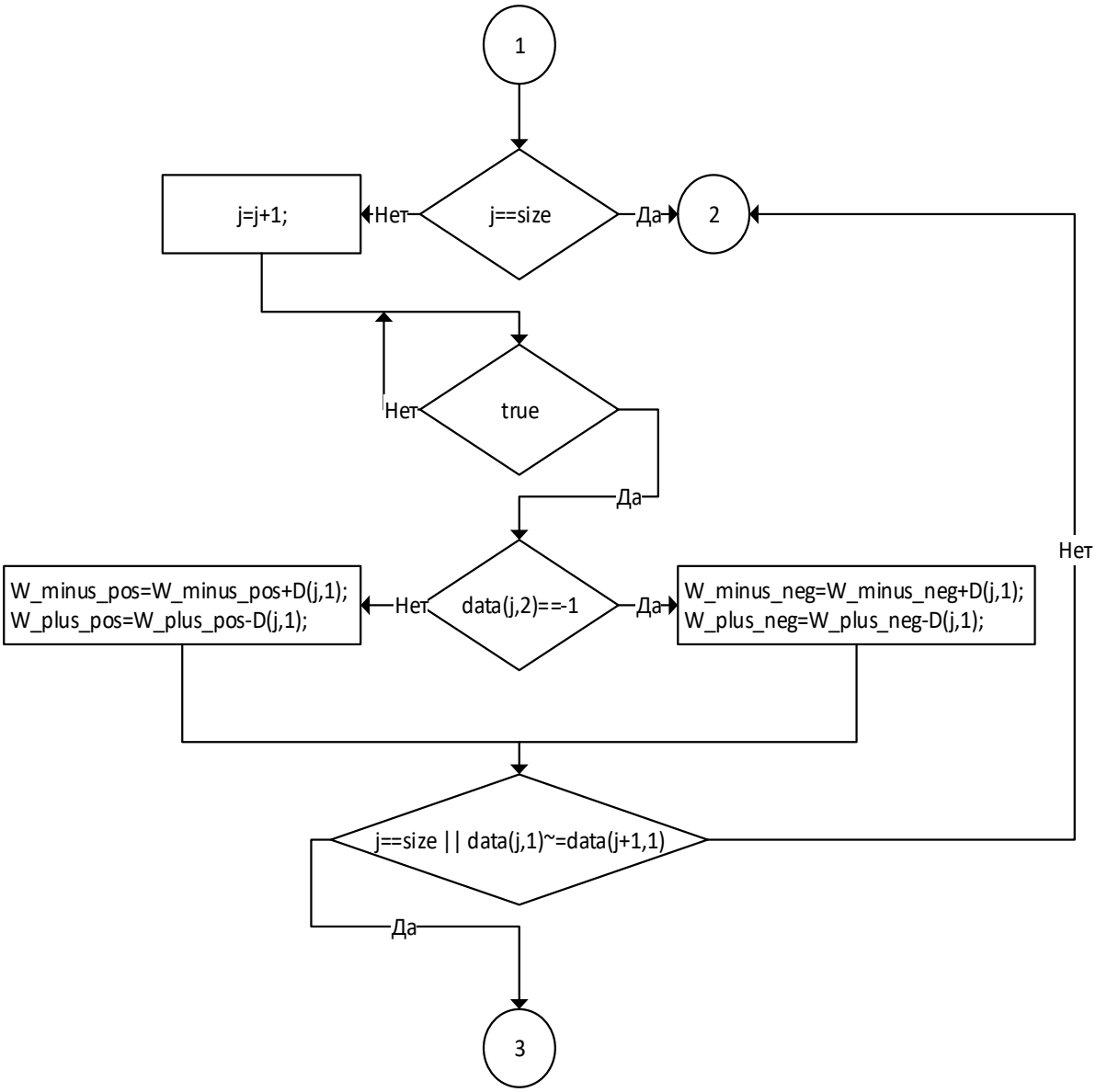

1
j==size
Нет
Да
2
j=j+1;
Нет
true
Да
Нет
W_minus_pos=W_minus_pos+D(j,1);
W_plus_pos=W_plus_pos-D(j,1);
Нет
data(j,2)==-1
Да
W_minus_neg=W_minus_neg+D(j,1);
W_plus_neg=W_plus_neg-D(j,1);
j==size || data(j,1)~=data(j+1,1)
Да
3

Рис. 27 (b)

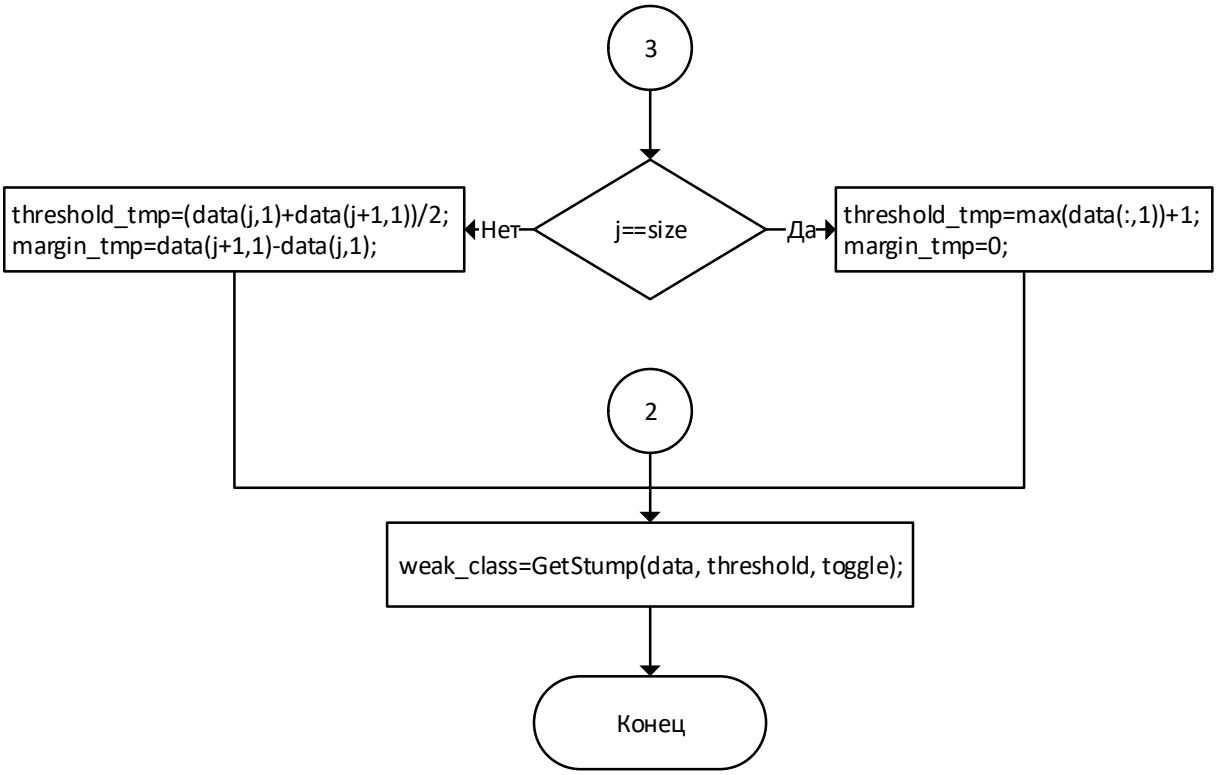


Рис. 28 (c)

Результатом работы алгоритма является слабый классификатор(weak_class), который является результатом функции GetStump, пороговое значение(threshold) и знак(toggle) определяющий, с какой стороны находятся неправильно классифицированные значения.

В ходе работы алгоритма инициализируются необходимые значения:

- Threshold задается как минимальное значение из всех значений признаков уменьшенное на единицу.
- Margin – расстояние между соседними признаками, между которыми находится значение threshold

- Error – значение ошибки классификации
- W_plus_pos – сумма весов признаков, принадлежащих классу 1, которые больше заданного порога
- W_plus_neg - сумма весов признаков, принадлежащих классу -1, которые больше заданного порога
- W_minus_pos - сумма весов признаков, принадлежащих классу 1, которые меньше заданного порога
- W_minus_neg - сумма весов признаков, принадлежащих классу -1, которые меньше заданного порога

Error_plus и Error_minus позволяют определить ошибку классификации и знак, определяющий направление знака неравенства.

В конце получаем два значение признака между которыми будет лежать пороговое значение, определяющееся как среднее арифметическое между двумя признаками.

3.2.3.4. Алгоритм функции GetStump

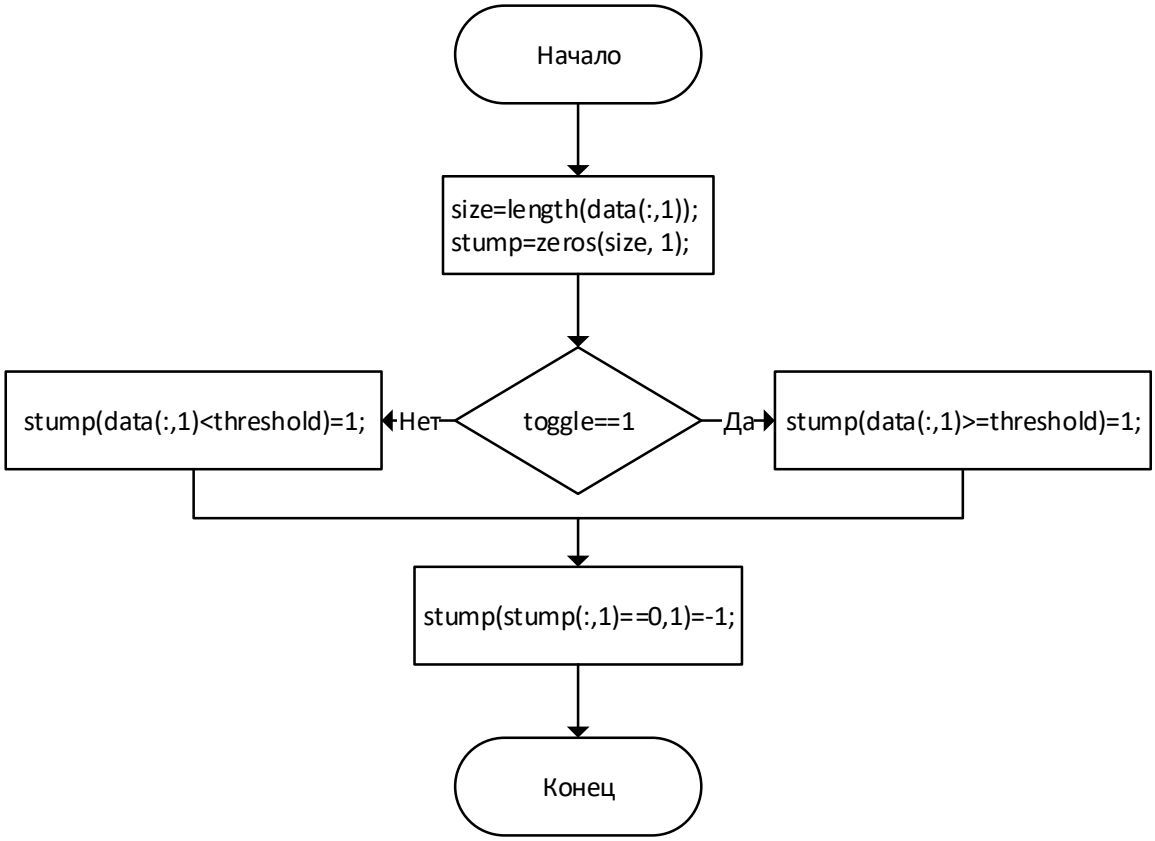


Рис. 29 Схема алгоритма GetStump

Функция GetStump формирует вектор 1 и -1 описывающих принадлежность значения признака к тому или иному классу.

3.2.3.5. Алгоритм Classify

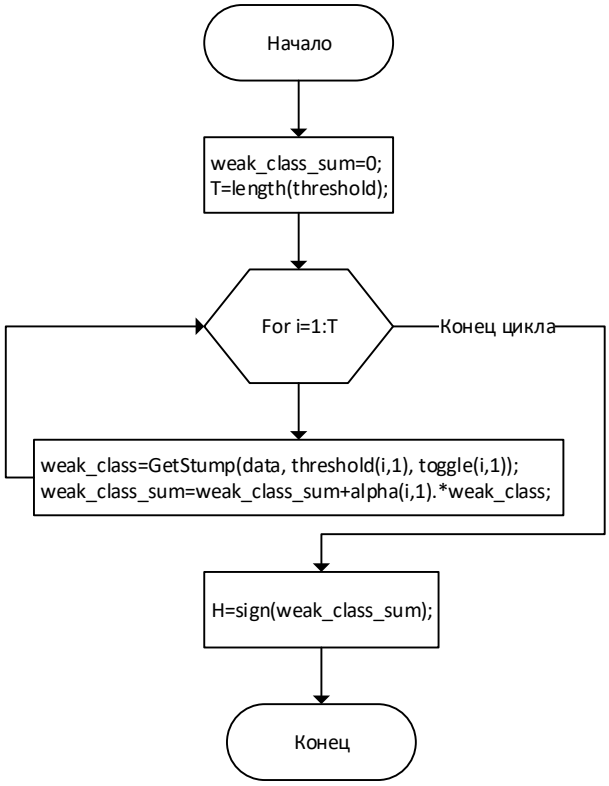


Рис. 30 Схема алгоритма Classify

Результатом работы алгоритма является сильный классификатор, определенный по формуле: $H(x) = sign(\sum_{t=1}^{T} \alpha_t \, h_t(x))$

3.2.4. Выбор лучших признаков

Для того, чтобы ускорить процесс классификации изображения необходимо разбить полученные признаки на отдельные группы, которые будут классифицировать выбранную область. Сам алгоритм построения каскада классификаторов достаточно ресурсоемкий и на обычной машине будет выполнятся очень долго. Для примера, на восьми ядерном процессоре и оперативной памяти 48 гигабайт обучение выборки из 3451 объектов составило примерно 24 часа. [20] Поэтому было принято решение выбрать лучшие признаки, которые имеют хорошие показатели классификации для тренировочной выборки, т.е. доля правильно классифицированных элементов больше 0.95. После этого было произведено разбиение признаков на группы, состоящие из первых 10, следующих 20 и так далее до конца. В результате такого подхода классификация выполняется быстро, поскольку, в случае перевешивания «голосов» в сторону не лица, процесс классификации прекращается и сканирующее окно переходит к следующему изображению.

3.2.5. Сканирующее окно

Для испытания алгоритма в «полевых условиях» было использовано сканирующее окно, имеющее стандартный размер 24x24 и шаг в один пиксель. К каждому окну применяются все необходимые фильтры, как нормализация и интегральное преобразование, затем происходит сжатие изображения к стандартному размеру и классификация. После сканирования всего изображения размер окна увеличивается в 1.25 раз и процесс начинается заново. Это продолжается пока размер окна не станет больше изображения.

4. ТЕСТИРОВАНИЕ

Для того чтобы определить насколько хорошо происходит детектирование лиц необходимо провести необходимые результаты, такие как определение ошибок первого и второго рода. Для тренировки использовалось 960 изображений, 320 из которых – изображения лиц, 640 – изображения, не содержащие лица. В работе были использованы образцы лиц из базы данных «AT&T», которая содержит 40 людей, изображения которых содержат по 10 различных вариаций поворота головы и эмоций. Для тестирования были выбраны случайным образом по 2 изображения из каждого набора, таким образом в тестируемой выборке оказалось 80 изображений людей, которых нет в тренировочной выборке и изображений не лиц - 160.

Для непосредственного детектирования была использована полная библиотека лиц из 400 изображений. Образцов, не содержащих лиц – 800.

Поскольку алгоритм обучения AdaBoost направлен на формирование наилучшего разделения данных на два класса, то увеличением количества операций можно добиться лучшего результата. Также улучшения можно добиться путем увеличения количества элементов классов. [20]

4.1. True positive rate

True positive rate – отношение правильно классифицированных элементов к количеству элементов данного класса, в данном случае отношение количества правильно детектированных лиц к общему количеству лиц. [24]

$$P(H(X) = 1|Y = 1)$$

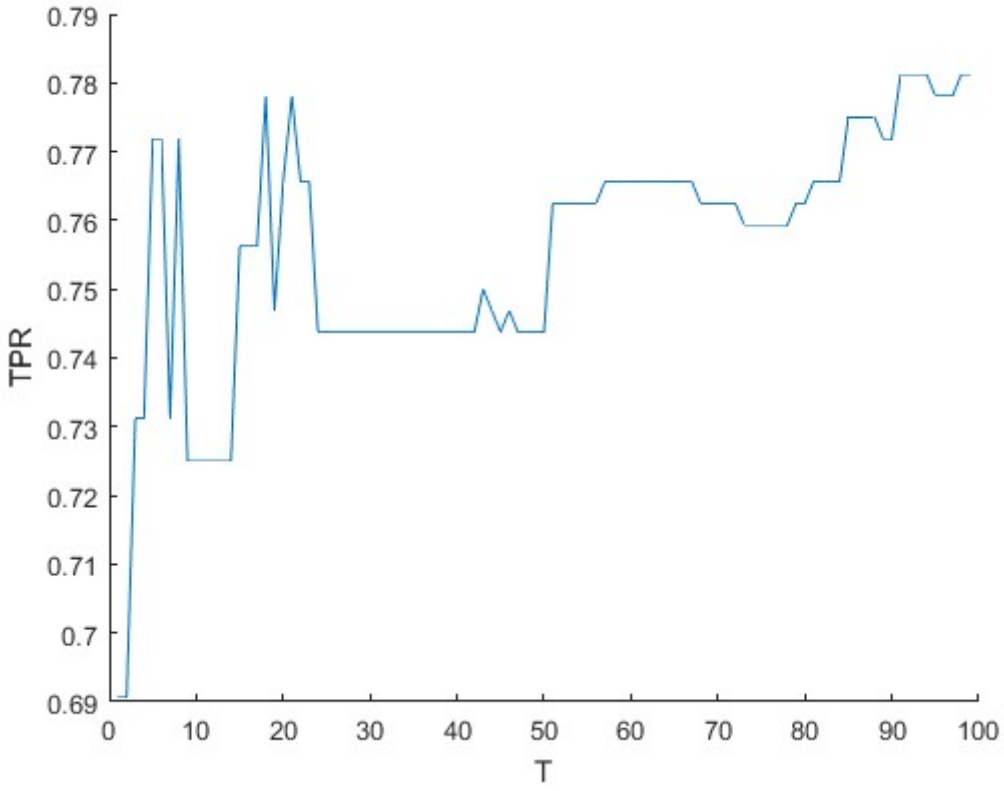


Рис. 31 Зависимость TRP от количества итераций AdaBoost

T – количество итераций алгоритма AdaBoost

4.2.False positive rate

False positive rate – отношение количества ошибочно классифицированных элементов к количеству элементов данного класса, в данном случае отношение количества лиц, которые не были детектированы, к общему количеству лиц. [24]

$$P(H(X) = 1|Y = -1)$$

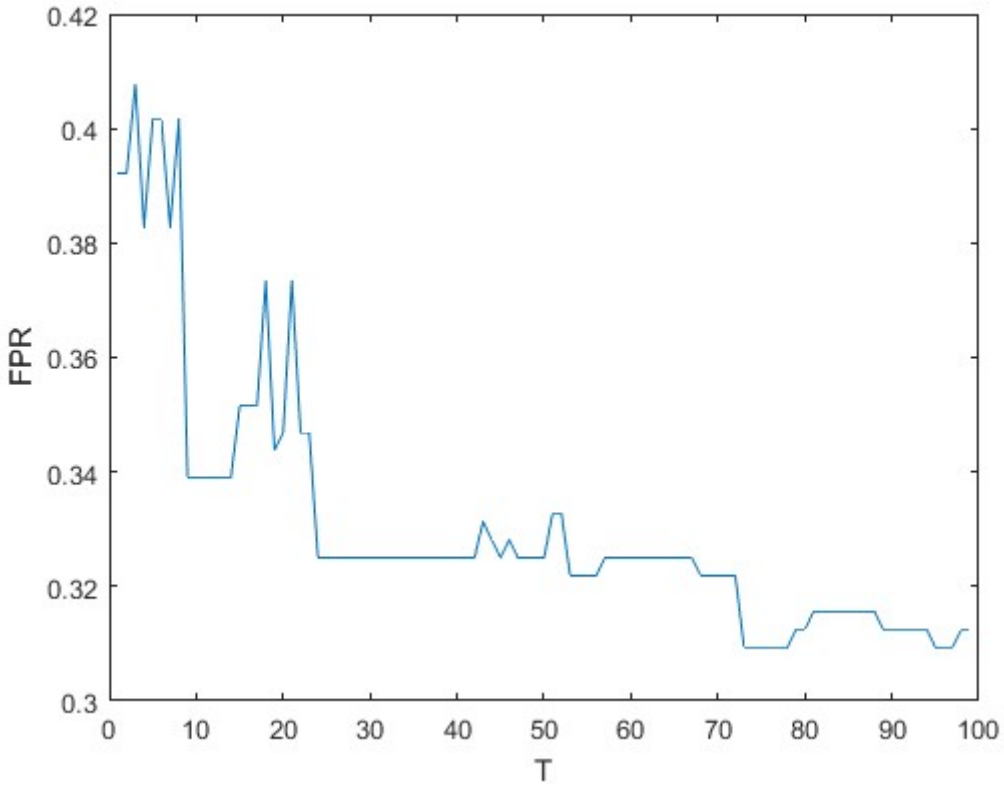


Рис. 32 Зависимость TRP от количества итераций AdaBoost

T – количество итераций алгоритма AdaBoost

Таким образом, судя по графикам можно сделать вывод, что алгоритм

работает правильно и соответствует примерам, приведенным в технической литературе(**Рис.** ***33***)[20]

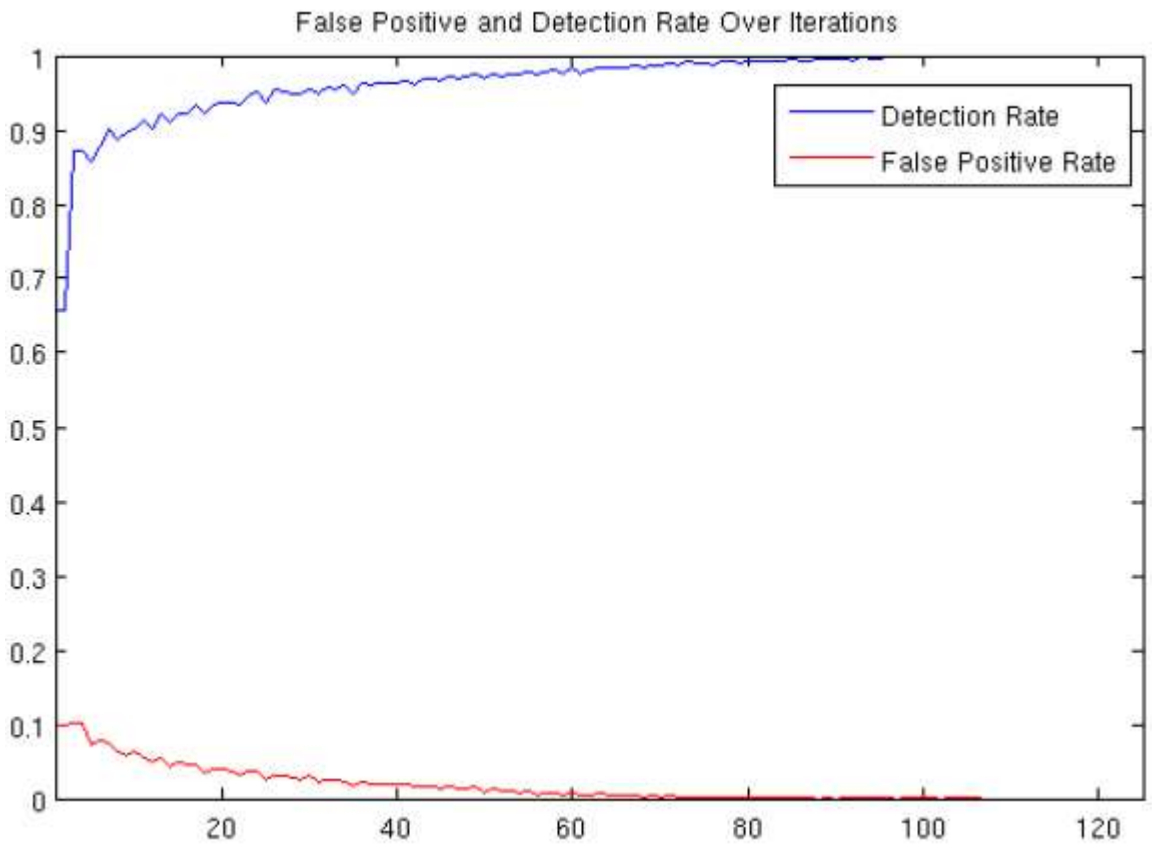


Рис. 33 График зависимости TPR(синий) и FPR(красный) от числа итераций

Результаты, приведенный на рисунке **Рис.** ***33*** лучше, чем на рисунке **Рис.** ***34***, поскольку для тренировки были использованы 1000 изображений лиц и 2451 изображений без лиц.[20]

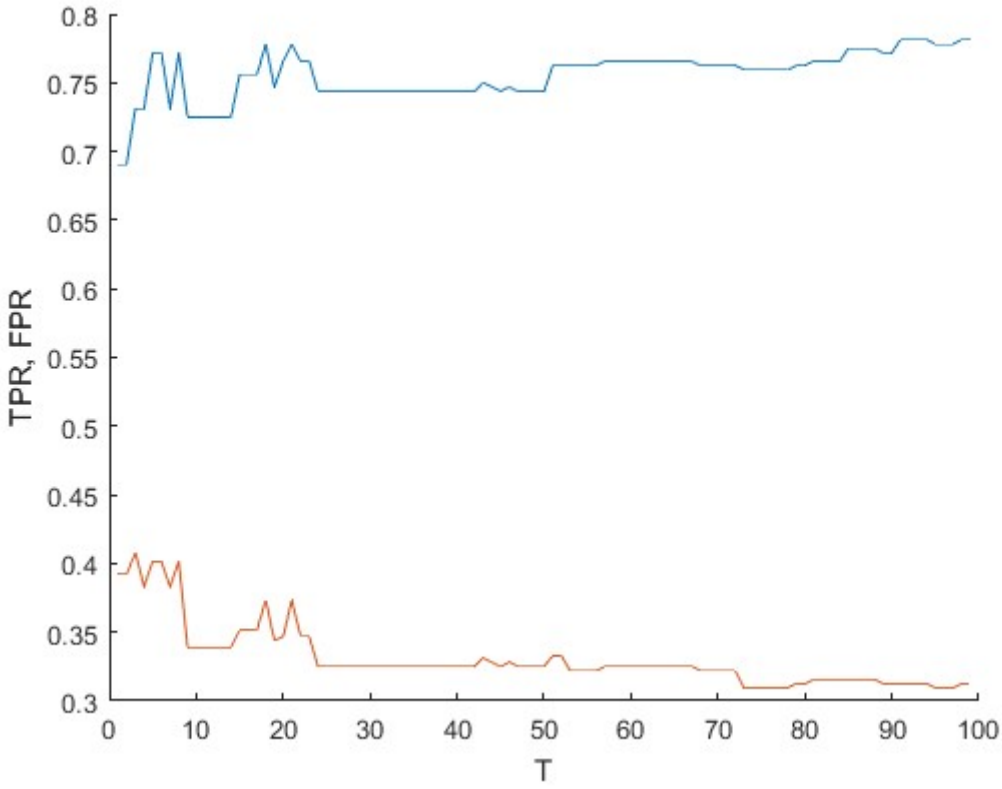


Рис. 34 Экспериментальные значения TPR(синий) и FPR(красный)

4.3.ROC – кривая

ROC-кривая — график, позволяющий оценить качество бинарной классификации, отображает соотношение между TPR и FPR. Количественную интерпретацию ROC даёт показатель AUC (англ. area under ROC curve, площадь под ROC-кривой) — площадь, ограниченная ROC-кривой и осью доли ложных положительных классификаций. Чем выше показатель AUC, тем качественнее классификатор. [24]

Для вычисления показателя AUC была использована функция perfcurve в пакете MatLab. [25] Для данного примера: AUC= 0.7344.

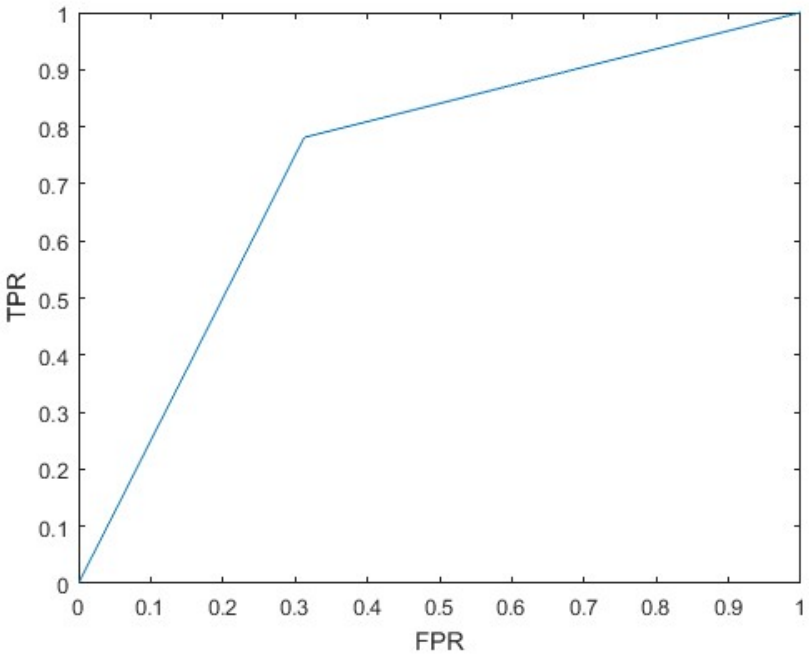


Рис. 35 ROC – кривая

4.4.Проверка тестовых изображений

Для того, чтобы дать примерную оценку качества детектирования, необходимо определить долю правильно классифицированных изображений. Для данного примера использовалось 1500 итераций алгоритма AdaBoost. Также для сравнения качества детектирования были использованы другие алгоритмы классификации, такие как Coarse Gaussian и Quadratic Disctiminant, с использованием средства классификации MatLab – Classification Learner(**Таблица *1***). По результатам сравнения можно судить, что лучшим среди представленных алгоритмов является AdaBoost, поскольку он имеет наилучший результат при детектировании позитивных образцов, т.е. лиц.

Таблица 1 Сравнение алгоритмов классификации

		Позитивных образцов	Негативных образцов
Размер тренировочной выборки		320	640
Размер тестовой выборки		80	160
AdaBoost	Правильно классифицировано	75	152
Coarse Gaussian		67	156
Quadratic Discriminant		68	154

Тестирование алгоритмов Coarse Gaussian и Quadratic Discriminant проводилось с использованием признаков, отобранных с помощью алгоритма AdaBoost.

4.5.Проверка работы алгоритма детектирования

Для того, чтобы убедится, что алгоритм справляется с поставленной задачей, был проведен опыт на изображении, содержащем несколько лиц.

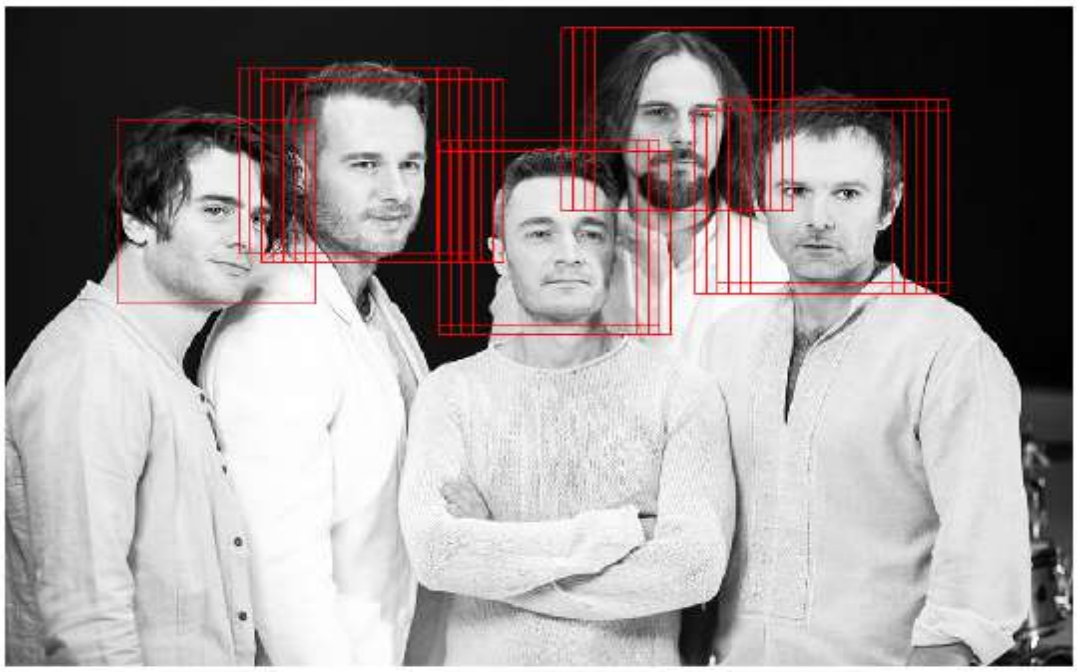


Рис. 36 Результат работы алгоритма на тестовом изображении

Таким образом можно сделать вывод, что для отбора лучших признаков Хаара необходимо провести достаточное количество итераций алгоритма AdaBoost для достижения высокого значения AUC, а также использовать большую выборку для тренировки. Однако эти две проблемы упираются в вычислительную мощность машины, на которой проводится обучение, и ограниченность памяти.

П.1. ПРИЛОЖЕНИЕ

П.1.1. Код MATLAB

П.1.1.2. Код заполнения структуры изображений

```
clear all;
clc;

% pos_num и neg_num - заранее известное количество позитивных
и негативных
% образцов
pos_num=80;
neg_num=160;
% num - общее количество тренировочной выборки
num=pos_num+neg_num;
k=1;
% test_var - структура, содержащая изображения
test_var=[];
test_var.i(num)=struct();

% Для ускорения работы необходимо заранее выделить необходимую
память
% Исходный размер изображения - матрица 25x25
for i=1:num
    test_var.i(i).img=zeros(25, 25);
end

txt='';

% Заполнение структуры изображениями лиц
tic
cd test_pos
files = dir('*.pgm');
for id = 1:pos_num
    if id>pos_num
        break;
    end
    txt=int2str(id);
    txt=strcat(txt,'.pgm');
% Изображение представлено в виде double, таким образом каждый
пиксель
% лежит не в диапазоне [0; 255], а в диапазоне [0; 1]
    I=im2double(imread(txt));
% Для нормализации изображения используется функци zscore
    I=zscore(I, 1);
% Преобразование изображения в интегральную форму с помощью
integralImage
```

```
    II=integralImage(I);
% Размер изображения должен соответствовать стандартному
размеру 25x25
    II=imresize(II, [25 25]);
    test_var.i(k).img=II;
    k=k+1;
end
cd ..\
toc

% Заполнение структуры изображениями не лиц
tic
cd test_neg
files = dir('*.jpg');
for id = 1:neg_num
    if id>neg_num
        break;
    end
    txt=int2str(id);
    txt=strcat(txt,'.jpg');
    I=im2double(imread(txt));
    I=zscore(I, 1);
    II=integralImage(I);
    II=imresize(II, [25 25]);
    test_var.i(k).img=II;
    k=k+1;
end
cd ..\
toc
cd Data
save('TestVarATT', 'test_var');
cd ..\
```

П.1.1.3. Код заполнения структуры признаков

```
clear all;
clc;

% Для формирования структуры признаков необходимо использовать
структуру
% изображений, использованных в GetImageStruct.m
cd Data
load ('TestVarATT', 'test_var');
cd ..\

% test_features - структура признаков для каждого изображения
n=length(test_var.i);
test_features(n)=struct();
tic
for i=1:n
```

```
% Результатом GetFeaturesValues является структура признаков
для одного
% изображения
    pfx=GetFeaturesValues(test_var.i(i).img);
    test_features(i).img=pfx;
end
toc

cd Data
save ('TestFeaturesATT', 'test_features', '-v7.3');
cd ..\
```

П.1.1.4. Код компоновки признаков

```
clear all;
clc;

% Для выполнения алгоритма нужна структура признаков,
полученная с помощью
% GetFeatureStruct.m
tic
cd Data
load ('TestFeaturesATT', 'test_features');
cd ..\
toc

% Количество признаков,feature_num, заранее известно.
% Также известно количество комбинаций каждого из признаков,
feature_sizes.
feature_num=162336;
test_var_num=length(test_features);
feature_sizes=[43200 27600 43200 27600 20736];

% для ускорения работы необходимо заранее выделить требуемый
размер массива
% признаков, px_test.
px_test=zeros(test_var_num, feature_num);
tic
for i=1:test_var_num
    tmp=1;
    for j=1:length(feature_sizes)
        px_test(i,tmp:tmp+feature_sizes(j)-
1)=test_features(i).img(j).val;
        tmp=tmp+feature_sizes(j);
    end
end
toc
% Результатом работы алгоритма является массив размером:
% (количество изображений)x(количество признаков)
% Это необходимо для увеличения скорости обучения, поскольку
доступ к
```

```
% массиву намного быстрее, чем доступ к полям структуры
cd Data
save ('CombineTestFeatures', 'px_test', '-v7.3');
cd ..\
```

П.1.1.5. Вычисление признаков для одного изображения

```
function d=GetFeaturesValues(II)
s=24;
% features - структура всех комбинаций каждого признака,
разделенных по
% типу: a, b, c, d, e
% Для ускорения работы алгоритма неоюходимо заранее выделить
необходимое
% количество памяти l
features(5)=struct();
tp=['a' 'b' 'c' 'd' 'e'];
l=[43200 27600 43200 27600 20736];

for i=1:5
    features(i).type=tp(i);
    features(i).i=zeros(1,l(i));
    features(i).j=zeros(1,l(i));
    features(i).w=zeros(1,l(i));
    features(i).h=zeros(1,l(i));
    features(i).val=zeros(1,l(i));
end

a=0;
k=1;
t=1;
% Вычисление признака типа a
for i=2:s+1
    for j=2:s+1
        for h=1:s
            for w=1:s
                if ((i+h-2)<=s) && ((j+2*w-2)<=s)
                    s1=II(i+h-1,j+w-1)+II(i-1,j-1)-II(i-1,j+w-
1)-II(i+h-1,j-1);
                    s2=II(i+h-1,j+2*w-1)+II(i-1,j+w-2)-II(i-
1,j+2*w-1)-II(i+h-1,j+2*w-2);
                    a=s1-s2;

                    features(t).i(k)=i-1;
                    features(t).j(k)=j-1;
                    features(t).w(k)=w;
                    features(t).h(k)=h;
                    features(t).val(k)=a;

                    k=k+1;
                else
```

```
                    break;
                end
            end
        end
    end
end

k=1;
t=t+1;
% Вычисление признака типа b
for i=2:s+1
    for j=2:s+1
        for w=1:s
            for h=1:s
                if i+h-2<=s && j+3*w-2 <=s
                    s1=II(i+h-1,j+w-1)+II(i-1,j-1)-II(i-1,j+w-
1)-II(i+h-1,j-1);
                    s2=II(i+h-1,j+2*w-1)+II(i-1,j+w-2)-II(i-
1,j+2*w-1)-II(i+h-1,j+w-2);
                    s3=II(i+h-1,j+3*w-1)+II(i-1,j+2*w-2)-II(i-
1,j+3*w-1)-II(i+h-1,j+2*w-2);
                    a=s1-s2+s3;

                    features(t).i(k)=i-1;
                    features(t).j(k)=j-1;
                    features(t).w(k)=w;
                    features(t).h(k)=h;
                    features(t).val(k)=a;

                    k=k+1;
                else
                    break;
                end
            end
        end
    end
end

k=1;
t=t+1;
% Вычисление признака типа c
for i=2:s+1
    for j=2:s+1
        for w=1:s
            for h=1:s
                if i+2*h-2<=s && j+w-2 <=s
                    s1=II(i+h-1,j+w-1)+II(i-1,j-1)-II(i-1,j+w-
1)-II(i+h-1,j-1);
                    s2=II(i+2*h-1,j+w-1)+II(i+h-2,j-1)-II(i+h-
2,j+w-1)-II(i+2*h-1,j-1);
```

```
                    a=s1-s2;

                    features(t).i(k)=i-1;
                    features(t).j(k)=j-1;
                    features(t).w(k)=w;
                    features(t).h(k)=h;
                    features(t).val(k)=a;

                    k=k+1;
                else
                    break;
                end
            end
        end
    end
end

k=1;
t=t+1;
% Вычисление признака типа d
for i=2:s+1
    for j=2:s+1
        for w=1:s
            for h=1:s
                if i+3*h-2<=s && j+w-2 <=s
                    s1=II(i+h-1,j+w-1)+II(i-1,j-1)-II(i-1,j+w-
1)-II(i+h-1,j-1);
                    s2=II(i+2*h-1,j+w-1)+II(i+h-2,j-1)-II(i+h-
2,j+w-1)-II(i+2*h-1,j-1);
                    s3=II(i+3*h-1,j+w-1)+II(i+2*h-2,j-1)-
II(i+2*h-2,j+w-1)-II(i+3*h-1,j-1);
                    a=s1-s2+s3;

                    features(t).i(k)=i-1;
                    features(t).j(k)=j-1;
                    features(t).w(k)=w;
                    features(t).h(k)=h;
                    features(t).val(k)=a;

                    k=k+1;
                else
                    break;
                end
            end
        end
    end
end

k=1;
t=t+1;
```

```
% Вычисление признака типа e
for i=2:s+1
    for j=2:s+1
        for w=1:s
            for h=1:s
                if i+2*h-2<=s && j+2*w-2 <=s
                    s1=II(i+h-1,j+w-1)+II(i-1,j-1)-II(i-1,j+w-
1)-II(i+h-1,j-1);
                    s2=II(i+2*h-1,j+w-1)+II(i+h-2,j-1)-II(i+h-
2,j+w-1)-II(i+2*h-1,j-1);
                    s3=II(i+h-1,j+2*w-1)+II(i-1,j+w-1)-II(i-
1,j+2*w-1)-II(i+h-1,j-1);
                    s4=II(i+2*h-1,j+2*w-1)+II(i+h-2,j+w-2)-
II(i+h-2,j+2*w-1)-II(i+2*h-1,j+w-2);
                    a=s1-s2-s3+s4;

                    features(t).i(k)=i-1;
                    features(t).j(k)=j-1;
                    features(t).w(k)=w;
                    features(t).h(k)=h;
                    features(t).val(k)=a;

                    k=k+1;
                else
                    break;
                end
            end
        end
    end
end
d=features;
end
```

II.1.1.6. AdaBoost

```
function [H, alpha, threshold, toggle]=AdaBoost(data, T)

data=sortrows(data);
size=length(data(:,1));
pos_size=length(data(data(:,2)==1,2));
neg_size=length(data(data(:,2)==-1,2));
D=zeros(size, 1);
D(data(:,2)==1,1)=1/pos_size;
D(data(:,2)==-1,1)=1/neg_size;

alpha=zeros(T,1);
threshold=zeros(T,1);
toggle=zeros(T,1);


for i=1:T
    [weak_class, threshold(i,1), toggle(i,1)]=BestStump(D,
data);

    error=sum(D(weak_class~=data(:,2),1));

    alpha(i,1)=(1/2)*log((1-error)/error);

    if error==0
        break;
    end

    D=(D./2).*((1/error).*(weak_class~=data(:,2))+(1/(1-
error)).*(weak_class==data(:,2)));
    D=D./sum(D);



end
H=Classify(data, alpha, threshold, toggle);

end
```

II.1.1.7. BestStump

```
function [weak_class, threshold, toggle]=BestStump(D, data)
size=length(data(:,1));
threshold=min(data(:,1))-1;
margin=0;
error=2;
W_plus_pos=sum(D(data(:,1)>threshold & data(:,2)==1,1));
W_plus_neg=sum(D(data(:,1)>threshold & data(:,2)==-1,1));
W_minus_pos=sum(D(data(:,1)<threshold & data(:,2)==1,1));
W_minus_neg=sum(D(data(:,1)<threshold & data(:,2)==-1,1));
j=0; threshold_tmp=threshold; margin_tmp=margin;
while 1
    error_plus=W_minus_pos+W_plus_neg;
    error_minus=W_plus_pos+W_minus_neg;
    if error_plus<error_minus
        error_tmp=error_plus;
        toggle_tmp=1;
    else
        error_tmp=error_minus;
        toggle_tmp=-1;
    end
    if error_tmp<error || (error_tmp==error && margin_tmp>margin)
        error=error_tmp;
        threshold=threshold_tmp;
        toggle=toggle_tmp;
    end
    if j==size
        break;
    end
    j=j+1;
    while 1
        if data(j,2)==-1
            W_minus_neg=W_minus_neg+D(j,1);
            W_plus_neg=W_plus_neg-D(j,1);
        else
            W_minus_pos=W_minus_pos+D(j,1);
            W_plus_pos=W_plus_pos-D(j,1);
        end
        if j==size || data(j,1)~=data(j+1,1)
            break;
        end
    end
    if j==size
        threshold_tmp=max(data(:,1))+1;
        margin_tmp=0;
    else
        threshold_tmp=(data(j,1)+data(j+1,1))/2;
        margin_tmp=data(j+1,1)-data(j,1);
    end
```

```
end
weak_class=GetStump(data, threshold, toggle);
end
```

П.1.1.8. GetStump

```
function stump=GetStump(data, threshold, toggle)
size=length(data(:,1));
stump=zeros(size, 1);
if toggle==1
    stump(data(:,1)>=threshold)=1;
else
    stump(data(:,1)<threshold)=1;
end
stump(stump(:,1)==0,1)=-1;
end
```

П.1.1.9. Classify

```
function H=Classify(data, alpha, threshold, toggle)
weak_class_sum=0;
T=length(threshold);
for i=1:T
    weak_class=GetStump(data, threshold(i,1), toggle(i,1));
    weak_class_sum=weak_class_sum+alpha(i,1).*weak_class;
end
H=sign(weak_class_sum);
end
```

П.1.1.10. GetRates

```
% Функция GetRates получает значения Detection positive rate,
Detection
% negative rate, False positive rate, False negative rate
function [dpr, dnr, fpr, fnr]=GetRates(data, H)
data=sortrows(data);
pos_size=length(data(data(:,2)==1,2));
neg_size=length(data(data(:,2)==-1,2));


H1=H(data(:,2)==1,1);
H1(H1(:,1)==-1,1)=0;
dpr=sum(H1)/pos_size;

H2=H(data(:,2)==-1,1);
H2(H2(:,1)==1)=0;
H2=abs(H2);
dnr=sum(H2)/neg_size;

fpr=1-dnr;

fnr=1-dpr;
end
```

СПИСОК ЛИТЕРАТУРЫ

1. Машинное зрение и цифровая обработка изображений // Журнал «СТА» («Современные технологии автоматизации») (дата обращения: 30.10.2016) (http://www.cta.ru/cms/f/435961.pdf).
2. Тимошенко Д. М. МЕТОДЫ АВТОМАТИЧЕСКОЙ ИДЕНТИФИКАЦИИ ЛИЧНОСТИ ПО ИЗОБРАЖЕНИЯМ ЛИЦ, ПОЛУЧЕННЫМ В НЕКОНТРОЛИРУЕМЫХ УСЛОВИЯХ: дис. ... канд. техн. наук: 05.13.18. - СПБ., 2014.
3. Upgrade Viola Jones // Хабрахабр (дата обращения: 19.11.2016) (https://habrahabr.ru/post/133909/).
4. Татаренков Д.А. Анализ методов обнаружения лиц на изображении. // Молодой ученый. 2015. №4 (84). С. 270.
5. Анализ существующих подходов к распознаванию лиц // Хабрахабр URL: https://habrahabr.ru/company/synesis/blog/238129/ (дата обращения: 20.12.2016).
6. Обнаружение и локализация лица на изображении // Информационно-коммуникационные технологии в образовании URL: http://ict.informika.ru/ft/002403/num2face.pdf (дата обращения: 20.12.2016).
7. Графические методы анализа данных // StatSoft URL: http://statsoft.ru/home/textbook/modules/stgraph.html (дата обращения: 20.12.2016).
8. Путятин Е. П., Гороховатский В. А., Кузьмин С. В РАСПОЗНАВАНИЕ ИЗОБРАЖЕНИЙ В ПРОСТРАНСТВЕ ИНВАРИАНТНЫХ ЛОКАЛЬНЫХ ПРИЗНАКОВ // Радиоэлектроника и информатика. 2006. №1.

9. Фильтр Gaussian Blur и передний план // PhotoGeek URL: http://www.photogeek.ru/groups/postprocessing/1199.html (дата обращения: 20.12.2016).
10. Матричные фильтры обработки изображений // Хабрахабр URL: https://habrahabr.ru/post/142818/ (дата обращения: 20.12.2016).
11. Методы и средства выделения лица человека на изображении // http://scicenter.online/ URL: http://scicenter.online/radiotehnika-elektronika-kompyuteryi/metodyi-sredstva-vyideleniya-litsa-cheloveka.html (дата обращения: 20.12.2016).
12. Поиск лица в растровом образе // 100byte URL: http://www.100byte.ru/stdntswrks/dbst/dbst.html (дата обращения: 20.12.2016).
13. Методы основанные на построении модели лица // http://scicenter.online/ URL: http://scicenter.online/radiotehnika-elektronika-kompyuteryi/metodyi-osnovannyie-postroenii-modeli.html (дата обращения: 20.12.2016).
14. Explaining AdaBoost // http://rob.schapire.net/ URL: http://rob.schapire.net/papers/explaining-adaboost.pdf (дата обращения: 02.01.2017).
15. Алгоритм AdaBoost // MachineLearning.ru URL: http://www.machinelearning.ru/wiki/index.php?title=AdaBoost (дата обращения: 02.01.2017).
16. Multiclass AdaBoost // pdfs.semanticscholar.org URL: https://pdfs.semanticscholar.org/7311/c23dbcc860b7b0993c861bf36f39255ac2e5.pdf (дата обращения: 02.01.2017).

17. AdaBoost Tutorial // mccormickml.com URL: http://mccormickml.com/2013/12/13/adaboost-tutorial/ (дата обращения: 02.01.2017).
18. Cascading classifiers // en.wikipedia.org URL: https://en.wikipedia.org/wiki/Cascading_classifiers (дата обращения: 02.01.2017).
19. Train a Cascade Object Detector // MathWorks URL: https://www.mathworks.com/help/vision/ug/train-a-cascade-object-detector.html (дата обращения: 03.01.2017).
20. Yi-Qing Wang An Analysis of the Viola-Jones Face Detection Algorithm // IPOL Journal · Image Processing On Line. 2014. №4.
21. MATLAB // Wikipedia URL: https://ru.wikipedia.org/wiki/MATLAB (дата обращения: 03.01.2017).
22. OpenCV // Wikipedia URL: https://ru.wikipedia.org/wiki/OpenCV (дата обращения: 03.01.2017).
23. Компьютерное зрение. Библиотека OpenCV // IT и Мультимедиа URL: http://itmultimedia.ru/kompyuternoe-zrenie-biblioteka-opencv/ (дата обращения: 03.01.2017).
24. ROC-кривая // Википедия URL: https://ru.wikipedia.org/wiki/ROC-кривая (дата обращения: 11.01.2017).
25. perfcurve // MathWorks URL: https://www.mathworks.com/help/stats/perfcurve.html (дата обращения: 11.01.2017).