МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование кафедры)

01.04.02 ПРИКЛАДНАЯ МАТЕМАТИКА И ИНФОРМАТИКА
(код и наименование направления подготовки)

МАТЕМАТИЧЕСКОЕ МОДЕЛИРОВАНИЕ
(направленность (профиль))

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

на тему Модификация метода Виолы-Джонса путём фильтрации входного потока с помощью оператора Лапласа

Студент Катанов И.Е. _____
(И.О. Фамилия) (личная подпись)

Научный руководитель Климов В.С. _____
(И.О. Фамилия) (личная подпись)

Руководитель программы д. ф.-м. н., доцент, С.В. Талалов _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Допустить к защите

Заведующий кафедрой к.т.н., доцент, А.В. Очеповский _____
(ученая степень, звание, И.О. Фамилия) (личная подпись)
«_____»_____20_____г.

Тольятти 2018

Оглавление

Введение 3
1 Степень разработанности темы исследования 7
1.1 Существующие исследования по теме 8
1.2 Выводы по существующим исследованиям 10
2 Современные подходы к решению задачи 11
2.1 Deformable Part Models 13
2.2 Scale-Invariant Feature Transform (SIFT) 16
2.3 Speeded Up Robust Features 19
2.4 Gradient Location Orientation Histogram (GLOH) 21
2.5 Histogram of Oriented Gradients (HOG) 22
2.6 Метод Виолы-Джонса 29
2.7 Выбор подхода 35
3 Варианты улучшения алгоритма 37
3.1 Сглаживающий фильтр Гаусса 37
3.2 Выделение границ 42
3.3 Оператор Превитта 47
3.4 Оператор Собеля 51
3.5 Оператор Лапласа 54
3.6 Детектор границ Канни 57
3.7 RASW – Runtime Adapting Sliding Window 60
3.8 Выбор способа улучшения метода Виолы-Джонса 62
Глава 4 Описание выбранного решения 64
4.1 Реализация 65
4.2 Результаты работы реализации 70
Заключение 82
Список используемой литературы 83

Введение

Способность видеть и различать объекты – нечто естественное и привычное для большинства людей. Однако, даже для сложных компьютерных систем на данный момент – это является чрезвычайно сложной задачей. Сейчас весьма большое количество исследовательских команд и даже целые корпорации предпринимают попытки обучить компьютер по крайней мере отчасти тому, что человек делает ежедневно, даже того не замечая.

Актуальность темы исследования:

Исследования, посвященные улучшению работы методов компьютерного зрения, являются крайне актуальными по многим причинам. Существует множество задач, решение которых не представляется возможным без использования методов компьютерного зрения, или же существенно упрощается с их помощью. Область применения машинного зрения весьма широка: от считывателей штрих-кодов и QR-кодов в супермаркетах до приложений дополненной реальности (Augmented Reality).

В данной работе речь пойдёт об улучшении методов компьютерного зрения для решения одной из таких задач, выполняемых методами компьютерного зрения, а именно – о задаче поиска пешеходов на изображении. При реализации интеллектуальных систем от разработчика часто требуется анализировать изображения с целью поиска на них определённых объектов, например автомобилей, людей, знаков дорожного движения и т.п. Существует множество стандартных алгоритмов для решения задач локализации объектов, например HOG-SVM, метод Виолы-Джонса, Deformable Part Models и т.д., однако, проблемой некоторых существующих алгоритмов в определённых задачах, например, метода Виолы-Джонса является их относительно-низкая точность работы, поэтому исследования, направленные на повышение точности работы алгоритма являются актуальными.

Научная проблема исследования:

При решении вышеописанных задач возникает множество проблем, связанных с действиями в экстренных ситуациях, таких как пешеход на проезжей части. Для решения подобных проблем используются методы компьютерного зрения, а именно – обнаружения объектов, в данном случае пешеходов, на изображении.

Абсолютное большинство методов компьютерного зрения разрабатываются и настраиваются на конкретную задачу с помощью методов машинного обучения, которые позволяют определить значимые признаки и их значения, соответствующие искомым объектам.

Следует отметить, что не существует идеальных методов компьютерного зрения, и даже если сконцентрироваться на какой-либо конкретной задаче, как, например, задача поиска пешеходов на изображении, не существует метода, который не имел бы каких-либо недостатков. Например, одним из распространённых недостатков большинства методов компьютерного зрения является то, что они имеют трудоёмкость, которая зависит квадратично или даже кубично от объёма входных данных, вследствие чего их работа при решении различных серьёзных задач может быть слишком медлительной. Методы, которые не создают большую вычислительную нагрузку в свою очередь имеют сравнительно низкую эффективность обнаружения, что создаёт проблему недостатка пригодных альтернатив методам с высокой вычислительной нагрузкой.

Таким образом, научная проблема и актуальность исследования обусловлены несовершенством современных методов поиска пешеходов на изображении и нехваткой пригодных для подобной задачи альтернативных методов.

Объект исследования – компьютерное зрение в задаче поиска пешеходов на изображении.

Предмет исследования – использование метода Виолы-Джонса в задаче поиска пешеходов на изображении и модификации, способствующие улучшению его работы.

Цель исследования – модификация метода Виолы-Джонса с целью улучшения результатов его работы при приложении к задаче локализации людей на изображении.

Задачами работы являются:

1) Изучить особенности работы алгоритма Виолы-Джонса;

2) Разработка математической алгоритмической модели улучшения метода Виолы-Джонса за счёт предобработки входного изображения;

3) Разработать программное обеспечение, демонстрирующее работу данного подхода;

4) Протестировать модифицированный метод на основе тестовой выборки;

Гипотеза – результаты эксперимента по модификации метода Виолы-Джонса покажут уменьшение количества ошибок первого и второго рода при его работе в задаче поиска пешеходов на изображении.

Методологическая основа данной работы – сравнение, измерение и эксперимент.

В данной работе используется подход к решению описанной проблемы, опирающийся на проведение абстрактного (оторванного от реальной ситуации на дороге) испытания метода Виолы-Джонса с использованием предложенных модификаций и без них на наборе тестовых изображений. Полученные результаты работы, дающие представление о эффективности методов, позволят определить насколько предложенная модификация окажется полезной на практике, чем определяется практическая значимость исследования. Результаты испытания представлены с помощью графика кривых ROC, который позволяет наглядно представить и оценить количество объектов, верно-классифицированных в процессе работы методов.

Новизна исследования: установлено, что использование оператора Лапласа вместе со сглаживающим фильтром Гаусса для предобработки изображения позволяет повысить точность работы метода Виолы-Джонса в задачах локализации человека на 15-20%.

В ходе исследования установлено, что применение оператора Лапласа в сочетании с размытием по Гауссу ко входному потоку метода Виолы-Джонса приводит к уменьшению количества ошибок второго рода на 15-20%. При этом использование данной модификации приводит к увеличению количества ошибок первого рода, которое, впрочем, может быть нивелировано путём увеличения обучающей выборки.

Объем и структура диссертации: диссертационное исследование состоит из введения, 4 глав, заключения, библиографии (64 наименования). Работа изложена на 92 страницах, содержит 54 рисунка.

1 Степень разработанности темы исследования

Такая область науки, как компьютерное зрение является весьма молодой и активно развивающейся. И несмотря на то, что существуют и более ранние труды, можно точно сказать, что пристальное внимание и активное изучение этой проблемы началось только в конце 1970-х. Однако, эти работы также были направлены на решение задач других областей, из чего можно сделать вывод, что формулировки проблемы компьютерного зрения в стандартном виде не существует. Также невозможно предоставить стандартную формулировку способов решения проблемы компьютерного зрения. Существует большое количество способов, созданных для решения определённых узких задач компьютерного зрения. Методы, применяемые в этих способах обычно зависят от конкретной задачи и в большинстве случаев слабо поддаются обобщению с целью расширить области их применения. Несмотря на то, что большинство из подобных способов находятся в зародыше и исследования по их разработке и улучшению являются фундаментальными, с каждым годом всё больше коммерческих продуктов и решений начинают использовать эти способы при решении каких-либо своих задач. В подобных решениях данные способы обычно представляют собой часть большой системы и вносят большой вклад в решение ею множества весьма сложных задач, например в медицине, или же в контроле качества при изготовлении определённых продуктов там, где человек в силу определённых причин не может проводить подобную работу. В большинстве случаев практического применения методов компьютерного зрения компьютеры предварительно запрограммированы и настроены, чтобы максимально эффективно решать чётко-определённые и узкие задачи, однако способы, использующие в своей основе накопление и последующее применение знаний, становятся популярнее с каждым годом и всё больше таких способов находит применение в широких кругах.

Нельзя также не сказать о такой области искусственного интеллекта, как автоматическое принятие решений. Оно играет немаловажную роль в развитии

и популяризации искусственного интеллекта, так как его методы часто используются в весьма важных системах, которые способны самостоятельно выполнять механическую работу. Например, существуют роботы, способные самостоятельно передвигаться и находить путь в некоторой среде. Работа подобных систем зачастую напрямую зависит от визуальных входных данных, предоставляемых различными системами компьютерного зрения, играющими роль визуального сенсора и способны предоставить роботу весьма точную информацию о окружающем его пространстве.

Также существует такая область искусственного интеллекта, напрямую зависящая от методов компьютерного зрения, как обучающие методы в приложении к распознаваниям объектов на изображении или к сравнению изображений с целью обнаружения совпадений.

Всё это позволяет нам рассматривать компьютерное зрение, как подобласть искусственного интеллекта, играющую важную роль в его развитии в целом, а также как важную область компьютерных наук как таковых. [2,7,8,9].

1.1 Существующие исследования по теме

Однако, работы над усовершенствованием и обобщением методов компьютерного зрения ведутся постоянно. Существует множество научных работ, посвящённых использованию узконаправленных методов компьютерного зрения в решении задач, для которых они не предназначались.

Так, например, существует множество работ, посвящённых использованию метода Виолы-Джонса, изначально предназначавшегося для детектирования лиц на изображениях, в различных других задачах, таких как детектирование транспортных средств на изображениях, как описано в [23-46]. В приведённых источниках описаны в основном способы модификации метода Виолы-Джонса, предусматривающие изменение обучающего алгоритма, строящего деревья принятия решений, например, замену алгоритма Ada-Boost на SVM, или использование нейронных сетей для обучения каскада классификаторов.

Уже создано множество работ, как например [10,17,55-64], посвящённых использованию других методов поиска объектов на изображениях, таких как Histogramm of Oriented Gradients (HOG), или Deformable Part Models (DPM), в задаче поиска пешеходов на изображении. В данных работах примером предлагаемых модификаций является сочетание алгоритма машинного обучения, такого как AdaBoost, при обучении классификаторов, основанных на дескрипторах особых точек, таких как HOG, DPM или других, или же дополнение основных методов классификации дескрипторами на основе признаков Хаара.

Также существуют работы как [22,47-54], посвящённые использованию метода Виолы-Джонса в задаче поиска пешеходов, в числе идей которых, также как в случае с приложением метода Виолы-Джонса к задаче поиска транспортных средств на изображении, является замена обучающего алгоритма на гипотетически более эффективные методы. Также представлен метод, предусматривающий использование информации не только о человеке на изображении, но и о характере его перемещений. Таким образом, в данном методе используется информация с двух следующих друг за другом кадров, как представлено на рисунке 1.1:

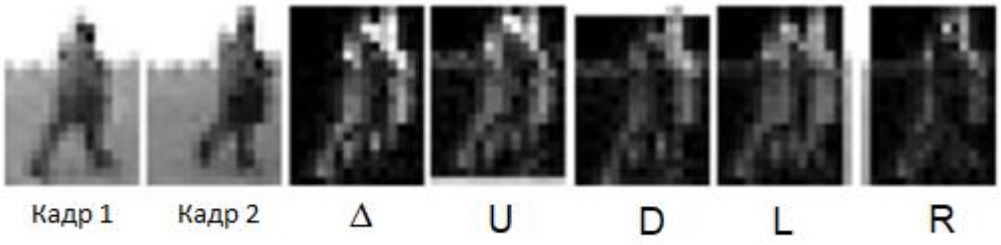

Рисунок 1.1 – Пример смещения объекта на изображении. На первых двух изображениях показаны два последовательных кадра с изображением пешехода в низком разрешении. Следующие пять изображений показывают величину смещения объекта в общем, вверх, вниз, влево и вправо

Существует также способ модификации метода Виолы-Джонса, предусматривающий использование вычислительных мощностей графического процессора, что позволяет существенно ускорить работу любых методов поиска объектов на изображении.

Встречаются и предложения модификации, предусматривающие предварительную обработку изображений с помощью выделения важных параметров изображения методами бинаризации изображения или выделения границ на изображении, которое в большинстве случаев осуществляется детектором границ Канни.

1.2 Выводы по существующим исследованиям

Приведённые примеры исследований демонстрируют улучшение результатов работы при использовании предлагаемых авторами методов улучшений, включая улучшения, предлагаемые для метода Виолы-Джонса в задаче поиска человека на изображении. Однако, работы, посвященные увеличению точности работы методов локализации человека на изображении путём предобработки входных изображений с помощью выделения границ на нём, встречаются редко, что создаёт недостаток исследований на данную тему, приводящий к недостаточности информации.

2 Современные подходы к решению задачи

Задача детектирования пешеходов, также, как и любых других объектов, на изображениях состоит в поиске местоположения всех объектов некоторого класса на изображении. Местоположение объекта можно понимать по-разному, например, как множество пикселей, составляющих объект, или, как координаты прямоугольной области, или контуров, окаймляющих его. В данной работе будет рассматриваться второй подход, то есть на выходе алгоритма детектирования требуется получить множество окаймляющих прямоугольников или контуров.

В связи с тем, что задача детектирования пешеходов актуальна и сложна, в данный момент можно перечислить множество подходов к ее решению, значительно отличающихся друг от друга.

Ярким примером таких систем является система автопилотирования Drive PX от компании NVIDIA. Принципы её работы основаны на использовании свёрточных нейронных сетей.

Нейронные сети в задаче компьютерного зрения являются относительно новым явлением, которое быстро захватывает популярность ввиду высокой точности их работы.

Однако, не смотря на высокое качество их работы, они не могут полностью вытеснить традиционные алгоритмы компьютерного зрения, основанные на машинном обучении и дескрипторах, так как те имеют определённые преимущества перед нейронными сетями. Перечислим их в таблице 2.1.

Таблица 2.1 – Преимущества традиционных алгоритмов компьютерного зрения перед глубокими нейронными сетями [12]

	Традиционные алгоритмы с машинным обучением	Глубокие нейронные сети
Размеры обучающей выборки	В среднем достаточно 10000 изображений, включающих в себя как положительные, так и отрицательные экземпляры. Обучение при этом длится в пределах двух суток.	При обучении необходимо несколько сотен тысяч изображений. Длительность обучения может превышать месяц.
Объёмы вычислений	На сравнительно старом аппаратном обеспечении (например, Pentium 3, 600 Mhz) достижимы результаты распознавания на скорости от двух кадров в секунду.	Для качественной работы необходимо мощное аппаратное обеспечение, в некоторых случаях – даже эксклюзивное от конкретного производителя (NVIDIA)

Также стоит отметить, что большинство традиционных алгоритмов гораздо проще в реализации, чем нейронные сети.

В целом, исходя из приведённых здесь данных мы можем сказать, что традиционные методы лучше методов, основанных на нейронных сетях, если рассматривать задачу со стороны экономии времени и средств [12].

В связи с этим мы будем использовать традиционные методы компьютерного зрения.

Существует множество традиционных алгоритмов компьютерного зрения, перечислим некоторые из них.

2.1 Deformable Part Models

Этот алгоритм является результатом исследований по применению деформируемых моделей, представляющих объекты с помощью локальных шаблонов частей и геометрических ограничений, накладываемых на возможные расположения частей к задаче обнаружения объектов на изображении [14].

Необходимость разработки данного подхода обозначена тем фактом, что обычные линейные классификаторы иногда бывают недостаточными, чтобы моделировать объекты, которые могут иметь существенные внешние отличия.

Данный подход предназначен, чтобы работать с объектами большой вариативности. В модели, используемой в данном подходе объекты описаны, как набор частей, собранных в деформируемую конфигурацию. Каждая часть этой модели кодирует локальную конфигурацию внешнего вида объекта, пример которой представлен на рисунке 2.1, в то время как общая деформируемая конфигурация характеризуется соединениями, похожими на пружины, между некоторыми парами частей, как представлено на рисунке 2.2.

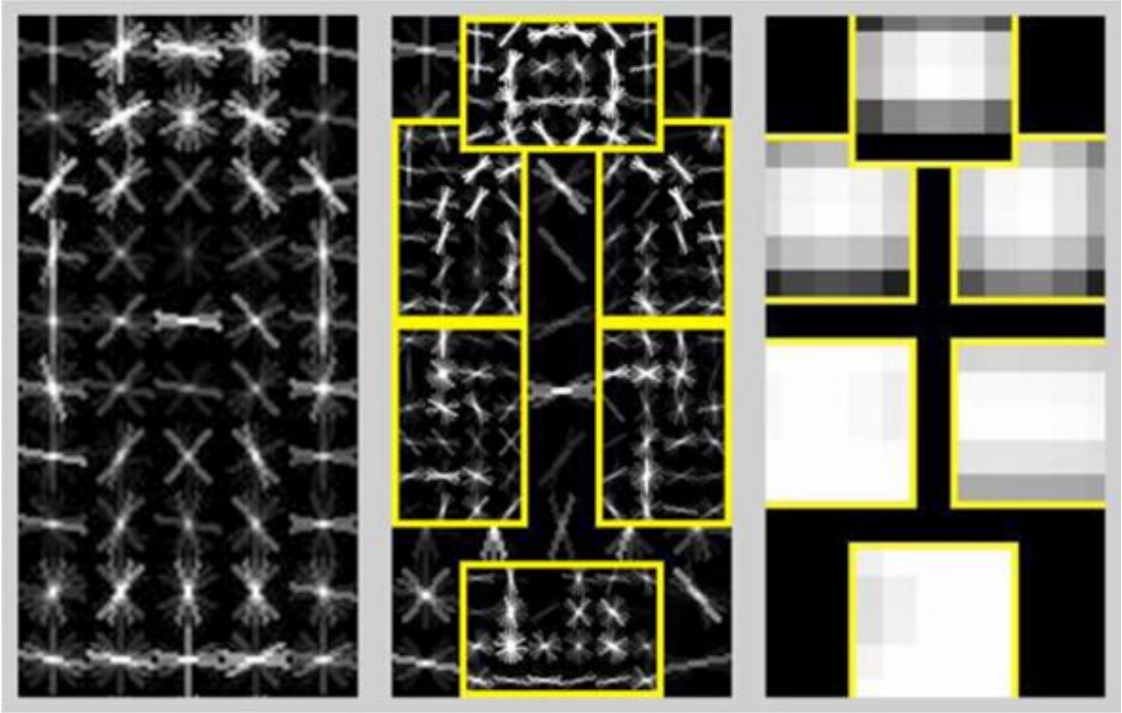

Рисунок 2.1 – Пример локальных конфигураций DPM

В целях обеспечения лучшей вариативности обнаруживаемых объектов (например, различные положения тела и так далее) используются смеси деформируемых моделей.

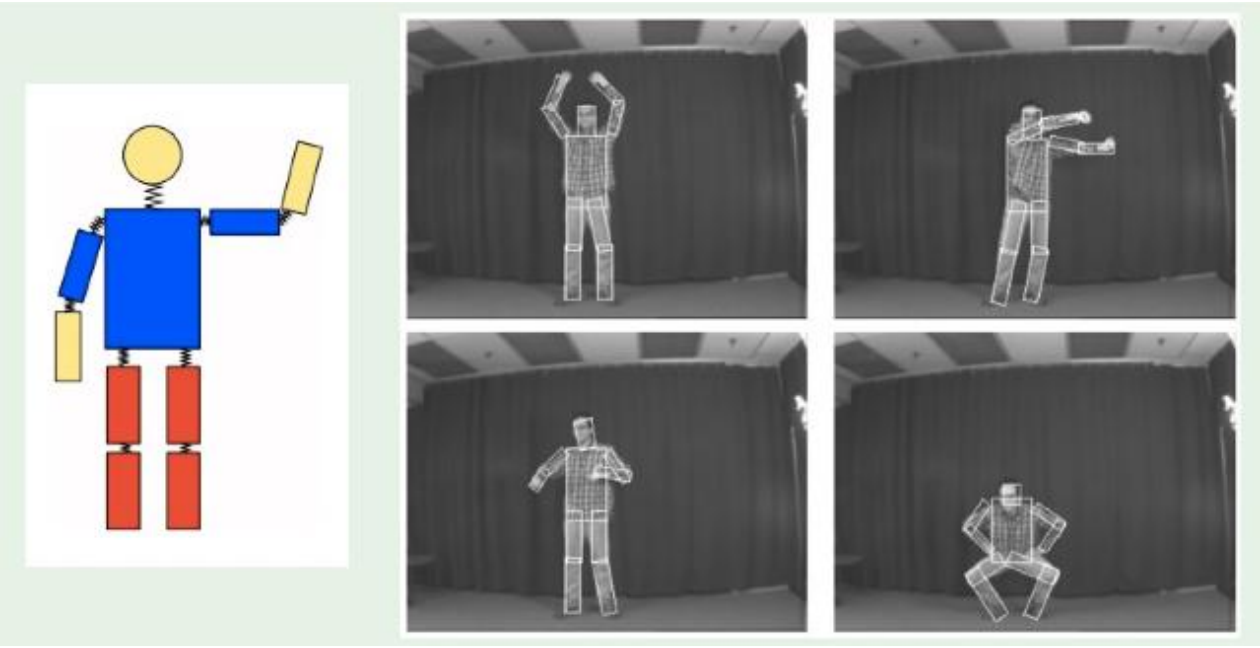

Рисунок 2.2 – Пример общей конфигурации DPM

В данном случае классификаторы рассматривают выбор компонентов смеси моделей и местоположения частей объекта как латентные переменные. Пускай x определяет изображение, а также позицию и размер в нём. Классификаторы вычисляют параметры результата так, как описано в уравнении (2.1):

$$f_\beta(x) = \max_z \beta * \Phi(x, z) \quad (2.1)$$

Здесь β – это вектор параметров модели, z – латентные значения и $\Phi(x,z)$ – вектор признаков.

Если значение функции выше определённой границы, то модель произведёт обнаружение объекта на x. Примеры работы метода Deformable Part Models, обученного на различных объектах приведены на рисунке 2.3.

Рисунок 2.3 – Примеры результатов работы DPM

В итоге главным положительным качеством данного метода является большая вариативность внешнего вида объектов, которые он может обнаруживать.

Однако у данного подхода есть недостаток, состоящий в том, что работа алгоритма может быть недостаточно быстрой для нашей задачи, так как он основан на алгоритме Histogram of Oriented Gradients (HOG) [14,26,27].

2.2 Scale-Invariant Feature Transform (SIFT)

Данный алгоритм компьютерного зрения предназначен для поиска и описания локальных признаков на изображениях.

При создании дескриптора SIFT сначала вычисляются параметры магнитуды и направления градиента для каждого пикселя, принадлежащего области особой точки, имеющей размер 16 на 16 пикселей. Магнитуды градиентов рассчитываются с использованием весов, пропорциональных значению функции плотности нормального распределения с математическим ожиданием в заданной особой точке и стандартным отклонением, равным половине ширины окрестности.

Для каждого квадрата, имеющего размер 4 на 4 пикселя, вычисляется гистограмма направленных градиентов с помощью суммирования взвешенной величины магнитуды градиента с одним из 8 бинов гистограммы. Чтобы избавиться от влияния "граничных" эффектов, которые связаны с отнесением схожих градиентов к разным квадратам используется билинейная интерполяция, а именно добавление значения магнитуды градиента в гистограммы, относящиеся к квадратам, являющимся соседними для квадрата, к которому относится данный пиксель. При этом при добавлении значения магнитуды учитывается вес, который пропорционален расстоянию от пикселя, в котором вычисляется данный градиент, до центра соответствующего квадрата. В итоге все полученные гистограммы объединяются в один вектор, имеющий размер, равный 128 = 8 (число бинов) * 4 * 4 (число квадратов).

Чтобы минимизировать возможные эффекты от изменчивости освещения полученный дескриптор преобразуется. Изменение контрастности изображения (значение интенсивности каждого пикселя умножается на некоторую

константу) даёт результат в виде таких же изменений в значениях магнитуд градиентов. Из этого следует, что данный эффект может быть нейтрализован, если нормализовать дескриптор так, чтобы его длина стала равна единице. Изменения яркости изображения (к значению интенсивности каждого пикселя прибавляется некоторая константа) не могут повлиять на величину магнитуд градиентов. Так, дескриптор SIFT, представленный на рисунке 2.4, инвариантен по отношению к аффинным изменениям освещенности. Однако, вследствие, например, различной ориентации освещения относительно поверхностей трёхмерного объекта, встречаются и нелинейные изменения освещенности. Данные эффекты могут вызвать большое изменение в отношении магнитуд некоторых градиентов (при этом оказывают незначительное влияние на ориентацию вектора градиента). Во избежание такой проблемы, используют отсечение с использованием некоторого порога (результаты экспериментов показали, что 0.2 является оптимальным значением). Его применяют к компонентам дескриптора после его нормализации. После отсечения по порогу дескриптор снова нормализуется. Это позволяет уменьшить значение больших магнитуд градиентов, одновременно с увеличением значений распределения направлений данных градиентов в окрестности особой точки.

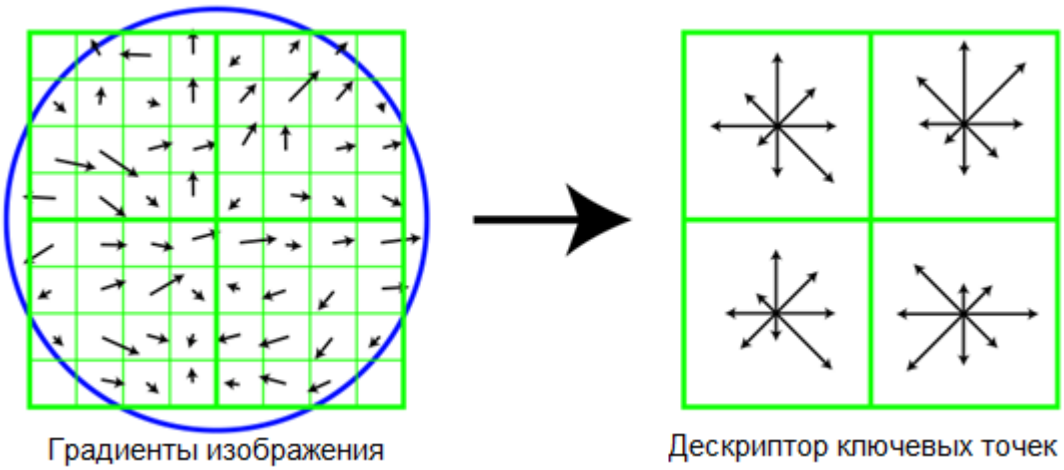

Рисунок 2.4 – Пример доминирующих градиентов изображения и соответствующего им дескриптора ключевых точек

Пример результатов работы подобного дескриптора приведён на рисунке 2.5:

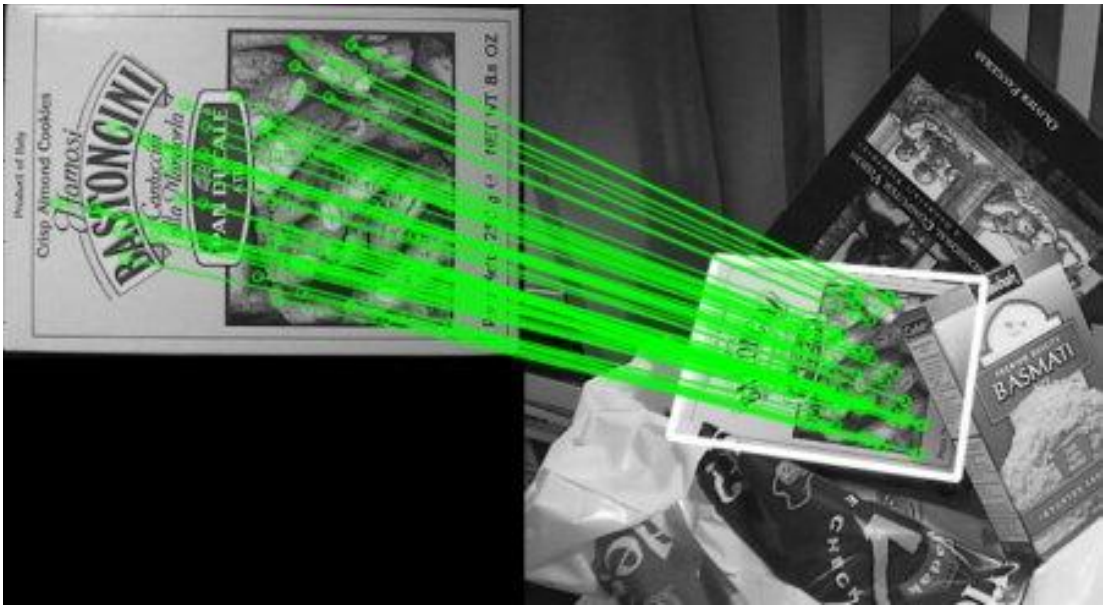

Рисунок 2.5 – Пример результата работы дескриптора SIFT

Перечислим положительные качества данного алгоритма:

- независимость от размера изображения и ориентации объекта на нём.

Негативные качества:

- очень большое количество вычислений на кадр, высокая вычислительная сложность [28]

Таблица 2.2 – вычислительная сложность различных этапов выполнения [15, 16]

Фаза работы	Сложность	Число операций
Gaussian Blurring	$\theta(N^2\omega^2 s)$	$4N^2\omega^2 s$
Difference of Gaussian	$\theta(sN^2)$	$4N^2 s)$
Scale-space Extrema Detection	$\theta(sN^2)$	$104sN^2$
Keypoint Detection	$\theta(\alpha sN^2)$	$100s\alpha N^2$
Orientation Assignment	$\theta(sN^2(1-\alpha\beta))$	$48sN^2$

Keypoint Descriptor Generation	$\theta(x^2N^2(\alpha\beta+\gamma))$	$1520x^2(\alpha\beta+\gamma)N^2$

2.3 Speeded Up Robust Features

Дескриптор SURF является одним из тех дескрипторов, которые позволяют выполнять поиск особых точек и создавать их описание, не зависящее от изменений масштаба и вращения, одновременно. Также, сам поиск ключевых точек инвариантен в том смысле, что, если повернуть объект сцены, набор особых точек останется тем же, что и у образца.

Поиск особых точек на изображении проводится с помощью матрицы Гессе. Использование Гессиана обеспечивает независимость относительно преобразований, типа поворотов, но не независимость относительно изменений масштаба. Поэтому в методе Speeded Up Robust Features при вычислении Гессиана применяются фильтры разного масштаба. Допустим, что изображение задается матрицей интенсивностей пикселей I, конкретный, рассматриваемый в данный момент, пиксель обозначим как X=(x,y), а масштаб фильтра обозначим как σ. Тогда матрица Гессе будет иметь вид (2.2):

$$H(X,\sigma) = \begin{matrix} L_{xx}(X,\sigma) & L_{xy}(X,\sigma) \\ L_{xy}(X,\sigma) & L_{yy}(X,\sigma) \end{matrix}, \quad (2.2)$$

где $L_{xx}(X,\sigma)$, $L_{xy}(X,\sigma)$, $L_{yy}(X,\sigma)$ – свертки аппроксимации второй производной Гауссова ядра с изображением *I*.

Детерминант матрицы Гессе достигает максимального значения, то есть экстремума, в точках, где изменение градиента яркости максимально. Поэтому Speeded Up Robust Features проходит фильтром, использующим ядро Гаусса по всему изображению и определяет точки, в которых достигается экстремум детерминанта матрицы Гессе. Следует отметить, что данный проход выделяет как темные элементы на белом фоне, так и светлые элементы на темном фоне.

После этого для всех обнаруженных особых точек вычисляется ориентация – доминирующее направление изменения яркости. Понятие

ориентации схоже с понятием направления градиента, но при определении ориентации особой точки используется фильтр Хаара.

На основании всей имеющейся информации строятся дескрипторы для всех особых точек:

- вокруг особой точки строится квадратная область размером 20 * s, где s – это масштаб, на котором было получено максимальное значение детерминанта матрицы Гессе;
- построенная квадратная область делится на блоки, в результате чего область разбивается на 4 x 4 региона;
- для каждого такого блока вычисляются более простые признаки, вследствие чего получается вектор, который содержит 4 компоненты: из которых 2 – это суммарный градиент по квадранту, и 2 – сумма модулей точечных градиентов;
- дескриптор создаётся в результате соединения взвешенных описаний градиента для 16 квадрантов вокруг особой точки. Все элементы дескриптора затем взвешиваются с помощью коэффициентов Гауссова ядра. Взвешивание необходимо, чтобы обеспечить большую устойчивость к шумам в удаленных точках;
- дополнительно к дескриптору вносится след матрицы Гессе. Это необходимо, чтобы иметь возможность различать темные и светлые пятна. Для светлых точек на темном фоне след будет отрицателен, в то время как для темных точек на светлом фоне след будет положителен.

Отметим, что несмотря на то, что SURF используется для поиска объектов, дескриптор никак не использует информацию об этих объектах. Метод SURF рассматривает изображение как единое целое и выделяет особенности для всего изображения в целом, что приводит к тому, что он плохо работает с объектами простой формы или объектами, имеющими достаточные отличия внутри своих контуров [16, 17].

Перечислим положительные качества:

- независимость от освещения, размера и ориентации объектов.

Отрицательные качества:

- несмотря на то, что данный алгоритм работает быстрее, чем SIFT, он всё ещё имеет большую вычислительную сложность [16,28,29].

2.4 Gradient Location Orientation Histogram (GLOH)

Дескриптор, называемый GLOH, или Gradient Location-Orientation Histogram по сути своей является модификацией такого метода, как SIFT, или Scale Invariant Feature Transform, о котором мы говорили выше. GLOH был разработан в целях повышения надежности и устойчивости исходного метода. Если разобрать отличия метода GLOH от SIFT, мы выясним, что их не много. Вычисляется такой же дескриптор, как и для метода SIFT, однако вместо квадратной сетки разбиения изображения, в нём используется полярная сетка, содержащая бины. Она представляет собой 3 радиальных блока, радиусы которых равны 6, 11 и 15 пикселей, и 8 секторов. В результате получается вектор, содержащий 272 компоненты, который проецируется на пространство размерности 128 с помощью анализа главных компонент (PCA).

Пример дескриптора значимой точки GLOH приведён на рисунке 2.6.

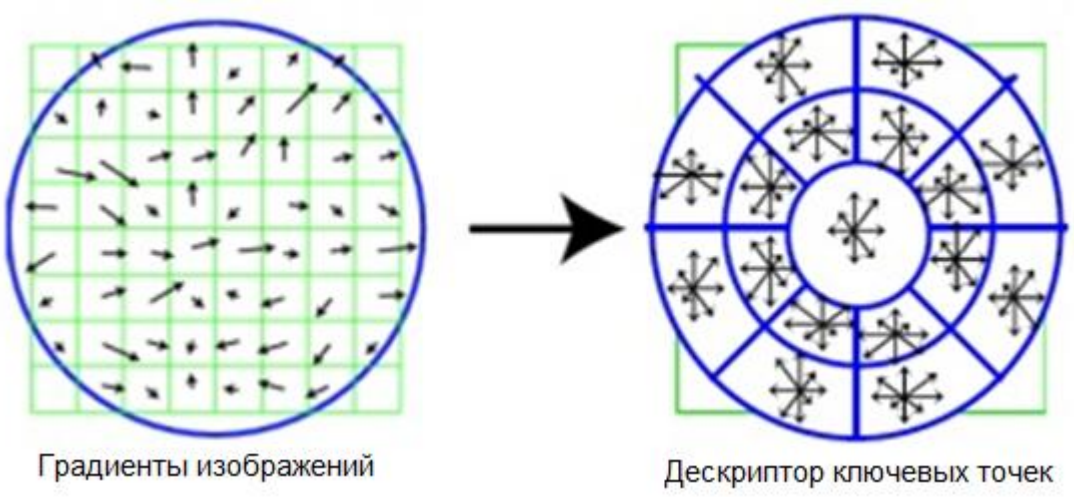

Рисунок 2.6 – Градиенты изображения и полученный дескриптор значимой точки GLOH

К положительным качествам можно отнести повышение надёжности относительно предка данного метода.

В силу того, что основан на SIFT, имеет те же недостатки, а именно высокую вычислительную нагрузку.

2.5 Histogram of Oriented Gradients (HOG)

Основной идеей, определяющей всю работу метода HOG, или Histogram Oriented Gradients, является то, что любой объект на изображении, будь то ваза, автомобиль, человек или что-либо ещё, может быть описан с помощью сетки, содержащей описание распределения интенсивности пикселей в определённых его частях. При этом нам не нужна информация для каждой точки, достаточно обладать аппроксимацией градиента яркости. Мы считаем, что она дифференцируема с помощью некоторой разностной схемы, но при этом известна только в узлах упоминаемой ранее сетки. Признаки HOG описываются с помощью дескрипторов, как и в случае большинства существующих методов компьютерного зрения, использующих машинное обучение. Способ составления данных дескрипторов во многом схож со способами, используемыми в таких методах компьютерного зрения, как SIFT и SURF,

которые мы описали выше. Однако, существует важное отличие. Если дескриптор SIFT представляет собой некоторое описание окрестности точки интереса, то дескриптор HOG описывает всё обрабатываемое изображение.

Основными элементами дескриптора, используемого в методе Histogram Oriented Gradients, считаются некоторые блоки, которые описывают некоторую область прямоугольной формы, имеющую строго определённые размеры, а именно 64 на 128 пикселей. При этом не имеет значения, какой размер имеет исходное изображение, главное – соотношение сторон 1 к 2. Например, на рисунке 2.7 показано большое исходное изображение размером 720 на 475 пикселей. Для создания дескриптора был выбран кусок изображения размером 100 на 200 пикселей, который после был преобразован к размеру 64 на 128.

Рисунок 2.7 – Пример преобразования участка изображения к необходимому размеру

Каждый блок дескриптора HOG состоит из ячеек 8 на 8 пикселей. При этом каждой ячейке соответствует гистограмма ориентаций (углов наклона относительно горизонтали, или направлений) градиентов из заданного количества направлений (бинов), получаемых с помощью сопоставления

значений величины градиента в пикселе изображения его направлению так, как показано на рисунке 2.8.

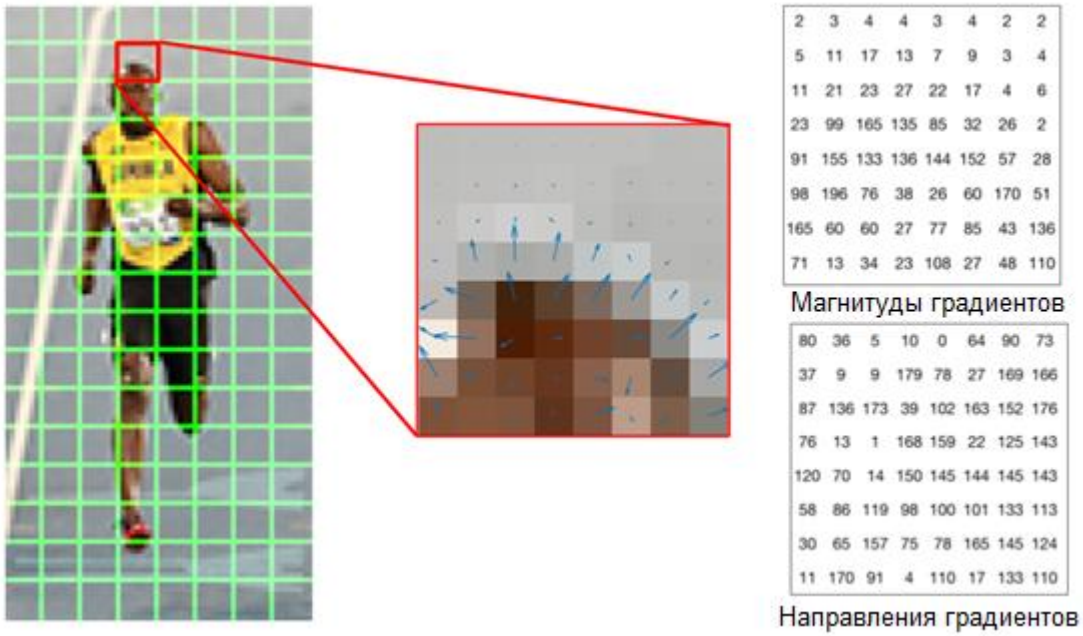

Рисунок 2.8 – Слева-направо: участок изображения в формате RGB, градиенты, представленные в виде стрелок на участке изображения величиной 8 на 8 пикселей и те же градиенты, представленные в виде значений величины и направления

При этом направление градиента считается беззнаковым, то есть наклон в α градусов и в $(2\pi - \alpha)$ градусов считаются эквивалентными. Величины градиентов для каждого пикселя составляют вместе некоторые бины гистограммы, принадлежащей ячейке, содержащей эти пиксели. Также, в целях улучшения результатов работы, эти величины также вносят некоторый вклад в бины гистограмм других ячеек, а именно, тех, что соседствуют с текущей ячейкой. Для этого в методе HOG применяется линейная интерполяция, прикладываемая к углу наклона, а также билинейная, которая прикладывается к относительному расположению ячейки в пространстве. Вдобавок к этому величины градиентов подвергаются взвешиванию с помощью свёртки гауссианом. Центр свёртки при этом находится точно по центру блока. Когда

для каждой ячейки вычислены гистограммы, их конкатенируют таким образом, что создаётся вектор, или массив признаков блока, которому принадлежат эти ячейки. Пример проведения данной операции показан на рисунке 2.9.

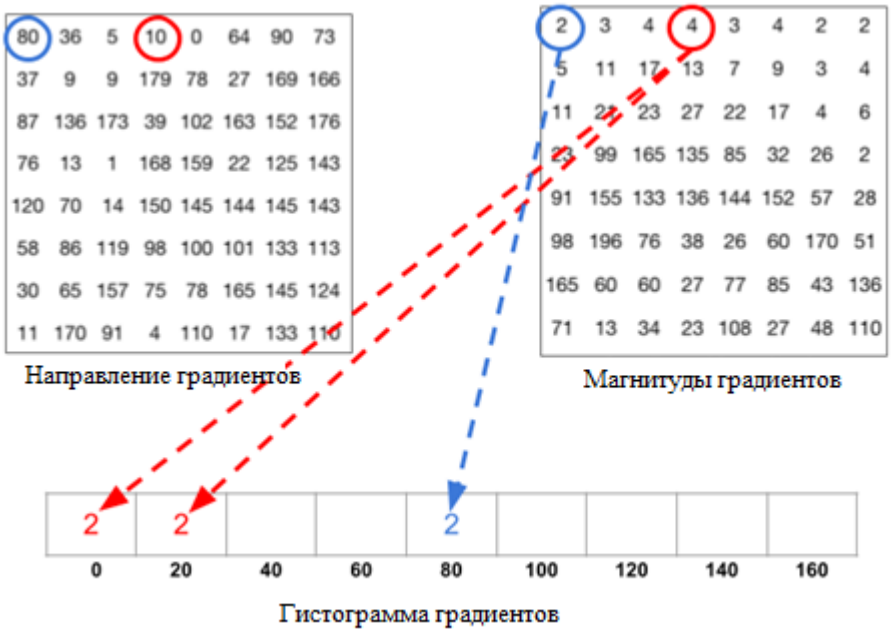

Рисунок 2.9 – Формирование гистограммы градиентов

Полученный вектор нормализуется. Такие дескрипторы формируются для всех блоков, которые содержатся на изображении. При этом их координаты должны соответствовать некоторым требованиям. Так, их левые верхние пиксели должны быть кратны некоторым заданным шагам. Кратность должна соблюдаться по вертикали и по горизонтали. При чём данные шаги обычно задаются так, что блоки перекрывают друг друга, что приводит к тому, что градиент пикселя используется при вычислении признаковых описаний для нескольких блоков. HOG-описание изображения получается путем сложения векторов признаков всех блоков. В итоге, мы получаем дескриптор, представленный на изображении 2.10.

Нетрудно заметить, что основные направления гистограмм следуют силуэту человека, использованного для построения дескриптора.

Рисунок 2.10 – Дескриптор HOG, представленный на изображении

Дескрипторы, создаваемые в процессе работы метода Histogram Oriented Gradients в дальнейшем прикладываются к задаче классификации изображений. Таким образом, так как основной задачей, для которой создавался данный метод, была локализация человека на изображении, рассмотрим её как пример задачи классификации. При классификации нам необходимо чётко разделить изображения на две группы, а именно те, что содержат на себе одного человека или нескольких людей и те, на которых людей нет. Именно для решения

подобных задач используется признаковое описание объекта, или дескриптор, который формируется в процессе обучения с использованием метода HOG. Сначала описание искомого объекта, предоставляемое дескриптором, ставится в соответствие входному изображению. После этого в действие вступают алгоритмы машинного обучения, которые позволяют обучить классификатор. Большинство алгоритмов машинного обучения с учителем (использующие обучающую выборку) могут работать только в пространстве признаков, которое имеет чётко фиксированную размерность. Это ограничение означает, что для работы этих обучающих алгоритмов дескрипторы HOG для любого изображения должны быть одинаковой длины. Для удовлетворения данного требования простейшим решением является использование изображений одинакового размера. При этом считается, что масштаб искомых объектов на изображениях, используемых при обучении и в процессе работы, будет как минимум схож от изображения к изображению.

Таким образом, при решении задачи локализации человека на изображении, мы считаем, что все вышеизложенные требования к размерам и масштабу объектов удовлетворены. В таком случае задача локализации сводится к классификации и может решаться с помощью алгоритмов машинного обучения, впрочем, удовлетворение данных требований в реальности не может быть истиной. Для максимального приближения к требуемым условиям классификации используется так называемый метод скользящего окна, который позволяет рассматривать локализацию объектов на изображении, как задачу по классификации множества подокон, образуемых при скольжении некоторой рамки (скользящего окна) по изображению. В таком случае считается, что на изображении ищутся объекты, имеющие определённые высоту и ширину $w \times h$. Для их поиска рассматриваются прямоугольные области, имеющие размер искомого объекта, а именно $w \times h$. При этом после выполнения классификации для данной прямоугольной области в определённом месте, окно сканирования смещается сначала по горизонтали на

некоторую величину dx. После достижения окном края изображения используется смещение по вертикали на величину dy. При этом dx и dy определены заранее в алгоритме. Таким образом, для прямоугольного окна, имеющего размеры $w \times h$, левый верхний угол которых имеет координаты $i * dx, j * dy$,$i = 0, n, j = 0, m$, то-есть, они кратны шагам dx и dy, мы можем производить классификацию. Однако, как быть в случае если изображения имеют разный масштаб и искомые объекты на них, соответственно, имеют отличные друг от друга размеры? Решение данной проблемы достигается с помощью масштабирования изображения. Так, сначала поиск проводится на изображении с оригинальным размером. В дальнейшем, изображение уменьшается с использованием определённого, заранее задаваемого коэффициента. На уменьшенном изображении снова производится операция классификации с использованием сканирующего окна. Эти действия производятся до тех пор, пока размер итогового изображения не станет меньше размера сканирующего окна. [18,19,20,21,24].

Рисунок 2.11 - Пример работы метода Histogram Oriented Gradients

Отметим положительные качества данного метода:

- независим от размеров и ориентации объекта;
- увеличена точность обнаружения на некоторых задачах.

Отрицательные качества:

- высокая вычислительная нагрузка [26,27].

Вычислительная нагрузка от работы алгоритма равна фильтру медиан: ($O(n^2)$) + получению гистограмм($O(4n^2)$) + сравнению гистограмм($O(k * \frac{n^2}{w^2})$)

2.6 Метод Виолы-Джонса

Метод Виолы-Джонса – алгоритм, позволяющий обнаруживать объекты на изображениях в режиме реального времени. Был предложен в 2001 году Полом Виолой и Майклом Джонсом. Основное предназначение алгоритма – обнаружение лиц на изображениях [11].

Алгоритм позволяет находить объекты на изображениях с высокой точностью и низким количеством ложных срабатываний [4].

Работа алгоритма основана на поиске некоторых признаков на изображении. Признаки, используемые алгоритмом Виолы-Джонса, похожи на признаки Хаара, однако они несколько сложнее и содержат больше прямоугольных областей. Эти признаки состоят из смежных прямоугольных областей, пример которых показан на рисунке 2.11. Они располагаются на изображении определённым образом, после чего в их областях суммируется интенсивность их пикселей. Значением признака будет разность этих сумм интенсивностей.

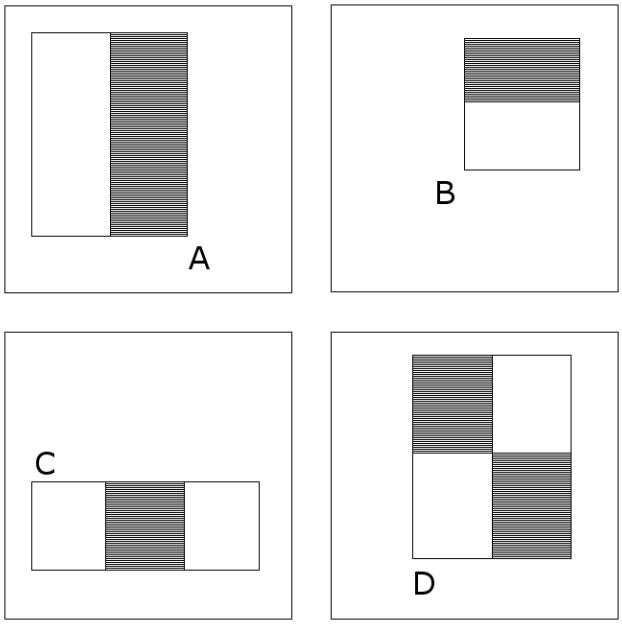

Рисунок 2.12 – Пример признаков, используемых в методе Виолы-Джонса

Данные признаки являются более примитивными, чем, так называемые, “steerable filters” и результат их поиска является более грубым и неточным.

Однако, важной деталью в данном случае является то, что алгоритм Виолы-Джонса подразумевает использование интегрального представления изображения с целью ускорения вычисления значений сумм интенсивности. При хранении изображения в виде интегрального представления проверка прямоугольного признака будет проводиться за константное время, зависящее только от количества прямоугольных областей в нём. Таким образом, для вычисления суммы одной прямоугольной области потребуется только 4 операции обращения к интегральному массиву, а так как все прямоугольные области в признаках смежны, расчёт признака из двух прямоугольников займёт 6 обращений к массиву, из 3 прямоугольников – 8 обращений, а из 4 прямоугольников – 9. Данный фактор позволяет компенсировать грубость используемых признаков крайне высокой скоростью их вычисления.

Однако, даже такая высокая скорость вычисления не может компенсировать значительное количество возможных признаков, подлежащих обработке. Например, при размере признака 24 на 24 пикселя (что является

стандартом при использовании данного метода), становятся возможными более 100 тысяч различных признаков. Естественно, рассчитывать их все будет слишком затратным по времени занятием. Именно поэтому в алгоритме Виолы-Джонса используется машинное обучение, а именно вариация алгоритма обучения AdaBoost, которая используется как для подбора используемых признаков, так и для настройки классификаторов.

Благодаря всему этому алгоритм Виолы-Джонса работает примерно в 15 раз быстрее большинства существующих алгоритмов, не теряя при этом в точности [11].

Остановимся на описанных выше определениях подробнее. Интегральное представление изображения – представление изображения, используемое для изображений в оттенках серого, представляемое в виде матрицы, в которой каждая ячейка содержит сумму интенсивностей пикселей, вычисляемую по формуле (2.3) [3,11]:

$$S\ x,y\ = I\ x,y\ + S\ x-1,y\ + S\ x,y-1\ - S\ x-1,y-1 \qquad (2.3)$$

где x – позиция пикселя по оси X, y – позиция пикселя по оси Y, S – сумма интенсивностей пикселей для данной позиции, I – интенсивность яркости пикселя для данной позиции.

Рисунок 2.13 – Пример вычисления интегрального изображения

Если значений матрицы не существует, они считаются равными нулю.

AdaBoost – алгоритм машинного обучения, предложенный Йоавом Фройндом и Робертом Шапиром. Характеризуется тем, что во время обучения строит целую конструкцию из основных, или базовых алгоритмов обучения с целью улучшить эффективность обучения. Является адаптивным алгоритмом потому, что в нём каждый следующий классификатор строится по тем объектом, которые предыдущие классификаторы не смогли классифицировать верно. Каждому объекту присваиваются некоторые “веса”, характеризующие его важность для алгоритма. Каждый раз, когда классификатор не смог классифицировать объект должным образом, его вес увеличивается, что заставляет алгоритм “фокусировать” на нём внимание.

Достоинства:

- хорошая способность к обобщению. В реальных задачах (в основном, но не всегда) при помощи данного алгоритма удаётся строить композиции, которые превосходят по качеству базовые алгоритмы, на которых он основан. Обобщающая способность также может улучшаться по мере увеличения числа базовых алгоритмов;

- простота реализации;
- собственные вычислительные расходы бустинга весьма малы. Время, за которое строится композиция практически полностью зависит от времени обучения базовых алгоритмов;
- возможность определять объекты, которые являются шумовыми выбросами;

Недостатки:

- AdaBoost весьма склонен к переобучению при наличии значительного уровня шума в исходных данных. Экспоненциальная функция потерь присваивает слишком высокие веса наиболее трудным объектам, на которых ошибаются большинство базовых алгоритмов. Однако, чаще всего именно эти объекты оказываются шумовыми выбросами, в результате чего AdaBoost начинает обучать на шум, что ведёт к переобучению. Проблема решается путём уменьшения числа выбросов или применения более мягких функций потерь;
- AdaBoost требует достаточно больших обучающих выборок. Другие методы линейной коррекции, как например, бэггинг, способны строить алгоритмы сопоставимого качества, используя меньшие выборки данных;
- жадная стратегия последовательного добавления может приводить к созданию неоптимального набора базовых алгоритмов. Для улучшения композиции можно периодически возвращаться к ранее построенным алгоритмам, чтобы обучать их заново. Для улучшения коэффициентов можно оптимизировать их ещё раз по окончании процесса бустинга с помощью какого-нибудь стандартного метода построения линейной разделяющей поверхности;
- бустинг зачастую приводит к построению громоздких композиций, состоящих из сотен алгоритмов. Такие композиции не допускают возможность содержательной интерпретации, требуют больших объёмов памяти для

хранения базовых алгоритмов, а также существенных затрат времени на обучение классификаций.

Пример работы метода Виолы-Джонса для поиска лиц приведён на рисунке 2.14:

Рисунок 2.14 – Пример работы метода Виолы-Джонса в задаче поиска лиц

Пример работы метода Виолы-Джонса в задаче поиска пешеходов приведён на рисунке 2.15:

Рисунок 2.15 – Пример результатов работы метода Виолы-Джонса в задаче поиска пешеходов

Как мы можем видеть на данном изображении, результаты работы метода Виолы-Джонса без модификаций в приложении к поиску пешеходов на

изображении нельзя назвать приемлемой, так как обнаружен только один пешеход из четырёх.

Перечислим положительные качества данного алгоритма:

- простота реализации;
- низкая вычислительная нагрузка: $O\ n^2$.

Перечислим отрицательные качества:

- чувствительность к ориентации и размерам объектов;
- ориентированность на конкретную задачу (поиск лиц), в связи с этим не все признаки эффективно применимы к задаче поиска пешеходов [11,13,30].

2.7 Выбор подхода

В предыдущем разделе было рассмотрено несколько существующих популярных методов поиска объектов на изображении. Были перечислены их положительные и отрицательные качества, на основе официальных данных о перечисленных методах и сторонних исследований.

При разработке и исследовании мы не располагаем ни большим бюджетом, ни запасом времени. По этой причине, мы будем отдавать предпочтение алгоритмам, удовлетворяющим следующим требованиям:

- могут быть реализованы без особых трудностей или имеют официальную общедоступную реализацию;
- не дают высокую вычислительную нагрузку на процессор;
- позволяют получить результаты обнаружения, близкие к 100%.

Исходя из положительных и отрицательных сторон методов, перечисленных в предыдущем разделе, автор данной работы сделал вывод о том, что выдвинутым требованиям в наибольшей мере соответствует метод Виолы-Джонса, так как все остальные методы дают слишком большую вычислительную нагрузку и имеют весьма сложный математический аппарат, что делает их разбор, реализацию и модификацию более сложным занятием.

В связи с этим как метод обнаружения пешеходов на изображении было решено использовать алгоритм Виолы-Джонса.

Однако, одновременно с удовлетворением первых двух требований к методам детектирования, изложенных автором, при практическом использовании метода Виолы-Джонса был обнаружен серьёзный недостаток. При приложении данного метода к задаче поиска пешеходов на изображении, он даёт не самые лучшие результаты работы, давая большое количество ошибок первого рода. В связи с этим было принято решение внести изменения во входные данные или в обработку данных метода Виолы-Джонса с целью улучшения результатов его работы при приложении к задаче поиска пешеходов на изображении.

3 Варианты улучшения алгоритма

В процессе рассмотрения метода Виолы-Джонса и проблем, которые возникли в ходе работы с ним, автор пришёл к выводу о том, что существует несколько способов улучшить результаты работы по поиску пешеходов на изображении.

Данные способы в общем можно разбить на две категории:

- предобработка входного потока;
- изменение работы самого алгоритма.

К предобработке входного потока можно отнести такие методы обработки изображения, как сглаживающий фильтр Гаусса, поиск границ на изображении на основе оператора Преввита, Собеля, или лапласиана, а также поиск границ на изображении с помощью метода Канни.

К изменениям работы алгоритма можно отнести изменение шага при сдвиге окна детектора.

Разберём каждый из перечисленных методов подробнее.

3.1 Сглаживающий фильтр Гаусса

Гауссово размытие – это тип фильтра для размытия изображений, который использует функцию Гаусса (которая также задаёт нормальное распределение вероятностей в статистике) для вычисления изменений, которые необходимо приложить к каждому пикселю изображения. Уравнение функции Гаусса для одного измерения (3.1):

$$G\ x\ = \frac{1}{2*\pi*\sigma^2} * e^{-\frac{x^2}{2*\sigma^2}} \quad (3.1)$$

Распределение весов для одномерного фильтра Гаусса представлено на рисунке 3.1:

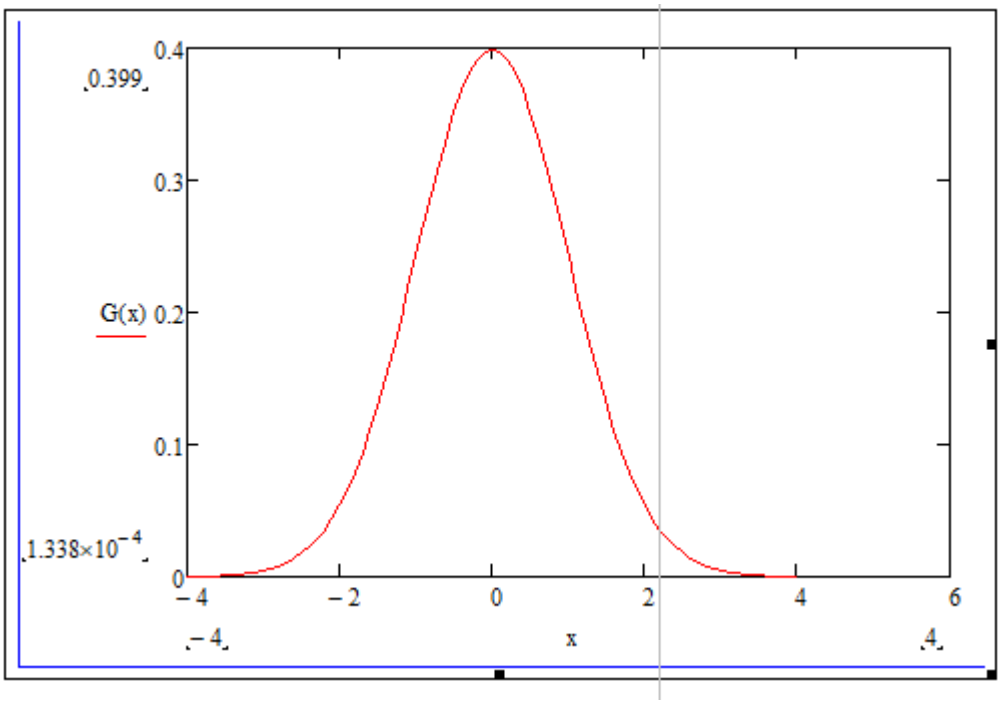

Рисунок 3.1 – Пример распределения весов для одномерного фильтра Гаусса, где по оси абсцисс значение x, а по оси ординат значение функции Гаусса для одного измерения (G(x))

Для двух измерений уравнение представляет собой два таких Гауссиана, по одному на каждое измерение. Данное уравнение представлено ниже (3.2):

$$G\ x,y\ = \frac{1}{2*\pi*\sigma^2} * e^{-\frac{x^2+y^2}{2*\sigma^2}}, \tag{3.2}$$

где x – это расстояние от точки начала координат по горизонтальной оси, y – это расстояние от точки начала координат по вертикальной оси и σ – это стандартное отклонение Гауссового распределения. Будучи приложенной к двумерной системе, данная формула производит поверхность, контуры на которой представляют собой концентрические окружности с распределением Гаусса от центральной точки. Значения этого распределения используются, чтобы построить матрицу свёртки, которая применяется к оригинальному изображению.

Матрица свёртки, получаемая с помощью распределения Гаусса, представлена в формуле 3.3:

$$\begin{matrix} 0.000789 & 0.006581 & 0.013347 & 0.006581 & 0.000789 \\ 0.006581 & 0.054901 & 0.111345 & 0.054901 & 0.006581 \\ 0.013347 & 0.111345 & 0.225821 & 0.111345 & 0.013347 \\ 0.006581 & 0.054901 & 0.111345 & 0.054901 & 0.006581 \\ 0.000789 & 0.006581 & 0.013347 & 0.006581 & 0.000789 \end{matrix} \quad (3.3)$$

Новое значение каждого пикселя приравнивается взвешенному среднему значению окрестности этого пикселя. Значение исходного пикселя получает наибольший вес (с самым высоким гауссовым значением) и пиксели окрестности получают тем меньший вес, чем дальше они расположены от исходного пикселя. Это приводит к размытию изображения, при котором границы и углы сохраняются лучше, чем при использовании других, более однородных фильтров размытия.

В теории, функция Гаусса в любой точке изображения не будет равна нулю, что значит, что для вычисления значения каждого пикселя будет задействовано всё изображение целиком. На практике, при вычислении дискретного приближения функции Гаусса, пиксели, находящиеся на расстоянии более 3σ имеют вес настолько малый, что могут считаться равными нулю. В связи с этим, влияние пикселей, находящихся0 вне данного радиуса, можно игнорировать.

В добавок к тому, что Гауссово размытие имеет круговую симметричность, оно может быть применено к двумерному изображению для двух независимых вычислений, в связи с чем, оно называется сепарабельным, или разделяемым, фильтром. Таким образом, эффект приложения двумерной матрицы также может быть достигнут приложением ряда одиночных одномерных гауссовых матриц в горизонтальном направлении, затем повторением данного процесса в вертикальном направлении. В плане вычислений это полезное свойство, так как вычисление может быть произведено за время (3.4)

$$O\ w_{kernel} w_{image} h_{image}\ + O(h_{kernel} w_{image} h_{image}), \quad (3.4)$$

где h это высота и w это ширина изображения.

Для сравнения, неразделяемые фильтры размытия имеют время вычисления (3.5):

$$O(w_{kernel} h_{kernel} w_{image} h_{image}) \quad (3.5)$$

Применение множества последовательных гауссовых размытий к изображению имеет такой же эффект, как применение одного, большего гауссова размытия с радиусом, равным квадратному корню суммы квадратов радиусов размытия, которые были фактически применены. К примеру, применение последовательных гауссовых размытий с радиусами 6 и 8 даёт такие же результаты, как применение одного гауссового размытия с радиусом, равным 10, так как корень суммы квадратов 6 и 8 равен 10.

Из-за этого отношения, вычислительное время не может быть сокращено моделированием размытия Гаусса с последовательными малыми размытиями, так как время, необходимое для вычислений, будет не меньше, чем выполнение одного большого размытия Гаусса.

Гауссово размытие часто используется при уменьшении размеров изображения. При уменьшении разрешения изображения, нормой является применение к изображению фильтра низких частот до изменения разрешения. Это делается для того, чтобы удостовериться, что ненужная высокочастотная информация не появится в уменьшенном изображении. Гауссово размытие имеет полезные свойства, такие как отсутствие острых краёв, что позволяет отфильтровать подобный высокочастотный шум.

Пример использования размытия Гаусса приведён на рисунке 3.2:

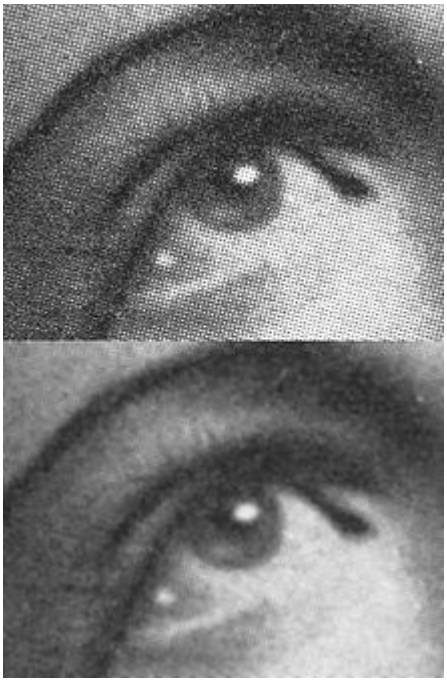

Рисунок 3.2 – Пример применения размытия Гаусса

На рисунке 3.3 приведены примеры результатов размытия изображения фильтром Гаусса с различными размерами матрицы свёртки:

Рисунок 3.3 – Пример результатов размытия Гаусса с различными размерами матрицы свёртки

Положительным качеством данного метода является то, что он сочетается со многими другими методами компьютерного зрения, так как позволяет уменьшить уровень шума на изображении.

3.2 Выделение границ

Все существующие методы поиска границ на изображении так или иначе основаны на использовании производных.

Чтобы понять, зачем нам нужны производные можно разобрать следующий пример. На рисунке 3.4 представлено одномерное изображение с кривой, отражающей интенсивность его пикселей:

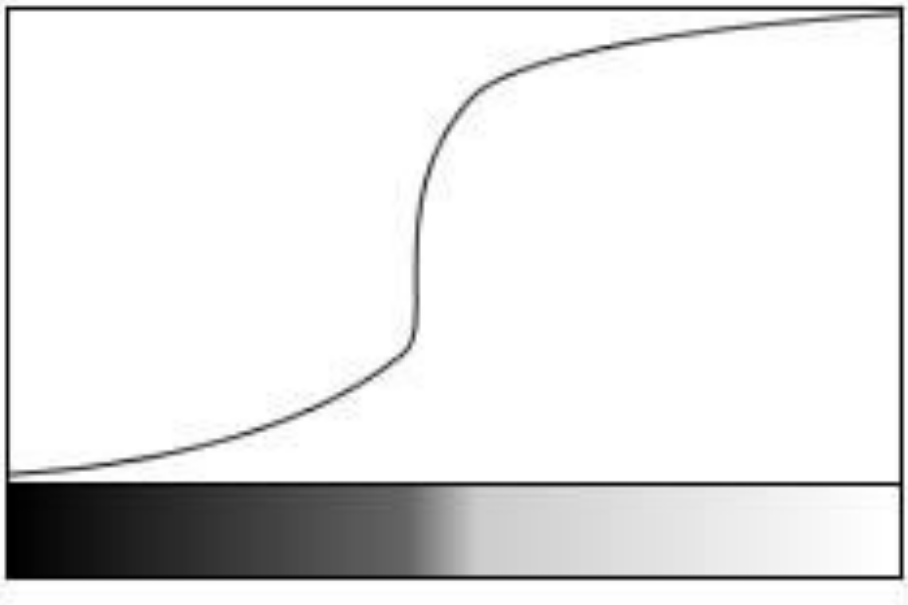

Рисунок 3.4 – Кривая, отображающая интенсивность яркости пикселей

Как можно заметить на приведённом изображении, кривая интенсивности поднимается на протяжении всего изображения, однако следует отметить, что в середине кривой есть резкий её подъём. Этот подъём отражает наличие границы на изображении.

Теперь, рассмотрим график первой производной от данной кривой. Он представлен на рисунке 3.5:

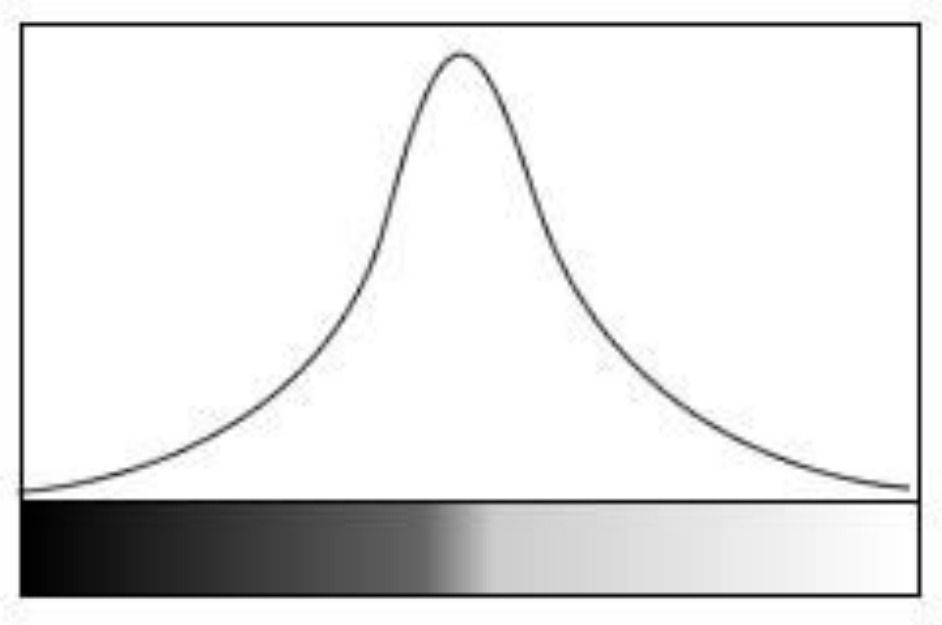

Рисунок 3.5 – Первая производная кривой интенсивности

Легко заметить, что график первой производной достиг максимума в той точке, в которой мы наблюдали резкий подъём интенсивности яркости пикселей на изображении 3.5. Подобное поведение графика легко объяснить, если вспомнить, что производная – есть скорость изменения функции в точке. Из этого следует, что своего максимального значения она достигнет в той точке кривой, разность высоты которой с соседней будет максимальной.

Именно эти точки с максимальным значением разности и ищут методы, основанные на использовании первой производной. Если разность достаточно велика, то производная в данной точке достигает пика, что позволяет выделить границу.

Однако, у подобных методов есть недостаток, а именно определение, что есть граница, а что – нет. Должна быть определённая грань, позволяющая определить, относится ли пик производной к границе на изображении, или его следует считать частью шума изображения. Теперь взглянем на кривую второй производной, представленную на рисунке 3.6:

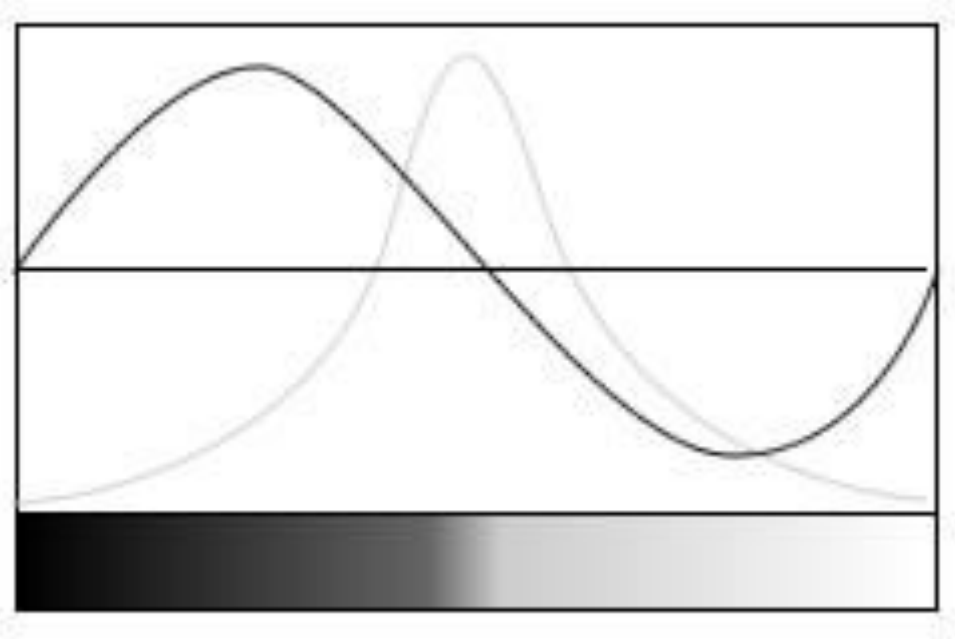

Рисунок 3.6 – Производная второго порядка

Как мы видим на рисунке 3.6, представляющем собой производную второго порядка, производная слева от максимума представляет собой положительно-направленный наклон, а справа от него – отрицательно-направленный. Это отражается на кривой производной второго порядка, которая отражает резкость этих наклонов, пересекая ноль в той точке, в которой положительный наклон переходит в отрицательный, то есть, в точке предполагаемой границы. Методы, основанные на использовании производных второго порядка, используют эту идею о наличии пересечения нуля для поиска границ. Разберём это с математической точки зрения.

Разберём пример одномерной функции *f(x)*. Для определения её первой производной нам следует вычислить разницу значений для всех соседних элементов на изображении (3.6):

$$\frac{\partial f}{\partial x} = f\ x+1\ - f(x) \tag{3.6}$$

Здесь используется частная производная. Она используется с целью упрощения работы с ассигнациями для переменных, то есть *f(x, y)*, в том случае,

если у нас возникает необходимость использования двух осей координат с частными производными.

Как понятно из названия, вторая производная прикладывается к значениям первой производной. Таким образом, вычисление второй производной представляет собой ту же самую операцию, что и вычисление первой производной, но в приложении к соседним значениям результатов её вычисления:

$$\frac{\partial^2 f}{dx^2} = f\ x+1\ + f\ x-1\ - 2f(x) \tag{3.7}$$

Основой для расчёта первой производной для изображения является использование дискретных приближений двумерного градиента. Как мы описывали в предыдущих главах, градиент изображения *f(x, y)* в точке *(x, y)* представляет собой некоторый вектор (3.8):

$$\nabla f = \frac{G_x}{G_y} = \frac{\frac{\partial f}{\partial x}}{\frac{\partial f}{\partial y}} \tag{3.8}$$

По определению производной, траектория подобного вектора будет совпадать с траекторией кривой скорости изменения значения функции *f* в точке *(x, y)*. Знак при этом не играет роли, в связи с чем используется модуль вектора. Таким образом, он играет важную роль в задаче определения контуров на изображении. Этот вектор обозначается символом ∇f и представлен на (3.9):

$$\nabla f = \nabla f = \overline{G_x^2 + G_y^2} \tag{3.9}$$

В теории всё вышеописанное позволит нам без проблем находить границы объектов на изображениях, однако, всё вышеперечисленное можно приложить лишь к непрерывным изображениям. Но таких изображений не существует, в связи с чем нам приходится апроксимизировать производные, основываясь на пиксельных изображениях, которые мы имеем. Это можно сделать с помощью матричных фильтров, то есть с помощью свёртки изображений.

Таким образом, каждый из методов нахождения границ на изображении базируется на процессе пространственной фильтрации. Одним из наиболее часто используемых способов пространственной области основан на применении фильтров (масок свертки, шаблонов, окон). Обычно маска фильтра представляет собой маленькую (к примеру, размерностью 3 на 3) двумерную систему. Коэффициенты для неё выбираются так, чтобы выявить заданное свойство изображения. На рисунках 3.7 и 3.8 представлены абстрактные примеры маски и её коэффициентов:

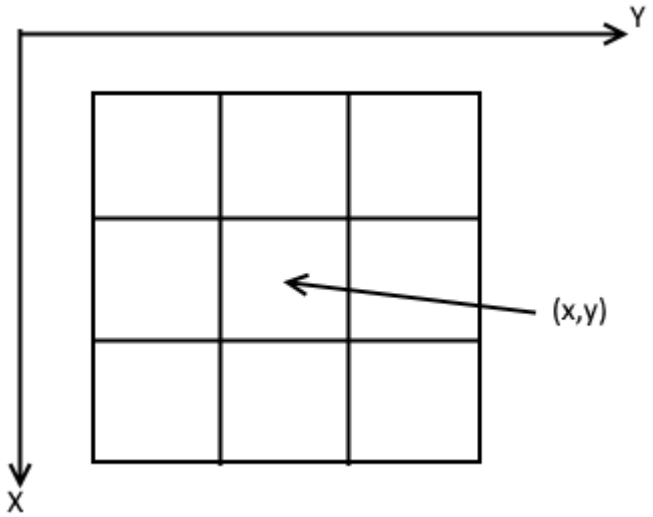

Рисунок 3.7 – Маска фильтра

w1 (x-1,y-1)	w2 (x-1,y)	w3 (x-1,y+1)
w4 (x,y-1)	w5 (x,y)	w6 (x,y+1)
w7 (x+1,y-1)	w8 (x+1,y)	w9 (x+1,y+1)

Рисунок 3.8 – Коэффициенты маски фильтра

Так, пространственная фильтрация проводится с помощью перемещения маски фильтра попиксельно по всему изображению. Каждый из этих пикселей приравнивается к сумме произведений интенсивности окружающих пикселей, попадающих под маску фильтра, на некоторые коэффициенты, содержащиеся в соответствующих позициях маски.

В большинстве способах определения границ на изображении используется матрица пикселей, имеющая размерность 3 на 3.

Матрицы используются при вычислении величин, из которых состоит градиент, а именно G_x и G_y. При этом, для определения значения самого градиента, нам следует использовать эти величины одновременно, как показано в формуле (3.10):

$$\nabla f \approx G_x + |G_y| \tag{3.10}$$

3.3 Оператор Превитта

Оператор Превитта используется для обнаружения границ на изображении. Он обнаруживает два типа границ:

- горизонтальные границы;
- вертикальные границы.

Границы вычисляются с использованием разницы между интенсивностями в соответствующих пикселях изображения. Все маски, которые используются для обнаружения границ на изображении называются производными масками. Изображение это по сути своей сигнал, разница между сигналами может быть может быть вычислена с помощью дифференциации, в связи с чем подобные операторы и называются производными масками или производными операторами.

Все производные маски должны обладать следующими свойствами:

- в маске должен присутствовать противоположный знак;
- сумма маски должна быть равна нулю;
- больший вес ведёт к обнаружению большего числа границ.

Оператор Превитта предоставляет нам две маски. Одна для определения границ в горизонтальном направлении и другая – для определения границ в вертикальном направлении.

$$M = \begin{matrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{matrix} \tag{3.11}$$

Маска, показанная выше, обнаружит края в вертикальном направлении благодаря тому, что нули расположены в виде вертикального столбца. Когда вы свернёте эту маску на изображении, вы получите вертикальные границы на изображении.

Когда мы применяем данную маску к изображению, мы получаем явные вертикальные границы. Оператор просто работает как производная первого порядка и вычисляет разницу в интенсивности пикселей в области границ. Так как центральный столбец маски состоит из нулей, он не учитывает исходные значения границы на изображении, а скорее вычисляет разницу между значениями пикселей слева и справа от границы. Это увеличивает интенсивность границы, и она становится более выраженной, чем на оригинальном изображении.

$$M = \begin{matrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{matrix} \tag{3.12}$$

Маска, приведённая выше, позволит обнаружить границы в горизонтальном направлении, так как нули расположены в горизонтальном направлении. Будучи приложенной к изображению, эта маска выделит яркие горизонтальные границы на изображении.

Эта маска выделит горизонтальные границы изображения. Она работает по тому же принципу, что и вертикальная маска и вычисляет разницу в интенсивности пикселей вокруг некоторой горизонтальной границы. Так как центральный ряд маски состоит из нулей, значения пикселей границы не

учитываются, вместо чего вычисляется разница интенсивности пикселей выше и ниже границы на изображении, таким образом, увеличивая интенсивность границ и делая их заметнее. Обе маски, приведённые выше следуют принципам производных масок. В обеих масках есть отрицательный знак, сумма обеих масок равна нулю. Третье условие не применимо к данному оператору, так как обе маски стандартизированы и не могут изменять свои значения.

Рассмотрим маски в действии.

Мы попеременно применим маски к следующему изображению, показанному на рисунке 3.9:

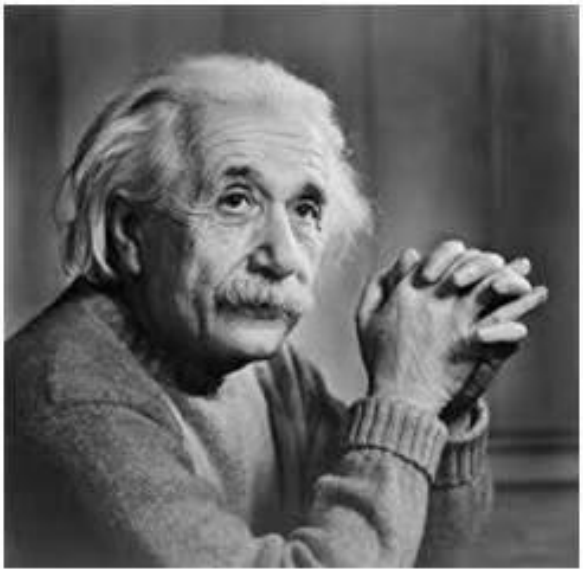

Рисунок 3.9 – Пример изображения до обработки

После применения вертикальной маски на данном изображении, будет получено изображение с рисунка 3.10:

Рисунок 3.10 – Изображение после обработки вертикальной маской

Данное изображение содержит вертикальные границы. Вы можете понять это точнее при сравнении с изображением горизонтальных границ.

При приложении горизонтальной маски к изображению, будет получено изображение, представленное на рисунке 3.11:

Рисунок 3.11 – Изображение горизонтальных границ

Как вы можете видеть, на изображении, полученном приложением вертикальной маски, все вертикальные границы заметнее, чем на оригинальном изображении. Таким же образом, на изображении, полученном приложением горизонтальной маски, горизонтальные границы видны лучше, чем на

оригинальном изображении. Видно, что таким образом мы можем обнаружить и вертикальные и горизонтальные границы на изображении.

Положительным качеством данного метода является его простота.

Отрицательными качествами являются:

- обнаруживает меньшее количество границ по сравнению с другими методами;
- зависит от направления границ, что важно в задаче поиска человека на изображении ввиду его геометрической сложности.

3.4 Оператор Собеля

Оператор Собеля очень похож на оператор Превитта. Он также представляет собой производные маски и используется для определения границ на изображении. Также, как и оператор Превитта, оператор Собеля используется при обнаружении двух типов границ на изображении:

- вертикальные границы;
- горизонтальные границы.

Главное отличие оператора Собеля от оператора Превитта состоит в том, что в операторе Собеля коофициенты в масках не фиксированны и могут быть подстроены в соответствии с нашими требованиями до тех пор, пока это не нарушает свойства производных масок.

Ниже представлена вертикальная маска оператора Собеля:

$$M = \begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix} \qquad (3.13)$$

Эта маска работает точно также, как вертикальная маска оператора Превитта. Единственная разница состоит в том, что в ней содержатся значения “2” и “-2” в центре первого и третьего столбца. Будучи применённой к изображению, данная маска выделит вертикальные границы.

Когда мы применяем эту маску к изображению, она выделяет вертикальные границы. Она работает как производная первого порядка и позволяет вычислить разницу интенсивности пикселей в области границы.

Так как центральный столбец состоит из нулей, значение самой границы не учитывается, вместо чего вычисляется разница пикселей слева и справа от границы. Также, центральные значения первого и третьего столбца равны 2 и -2 соответственно.

Это даёт больше веса значениям пикселей вокруг границы. Это увеличивает яркость границы, и она выделяется сильнее по сравнению с оригинальным изображением.

Ниже представлена горизонтальная маска оператора Собеля:

$$M = \begin{matrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{matrix} \quad (3.14)$$

Маска, показанная выше позволит обнаружить границы в горизонтальном направлении, так как нули в ней расположены горизонтально. Когда вы приложите данную маску к изображению, будут выделены горизонтальные границы изображения.

Данная маска выделит горизонтальные границы изображения. Она также работает по тем же принципам, что и вертикальная маска, и вычисляет разницу между интенсивностями пикселей вокруг некоторой границы. Так как центральный ряд данной маски состоит из нулей, значения самой границы не учитываются, вместо чего вычисляется разница интенсивности пикселей выше и ниже границы. Это увеличивает резкость перехода интенсивности и делает границу заметнее.

Приложим данные маски к изображению на рисунке 3.9.

После наложения вертикальной маски на рисунок 3.9, было получено изображение, представленное на рисунке 3.12:

Рисунок 3.12 – Изображение, полученное путём наложения вертикальной маски Собеля

При наложении горизонтальной маски на изображение с рисунка 3.9, было получено изображение, представленное на рисунке 3.13:

Рисунок 3.13 – Изображение, полученное путём наложения горизонтальной маски Собеля

Как вы можете видеть, на первом полученном изображении, в результате приложения вертикальной маски Собеля, все вертикальные границы выделены сильнее, чем на оригинальном изображении. Таким же образом, на втором полученном изображении, в результате приложения горизонтальной маски Собеля, все горизонтальные границы выделены сильнее, чем на оригинальном изображении.

Таким образом, как вы можете видеть, мы можем обнаружить как вертикальные, так и горизонтальные границы на изображении. Также, если вы сравните результаты работы оператора Собеля с оператором Превитта, вы обнаружите, что оператор Собеля позволил обнаружить больше границ и выделить их сильнее, чем оператор Превитта.

Это произошло потому, что для оператора Собеля мы выделили больший вес пикселям вокруг границ.

Видно, что если мы приложим больше веса к маске, то будет найдено больше границ. Также, как уже было сказано, в операторе Собеля нет фиксированных значений коэффициентов. Поэтому приведём пример другого взвешенного оператора:

$$M = \begin{matrix} -1 & 0 & 1 \\ -5 & 0 & 5 \\ -1 & 0 & 1 \end{matrix} \quad (3.15)$$

При сравнении результатов работы данной вертикальной маски с вертикальной маской Превитта становится понятно, что данная маска выделит больше границ, просто потому, что мы выделили больше веса на маску.

Положительным качеством являются хорошие результаты поиска границ

Отрицательными качествами являются:

- зависимость от направления границ, что важно ввиду геометрической сложности человеческой фигуры;
- относительно большая вычислительная сложность.

3.5 Оператор Лапласа

Лапласиан — это также производный оператор, который используется для поиска границ на изображении. Основное отличие Лапласиана от таких операторов, как оператор Превитта и Собеля в том, что это всё маски производных первого порядка, в то время как Лапласиан представляет собой маску производных второго порядка. Для этой маски у нас имеется две

классификации. Одна из них – положительный оператор Лапласа, другая – отрицательный оператор Лапласа.

Другим отличием Лапласиана от других операторов является то, что он не выделяет границы, расположенные в каком-то направлении, а выделяет границы по следующей классификации:

- внутренние границы;
- внешние границы.

Рассмотрим, как работает оператор Лапласа.

В позитивном операторе Лапласа у нас имеется стандартная маска, в которой центральный элемент маски должен быть негативным, а угловые элементы должны быть равны нулю.

$$M = \begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix} \tag{3.16}$$

Позитивный оператор Лапласа используется для получения внешних границ на изображении.

Для негативного оператора Лапласа мы также имеем стандартную маску, в центре которой находится позитивный элемент. Все элементы в углах должны быть равны нулю, а все остальные элементы маски должны быть равны -1.

$$M = \begin{matrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{matrix} \tag{3.17}$$

Негативный оператор Лапласа используется для получения внутренних границ изображения.

Оператор Лапласа — это производный оператор. При его использовании подчёркиваются разрывы уровня серого на изображении и отбрасываются регионы со слишком медленно меняющимся уровнем серого. Такая операция в результате производит изображения, на которых границы представлены серыми линиями на чёрном фоне. Это показывает внутренние и внешние границы на изображении.

Что важно – как применить эти фильтры к изображению. Помните, что мы не можем применить положительный и отрицательный оператор Лапласа к одному и тому же изображению. Мы должны применять только один, но мы должны помнить, что если мы хотим получить изображение с обострёнными границами, то при применении позитивного оператора Лапласа, мы должны вычесть изображение, полученное в результате из исходного изображения, а при применении отрицательного оператора Лапласа нам следует добавить изображение, полученное в результате обработки к оригинальному.

Приведём примеры использования этих операторов. В качестве исходного изображения мы рассмотрим изображение на рисунке 3.9.

При применении позитивного оператора Лапласа мы получаем изображение, представленное на рисунке 3.14:

Рисунок 3.14 – Результат использования позитивного оператора Лапласа

При приложении негативного оператора Лапласа мы получаем изображение, представленное на рисунке 3.15:

Рисунок 3.15 – Результат использования негативного оператора Лапласа

Положительными качествами данного метода являются:

- качественное обнаружение границ на изображении;
- независимость от направления границ на изображении, что важно, так как человек является геометрически-сложным объектом;
- скорость работы. Поиск границ оператором Лапласа производится за один проход, в отличие от операторов Превитта, Собеля, или детектора границ Канни.

Отрицательными качествами данного метода являются:

- чувствительность к шуму, более высокая, чем у методов, основанных на использовании первой производной;
- границы, найденные оператором Лапласа, зачастую имеют большую толщину.

3.6 Детектор границ Канни

Одним из самых популярных и эффективных алгоритмов обнаружения границ является алгоритм, описанный Джоном Канни, и названный в его честь. При составлении и описании этого алгоритма, Канни руководствовался тремя критериями, которые, по его мнению, являются залогом хорошей работы детектора границ на изображении:

1) Высокое качество результатов обнаружения, то есть, повышенное отношение сигнала к шуму;

2) Высокое качество локализации границ;

3) Каждая граница на изображении должна вызывать один, и только один отклик алгоритма.

С помощью этих критериев была определена функция стоимости ошибок, с помощью минимизации значений которой, был найден “идеальный” линейный оператор, с помощью которого проводится свёртка с изображением.

В отличие от большинства других алгоритмов, детектор границ Канни не только вычисляет градиент сглаженного изображения, а также удаляются точки, лежащие рядом с границей, для чего используется информация о направлениях границ, что позволяет выделять границу целиком, без разрывов, возможных вблизи локальных максимумов градиента. Также используются два порога, что позволяет избавиться от всяческих слабых границ. Фрагмент границы при этом обрабатывается как целое, что значит, что если на протяжении всего этого фрагмента ни разу не будет удовлетворено условие преодоления этих двух порогов, то этот фрагмент удаляется. Это позволяет уменьшить количество разрывов в конечных границах. Подобное использование шумоподавления с одной стороны создаёт более чёткие и устойчивые результаты, а с другой увеличивает вычислительную нагрузку, создаваемую алгоритмом, а также может привести к излишней потере деталей границ. Таким образом могут слишком сильно закруглиться углы у объектов на изображении, а также разрушиться границы в различных точках соединений.

С учётом всего вышеописанного детектор границ Канни работает следующим образом: сначала происходит сглаживание исходного изображения, которое можно представить в виде функции $f(x, y)$ (3.18):

$$g\ x, y\ = G_\sigma\ x, y\ {}^* f\ x, y \tag{3.18}$$

$$G_\sigma\ x, y\ = \frac{1}{2\pi\sigma^2}\exp\ -\frac{x^2+y^2}{2\sigma^2}\ , \tag{3.19}$$

где σ – это коэффициент сглаживания, чем он больше, тем сильнее сглаживается входное изображение.

После этого производится вычисление градиентов сглаженного изображения. Оно производится в двух направлениях, а именно, в вертикальном g_v и горизонтальном g_H. Это совершается с помощью таких операторов, основанных на использовании первой производной, как операторы Собеля и Превитта, о которых мы говорили ранее. Значения, полученные в результате подобной обработки объединяются по формуле (3.20):

$$E\ i,j\ = \sqrt{g_V\ i,j\ ^2 + g_H\ i,j\ ^2} \tag{3.20}$$

Направление градиента (3.21):

$$\theta\ i,j\ = \tan^{-1}\ \frac{g_V\ i,j}{g_H\ i,j} \tag{3.21}$$

$$E_T\ i,j\ = \begin{matrix} E\ i,j\ \text{если}\ E\ i,j\ > T \\ 0\ \text{иначе} \end{matrix},$$

тут T подбирается так, что элементы границы будут выделены и останутся в конечном результате, одновременно с удалением большей части шумов.

Далее выполняется операция подавления не-максимумов. Для этого осуществляется обнуление градиентов по двум порогам T_1 и T_2, причем с выполнением условия $T_1 > T_2$. Градиент не обнуляется до тех пор, пока его значение в точке меньше, чем T_1 и больше, чем T_2. Эти действия производятся с целью обнаружения наиболее значимых контуров на изображении.

Пример работы детектора границ Канни представлен на рисунке 3.16:

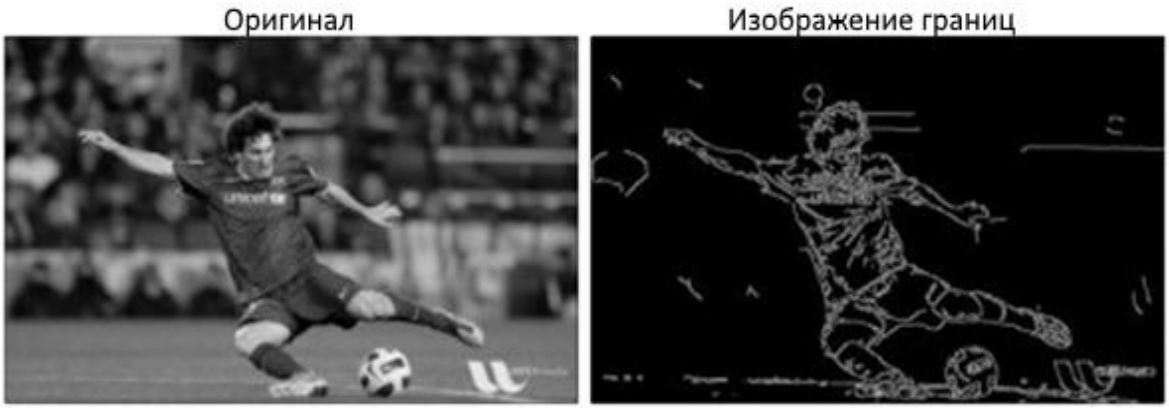

Рисунок 3.16 – Пример работы детектора границ Канни

Положительным качеством данного метода являются очень низкие показатели ошибок при поиске.

Отрицательными качествами являются:

- метод Канни работает медленнее, чем другие методы поиска границ ввиду более высокой вычислительной сложности;
- относительно высокая чувствительность к шуму для некоторых реализаций;
- метод Канни очень часто используется в сочетании с методом Виолы-Джонса, в связи с чем, его применение не внесёт новизны для данной работы.

3.7 RASW – Runtime Adapting Sliding Window

Для метода Виолы-Джонса, объем требуемых вычислений увеличивается с увеличением размера обрабатываемого изображения, что не хорошо при обработке изображений в реальном времени. В связи с этим был разработан метод RASW, который позволяет проводить адаптивное изменение шага сканирующего окна.

В методе Виолы-Джонса это окно двигается по изображению с шагом сканирования Δ. RASW подразумевает различение размера шага вдоль оси x и оси y, т.е. Δ_X и Δ_Y соответственно. Для оставим обозначение Δ, для случаев, когда $\Delta_X = \Delta_Y$. Размер шага сканирования влияет на вероятность обнаружения и на скорость обработки изображения. Большее значение увеличивает скорость обработки, но и также увеличивает вероятность ошибки. Для перемещения сканирующего окна разработчиками метода RASW был найден такой критерий, что оно двигается на больший шаг в тех областях, которые не содержат искомые объекты, и медленнее в близости от объекта. Проведенный создателями RASW анализ позволил обнаружить взаимосвязь между наличием искомого объекта в области изображения и номером классификатора каскада, на котором данное окно считается отброшенным. В

частности, в областях, в которых есть только фон, окно отвергается раньше, чем вблизи с искомым объектом. И чем ближе искомый объект, тем выше ступень выхода.

Ниже представлен алгоритм, приведённый в статье [31] и описывающий работу метода RASW.

```
Require: scaled input image X, classifier cascade S
Ensure: vector V of detected faces (bounding boxes)
for y <- 0 to X.height – subwindow.height do
    Δx <- 1
    for x <- 0 to X.width – subwindow.width do
        Δx <- Δx – 1
        Δy[x] <- Δy[x] – 1
        if(Δx = 0 AND Δy[x] = 0 then
            exit-stage <- S(x,y)
            if exit-stage = n then
                R <- (x,y,width,height)
                push R into V
            end if
            if exit-stage < Δx,t1 then
                Δx = Δx,max
            else if Δx,t1 <= exit-stage < Δx,t2 then
                Δx = Δx,nom
            else
                Δx = Δx,min
            end if
        else if Δx = 0 then
            Δx <- Δx,min
        else if Δy[x] = 0 then
            Δy[x] <- Δy,min
        end if
    end for
end for
```

При работе данного алгоритма классификация производится тем же образом, что и в случае классического метода Виолы-Джонса. Однако, в данном случае в результате классификации классификатор возвращает номер шага, на котором окно было отброшено. Если этот номер равен n, то-есть числу классификаторов, используемых при построении каскада, считается, что

искомый объект найден. При этом положение и размер этого окна записываются в вектор результатов V (строка 10). Окно движется по изображению с определённым шагом, а именно: $\Delta_{x,max}$, $\Delta_{x,min}$ и $\Delta_{x,nom}$, которые используются в зависимости от того, на каком номере классификатора окно было отброшено в предыдущий раз. Для правильного присвоения величины шага в алгоритме хранится информация о номере классификатора, который пометил предыдущее окно, как не содержащее искомый объект. В данной реализации информация, используемая для движения по оси x, хранится в целочисленной переменной Δ_X. Также, к моменту достижения окном края изображения по оси x, должна быть доступна информация о шаге по оси y, применимая ко всей строке. То-есть, Δ_Y — массив целых чисел. На каждом шаге выполнения алгоритма Δ_X и $\Delta_Y[x]$ уменьшаются на единицу, чтобы в тот момент, когда их значение станет равным нулю, окно, на котором выполнится данное условие, было обработано классификатором. При условии, что только одно из этих значений станет равно нулю, это значение принимается за минимальную величину шага (строки 20 и 22).

Положительным качеством данного метода является ускорение работы алгоритма.

Отрицательными качествами данного метода являются:

- сложность реализации;
- при высоком уровне шума на изображении эффективность улучшений данного метода будет уменьшаться.

3.8 Выбор способа улучшения метода Виолы-Джонса

Основываясь на положительных и отрицательных качествах перечисленных способов улучшения работы метода Виолы-Джонса, а также учитывая основные проблемы, с которыми приходится столкнуться при приложении метода Виолы-Джонса к задаче поиска пешеходов на изображении, автор данной работы принял решение о том, что будет

использоваться один из способов изменения входного потока данных. Данное решение было принято в связи с тем, что изменение шага сканирующего окна может привести к результатам, прямо противоположным тем, которых мы стремимся достичь в результате выполнения данной работы, а именно улучшения результатов обнаружения пешеходов на изображении, а также потому, что реализация данного способа представляется весьма громоздкой и долгосрочной, что не допустимо в нашей ситуации.

Из методов модификации входного потока, основываясь на их положительных и отрицательных качествах, автор данной работы принял решение о том, что наилучшим решением будет использовать оператор Лапласа для выделения границ на входных изображениях. Такое решение было принято потому, что ввиду того, что человек на изображении – геометрически-сложный объект, выделение его границ способом, не зависящим от ориентации границ в пространстве будет более предпочтительным, а также потому, что данный способ работает быстрее, чем детектор границ Канни и автор не нашёл примеров использования данного способа модификации входного потока в сочетании с методом Виолы-Джонса.

Однако, оператор Лапласа является очень уязвимым к шумам способом поиска границ на изображении. В связи с этим было принято решение использовать также сглаживающий фильтр Гаусса.

Таким образом, было принято решение использовать сочетание лапласиана и гауссиана, которое представляет собой последовательное использование сглаживающего фильтра Гаусса для уменьшения шумов на изображении и оператора Лапласа для поиска границ на изображении.

Глава 4 Описание выбранного решения

Прежде всего, рассмотрим сочетание выбранных нами методов со стороны алгоритмов.

Было принято решение использовать лапласиан гауссиана на изображении, прежде чем использовать метод Виолы-Джонса, как перед обучением классификатора, так и перед использованием метода Виолы-Джонса для классификации изображения.

Таким образом, приведём модель, получившуюся в результате.

Сначала применяется Гауссово размытие, которое имеет следующую формулу:

$$G_\sigma\ x,y\ = \frac{1}{2\pi\sigma^2} * e^{-\frac{x^2+y^2}{2\sigma^2}}\ , \tag{4.1}$$

где x – это расстояние от начала координат по горизонтальной оси, а y – это расстояние от начала координат по вертикальной оси, σ – это стандартное отклонение гауссова распределения.

Данная операция совершается, чтобы подавить шумы перед использованием Лапласиана для обнаружения границ. Лапласиан представлен следующей формулой:

$$\nabla^2 I\ x,y\ = L\ x,y\ = \frac{\partial^2 I\ x,y}{\partial x^2} + \frac{\partial^2 I\ x,y}{\partial y^2}, \tag{4.2}$$

где x – это расстояние от начала координат по горизонтальной оси, y – расстояние от начала координат по вертикальной оси, $I(x,y)$ – значение интенсивности пикселя с координатами (x,y).

Таким образом, при сочетании данных методов, мы получаем формулу:

$$LoG\ x,y\ = -\ \frac{1}{\pi\sigma^4}\ \ 1-\ \frac{x^2+y^2}{2\sigma^2}\ \ e^{-\frac{x^2+y^2}{2\sigma^2}}\ , \tag{4.3}$$

которую можно коротко записать как применение ядра лапласиана

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

к G_σ:

$$\nabla^2 G_\sigma = \begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix} * G_\sigma, \quad (4.4)$$

где ∇^2 – это лапласиан, G_σ – результат гауссова размытия.

После чего мы можем использовать полученное изображение при расчёте интегрального представления изображения:

$$S\ x,y\ = \sum_{i=0,j=0}^{i\le x,j\le y} \nabla^2 G_\sigma(i,j), \quad (4.5)$$

которое будет использоваться при построении классификатора и непосредственно при классификации изображения с использованием признаков Хаара для каждого признака, выбранного в классификаторе.

4.1 Реализация

Так как описанные выше методы обработки изображений, а именно, гауссово размытие, оператор Лапласа и метод Виолы-Джонса, уже реализованы в свободно-распространяемой библиотеке компьютерного зрения с открытым исходным кодом – OpenCV [1,3,4,5,6], было принято решение использовать её при реализации программы для данной научной работы, чтобы быть уверенными в оптимизированности отдельных её компонент.

Таким образом, в качестве каскада классификаторов использовались специфичные документы XML, пример которых представлен ниже:

```
<stages>
   <!-- stage 0 -->
   <_>
    <maxWeakCount>3</maxWeakCount>
    <stageThreshold>-1.096e+00</stageThreshold>
    <weakClassifiers>
      <_>
        <internalNodes>
          0 -1 28 -5.188e-02</internalNodes>
        <leafValues>7.349e-01 -4.722e-01</leafValues></_>
      <_>
        <internalNodes>
          0 -1 421 -5.404e-02</internalNodes>
        <leafValues>
```

```
            7.812 -2.558</leafValues></_>
    </weakClassifiers></_>
</stages>
```

где *maxWeakCount* – число слабых классификаторов на уровне каскада, *stageThreshold* – максимальный порог яркости для текущего уровня каскада, *weakClassifiers* – список, представляющий собой описание слабых классификаторов, участвующих в принятии решения о том, содержит ли обрабатываемое окно объект, *internalNodes* – параметры каждого такого классификатора, где третье число является номером признака из общей таблицы признаков Хаара, в то время как четвёртое число является пороговым значеним классификатора, которое используется для отсеивания признаков, значение которых ниже данного порога. Если признак отсеен, то выбирается первое взвешенное значение *leafValues*, в то время как если значение признака больше порога, то выходное значение классификатора принимается как второе взвешенное значение leafValues.

Обучение классификаторов, как обычного, так и для модифицированного метода, происходило с помощью выборки изображений, состоящей из 1000 изображений, на которых были изображения пешеходов, и 2000 изображений, на которых их не было.

Таким образом, было получено два классификатора, каждый из которых использовался двумя разными методами, с использованием предложенной модификации и без неё. Приведём ниже фрагмент реализации метода без модификаций:

```
CascadeClassifier fullbody_cascade;
//Инициализация каскада классификаторов, с передачей пути
до каскада в формате XML в переменной
fullbody_cascade_name
fullbody_cascade.load(fullbody_cascade_name);
std::vector<Rect> full_body;
Mat frame_gray;
Mat frame;
//Приведение изображения к оттенкам серого
cvtColor(image, frame_gray, COLOR_BGR2GRAY);
```

```
//Конвертация изображения к восьмибитной глубине
frame_gray.convertTo(frame_gray, CV_8U);
//Поиск искомых объектов на изображении frame_gray, в
вектор full_body записываются координаты каждого
найденного объекта
fullbody_cascade.detectMultiScale(frame_gray, full_body,
1.05, 3, 0, Size(100, 200), Size(300, 800));
if (full_body.size() != 0) {
    for (size_t i = 0; i < full_body.size(); i++)
    {
        //Перебор найденных объектов, вычисление
параметров для выделения их рамками на исходном
изображении
        Point center(full_body[i].x + full_body[i].width
/ 2, full_body[i].y + full_body[i].height / 2);
        ellipse(frame_gray, center,
Size(full_body[i].width / 2, full_body[i].height / 2), 0,
0, 360, Scalar(255, 0, 255), 4, 8, 0);
    }
}
//Запись изображения с выделением найденных искомых
объектов
imwrite(endname, frame_gray);
```

В приведённом коде мы видим, что изображение, посылаемое на вход, без каких-либо изменений переходит к поиску искомых объектов, в процессе которого изображение приводится к интегральному представлению, которое затем подвергается обработке каскадом классификаторов *fullbody_cascade*, на основе работы которой принимается решение о наличии и положении искомых объектов.

Теперь, приведём фрагмент кода, в котором исходное изображение подвергается предложенным автором модификациям, а именно размытию фильтром Гаусса и последующему выделению важных параметров изображения с помощью выделения границ оператором Лапласа:

```
CascadeClassifier fullbody_laplacian_cascade;
fullbody_laplacian_cascade.load(fullbody_laplacian_cascad
e_name);
std::vector<Rect> fullbody_laplacian;
Mat frame_gray;
```

```
Mat frame;
Mat dst;
cvtColor(image, frame_gray, CV_RGB2GRAY);
//Применение гауссова размытия к изображению
GaussianBlur(frame_gray, frame_gray, Size(7, 7), 1.4,
1.4, BORDER_DEFAULT);
//Создание изображения dst, содержащего границы размытого
гауссианом исходного изображения
Laplacian(frame_gray, dst, CV_16S, 3, 1, 10,
BORDER_DEFAULT);
//Конвертация изображения к восьмибитной глубине
dst.convertTo(dst, CV_8U);
//Поиск искомых объектов на изображении dst, в вектор
fullbody_laplacian записываются координаты каждого
найденного объекта
fullbody_laplacian_cascade.detectMultiScale(dst,
fullbody_laplacian, 1.05, 3, 0, Size(100, 200), Size(300,
800));
if (fullbody_laplacian.size() != 0) {
    for (size_t i = 0; i < fullbody_laplacian.size();
i++)
    {
        //Перебор найденных объектов, вычисление
параметров для выделения их рамками на исходном
изображении
        Point center(fullbody_laplacian[i].x +
fullbody_laplacian[i].width / 2, fullbody_laplacian[i].y
+ fullbody_laplacian[i].height / 2);
        ellipse(frame_gray, center,
Size(fullbody_laplacian[i].width / 2,
fullbody_laplacian[i].height / 2), 0, 0, 360, Scalar(255,
0, 255), 4, 8, 0);
    }
}
imwrite(endname, frame_gray);
```

В приведённых примерах мы видим, что параметры, передаваемые функции, реализующей метод Виолы-Джонса, помимо исходного изображения и массива для записи найденных объектов, следующие:

- scaleFactor, являющийся коэффициентом, определяющим, насколько будет уменьшено изображение при каждом пересчёте его размера. В нашем случае данный параметр выбран равным 1.05, в то время как

значение по-умолчанию равно 1.1. Данный выбор был сделан, чтобы минимизировать потерю значимых деталей изображения при поиске объектов;

- minNeighbors, являющийся параметром, определяющим, сколько у определённого обнаруженного прямоугольника должно быть соседних прямоугольников, чтобы классификатор определил его, как содержащий искомый объект. Установлено значение, равное 3, чтобы уменьшить количество ложно-положительных результатов работы;
- flags – служебный параметр OpenCV, отвечает за формат используемого каскада;
- minSize – минимальный размер искомого объекта на изображении в пикселях. Установлен размер 100 на 200 пикселей, так как объекты меньшего размера в выборке не ожидаются, в то время как установка оптимального размера искомого объекта позволяет ускорить поиск;
- maxSize – максимальный размер искомого объекта на изображении в пикселях. Установлен размер 300 на 800 пикселей, так как даже на максимальном разрешении тестовых изображений, то есть при отсутствии масштабирования, размеры искомых объектов на них не превышают данных значений.

Работа с тестовыми изображениями производится в цикле, показанном ниже:

```
//Цикл, обрабатывающий все тестовые изображения. В
векторе arr содержатся имена всех файлов тестовых
изображений.
for (int i = 0; i < arr.size(); i++) {
    //Файлы изображений находятся по относительному пути
в папке программы, переданном как элемент вектора arr
    char path[50];
    strcpy(path, arr[i].c_str());
    image = imread(path, IMREAD_COLOR);
    if (image.empty()) {
        cout << "couldn't open image" << endl;
```

```
    }
    //Происходит последовательный поиск искомых объектов
на открытом изображении, сначала - классическим методом,
потом - модифицированным.
    detectAndDisplay(image);
    detectAndDisplayLaplacian(image);
    //В результате работы методов создаются новые файлы
изображений в оттенках серого, на которых овалами
отмечены обнаруженные объекты.
    //Данные файлы имеют имена, зависящие от метода,
которым они были созданы
    //По завершению обработки изображения, оно
освобождается, во избежание утечек памяти
    image.release();
}
```

В итоге работы реализованной программы мы получаем два набора изображений, один – обработанный классическим методом Виолы-Джонса, а другой – обработанный модифицированным методом, использующим размытие и выделение границ. Результаты работы программы рассмотрены в следующем подразделе.

4.2 Результаты работы реализации

Таким образом, было обучено два классификатора со всеми одинаковыми входными параметрами для обучения, кроме изображений. Для классификатора, который должен использоваться совместно с предлагаемой модификацией, обучающая выборка претерпела обработку в виде преобразования её изображений в изображения границ, путём использования на них фильтра размытия Гаусса и оператора Лапласа, в то время как для контрольного классификатора использовались обычные изображения. Использовалось 1000 изображений, на которых присутствовали искомые объекты и 2000 изображений, не имеющих искомых объектов.

На вход программе, идентифицирующей изображения двумя способами – модифицированным с помощью лапласиана методом Виолы-Джонса и контрольным, не претерпевшим изменений методом, было передано 289 изображений, содержащих искомые объекты, а именно людей.

Приведём сравнительные примеры работы контрольного и модифицированного методов на рисунках с 4.1 по 4.20:

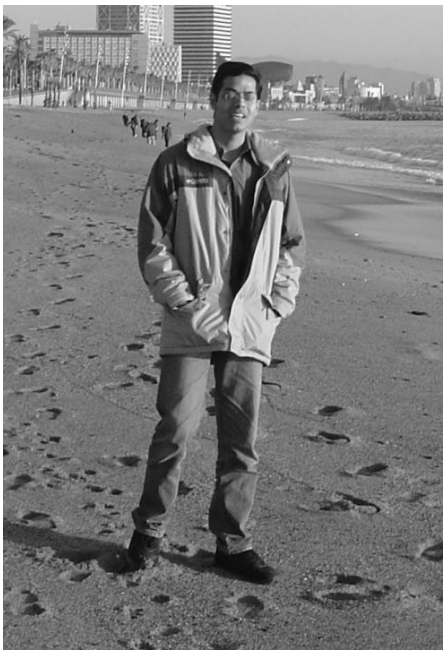

Рисунок 4.1 – Тестовое изображение №1. Обработка контрольным методом

Рисунок 4.2 – Тестовое изображение №1. Обработка модифицированным методом

Как можно увидеть на примере рисунков 4.1 и 4.2, модифицированный метод оказался способен обнаружить искомый объект на изображении, в то время как контрольный метод не обнаружил ничего. Однако, мы видим, что модифицированному методу всё ещё не хватает точности для более чёткого определения позиции объекта на данном изображении.

Рисунок 4.3 – Тестовое изображение №2. Обработка контрольным методом

Рисунок 4.4 – Тестовое изображение №2. Обработка модифицированным методом

Примеры, приведённые выше показывают, что модифицированный метод позволяет получать больше удовлетворительных значений признаков хаара в классификаторе, чем контрольный метод, что в свою очередь позволяет модифицированному методу обнаруживать искомые объекты там, где их не обнаруживает контрольный метод.

Рисунок 4.5 – Тестовое изображение №3. Обработка контрольным методом

Рисунок 4.6 – Тестовое изображение №3. Обработка модифицированным методом

Пример, приведённый на рисунках 4.5 и 4.6 показывает, что модифицированный метод имеет более высокую вероятность ложноположительных результатов, чем контрольный.

Рисунок 4.7 – Тестовое изображение №4. Обработка контрольным методом

Рисунок 4.8 – Тестовое изображение №4. Обработка модифицированным методом

Пример, приведённый на рисунках 4.7 и 4.8 подкрепляет пример тестового изображения №3, приведённый на рисунках 4.5 и 4.6. Модифицированный метод демонстрирует более высокую вероятность ложноположительных результатов, что приводит к обнаружению несуществующих объектов, как мы видим на рисунке 4.8.

Рисунок 4.9 – Тестовое изображение №5. Обработка контрольным методом

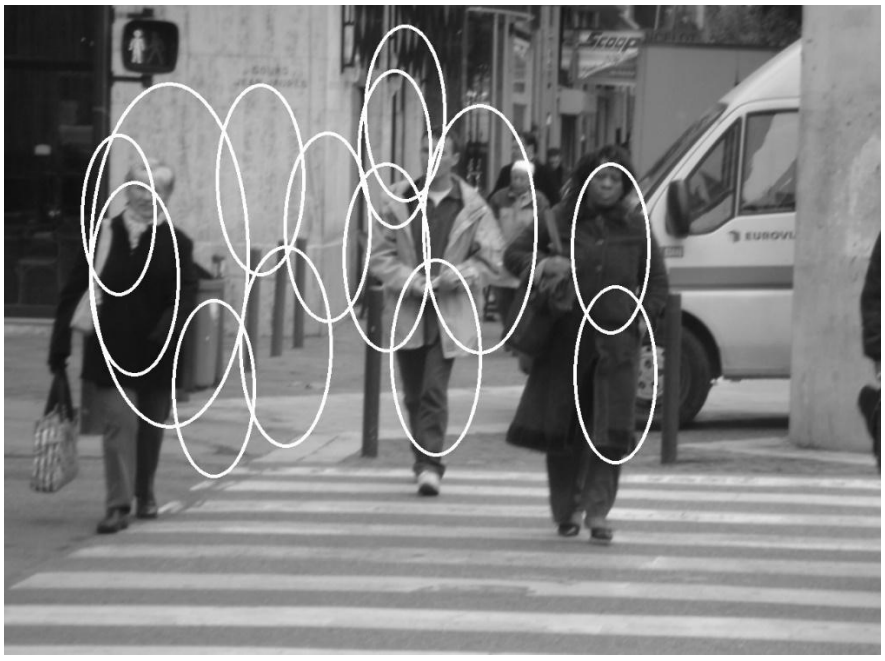

Рисунок 4.10 – Тестовое изображение №5. Обработка модифицированным методом

Рисунки 4.9 и 4.10 демонстрируют ещё более высокий процент ложноположительных результатов работы модифицированного метода. В то время как все искомые объекты были найдены на изображении, количество ложноположительных результатов слишком велико.

Рисунок 4.11 – Тестовое изображение №6. Обработка контрольным методом

Рисунок 4.12 – Тестовое изображение №6. Обработка модифицированным методом

Рисунки 4.11 и 4.12 демонстрируют менее высокую вероятность ложноотрицательных результатов классификации, что позволяет обнаруживать большее число искомых объектов.

Рисунок 4.13 – Тестовое изображение №7. Обработка контрольным методом

Рисунок 4.14 – Тестовое изображение №7. Обработка модифицированным методом

Рисунки 4.13 и 4.14 демонстрируют низкую вероятность ложноотрицательных результатов для модифицированного метода.

Рисунок 4.15 – Тестовое изображение №8. Обработка контрольным методом

Рисунок 4.16 – Тестовое изображение №8. Обработка модифицированным методом

Данное тестовое изображение демонстрирует обнаружение большего числа искомых объектов модифицированным методом, однако, при этом отмечается наличие ложноположительных результатов.

Рисунок 4.17 – Тестовое изображение №9. Обработка контрольным методом

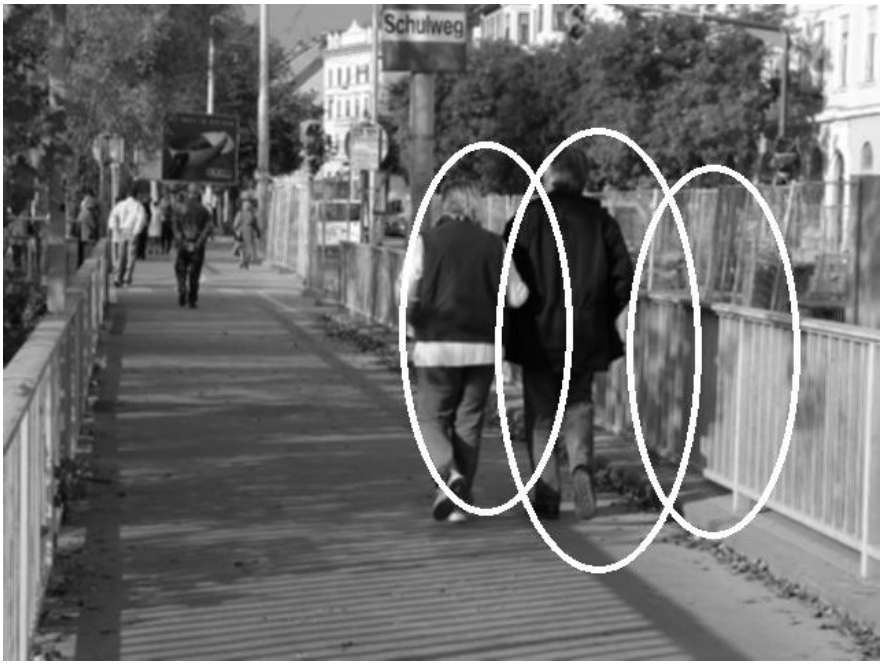

Рисунок 4.18 – Тестовое изображение №9. Обработка модифицированным методом

Данные рисунки также демонстрируют более низкую вероятность ложноотрицательного результата у модифицированного метода по сравнению с контрольным, однако и более высокую вероятность ложноположительного

результата ввиду того, что модифицированный метод более чувствителен к помехам вроде теней или вертикально-стоящих объектов.

Рисунок 4.19 - Тестовое изображение №10. Обработка контрольным методом

Рисунок 4.20 - Тестовое изображение №10. Обработка модифицированным методом

Рисунки 4.19 и 4.20 показывают, что при достаточном удалении от камеры, ни контрольный, ни модифицированный методы не смогут обнаружить искомые объекты, при этом принимая ложноотрицательные выводы.

В результате проверки работы контрольного метода и модифицированного метода, были получены кривые ROC, изображённые на рисунке 4.21:

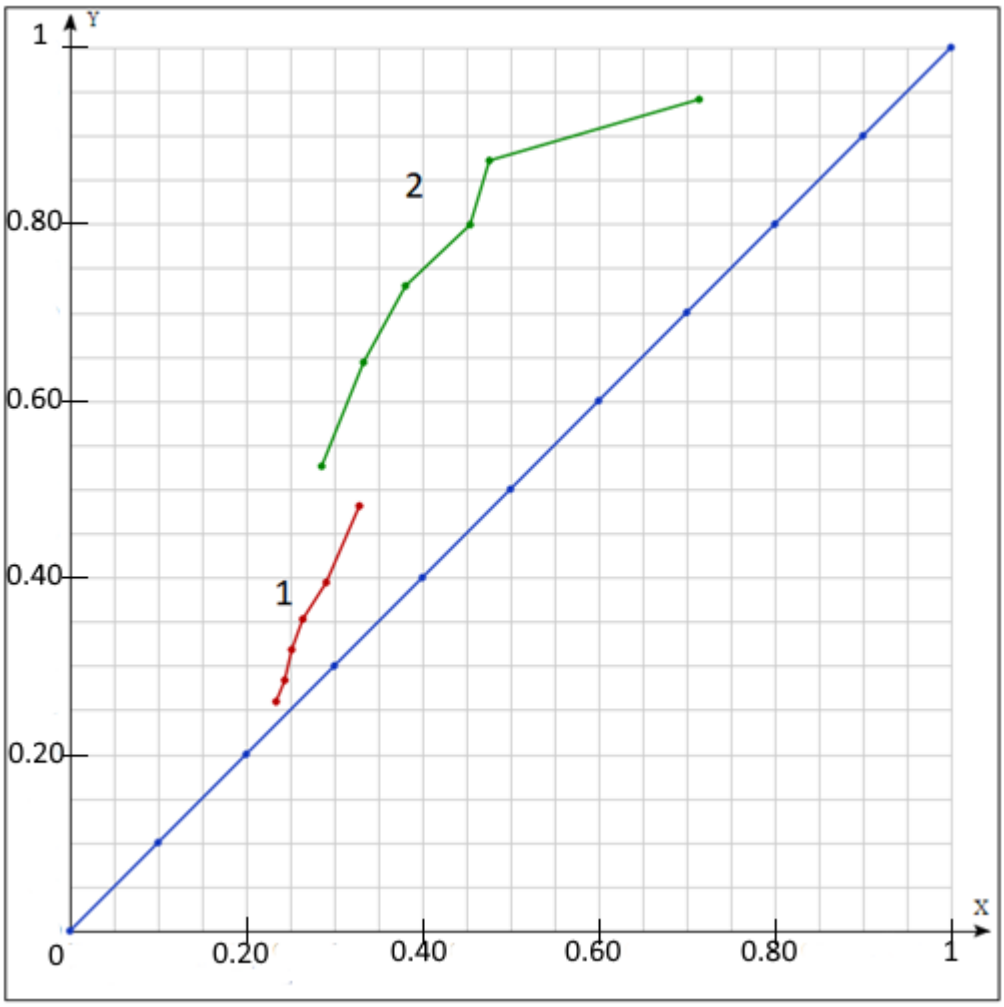

Рисунок 4.21 – кривая ROC проверки работы методов. Кривая под цифрой 1 описывает работу контрольного метода. Кривая под цифрой 2 описывает работу модифицированного метода

Заключение

На основании приведённых примеров установлено, что применение предложенного автором метода приведёт к увеличению TPR (True Positive Response, или верный позитивный ответ) выводов о наличии человека на изображении, однако при этом увеличивается FPR (False Positive Response, или ложный положительный ответ), что недопустимо в задаче поиска человека на изображении.

Однако, проблема высокого FPR, при использовании методов машинного обучения решаема с помощью увеличения обучающей выборки. Подобные результаты эксперимента позволяют сделать вывод о том, что выбранная модификация метода Виолы-Джонса требует большой обучающей выборки, чтобы избежать проблемы высокого FPR, однако даже в таком виде, она показывает результаты работы, превосходящие результаты работы классического метода, что позволяет говорить о возможности применения данного метода в различных прикладных задачах.

Список используемой литературы

1. Bradsky, G. Learning OpenCV: Computer Vision with OpenCV Library / G. Bradsky, A. Kaehler. – O'Reilly Media, 2008. – 555 p.

2. Ponce, J. Computer Vision: A Modern Approach / J. Ponce, D.A. Forsyth. - Prentice Hall Professional Technical Reference, 2002. – 720 p.

3. Szeliski, R. Coputer Vision: Algorithms and Applications / R. Szeliski. – Springer, 2011. - 812 p.

4. Форсайт Д.А. Компьютерное зрение. Современный подход / Д.А. Форсайт. –М.: Вильямс, 2004. – 928 с.

5. Гонсалес Р. Цифровая обработка изображений в среде MATLAB / Р. Гонсалес, Р. Вудс, С. Эддинс. –М.: Техносфера, 2006. - 616 с.

6. Дьяконов В.П. Matlab 6.5 SP1/7 + Simulink 5/6. Работа с изображениями и видеопотоками / В.П. Дьяконов. -М.: СОЛОН-Пресс, 2005. - 400с.

7. Яншин В.В. Анализ и обработка изображений, принципы и алгоритмы / В.В. Яншин. –М.: Машиностроение, 1995. – 111с.

8. Визильтер Ю.В. Обработка и анализ изображений в задачах машинного зрения / Ю.В. Визильтер. –М.: Физматкнига, – 2010. – 689с.

9. Фисенко В. Т. Компьютерная обработка и распознавание изображений: учебное пособие / В.Т. Фисенко. –СПб: СПбГУ ИТМО, 2008. -195 с.

10. Papageorgiou C., Poggio T. A trainable pedestrian detection system[текст] / C. Papageougiou, T. Poggio // Intelligent Vehicles. – October 1998, pages 241-246.

11. Viola P., Jones M. Rapid Object Detection using a Boosted Cascade of Simple Features[текст] / P. Viola, M. Jones // Accepted Conference on Computer Vision and Pattern Recognition. – 2001. – 9 p.

12. Lee A. Comparing Deep Neural Networks and Traditional Vision Algorithms in Mobile Robotics[текст] / A. Lee // Swarthmore College. – 2015. – 9 p.

13. Yi-Qing W. An Analysis of the Viola-Jones Face Detection Algorithm[текст] / W. Yi-Quing // Image Processing On Line. – 2014. – 148 p.

14. Pedro F. Visual Object Detection with Deformable Part Models[текст] / F. F. Pedro // Communications of the ACM. 2013. – 97-105 p.

15. David G. L. Distinctive Image Features from Scale-Invariant Keypoints[текст] / G. L. David // Computer Science Department University of British Columbia. 2004. – 28 p.

16. Herbert B., Andreas S. Speeded-Up Robust Features (SURF)[текст] / B. Herbert, S. Andreas // Katholieke Universiteit Leuven. 2008. – 14 p.

17. Pablo N. Benchmarking Haar and Histograms of Oriented Gradients Features Applied to Vehicle Detection[текст] / N. Pablo // Universite Pierre et Marie Curie-Paris. 2007. – 6 p.

18. Navneet D., Bill T. Histograms of Oriented Gradients for Human Detectionx[текст] / D. Navneet, T. Bill // IEEE Computer Society Washington, DC, USA. 2005. – 8 p.

19. Alberto D. C., Roberto A. V. Intelligent Computing Systems[текст] / D. C. Alberto, A. V. Roberto // Springer International Publishing. 2016. – 153 p.

20. Abdul-Lateef Y. Advances in Computer Science and its Applications[текст] / Y. Abdul-Lateef // Springer Berlin Heidelberg. 2014. – 1415 p.

21. Simone F. VOCUS: A Visual Attention System for Object Detection and Goal-Directed Search[текст] / F. Simone // Springer Berlin Heidelberg. 2006. – 218 p.

22. Modesto C., Oscar D., Daniel H., Javier L. A comparison of face and facial feature detectors based on the Viola–Jones general object detection framework[текст] / C. Modesto, D. Oscar, H. Daniel, L. Javier // Springer-Verlag. 2011. – 14 p.

23. Alexandre P. M. Bioinspired Computation in Artificial Systems[текст] / P. M. Alexandre // Springer International Publishing. 2015. – 485 p.

24. Ryoji K., Hiroki S., Masayuki H. Hardware Architecture for HOG Feature Extraction[текст] / K. Ryoji, S. Hiroki, H. Masayuki // IEEE Computer Society. 2009. – 34 p.

25. Forrest N. I., Matthew W. M., Kurt K. Energy-Efficient Histogram of Oriented Gradient Computation[текст] / N. I. Forrest, W. M. Matthew, K. Kurt // University of California, Berkeley. 2015. – 7 p.

26. Li-Chuan G., Ze-xun G., Guo-xi W. Compressed Binary Discriminative Feature for Fast UAV Image Registration[текст] / G. Li-Chuan, G. Ze-xun, W. Guo-xi // Springer International Publishing. 2015. – 16 p.

27. Kyung-mo K., Eui-young C. A Multiple Object Tracking Method using SURF and Color Histogram[текст] / K. Kyung-mo, C. Eui-young // Computer Engineering, Pusan National University. 2014. – 4 p.

28. Paul V., Michael J. J. Robust Real-Time Face Detection[текст] / V. Paul, J. J. Michael // Kluwer Academic Publishers. 2004. – 18 p.

29. Francesco C., Sander S., Twan B., Henk C. RASW: a Run-time Adaptive Sliding Window to Improve Viola-Jones Object Detection[текст] / C. Francesco, S. Sander, B. Twan, C. Henk // Electronic Systems Group, Eindhoven University of Technology. 2014. – 6 p.

30. Moghimi M.M., Nayeri M., Pourahmadi M., Moghimi M.K. Moving Vehicle Detection Using AdaBoost and Haar-Like Feature in Surveillance Videos[текст] / M.M. Moghimi, M. Nayeri, M. Pourahmadi, M.K. Moghimi // Department of Electrical Engineering, Shahed University, Tehran, Iran. 2018. – 13 p.

31. Lee D.C. Boosted Classifier for Car Detection[текст] / D.C. Lee // ECE Department, Carnegie Mellon University, U.S.A. 2007. – 4 p.

32. Yongzheng X., Guizhen Y., Yunpeng W., Xinkai W., Yalong M. A Hybrid Vehicle Detection Method Based on Viola-Jones and HOG + SVM from UAV Images[текст] / X. Yongzheng, Y. Guizhen, W. Yunpeng, W. Xinkai, M. Yalong // Beijing Key Laboratory for Cooperative Vehicle Infrastructure Systems and Safety

Control, School of Transportation Science and Engineering, Beihang University, Beijing 100191, China; Jiangsu Province Collaborative Innovation Center of Modern Urban Traffic Technologies, SiPaiLou #2, Nanjing 210096, China. 2016. – 23 p.

33. Choudhury S., Chattopadhyay S.P., Hazra T.K. Vehicle detection and counting using haar feature-based classifier[текст] / S. Choudhury, S.P. Chattopadhyay, T.K. Hazra // Industrial Automation and Electromechanical Engineering Conference (IEMECON), Bangkok, Thailand. 2017. – 4 p.

34. Almehmadi T., Htike Z.Z. Vehicle Classification System Using Viola Jones and Multi-Layer Perception[текст] / T. Almehmadi, Z.Z. Htike // Department of Electrical Engineering, University of Malaya, Malaysia; Department of Mechatronics, International Islamic University, Malaysia. 2015. – 7 p.

35. Choi J. Realtime On-Road Vehicle Detection with Optical Flows and Haar-Like Feature Detectors[текст] / J. Choi // Department of Computer Science, University of Illinois at Urbana-Champaign, Urbana. 2006. – 7 p.

36. Moutarde F., Stanciulescu B., Breheret A. Real-time visual detection of vehicles and pedestrians with new efficient adaBoost features[текст] / F. Moutarde, B. Stanciulescu, A. Breheret // 2nd Workshop on Planning, Perception and Navigation for Intelligent Vehicles (PPNIV), at IEEE International Conference on Intelligent Robots Systems (IROS), Nice, France. 2008. – 7 p.

37. Caraffi C., Vojir T., Trefny J., Sochman J., Matas J. A System for Real-time Detection and Tracking of Vehicles from a Single Car-mounted Camera[текст] / C. Caraffi, T. Vojir, J. Trefny, J. Sochman, J. Matas // The Center for Machine Perception, Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Czech Re public. 2012. – 8 p.

38. Hermawati F.A., Budianto H. A Video Based License Plate Detection System Using Viola-Jones Method[текст] / F.A. Hermawati, H. Budianto // Department of Informatics, Faculty of Engineering, University of 17 Agustus 1945 of Surabaya Jl. Semolowaru 45, Surabaya 60119, Indonesia. 2013. – 10 p.

39. Neumann D., Langner T., Ulbrich F., Spitta D., Goehring D. Online Vehicle Detection using Haar-like, LBP and HOG Feature based Image Classifiers with Stereo Vision Preselection[текст] / D. Neumann, T. Langner, F. Ulbrich, D. Spitta, D. Goehring // Department of Computer Science, Freie University, Berlin, Germany. 2017. – 6 p.

40. Han S., Han Y., Hahn H. Vehicle Detection Method using Haar-like Feature on Real Time System[текст] / S. Han, Y. Han, H. Hahn // Electronic Engineering Department, Soongsil University, Seoul, Korea. 2009. – 5 p.

41. Elkerdawi S.M., Sayed R., Elhelw M. Real-Time Vehicle Detection and Tracking Using Haar-Like Features and Compressive Tracking[текст] / S.M. Elkerdawi, R. Sayed, M. Elhelw // Nile University, Ubiquitous Computing Group, Cairo, Egypt. 2014. – 9 p.

42. Liu K., Mattyus G. Fast multiclass vehicle detection on aerial images[текст] / K. Liu, G. Mattyus // Remote Sensing Technology Institute of the German Aerospace Center. 2015. – 5 p.

43. Hao P., Bailing Z. An Integrative Approach to Accurate Vehicle Logo Detection[текст] / P. Hao, Z. Bailing // Journal of Electrical and Computer Engineering. 2013. – 12 p.

44. Keller C.G., Sprunk C., Bahlmann C., Giebel J., Baratoff G. Real-time Recognition of U.S. Speed Signs[текст] / C.G. Keller, C. Sprunk, C. Bahlmann, J. Giebel, G. Baratoff // Intelligent Vehicles Symposium. 2008. – 6 p.

45. Sebastian H., Johannes S., Jan S., Marc S., Christian I. Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark[текст] / H. Sebastian, S. Johannes, S. Jan, S. Marc, I. Christian // The 2013 International Joint Conference on Neural Networks (IJCNN). 2013. – 8 p.

46. Marcus M., Radu T., Rodrigo B., Luc V.G. Traffic sign recognition — How far are we from the solution?[текст] / M. Marcus, T. Radu, B. Rodrigo, V.G.

Luc // The 2013 International Joint Conference on Neural Networks (IJCNN). 2013. – 4 p.

47. Zhang Y., Wang G., Gu X., Zhang S., Hu J. Real-time pedestrian detection with the videos of car camera[текст] / Y. Zhang, G. Wang, X. Gu, S. Zhang, J. Hu // Advances in Mechanical Engineering. 2015. – 9 p.

48. Adrian W.Y.W., Shahirina M.T., Yoong C.C. GPU Acceleration of Real Time Viola-Jones Face Detection[текст] / W.Y.W. Adrian, M.T. Shahirina, C.C. Yoong // IEEE International Conference on Control System, Computing and Engineering, Penang, Malaysia. 2015. – 6 p.

49. Paul A.V., Michael J.J., Daniel S. Detecting Pedestrians Using Patterns of Motion and Appearance[текст] / A.V. Paul, J.J. Michael, S. Daniel // International Journal of Computer Vision. 2005. – 9 p.

50. Chen Y., Chen N., Zhou Y., Zhang W. Pedestrian Detection and Tracking for Counting Applications in Metro Station[текст] / Y. Chen, N. Chen, Y. Zhou, W. Zhang // Discrete Dynamics in Nature and Society. 2014. – 12 p.

51. Kashif Q. Video tracking and person categorisation system[текст] / Q. Kashif // Jazan University. 2013. – 5 p.

52. Goncalo M., Paulo P., Urbano N. Vision-based pedestrian detection using Haar-like features[текст] / M. Goncalo, P. Paulo, N. Urbano // Institute of Systems and Robotic, University of Coimbra, Coimbra, Portugal. 2018. – 6 p.

53. Javier H.A., Leopoldo A., Jose D.P. Pedestrian Detection in Crowded Environments through Bayesian Prediction of Sequential Probability Matrices[текст] / H.A. Javier, A. Leopoldo, D.P. Jose // Hindawi Publishing Corporation, Journal of Sensors. 2016. – 9 p.

54. Rodrigo B., Mohamed O., Jan H., Bernt S. Ten Years of Pedestrian Detection, What Have We Learned?[текст] / B. Rodrigo, O. Mohamed, H. Jan, S. Bernt // Max Planck Institute for Informatics, Saarbrucken, Germany. 2014. – 17 p.

55. Vijay G., Shashikant L. Vision Based Pedestrian Detection for Advanced Driver assistance[текст] / G. Vijay, L. Shashikant // Singhad College of Engineering Research center, Savitribai Phule Pune University, Pune 411 041, Maharashtra, India. 2015. – 8 p.

56. Xiaofei L., Lingxi L., Fabian F., Jianqiang W., Hui X., Morys B., Shuyue P., Dariu M.G., Keqiang L. A Unified Framework for Concurrent Pedestrian and Cyclist Detection[текст] / L. Xiaofei, L. Lingxi, F. Fabian, W. Jianqiang, X. Hui, B. Morys, P. Shuyue, M.G. Dariu, L. Keqiang // IEEE Transactions on Intelligent Transportations Systems. 2016. – 13 p.

57. Bin Y., Junjie Y., Zhen L., Stan Z.L. Aggregate Channel Features for Multi-view Face Detection[текст] / Y. Bin, Y. Junjie, L. Zhen, Z.L. Stan // Center for Biometrics and Security Research & National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, China. 2014. – 8 p.

58. Yonglong T., Ping L., Xiogang W., Xiaoou T. Deep Learning Strong Parts for Pedestrian Detection[текст] / T. Yonglong, L. Ping, W. Xiogang, T. Xiaoou // 2015 IEEE International Conference on Computer Vision (ICCV). 2015. – 9 p.

59. Htike K.K. Hidden-Layer Ensemble Fusion of MLP Neural Networks for Pedestrian Detection[текст] / K.K. Htike // School of Information Technology, UCSI University, Kuala Lumpur, Malaysia. 2017. – 12 p.

60. Antonio B., Domenico B., Gianpaolo F.T., Vitoantonio B. Computer vision and deep learning techniques for pedestrian detection and tracking: A survey[текст] / B. Antonio, B. Domenico, F.T. Gianpaolo, B. Vitoantonio // Department of Electrical and Information Engineering (DEI), Polytechnic University of Bari, Italy. 2018. – 16 p.

61. Vincent F., Manh T.B., Djamal B., Pierrick L. Vision-Based People Detection System for Heavy Machine Applications[текст] / F. Vincent, T.B. Manh, B. Djamal, L. Pierrick // Sorbonne Universités, Université de Technologie de

Compiègne, CNRS, UMR 7253, Heudiasyc-CS 60 319, 60 203 Compiègne Cedex, France. 2016. – 30 p.

62. Yu J., Miyamoto R., Onoye T. A speed-up scheme based on multiple-instance pruning for pedestrian detection using a support vector machine[текст] / J. Yu, R. Miyamoto, T. Onoye // IEEE Transactions on Image Processing. 2013. – 9 p.

63. Chao Z., Yuxin P. Group Cost-Sensitive Boosting for Multi-Resolution Pedestrian Detection[текст] / Z. Chao, P. Yuxin // Proceedings of the Thirtieth AAAI Conference on Artifitial Intelligence (AAAI-16). 2016. – 7 p.

64. Girish N.C., Daruwata R.D., Manoj S.G. Comparisions of Robert, Prewitt, Sobel operator based edge detection methods for real time uses on FPGA[текст] / N.C. Girish, R.D. Daruwata, S.G. Manoj // 2015 International Conference on Technologies for Sustainable Development (ICTSD), Mumbai, India. 2015. – 4 p.