Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей
Кафедра электронных вычислительных машин

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к курсовому проекту
на тему:

СИСТЕМА ДЕТЕКТИРОВАНИЯ ДВИЖЕНИЙ С ИСПОЛЬЗОВАНИЕМ БИБЛИОТЕКИ OPENCV ДЛЯ ОС WINDOWS

Студент: Д. В. Дворник
Руководитель: Д. А. Лавникевич

Минск 2015

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время вычислительная техника используется во многих областях человеческой деятельности, являясь удобным и многофункциональным инструментом решения широкого круга задач. Многие отрасли техники, имеющие отношение к получению, обработке, хранению и передаче информации, в значительной степени ориентируются в настоящее время на развитие систем, в которых информация имеет характер изображений или видео.

Вместе с тем, решение научных и инженерных задач при работе с визуальными данными требует особых усилий, опирающихся на знание специфических методов, поскольку традиционные методы обработки одномерных сигналов мало пригодны в этих случаях. В особой мере это проявляется при создании новых типов информационных систем, решающих такие проблемы, которые до сих пор в науке и технике не решались, и которые решаются сейчас благодаря использованию информации визуального характера. Эти проблемы решает теория компьютерного зрения.

Компьютерное зрение является динамично развивающимся направлением современной науки, востребованным в различных областях, начиная с интеллектуальных человеко-машинных интерфейсов, принятия решений роботами и заканчивая системами автоматического контроля на производстве. Неотъемлемой частью компьютерного зрения является распознавание образов, решающее задачу определения принадлежности входного изображения к одному из хранимых эталонных изображений объектов. При создании интеллектуальных систем также часто требуется отслеживать положение подвижных объектов в реальном времени на основе зрительной информации, полученной от видеокамеры. Располагая рядом последовательных по времени цифровых изображений или видеопотоком, можно выделить специальную информацию об объекте и затем использовать ее для обнаружения текущего положения объекта и отслеживания его перемещений.

Для обеспечения отслеживания движения объектов на видео, достаточно будет организовать последовательную передачу кадров видеопотока с камеры в программу для их обработки, с последующим использованием алгоритмов обнаружения для данной последовательности кадров.

К задачам компьютерного зрения можно отнести и проблему обнаружения на фото или видео человеческих лиц. Интерес к процессам отслеживания и распознавания лиц, всегда был значительным, особенно в связи с все возрастающими практическими потребностями: системы охраны, верификация кредитных карточек, криминалистическая экспертиза, телеконференции

и т.д. Несмотря на ясность того житейского факта, что человек хорошо идентифицирует лица людей, совсем не очевидно, как научить этому компьютер, в том числе как декодировать и хранить цифровые изображения лиц. Так в последние десять лет или около того, распознавание лиц стало популярной областью исследований в компьютерном зрении и одним из самых успешных применений анализа изображений. Данная область привлекает не только исследователей в области информатики, а также неврологов и психологов.

Объектом изучения в рамках данного проекта будет библиотека алгоритмов компьютерного зрения OpenCV (open source computer vision) и методы обнаружения на видео движения и лиц.

В рамках проекта будет создана программа реализующая алгоритмы компьютерного зрения для поиска объектов в видеопотоке, получаемом с веб-камеры.

1 ОБЗОР ЛИТЕРАТУРЫ

В данном разделе будет произведён обзор предметной области задачи, решаемой в рамках курсового проекта; рассмотрены вопросы о способах обнаружения движений, методы анализа и алгоритмы обнаружения лиц и рассмотрены библиотеки обработки визуальной информации.

1.1 Обнаружение движений

Видеоданные проверяются на наличие определенного условия. Например, обнаружение возможных неправильных клеток или тканей в медицинских изображениях. Обнаружение, основанное на относительно простых и быстрых вычислениях, иногда используется для нахождения небольших участков в анализируемом изображении, которые затем анализируются с помощью приемов, более требовательных к ресурсам, для получения правильной интерпретации. Существует несколько специализированных задач, основанных на распознавании текстов, например: Поиск изображений по содержанию: нахождение всех изображений в большом наборе изображений, которые имеют определенное содержание. Содержание может быть определено различными путями, например, в терминах схожести с конкретным изображением (найдите мне все изображения похожие на данное изображение), или в терминах высокоуровневых критериев поиска, вводимых как текстовые данные (найдите мне все изображения, на которых изображено много домов, которые сделаны зимой и на которых нет машин).Оценка положения: определение положения или ориентации определенного объекта относительно камеры. Примером применения этой техники может быть содействие руке робота в извлечении объектов с ленты конвейера на линии сборки [1]. Оптическое распознавание знаков: распознавание символов на изображениях печатного или рукописного текста, обычно для перевода в текстовый формат, наиболее удобный для редактирования или индексации (например, ASCII).

Рассмотрим подробнее методы обнаружения движений

1.1.1 Межкадровая разность. Вычисление межкадровой разности является распространённым методом первичного обнаружения движения, после выполнения которого можно определить присутствует ли в потоке кадров движение [2]. Алогоритм вычисления межкадровой разности выглядит следующим образом: на вход поступают два видеоряда, представляющие собой две последовательности байт. Производится вычисление попиксельных

межкадровых разностей по следующей схеме

$$d_t\,(x,y) = I_t\,(x,y) - I_{t-1}\,(x,y) \tag{1.1}$$

Разность сравнивается с заданным порогом. В результате сравнения формируется двоичная маска:

$$m_t\,(x,y) = \begin{cases} 0, d_t\,(x,y) < T \\ 1, d_t\,(x,y) > T \end{cases} \tag{1.2}$$

где m_t — значение t-го элемента маски, T — порог сравнения, иногда называемый также порогом или уровнем чувствительности.

Однако такой подход даёт достаточно грубую оценку, приводя к наличию неизбежной ложной реакции детектора на шум регистрирующей аппаратуры, смену условий освещения, лёгкое качание камеры и пр. Таким образом, видеокадры должны быть предварительно обработаны перед вычислением разности между ними. Например, преобразование цветов изображения к серой гамме даст большую скорость выполнения операций [3].

Для сравнения могут быть использованы соседние кадры или текущий и базовый кадр (базовый кадр содержит фон).

1.1.2 Метод вычитания фона. Суть алгоритма заключается в разделении изображения на фоновую область и передний план [4]. Существует множество алгоритмов метода вычитания фона, но большинство имеют примерно одинаковую структуру. Основными этапами данного алгоритма являются предобработка, моделирование фона, обнаружение движения и постобработка.

Типовая схема реализации метода вычитания фона представлена на рисунке 1.1.

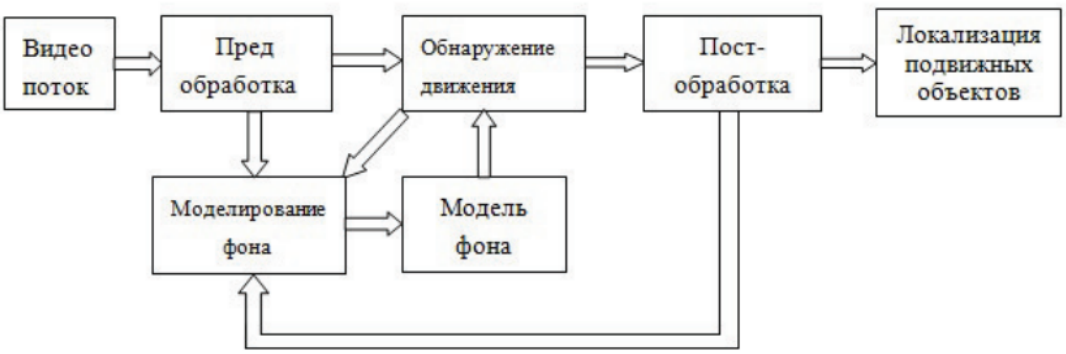

Рисунок *1.1* – Схема обнаружения подвижных объектов.

Моделирование фона.

Фон можно использовать фиксированный - заранее подготовленный кадр. Для всех остальных кадров применяется порог к модулю разности текущего и сохраненного изображения по каждому пикселю.

Для моделирования фона чаще используется метод усреднения. Модель фона будет среднее от первых n кадров:

$$BG\,(x,y) = \frac{1}{n} \cdot \sum_{i=1}^{n} I_i\,(x,y) \tag{1.3}$$

Для того чтобы алгоритм был устойчив к изменению фона, необходимо обновлять фон в течении всего времени на основе начальной модели фона [5].

1.2 Обнаружение лиц

По сути, отслеживание и распознавание лиц является практическим применением теории распознавания образов, в задачу которого входит автоматическая локализация лица на фотографии или видеопотоке и, в случае необходимости, идентификация персоны по лицу. В настоящее время функцию идентификации людей на фотографиях уже активно используют в программном обеспечении для управления фотоальбомами (Picasa, iPhoto и др.) [6]

1.2.1 Метод Виолы-Джонса. В настоящее время метод Виолы—Джонса является самым популярным методом для поиска области лица на изображении в силу своей высокой скорости и эффективности. В основе метода Виолы—Джонса по поиску лица лежат идеи: интегральное представление изображения по признакам Хаара, метод построения классификатора на основе алгоритма адаптивного бустинга, и метод комбинирования классификаторов в каскадную структуру [7]. Эти идеи позволяют осуществлять поиск лица в режиме реального времени.

Признак Хаара состоит из смежных прямоугольных областей. Эти области позиционируются на изображении, далее происходит суммирование интенсивности пикселей в областях, затем между суммами вычисляется разность. Значение полученной разности и является значением определенного признака, определенного размера, определенным образом расположенного на изображении. На рисунке 1.2 представлены граничные, центральные и линейные признаки Хаара. Рисунок 1.3 демонстрирует пример использования признаков Хаара. На пример, для всех изображений, область в районе глаз тем-

нее, чем область в районе щек. Следовательно, общим признаком Хаара для лиц является 2 смежных прямоугольных региона, лежащих на глазах и щеках.

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

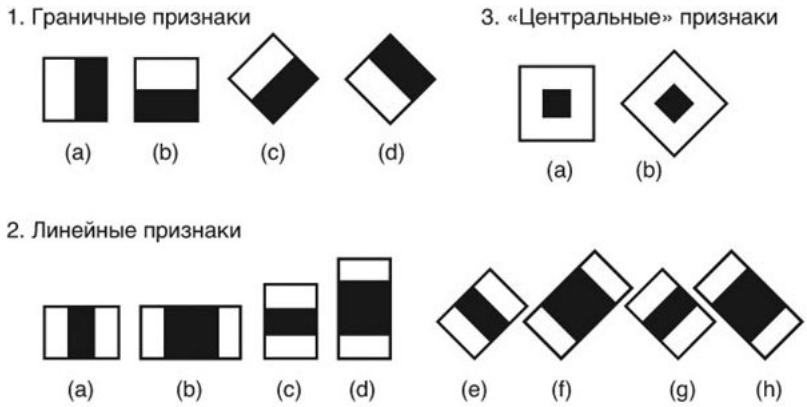

Рисунок *1.2* – Признаки Хаара

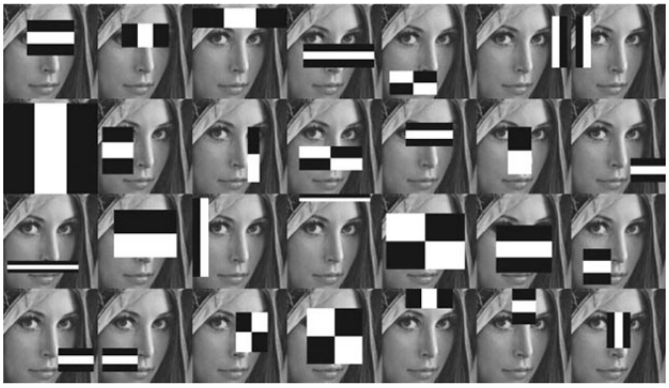

Рисунок *1.3* – Примеры использования признаков Хаара.

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

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

падает. В стандартной реализации метода указанная особенность не позволяет обнаруживать лицо человека, повернутое под произвольным углом, что в значительной мере затрудняет или делает невозможным использование данного метода в современных системах.

1.2.2 Метод главных компонент. Главная идея этого метода состоит в представлении изображений лиц людей в виде набора главных компонент изображений, называемых "собственные лица"(Eigenfaces). Полезным свойством собственных лиц является то, что изображение, соответствующее каждому такому компоненту имеет лицеподобную форму, как представлено на рисунке 1.4 Вычисление главных компонент основывается на вычислении собственных векторов и собственных значений ковариационной матрицы, которая рассчитывается из изображения. Сумма главных компонент, умноженных на соответствующие собственные вектора, является реконструкцией изображения [7].

Рисунок *1.4* – Пример изображений собственных векторов.

Для каждого изображения лица вычисляются его главные компоненты. Обычно берётся от 5 до 200 главных компонент. Остальные компоненты кодируют мелкие различия между лицами и шумами. Процесс распознавания заключается в сравнении главных компонент неизвестного изображения с компонентами всех известных изображений. При этом предполагается, что изображения лиц, соответствующих одному человеку, сгруппированы в кластеры в собственном пространстве. Из базы данных выбираются изображения кандидаты, имеющие наименьшее расстояние от входного (неизвестного) изображения.

1.3 Обработка изображений

Для обработки изображений в ЭВМ можно использовать вычислительные мощности CPU или GPU. Обработка изображений в центральном процессоре выполняется отдельно для каждой цветовой компоненты пикселя, то есть сложность алгоритма оценивается сложностью обработки одного пикселя умноженной на общее количество компонентов в пикселе и умноженной на

количество пикселей в изображении. К этому можно добавить влияние скорости передачи информации из памяти в процессор и обратно. Для решения данной задачи в процессор были добавлены инструкции по обработке пакетной мультимедийной информации — SIMD инструкции.

Расширенный набор команд MMX впервые появился в процессорах Intel Pentium MMX и с тех пор поддерживается всеми более современными процессорами фирмы Intel, а также процессорами фирмы AMD, начиная с AMD K6. Инструкции SSE появились в процессорах Intel Pentium III. Эти расширения стандартных наборов команд процессоров 6-го поколения позволяют обрабатывать одной инструкцией сразу несколько пар чисел. Такая пакетная обработка может дать значительное ускорение работы алгоритма, когда необходимо много раз выполнять одни и те же операции над различными данными. Это как раз имеет место при обработке изображений.

Блок команд MMX использует 64-битные MMX-регистры, большинство команд расширения SSE выполняются над содержимым 128-битных регистров блока XMM. Соответственно это позволяет эффективно обрабатывать 1-2 пикселя за раз при использовании команд блока MMX и до 4 пикселей при использовании SSE. Заметим, что, применяя обычные операции, мы могли обрабатывать всего одну из 3-х компонент цвета пикселя за раз.

Кроме того, в блоке команд SSE есть удобные инструкции, позволяющие переписывать содержимое регистров MMX и XMM напрямую в память, минуя кэш. Это полезно, в том случае, если заранее известно, что содержимое регистра, переписанное в память понадобится не скоро. Использование таких операций позволяет существенно сократить накладные расходы на работу с кэш-памятью.

Хотя и описанные выше способы и позволяют в 5- 6 раз ускорить выполнение обработки, но для больших изображений этого недостаточно. Для обработки графических данных выгоднее использовать GPU или графический процессор. Современные графические процессоры очень эффективно обрабатывают и отображают компьютерную графику. Благодаря специализированной конвейерной архитектуре они намного эффективнее в обработке графической информации, чем типичный центральный процессор. Графический процессор в современных видеоадаптерах применяется в качестве ускорителя трёхмерной графики.

Высокая вычислительная мощность GPU объясняется особенностями архитектуры. Если современные CPU содержат несколько ядер, графический процессор изначально создавался как многоядерная структура, в которой количество ядер может достигать несколько тысяч. Разница в архитектуре обу-

словливает и разницу в принципах работы. Если архитектура CPU предполагает последовательную обработку информации, то GPU исторически предназначался для обработки компьютерной графики, поэтому рассчитан на массивно параллельные вычисления.

Если алгоритм решения задачи может быть распараллелен на тысячи отдельных потоков, то эффективность решения такой задачи с применением GPU может быть выше, чем ее решение средствами только процессора общего назначения. Однако нельзя так просто взять и перенести решение какой-то задачи с CPU на GPU, хотя бы просто потому, что CPU и GPU используют разные команды. То есть когда программа пишется под решение на CPU, то применяется набор команд x86 (или набор команд, совместимый с конкретной архитектурой процессора), а вот для графического процессора используются уже совсем другие наборы команд, которые опять-таки учитывают его архитектуру и возможности.

Когда стали предприниматься первые попытки реализовать неграфические вычисления на GPU, возник компилятор BrookGPU. До его создания разработчикам приходилось получать доступ к ресурсам видеокарты через графические API OpenGL или Direct3D, что значительно усложняло процесс программирования, так как требовало специфических знаний —- приходилось изучать принципы работы с 3D-объектами (шейдерами, текстурами и т.п.). Это явилось причиной весьма ограниченного применения GPU в программных продуктах. BrookGPU стал своеобразным «переводчиком». Эти потоковые расширения к языку Си скрывали от программистов трехмерный API и при его использовании надобность в знаниях 3D-программирования практически отпала. Вычислительные мощности видеокарт стали доступны программистам в виде дополнительного сопроцессора для параллельных расчетов. Компилятор BrookGPU обрабатывал файл с кодом Си и расширениями, выстраивая код, привязанный к библиотеке с поддержкой DirectX или OpenGL.

Во многом благодаря BrookGPU, компании NVIDIA и ATI (ныне AMD) обратили внимание на зарождающуюся технологию вычислений общего назначения на графических процессорах и начали разработку собственных реализаций, обеспечивающих прямой и более прозрачный доступ к вычислительным блокам 3D-ускорителей. В результате компания NVIDIA разработала программно-аппаратную архитектуру параллельных вычислений CUDA (Compute Unified Device Architecture). Архитектура CUDA позволяет реализовать неграфические вычисления на графических процессорах NVIDIA [8].

Компания AMD (ATI) также разработала свою версию технологии GPU,

которая ранее называлась ATI Stream, а теперь — AMD Accelerated Parallel Processing (APP). Основу AMD APP составляет открытый индустриальный стандарт OpenCL (Open Computing Language). Стандарт OpenCL обеспечивает параллелизм на уровне инструкций и на уровне данных и является реализацией техники GPU. Это полностью открытый стандарт, его использование не облагается лицензионными отчислениями. Отметим, что AMD APP и NVIDIA CUDA несовместимы друг с другом, тем не менее, последняя версия NVIDIA CUDA поддерживает и OpenCL.

Рассмотренные выше технологии позволяют использовать аппаратные вычислительные мощности на низком уровне. Однако существуют библиотеки, реализующие алгоритмы и методы работы с аппаратными средствами на более высоком уровне.

OpenCV — самая популярная библиотека компьютерного зрения. Она написана на C/C++, ее исходный код открыт. библиотека включает более 1000 функций и алгоритмов. Она разрабатывается с 1998 г., сначала в компании Intel, теперь в Itseez при активном участии open source сообщества [9].

Существуют библиотеки, более продвинутые по функциональности, например, Halcon, Matrox Imaging Libraryили Open eVision. Есть библиотеки более специализированные, делающие акцент на какой-либо конкретной задаче, например, libmv. OpenCV – самая большая библиотека по широте тематики.

В библиотеке более 250 функций были портированы для использования CUDA, обеспечивая ускорение от 5 до 100 раз.

Библиотека распространяется по лицензии BSD, что означает, что ее можно свободно и бесплатно использовать как в открытых проектах с открытым кодом, так и в закрытых, коммерческих проектах. Библиотеку не обязательно копировать целиком в свой проект, можно использовать куски кода. Единственное требование лицензии – наличие в сопровождающих материалах копии лицензии OpenCV.

Основные модули библиотеки можно отнести к 4 группам (разделам):

– Модули core, highgui, реализующие базовую функциональность (базовые структуры, математические функции, генераторы случайных чисел, линейная алгебра, быстрое преобразование Фурье, ввод/вывод изображений и видео, ввод/вывод в форматах XML, YAML и др.).

– Модули imgproc, features2d для обработки изображений (фильтрация, геометрические преобразования, преобразование цветовых пространств, сегментация, обнаружение особых точек и ребер, контурный анализ и др).

– Модули video, objdetect, calib3d (калибровка камеры, анализ движе-

ния и отслеживание объектов, вычисление положения в пространстве, построение карты глубины, детектирование объектов, оптический поток).

– Модуль ml, реализующий алгоритмы машинного обучения (метод ближайших соседей, наивный байесовский классификатор, деревья решений, бустинг, градиентный бустинг деревьев решений, случайный лес, машина опорных векторов, нейронные сети и др.).

2 ПРОЕКТИРОВАНИЕ ПРОГРАММЫ

Разработка программного обеспечения будет выполняться на языке C++. Выбран этот язык, так как библиотека OpenCV разработана изначально на нем. C++ позволяет разработать программы более быстродейственные, так как в процессе анализа и обработки видеопотока необходима производительность. В программе будут реализованы следующие алгоритмы обнаружения: обнаружение движений будет выполняться двумя ранее представленными методами: методом межкадровой разницы и методом вычитания фона, а обнаружение лиц методом Виолы—Джонса. Метод вычитания фона для обнаружения движений использует алгоритм ассоциативной модели гауссова смешения для моделирования фона. Алгоритм реализован в классе BackgroundSubtractor библиотеки OpenCV.

2.1 Элементы библиотеки OpenCV.

Рассмотрим подробнее структуры и функции входящие в библиотеку OpenCV, необходимые для выполнения в рамках данного проекта [10].

2.1.1 Основные структуры данных. Для хранения изображений в памяти используется класс Mat, представляющий собой двухмерные матрицы.

Для хранения одномерных массивов данных используется класс Scalar. В проекте используется для хранения координат цвета в цветовой модели RGB в виде массива из трех координат.

Класс Point используется для хранения информации о точке — координаты X и Y.

2.1.2 Работа с веб-камерой. Для получения изображения с веб-камеры используется класс VideoCapture. Конструктор класса позволяется получить доступ к любой камере установленной в системе. Для получения доступа к основной камере конструктор запускается с параметром 0. Для проверки подключения камеры в классе существует метод isOpened.

Для получения изображения с камеры используется оператор чтения из потока $>>$.

2.1.3 Обработка изображений. Для преобразования цветовой модели изображения используется функция cvtColor. В параметрах описываются входная и выходная матрицы и константа указывающая преобразование.

Функция absdiff используется для формирования изображения на основе сравнения двух других. На входе — два изображения. На выходе — изображение с пикселями, которые не совпадают на двух исходных образцах.

Для получения одинаковых пикселей на изображениях используется функция bitwise_and.

Для выравнивания значений пикселей изображения по порогу используется функция threshold. На входе задаются входная и выходная матрица, пороговое значение пикселя, максимальное значение пикселя тип пороговой функции. Значение пикселя в данном случае — значение элемента матрицы.

Функция erode позволяет убирать шумы с изображения посредством объединения соседних пикселей в группы, задаваемые ядром — геометрическим примитивом определенного размера. На входе задаются входное и выходное изображение и ядро.

Функция dilate выполняет операцию свертки изображения и ядра. Можно говорить что эта функция используется для расширения ярких участков изображения.

Функция resize изменяет размеры изображения.

Функция equalizeHist выравнивает гистограмму изображения, заданного в серых тонах. Алгоритм, используемый в функции, нормализует яркость и улучшает контраст изображения.

2.1.4 Поиск контуров. Для поиска контуров объектов на изображении используется функция findContours. Она позволяет найти контуры на одноканальном (монохромном) изображении. На выходе функция формируем массив контуров. Нарисовать контуры на изображении позволяет функция drawContours.

2.1.5 Средсва рисования. Для рисования на экран графических примитивов испольуются следующие функции: rectangle — для рисования на изображение прямоугольника, circle — для рисования круга.

Для отображения одного изображения на другом необходимо выделить на исходном изображении область в которую следует поместить другое изображение, а затем используя метод copyTo класса Mat скопировать изображение в область другого. В проекте используется для отрисовки кнопок, а также наложения контуров определяемых объектов.

2.1.6 Поиск объектов на изображении. Поиск объектов выполняется, используя классификатор называемый каскадом(полностью: каскад классификаторов использующих признаки Хаара). В OpenCV за данный класси-

фикатор отвечает класс CascadeClassifier. Метод класса load загружает готовый классификатор из файла. Метод detectMultiScale обнаруживает объекты различных размеров на входном изображении и возвращает список прямоугольников с обнаруженными объектами.

2.1.7 Выделение фона. Класс BackgroundSubtractorMOG2 отвечает за выделение фона. В конструкторе указываются конкретные параметры алгоритма, такие как порог чувствительности и количество кадров в истории, формирующих фон. Оператор круглые скобки () добавляет кадр в историю и выделяет измененные области.

2.1.8 Управление окнами и устройствами ввода. Для создания именованного окна используется функция namedWindow. Обращение к окну реализуется через имя этого окна. Функция moveWindow изменяет положение окна. Для управления мышью в пределах окна используется функция setMouseCallback, которая при изменении событий мыши вызывает функцию заданную пользователем. Для получения кода нажатой клавиши нужно вызвать функцию waitKey.

2.1.9 Работа с GPU. Работа с GPU осуществляется с использованием технологии CUDA. Класс GpuMat хранит изображения. Метод класса upload преобразует Mat в GpuMat. Для обнаружения образов через касказы Хаара используют класс CascadeClassifier_GPU. Работа с классами аналогична работе их CPU аналогов.

2.2 Компоненты приложения.

Диаграмма компонентов приложения представлена на рисунке 2.1

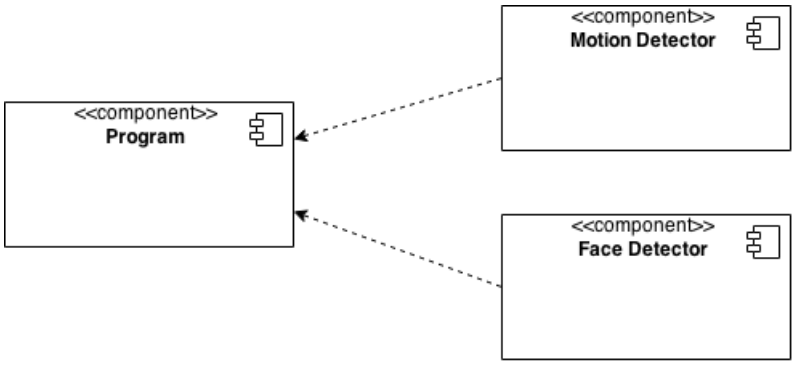

Рисунок *2.1* – Диаграмма компонентов приложения.

Program — компонент реализует графический интерфейс, получение и вывод изображения, а также переключение режимов приложения. Компонент представлен в исходных файлах DetectionSystem.h и DetectionSystem.cpp.

MotionDetector — компонент реализует режим обнаружения движений в приложении. Компонент представлен в исходных файлах motiondetect.h и motiondetect.cpp.

FaceDetector — компонент реализует режим обнаружения лиц в приложении.

Компонент представлен в исходных файлах facedetect.h и facedetect.cpp.

Переключение между режимами в приложении реализовано в виде изменения состояния приложения.

2.3 Блок-схемы алгоритмов.

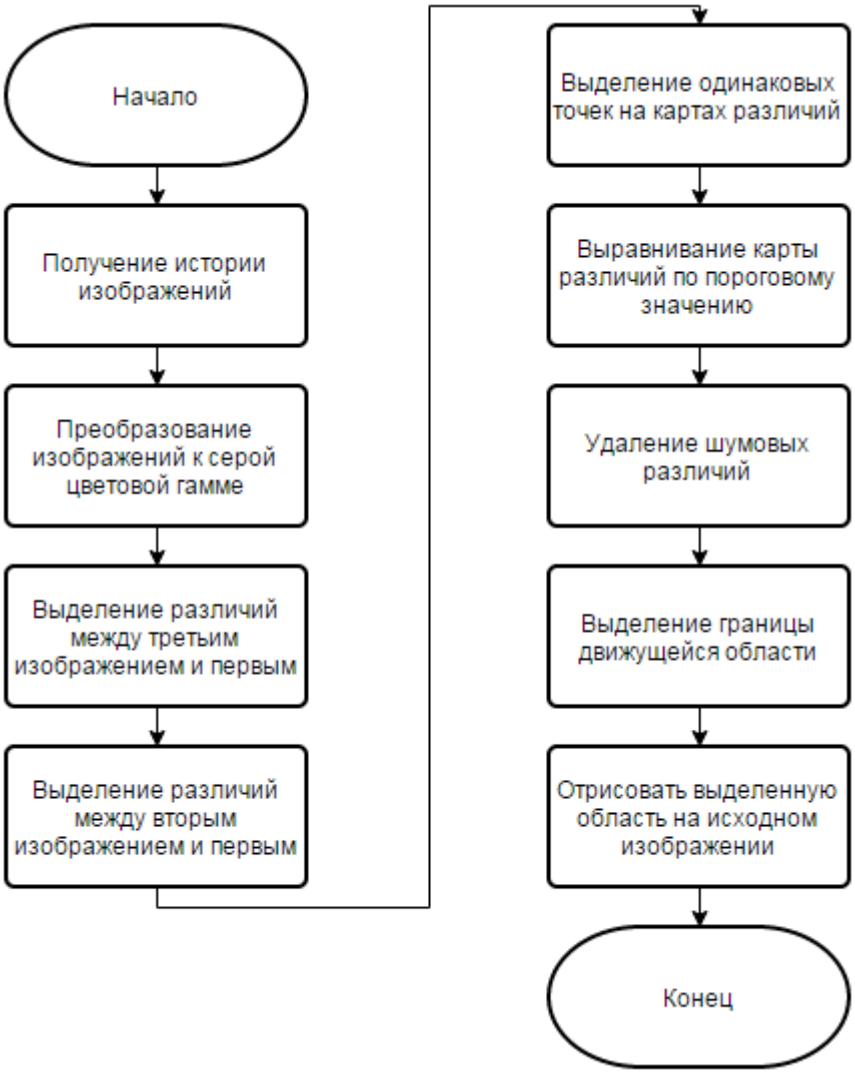

Рисунок *2.2* – Блок-схема алгоритма сравнения изображений.

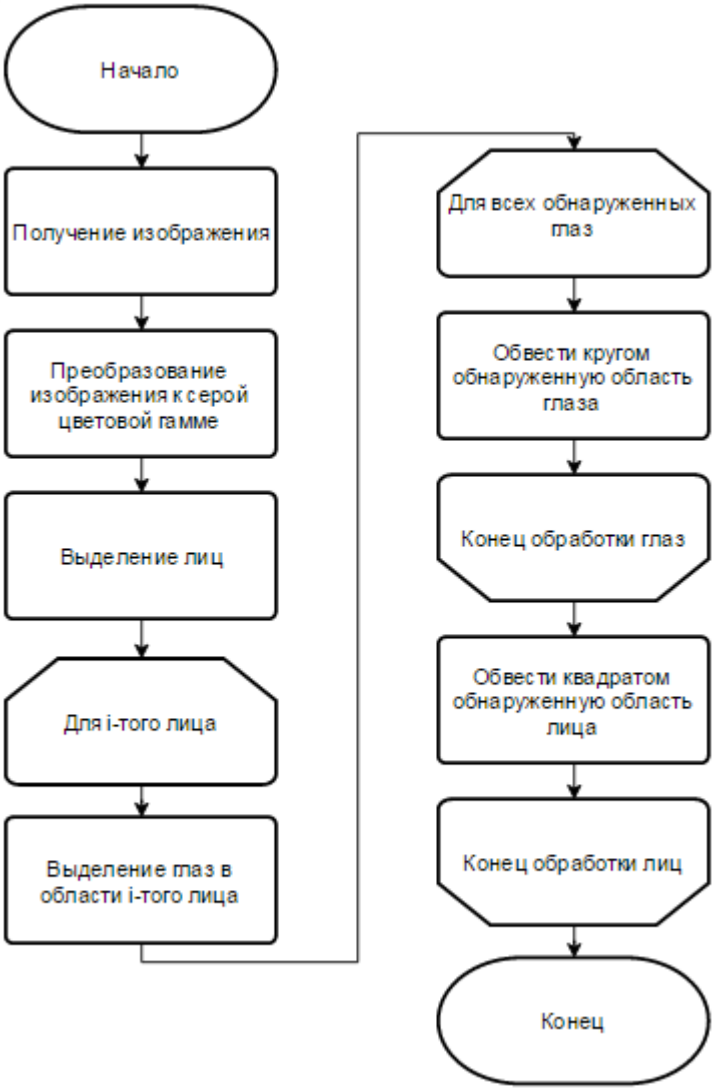

Рисунок *2.3* – Блок-схема алгоритма выделения лиц.

3 РАЗРАБОТКА СРЕДСТВ СИМУЛЯЦИИ И ВИЗУАЛИЗАЦИИ

В данном разделе будет произведен обзор средств симуляции работы оборудования а также средств визуализации данных, собранных дроном в процессе работы. Будет рассмотрен процесс дизайна и создания инструментов обработки данных, прошедших первичную обработу и восстановление облака точек, описывающего пространство, исследованное дроном.

3.1 Общие сведения

Компонент для симуляции и визуализации состоит из двух основных частей. Первая часть - модуль симуляции полета дрона с возможностью установки различных параметров его положения в пространстве с помощью соответствующего API. Вторая часть - модуль работы с облаком точек а также софт для визуализации облака точек. Процесс взаимодейтсвия этого компонента с системами дрона таков:

– Дрон собирает информацию об окружении.

– Компонент **TcpDroneUpdater** следит за дроном в реальном времени, проецируя его крен, тангаж и рысканье и ускорение на визуализационную модель.

– Компонент **TcpCloudUpdater** следит за собранной дроном информацией и в реальном времени отображает облако точек, собранное дроном. Активация компонента опциональна и не всегда возможна ввиду большого количества данных, генерируемых дроном. Облако точек, генерируемое таким образом имеет пониженное разрешение.

– После окончания сбора информации, вся собранная дроном информация (сохраненная на внутреннее хранилище) выгружается в софт визуализации, где из этих данных собирается полноразмерное облако точек.

Для упрощения процесса разработки инструментов и в виду не очень жестких требований к производительности данных инструментов, было принято решение разрабатывать инструментарий с применением языка C# с последующим выполнением в среде **Mono** для Linux. В качестве фреймворка для визуализации трехмерных данных было принято решение использовать пакет Unity (Personal Edition) из-за его мультиплатформенности, простоты организации его компонентной модели а также возможности реализации логики визуализации на языке C#, то есть возможности простой интеграции компонентов, считывающих данные о дроне в реальном времени с средствами

визуализации.

3.2 Симуляция полета дрона

Для визуализации положения дрона в пространстве используются показания внутренних систем гироскопа, акксселерометра и компаса. Посредством беспроводной связи поисходит постоянная передача показаний этих сенсоров в виде следующих TCP-пакетов:

```
struct RPYAccPacket {
  // Drone roll in degrees.
  float Roll;

  // Drone pitch in degrees.
  float Pitch;

  // Drone yaw in degrees.
  float Yaw;

  // Acceleration on X−axis.
  float AccX;

  // Acceleration on Y−axis.
  float AccY;

  /Acceleration on Z−axis.
  float AccZ;

  // Drone heading in degrees.
  float Heading;
}
```

Поскольку симуляция использует Lock-step схему при передаче данных о положении дрона, решено использовать TCP соединение для избежания возможных рассинхронизаций положения/ориентации дрона внутри симуляции и в пространстве. Параметр Heading в данном случае является вспомогательным и калибровочным и позволяет установить связь между поворотом дрона и фактическим направлением его обзора.

3.3 Обновление в реальном времени. Компонент TcpDroneUpdater

Для сбора данных о положении дрона было решено использовать протокол TCP из-за Lockstep природы алгоритмов визуализации и их аппроксима-

ционного характера в процессе сбора данных, когда происходит визуализация в реальном времени. Использование протокола TCP позволяет использовать встроенный в .NET тип TcpClient, позволяющий представить сетевой канал в виде потока данных (Порядок следования данных строго упорядочен пока соединение установлено, передача данных гарантируется). Пакет, будучи структурой фиксированной величины, можно обработать с помощью следующего компонента:

```
1  [RequireComponent(typeof(DroneController))]
2  public class TcpDroneUpdater : MonoBehaviour {
3      public string Hostname;
4      public ushort Port;
5
6      public readonly int PACKET_LENGTH = 7 * sizeof(float);
7      public readonly int UPDATE_EVERY = 50;
8
9      private DroneController drone;
10     private TcpClient client;
11     private NetworkStream dataStream;
12
13     private byte[] dataBuffer = new byte[PACKET_LENGTH];
14     private int alreadyRead;
15     private int packetNo = 0;
16
17     private void Awake() {
18         drone = GetComponent<DroneController>();
19         client = new TcpClient(Hostname, Port);
20         dataStream = client.GetStream();
21     }
22
23     private void ProcessPacket() {
24         var updatePacket = new RPYAccPacket();
25
26         // Details ommitted...
27
28         drone.SetAcceleration(updatePacket.AccX, updatePacket.AccY, updatePacket.AccZ);
29         drone.SetRotations(updatePacket.Roll, updatePacket.Pitch, updatePacket.Yaw);
30         if (packetNo % UPDATE_EVERY == 0)
31             drone.AdjustHeading(updatePacket.Heading);
32     }
33
34     private void Update() {
35         if (!dataStream.CanRead)
36             return;
37
```

```
38          if (dataStream.DataAvailable)
39          {
40              alreadyRead += dataStream.Read(dataBuffer, alreadyRead,
41                                             PACKET_LENGTH - alreadyRead);
42          }
43
44          if (alreadyRead == PACKET_LENGTH)
45          {
46              ProcessPacket();
47              alreadyRead = 0;
48              packetNo++;
49          }
50      }
51  }
```

Поскольку при использовании такого метода, задержка одного из пакетов либо скачки в показании приборов могут превести к внешним ”рывкам” при симуляции, на стороне контроллера объекта дрона реализованы алгоритмы линейной интерполяции данных параметров, так что ”промахи” показаний датчиков практически не заметны.

3.4 Визуализация облака точек

Вторая часть систем визуализации и симуляции - система отображения облака точек - служит для отображения данных, собранных дроном в процессе его работы. Для отображения точек используется специальный шейдер, позволяющий отображать только вершины mesh’а с опциональной возможностью изменения размеров и цвета отображаемых точек. Полученные mesh’ы могут содержать до десяти тысяч точек каждый. При достижении определенной плотности данных, облако точек можно использовать для восстановления трехмерной поверхности с адекватной точностью.

Формирование точек для визуализации происходит по следующему алгоритму:

а) На вход алгоритма приходят двухмерные массивы-”кадры” (frame) а так-же ассоциированное с кадром положение дрона в пространстве (описанная выше структура RPYAccPacket).

б) Расчитать поправочные коэффициенты xAdj, yAdj для вычисления угловых координат точек при проецировании на сферу вокруг дрона.

в) Для каждой точки в массиве frame

1) Умножить координаты (xi, yi) i-ой точки на соответствующие поправочные коэффициенты xAdj, yAdj.

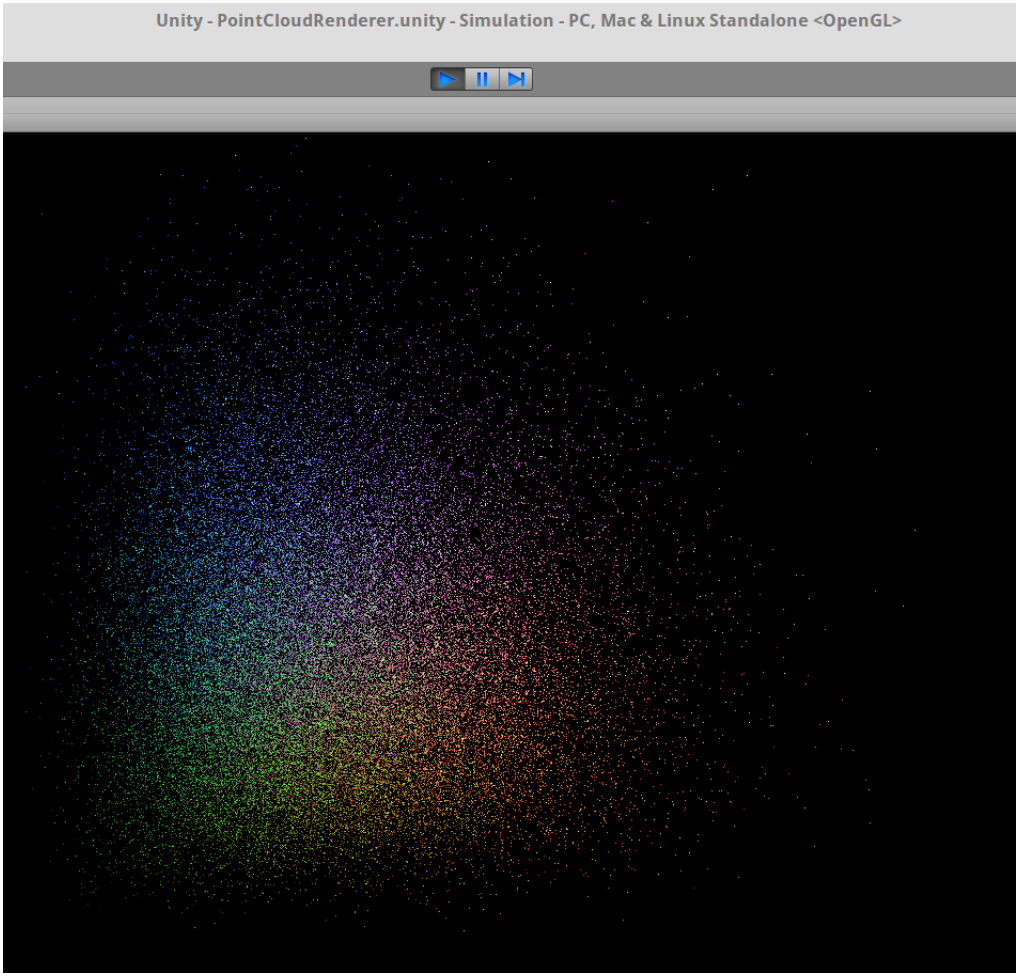

Рисунок *3.1* – Пример случайного облака точек с гауссовой плотностью распределением

2) Провести луч из точки, находящейся между камерами дрона и направленного в направлении, соответствующем i-ому элементу массива..

3) На расстоянии (1 - frame[xi, yi]) * Rсферы от камеры в направлении луча поставить точку и добавить ее на облако.

Каждый кадр формирует свой набор точек, зависящий от положения дрона и содержимого массива frame. Для вычисления поправочных коэффициентов нужно учитывать такие параметры как угол обзора используемой камеры и разрешение изображения, передаваемого с камеры. Каждый элемент массива, являясь числом от 0 до 1 показывает удаление ассоциированной точки от камеры и зависит от радиуса сферы, на которую происходит проецирование точек. Так точки с нулевым значением показывают, что если предмет в них и находится, то он за пределами сферы, а значит на основе этой точки нельзя сделать точных выводов о расстоянии. (Ограничение радиуса сферы зависит в первую очередь от параметров аппаратной части не рассматривает-

ся в данной главе).

Поскольку отображение отдельных точек Mesh'а - нетипичная задача машинной графики, для отображения облака точек был разработан специальный шейдер (совместимый с Direct3D и OpenGL), чей листинг представлен ниже:

```
Shader "PointCloud/VertexSizeAndColor" {
    Properties {
        _PointSize("PointSize", Float) = 3000
    }
    SubShader {
        Tags { "Queue"="Transparent" "IgnoreProjector"="True" "RenderType"="Transparent"}
        LOD 200
        Pass {
        Cull Off ZWrite On Blend SrcAlpha OneMinusSrcAlpha

            CGPROGRAM

                #pragma exclude_renderers flash
                #pragma vertex vert
                #pragma fragment frag


                struct appdata {
                    float4 pos : POSITION;
                    fixed4 color : COLOR;
                };


                struct v2f {
                    float4 pos : SV_POSITION;
                    float size : PSIZE;
                    fixed4 color : COLOR;
                    };
                float _PointSize;

                v2f vert(appdata v){
                    v2f o;
                    o.pos = mul(UNITY_MATRIX_MVP, v.pos);
                    o.size = _PointSize;
                    o.color = v.color;
                    return o;
                }

                half4 frag(v2f i) : COLOR0
                {
```

```
41                          return i.color;
42                      }
43                  ENDCG
44              }
45      }
46  }
```

ЗАКЛЮЧЕНИЕ

В ходе создания проекта было создано приложения для операционной системы Microsoft Windows, которое было написано в среде разработки Visual Studio 2012 с использованием библиотеки компьютерного зрения OpenCV.

В рамках изучения предметной области были рассмотрены основные методы обнаружения движений: метод межкадровой разности и метод вычитания фона. Также рассмотрены следующие методы обнаружения лиц: метод Виолы-Джонса и метод главных компонент.

В процессе работы с библиотекой OpenCV были использованы следующие её элементы: инструменты для работы с устройствами ввода и вебкамерой, инструменты обработки изображений, алгоритмы компьютерного зрения, инструменты работы с окнами. Однако этим библиотека не ограничивается, и возможности её использования неизмеримы. Плюсом данной библиотеки является и то, что она является open source продуктом — это значит что любой программист, заинтересованный в развитии библиотеки может принять участие в этом.

В ходе тестирования были выявлены ошибки в проектировании системы и места в программе, которые замедляют ее. Все данные недочеты были исправлены, что подтверждается в результате многочисленных тестовых запусках приложения.

Приложение можно использовать как визуализацию методов изученных ранее, и использовать как основу для домашней сигнализации и системы видеонаблюдения. Вместо дорогостоящих комплексных систем можно самому создать и использовать продукт, который решает конкретные задачи, поставленные перед системой видеонаблюдения.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Горелик А. Л., Скрипкин В. А. Методы распознавания. — 4-е изд. / Скрипкин В. А Горелик А. Л. — М. : Springer, 2004. — 262 с.

[2] R.Brooks, C.Breazeal, M.Marjanovic, B.Scassellat. Motion Detection and Segmentation. [Электронный ресурс]. — 1998. — Режим доступа: http://www.ai.mit.edu/projects/cog/VisionSystem/motion_detection.html. — Дата доступа: 16.03.2015.

[3] В.А., Сойфер. Компьютерная обработка изображений. Часть 2. Методы и алгоритмы. / Сойфер В.А. // Соросовский образовательный журнал. — 1996. — Т. №3. — с. 110–121.

[4] R, Fisher. CVOnline: Motion and time sequence Analysis [Электронный ресурс]. — 2002. — Режим доступа: http://homepages.inf.ed.ac.uk/rbf/CVonline/motion.htm#chgdet. — Дата доступа: 28.04.2015.

[5] Z.Zivkovic. Improved adaptive Gausian mixture model for background subtraction. — 2004. — August.

[6] Wikipedia, the free encyclopedia. Face detection [Электронный ресурс]. — Режим доступа: http://en.wikipedia.org/wiki/Face_detection. — Дата доступа: 10.05.2015.

[7] Буй Тхи Тху Чанг В.Г. Спицын, Фан Нгок Хоанг. Распознавание лиц на основе применения метода Виолы-Джонса, Вейвлет-преобразования и метода главных компонент / Фан Нгок Хоанг Буй Тхи Тху Чанг, В.Г. Спицын // Известия Томского политехнического университета. — 2012. — Т. 320 №5. — с. 110–121.

[8] Боресков, А. В. Параллельные вычисления на GPU.Архитектура и программная модель CUDA: Учебное пособие. / А. В. Боресков. — М. : Издательство Московского университета, 2012. — 336 с.

[9] OpenCV, Learning. Learning OpenCV [Электронный ресурс]. — 2013. — Режим доступа: http://locv.ru/wiki. — Дата доступа: 21.03.2015.

[10] OpenCV documentation [Электронный ресурс]. — Режим доступа: http://docs.opencv.org/. — Дата доступа: 22.03.2013.