МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ 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)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Прикладная информатика в социальной сфере
(направленность (профиль)/специализация

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему: «Разработка приложения для автоматизации процесса выбора субтрактора фона при создании систем компьютерного зрения»

Студент Н.Н. Латухин
(И.О. Фамилия) (личная подпись)

Руководитель А.П. Тонких
(ученая степень, звание, И.О. Фамилия)

Консультант А.В. Москалюк
(ученая степень, звание, И.О. Фамилия)

Тольятти 2020

Аннотация

Тема бакалаврской работы: «Разработка приложения для автоматизации процесса выбора субтрактора фона при создании систем компьютерного зрения».

В данной бакалаврской работе исследуются способы автоматизации процесса разработки систем компьютерного зрения, основанных на использовании субтракторов фона.

В данном исследовании разработано программное обеспечение, позволяющее разработчикам систем компьютерного зрения заблаговременно произвести выбор субтрактора фона, наиболее подходящего под условия работы системы. Для этого в программу загружается фрагмент видеоряда снятого в условиях будущего функционирования системы компьютерного зрения, затем программа предоставляет пользователю результаты обработки видеофрагмента с помощью всех существующих в настоящее время субтракторов фона – GMG, MOG, MOG2, KNN, CNT. По результатам работы программы делается выбор о том, какой субтрактор наиболее оптимален в заданных условиях работы.

Структура бакалаврской работы представлена введением, тремя главами, заключением, списком литературы.

Во введении дается краткая характеристика проделанной работы.

В первой главе проводится анализ процесса разработки систем компьютерного зрения.

Во второй главе описываются особенности работы субтракторов.

В третьей главе описывается разработанное программное обеспечения, приводится пример его практического использования.

В заключении представлены выводы по проделанной работе.

В работе использовано 30 рисунков, список литературы содержит 20 литературных источников. Общий объем выпускной квалификационной работы составляет 46 страниц.

Abstract

The title of the graduation work is *Development of an application to automate the process of selecting the background subtractor when creating computer vision systems.* Computer vision systems are used to solve various practical problems such as analysis of medical images and video surveillance. The variety of image analysis algorithms forces developers to check at the testing stage which of the selected algorithms are suitable for solving the tasks. The issue of the graduation work is the problem of choosing background subtractors - algorithms for separating foreground objects from background.

The aim of the work is to automate the process of developing a computer vision system by developing a program for testing background subtractors on a video sequence. The following tasks are completed: modeling the process of developing computer vision systems based on the use of subtractors, analysis of modern background subtractors, developing and testing the program for automating the process of developing computer vision systems. In this work a system computer vision process model is developed which describes the processes performed during the development of systems. A code is also developed for comparing subtractors.

In the result of this work, the software program has been developed that allows developers of computer vision systems to select in advance the most suitable background subtractors for the system. To do this, a fragment of the video sequence shot under the conditions of the future operation of the computer vision system is loaded into the program, and the program provides the user with the results of image processing using all currently existing background subtractors - GMG, MOG, MOG2, KNN, CNT. Based on the results of the program's work, the choice is made as to which subtractor is optimal in the given working conditions. This program can be used by developers of computer vision systems to select the optimal background subtractor.

Содержание

Введение

При разработке систем компьютерного зрения часто требуется решать задачу определения перемещения объектов на основе видеоряда (например, при анализе дорожного трафика). Алгоритмы, решающие данную задачу, называются субтракторами. При тестировании системы компьютерного зрения разработчикам часто приходится вносить изменения в работу их алгоритма и переписывать программный код из-за неудачного выбора субтрактора заднего фона.

Целью работы является разработка приложения для автоматизации процесса выбора субтрактора фона при создании систем компьютерного зрения.

Объектом исследования является деятельность разработчиков при создании систем компьютерного зрения.

Предметом исследования является автоматизация процесса выбора субтрактора фона при разработке систем компьютерного зрения.

Поставленная цель в работе достигнута путем решения следующих задач:

1. Моделирование процесса разработки систем компьютерного зрения, основанных на использовании субтракторов фона.

2. Анализ принципов работы субтракторов фона

3. Разработка и тестирование приложения для автоматизации процесса выбора субтрактора фона при создании систем компьютерного зрения.

В ходе выполнения работы на языке Python было разработано программное обеспечение, позволяющее определять наиболее оптимальный субтрактор для видеофрагмента. Использование программного обеспечения позволяет автоматизировать процесс разработки систем компьютерного зрения, снизить трудоемкость и повысить эффективность работы разработчиков за счет возможности заблаговременного выбора наиболее подходящего субтрактора заднего фона.

Работа программного обеспечения протестирована на данных видеоряда (результаты тестирования представлены в третьем разделе работы).

1 Анализ процесса разработки систем компьютерного зрения

1.1 Постановка задачи

С развитием искусственного интеллекта расширяется класс систем компьютерного зрения, направленных на решение практических задач на основе информации, получаемой с видеокамер. Компьютерное зрение используется [1-3]:

– в системах управления процессами (в составе промышленных роботов, автономных транспортных средств);

– в системах видеонаблюдения;

– в системах организации информации (например, при индексации видеороликов);

– в системах моделирования объектов или окружающей среды (анализ медицинских изображений, топографическое моделирование);

– в системах человеко-машинного взаимодействия;

– в системах дополненной реальности.

Ввиду того, что человеку сложно спрогнозировать результаты работы субтракторов фона, разработчикам систем компьютерного зрения на этапе тестирования часто приходиться возвращаться на предыдущие этапы разработки, чтобы поменять комбинацию используемых в системе алгоритмов анализа изображений [4-5].

Часто разработчики систем компьютерного зрения не учитывают чувствительность алгоритмов анализа изображения:

– к динамическому шуму, появление которого связанно с несовершенством цифровых сенсоров, применяемым в камерах;

– к пульсациям искусственного света, невидимым для глаза, но фиксируемым камерой;

– к артефактам изображения, появляющимся при сжатии видеоданных с использованием кодеков;

– к бликам света на объектах;

– к слабому дрожанию камеры от вибраций (например, ветра);

– к кратковременному попаданию в сенсор камеры блика;

– к наличию на изображении резких переходов по цвету;

– и другие особенности, которые еще не раскрыты в силу новизны используемых алгоритмов.

Это во многом усложняет процесс разработки систем компьютерного зрения. Предполагается, что если бы разработчики системы компьютерного зрения могли бы заранее выбрать оптимальный субтрактор фона на видеоряде полученным в условиях работы будущей системы, то это позволило бы еще до этапа написания программного кода понять какой алгоритм субтрактора больше подходит под заданные условия. Это впоследствии исключило или снизило бы количество доработок, вносимых на этапе тестирования системы компьютерного зрения [6-10].

С учетом вышесказанного становится ясно, что для автоматизации процесса разработки систем компьютерного зрения необходима разработка программы, позволяющей выбирать субтрактор путем анализа видеоданных, полученных в реальных условиях.

Так как задач, требующих решения при анализе видеоданных большое множество, то в рамках данного исследования будет рассмотрен один класс подзадач – сегментация изображения для обнаружения перемещений движущихся объектов.

Алгоритмы, направленные на определение перемещений движущихся объектов, называются субтракторами заднего фона.

Таким образом, целью работы является – автоматизация процесса разработки систем компьютерного зрения за счет разработки программы для выбора субтрактора фона на фрагменте видеоданных.

Поставленная цель будет достигнута путем решения следующих задач:

1. Моделирование процесса разработки систем компьютерного зрения, основанных на использовании субтракторов фона.

2. Анализ принципов работы субтракторов фона.

3. Разработка и тестирование программного обеспечения для автоматизации процесса разработки систем компьютерного зрения.

1.2 Моделирование процесса разработки систем компьютерного зрения

Моделирование процесса разработки систем компьютерного зрения необходимо начать с определения контекста, то есть наиболее абстрактного уровня описания системы разработки в целом. В контекст входит определение субъекта моделирования, цели и точки зрения на модель. С точки зрения разработчиков системы компьютерного зрения контекстная диаграмма «Разработка системы компьютерного зрения» описывает разработку наиболее обобщённым образом (рисунок 1.1).

Информация о предметной области представлена в виде неформализованных описаний. Извлечение экспертных знаний о предметной области не входит в задачу разработки системы компьютерного зрения и поэтому лежит за пределами модели.

Техническое задание предполагается уже утверждённым.

Предполагается, что разработчиками принята некоторая методология, которой они придерживаются при проектировании системы компьютерного зрения. Эта методология включает, в частности, построение функциональной, статической и динамической моделей системы компьютерного зрения. Поскольку методология разработки оказывает управляющее воздействие на все блоки модели, для удобства чтения диаграмм она «убрана в туннель».

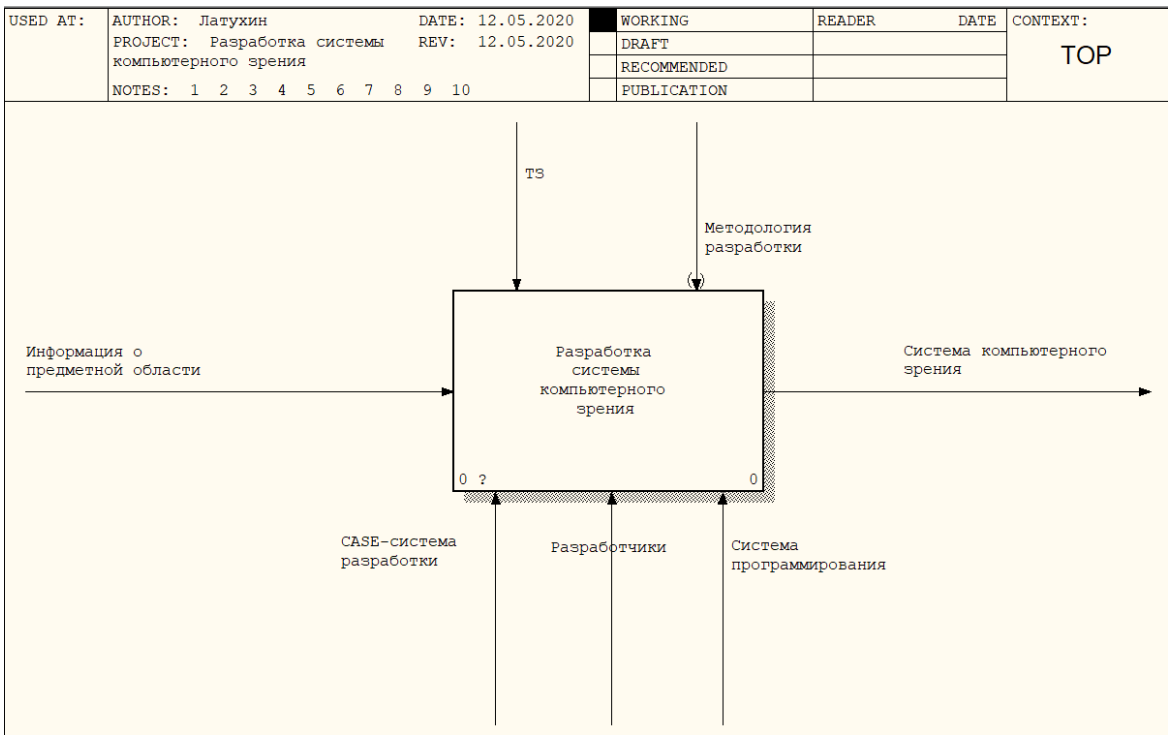


Рисунок 1.1 – Контекстная диаграмма «Разработка системы компьютерного зрения» для модели «Как есть»

CASE-система – механизм, автоматизирующий разработку системы компьютерного зрения.

Диаграмма декомпозиции «Разработка системы компьютерного зрения» описывает разработку системы компьютерного зрения как итерационный процесс (рисунок 1.2).

Под спецификациями понимаются детальные формальные описания интерфейсов компонентов системы компьютерного зрения.

Модель системы компьютерного зрения подробно описывает функционирование системы, её структуру и взаимодействие компонентов.

Разработчики включают аналитиков, программистов и тестировщиков. Аналитики проектируют и тестируют систему компьютерного зрения, программисты занимаются кодированием и отладкой системы. Тестировщики занимаются тестированием системы компьютерного зрения. Среди

тестировщиков допускается наличие представителей заказчика, внедрённых в коллектив разработчиков.

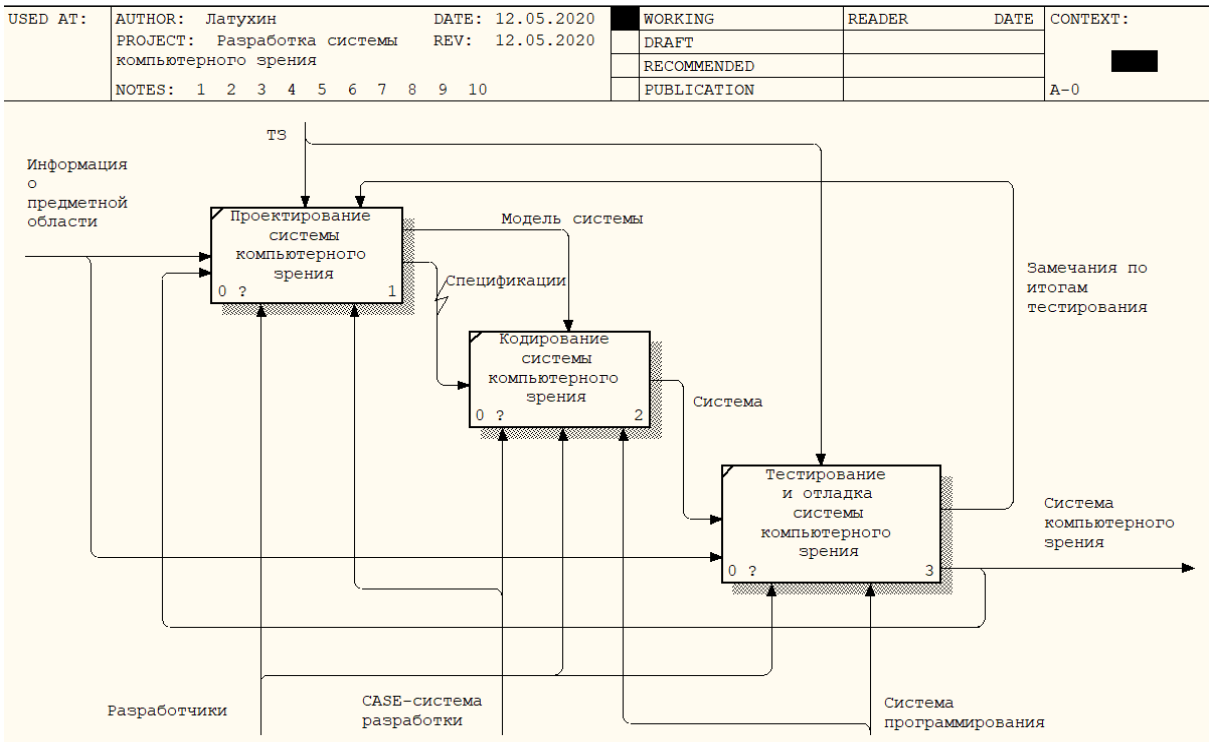


Рисунок 1.2 – Диаграмма «Разработка системы компьютерного зрения» (A0) для модели «Как есть»

Диаграмма декомпозиции «Тестирование и отладка системы компьютерного зрения» описывает последовательность проведения тестирования системы компьютерного зрения (рисунок 1.3).

В результате анализа замечаний по итогам тестирования могут быть внесены изменения в проект, касающихся различных моделей системы компьютерного зрения.

Выбор субтрактора осуществляет разработчик путём перебора.

В соответствии с моделью «Как должно быть» выбор подходящего субтрактора будет осуществляться с помощью разработанного программного обеспечения. Контекстная диаграмма для модели «Как должно быть» представлена на рисунке 1.4.

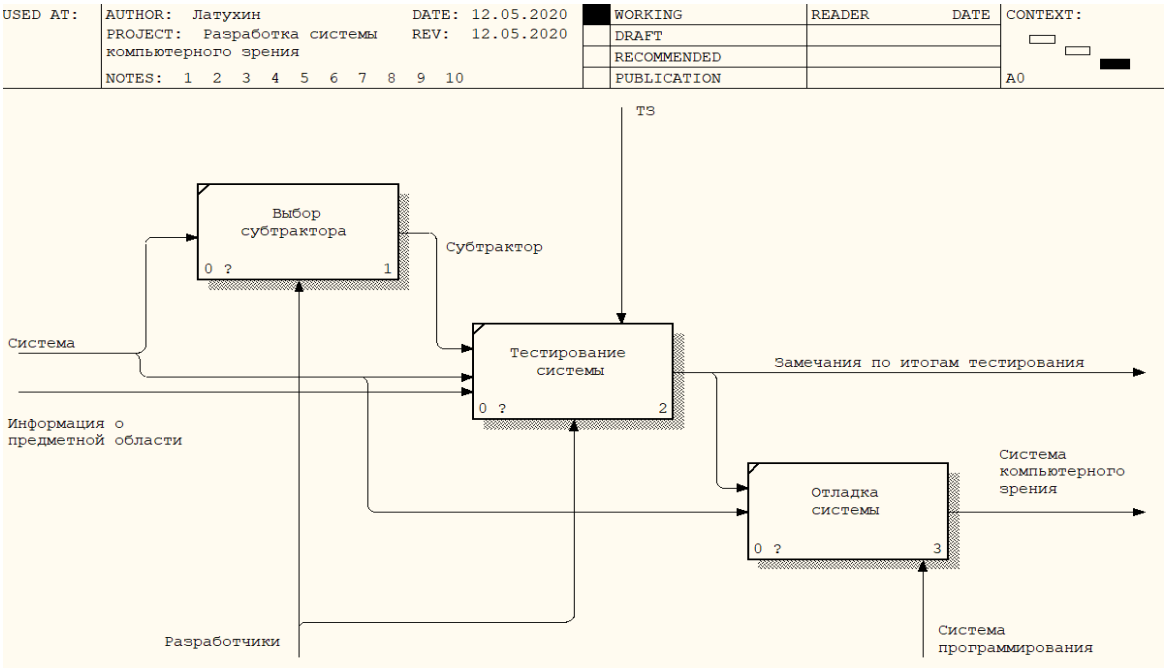


Рисунок 1.3 – Диаграмма «Тестирование и отладка системы компьютерного зрения» для модели «Как есть»

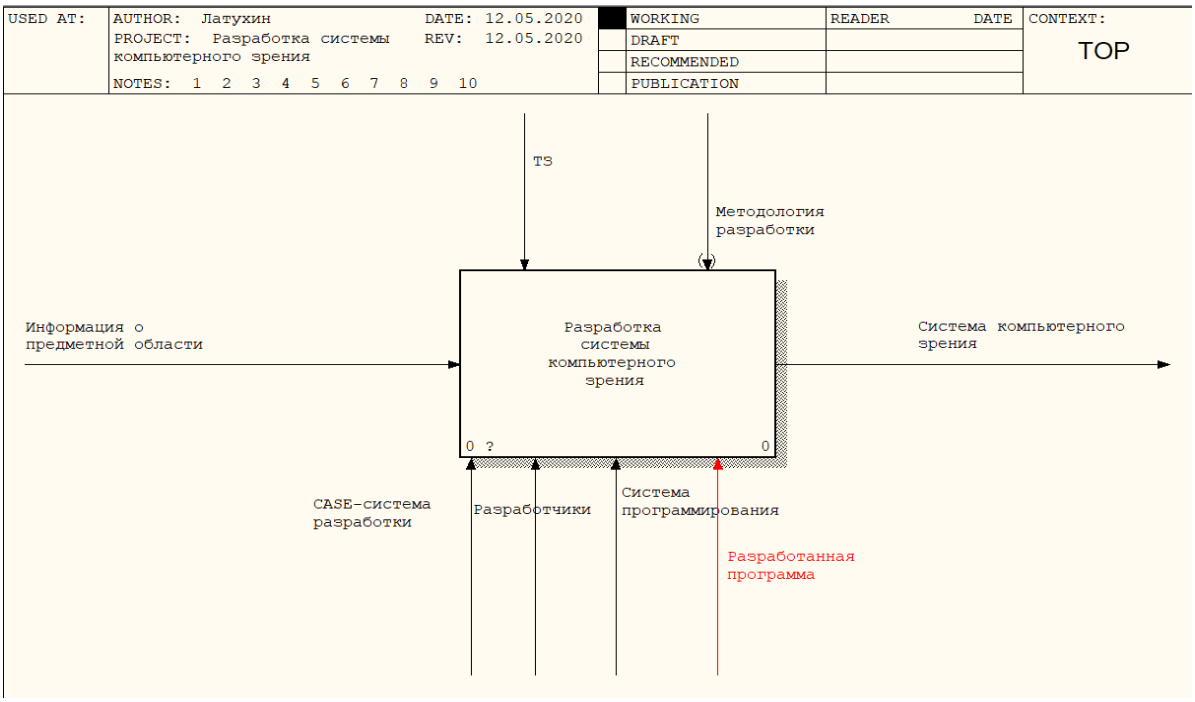


Рисунок 1.4 – Контекстная диаграмма «Разработка системы компьютерного зрения» для модели «Как должно быть»

Диаграмма декомпозиции «Разработка системы компьютерного зрения» для модели «Как должно быть представлена на рисунке 1.5.

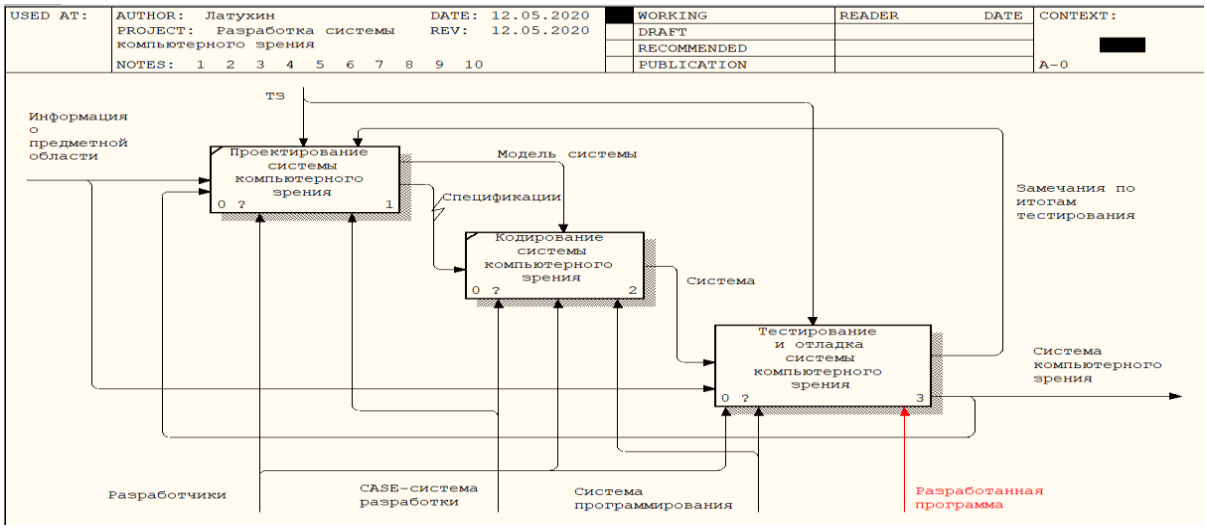


Рисунок 1.5 – Диаграмма «Разработка системы компьютерного зрения» (A0) для модели «Как должно быть»

На рисунке 1.6 представлена диаграмма декомпозиции «Тестирование и отладка системы компьютерного зрения» для модели «Как должно быть».

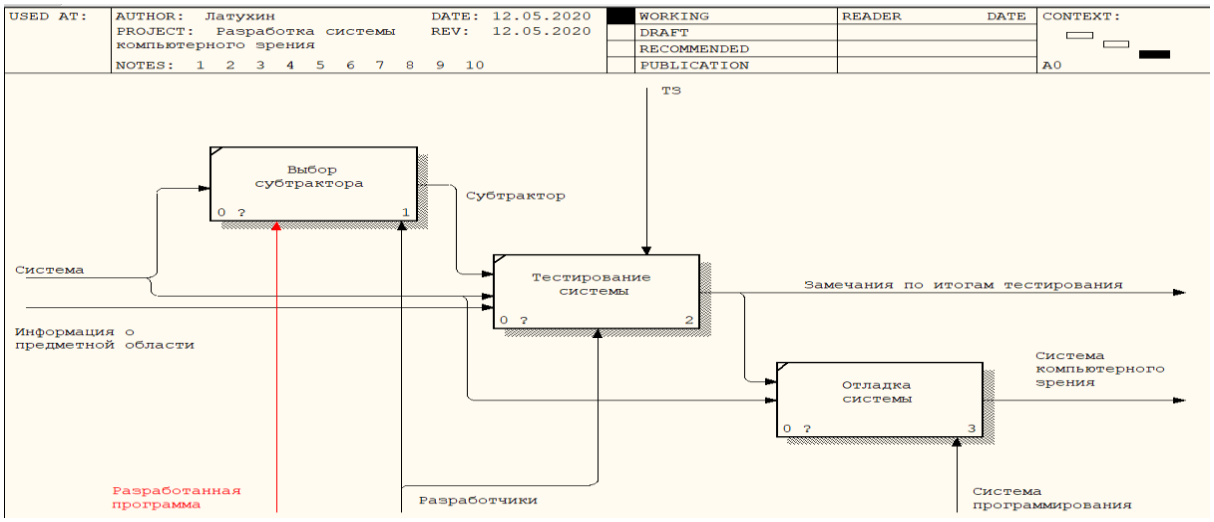


Рисунок 1.6 – Диаграмма «Тестирование и отладка системы компьютерного зрения» для модели «Как должно быть»

Таким образом, усовершенствование исследуемого бизнес-процесса разработки системы компьютерного зрения достигается путём внедрения программы, позволяющей автоматизировать выбор субтрактора для системы компьютерного зрения.

В качестве выводов можно отметить следующее:

1. По результатам материалов, представленных в данной главе, была поставлена задача автоматизации процесса выбора субтрактора фона при создании систем компьютерного зрения, сформулированы цели и задачи исследования. Требуется разработать программу для выбора оптимального субтрактора на основе видеоряда, полученного в режиме реального времени.

2. Были составлены диаграммы модели «Как есть» бизнес-процесса разработки систем компьютерного зрения. По результатам анализа модели установлено, что главным его недостатком бизнес-процесса является ручной перебор субтракторов фона.

Усовершенствование исследуемого бизнес-процесса достигается путём разработки и внедрения программного обеспечения, которое позволит выбирать необходимый субтрактор фона без ручного перебора разработчика.

2 Анализ принципов работы субтракторов фона

2.1 Принципы работы субтракторов фона

Субтракторы фона – это алгоритмы, используемые при анализе последовательности видеокадров для отделения перемещения движущихся объектов переднего плана от заднего фона [11-13].

Каждый кадр видео можно разделить на две разные области: передний план, чьи группы пикселей являются частью перемещающихся объектов (пешеходы, транспортные средства или велосипедисты), и фон, который группирует все пиксели, не являющиеся частью перемещающихся объектов, такие как тротуар, деревья, небо, здания. От алгоритмов (субтракторов) требуется уметь для любого кадра рассчитывать перемещения объектов [14-16].

Обнаружение перемещения движущихся объектов на видео обычно выполняется на основе таких методов как вычитание фона, создание оптического потока и временной дифференциации (контроль изменения распределения пикселей по цвету). Самой популярной техникой для обнаружения перемещения движущегося объекта на видеоряде является вычитание фона. Этот подход использует математическую модель статического фона и сравнивает её с каждым новым кадром видеопоследовательности.

Надежный субтрактор фона должен справляться с нестабильностью освещения днём и ночью. Также субтрактор фона должен быстро реагировать на изменения фона, возникающие при движении транспортных средств.

Процесс работы любого субтрактора фона можно представить как выполенение следующих этапов: предварительная обработка, моделирование фона, обнаружение переднего плана, представление результатов. Более подробно принцип действия субтракторов фона представлен на рисунке 2.1.

Рисунок 2.1 – Диаграмма работы субтракторов фона

Первый этап, предварительная обработка – это процесс подготовки данных о текущем кадре и собранной информации о предыдущих кадрах в том формате, который требуется выбранному типу субтрактора для выполнения последующих расчетов. На следующем этапе, с учетом полученной информации, субтрактор обновляет модель заднего фона. Затем с учетом обновленной модели производится сегментация изображения, при которой каждый пиксель относится алгоритмом или к заднему фону, или к переднему. Сегментация выполняется путем расчета маски заднего фона. Маска заднего фона – это матрица, размерность которой равна размерности изображения в пикселях. Компоненты матрицы чаще всего могут принимать 2 значения – «1» (пиксель переднего фона), «0» (пиксель объекта переднего плана). В некоторых субтракторах фона присутствует третье значение, обозначающее тень объекта переднего плана. Затем результаты сегментации изображения выводятся пользователю [17-20].

2.2 Субтрактор Godbehere-Matsukawa-Goldberg (GMG)

Алгоритм GMG моделирует фон с помощью комбинации Байесовского вывода и фильтра Калмана.

На первом шаге работы алгоритма на основе предыдущих кадров проводится расчет для каждого пикселя взвешенного значения, определяющего как долго цвет пикселя оставался неизменным. Для каждого следующего кадра эти значения обновляются. Пиксель, цвет которого в течение определенного времени не менялся, алгоритмом относится к фону.

В алгоритме применяется фильтр Калмана для фильтрации пикселей, отнесенных к переднему плану, чтобы снизить влияние цветового шума (например, вносимого цифровой камерой при съемке в темное время суток).

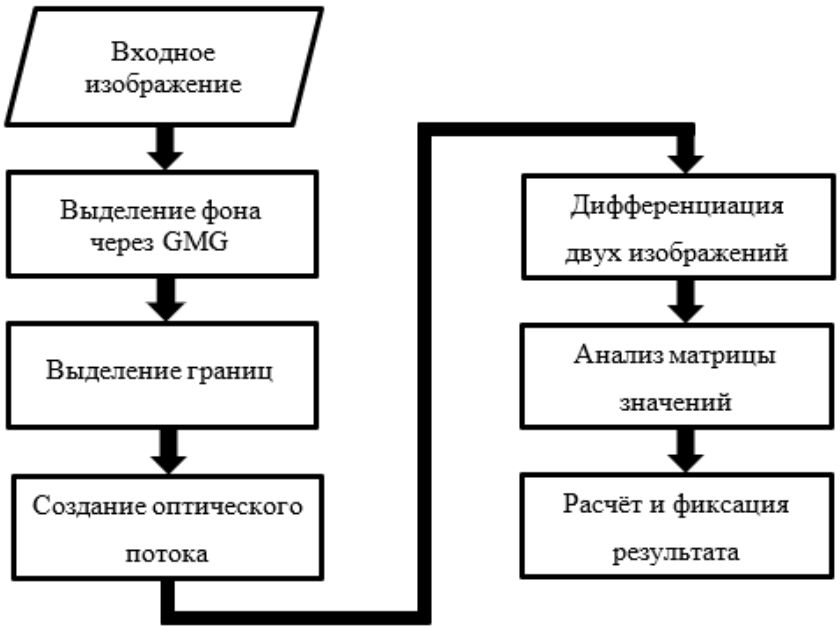


Рисунок 2.2 – Работа субтрактора GMG

Как видно из рисунка 2.2, алгоритм повторяет общий паттерн поведения субтракторов фона.

Далее на рисунке 2.3 представлен один из кадров видеоряда, к которому применен субтрактор фона GMG.

Рисунок 2.3 – Анализируемый кадр видеоряда

Рисунок 2.4 – Маска заднего фона после применения субтрактора GMG

Как можно видеть из рисунка 2.4 вместе с передним планом субтрактор фона GMG также выделил тени от деревьев и машин.

2.3 Субтрактор Mixture of Gaussians (MOG)

В этом методе каждый фоновый пиксель моделируется смесью k гауссовских распределений со значениями для k от 3 до 5. Различные распределения представляют собой разные цвета заднего фона и переднего плана. Вес каждого из используемых распределений на модели пропорционален количеству времени, в течение которого каждый цвет остается на этом пикселе (такая оценка производится на основе нескольких предыдущих кадров). Следовательно, когда вес распределения пикселей низкий (это происходит, когда пиксель на рассматриваемом множестве кадров менял свой цвет в широком цветовом диапазоне), этот пиксель распознается как передний план.

В MOG задний фон представляется в качестве параметрического набора значений. Каждый пиксель представлен функцией с набором гауссовых компонент, как указано в формуле 1.

$$\mathrm{F}\,(\,i_t\,=\,\mu)\,=\,\sum_{i=-1}^{k}\omega_{i,t}*\eta(\mu,\sigma)\quad, \tag{1}$$

где η – i-ая гауссовая компонента, μ – средняя интенсивность, σ – среднеквадратичное отклонение, $\omega_{i,t}$ – количество данных, накопленное i-ым компонентом.

В алгоритме MOG, пиксель считается частью заднего фона на основе оценки коэффициента масштабирования среднеквадратичного отклонения. Алгоритм MOG больше подходит для анализа кадров снятых на открытом воздухе, чем алгоритм GMG. При этом динамическое изменение освещения затрудняет работу алгоритма MOG.

Ключевую роль в алгоритме играют пять основных параметров, оказывающих значительное влияние на точность и эффективность работы алгоритма:

– T_s – порог веса фонового компонента;

– D – коэффициент масштабирования среднеквадратичного отклонения;

– ρ – скорость адаптации к изменениям;

– K – общее количество гауссовых компонентов;

– M – максимальное количество компонентов M в фоновой модели.

Правильный выбор параметров MOG для вычитания фона может обеспечить улучшенные результаты при обнаружении передвижения движущихся объектов на открытом воздухе.

На рисунке 2.5 представлен результат применения субтрактора MOG к кадру, показанному на рисунке 2.3.

Рисунок 2.5 – Маска заднего фона после применения субтрактора MOG

На рисунке 2.5 можно видеть, что на переднем плане, выделенным субтрактором фона MOG, почти отсутствует шум, но при этом некоторые динамические области (части автомобилей) были неверно отнесены к фону.

2.4 Субтрактор Mixture of Gaussians 2 (MOG2)

Алгоритм MOG2 основан на MOG. Его отличие от предшественника заключается в решении проблемы фиксированного количества используемых гауссовых компонент.

Благодаря использованию в MOG2 переменного количество распределений Гаусса, алгоритм достигает хороших результатов в условиях наличия в кадре произвольного количества цветовых сочетаний.

На рисунке 2.6 представлен результат применения субтрактора фона MOG2 к кадру, представленному на рисунке 2.3.

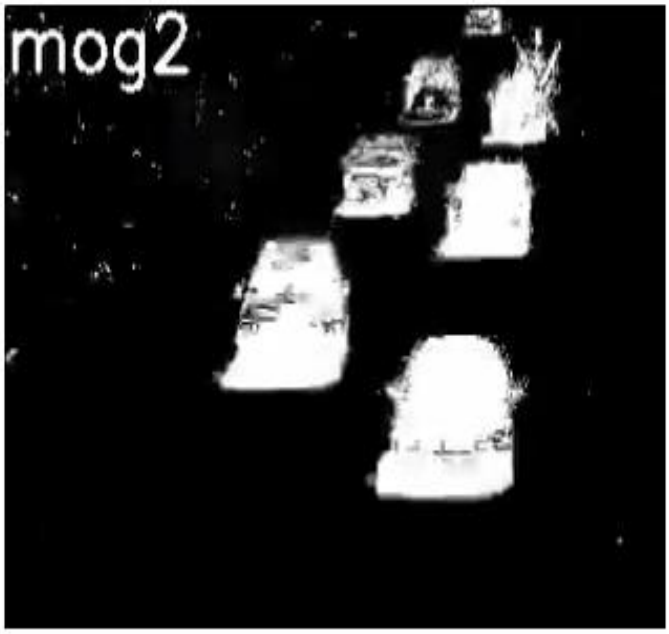


Рисунок 2.6 – Маска заднего фона после применения субтрактора MOG2

Если сравнить результата работы субтрактора MOG (рисунок 2.5) и MOG2 (рисунок 2.6), то можно увидеть, что последний лучше справился с определением перемещающихся объектов кадра (движущиеся автомобили). При этом уровень шума в маске заднего фона полученном с помощью MOG2 намного ниже, чем при использовании GMG (рисунки 2.4 и 2.5).

2.5 Субтрактор K-Nearest Neighbors (KNN)

Данный алгоритм выполняет определение перемещения движущихся объектов. На рисунке 2.7 слева представлен анализируемый кадр видео ряда, а справа – маска заднего фона сгенерированная субтрактором фона KNN.

Как видно из рисунка 2.7, субтрактор фона KNN успешно справляется с выделением большого количества перемещающихся объектов на изображении. При этом уровень шума в маске заднего фона полученном с помощью KNN сопоставим с уровнем шума с помощью MOG2.

Рисунок 2.7 – Исходный кадр (слева) и маска заднего фона полученная субтрактором KNN

Для определения перемещения объектов алгоритм выполняет следующие этапы:

– на первом этапе исходное изображение подвергается размытию путем применения фильтра Гаусса. Это шаг необходим, чтобы сгладить резкие цветовые переходы на изображении и очистить его от возможных шумов;

– на втором этапе к получившемуся изображению применяется маска KNN, которая направлена на выделение перемещающихся объектов в отдельные подобласти. При этом субтрактор KNN умеет выделять тени перемещающихся объектов, которые на маске KNN обозначены серым цветом.

Т.к. чаще всего тень не интересна для дальнейшего рассмотрения, то следующим шагом является применение к маске KNN пороговой функции;

– на последнем этапе для сглаживания остаточных шумов и жестких границ перемещающихся объектов также применяет размытие с помощью фильтра Гаусса с повторным применением пороговой функции.

Поэтапная работа данного алгоритма представлена на рисунке 2.8.

Отдельно стоит отметить, что субтрактор KNN является относительно новым, поэтому в открытых источниках затруднительно найти информацию об ограничениях и практических особенностях применения данного алгоритма.

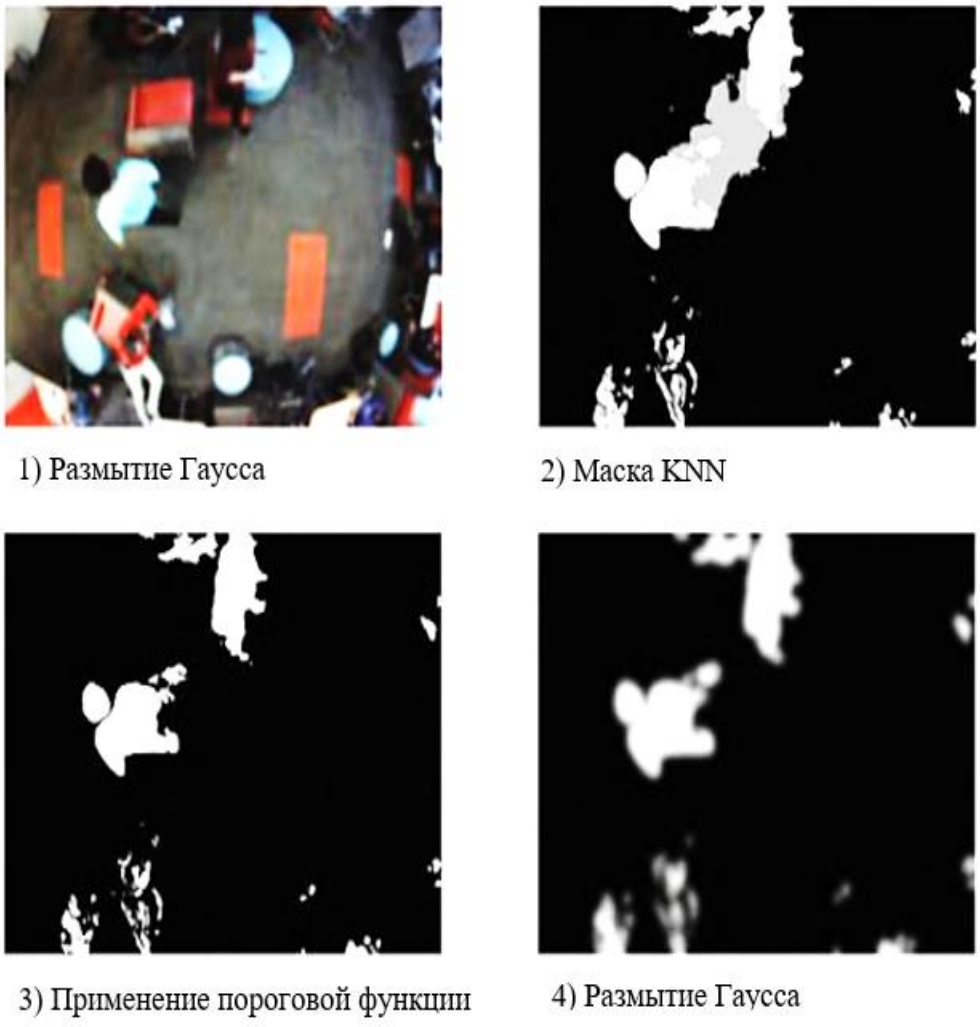


Рисунок 2.8 – изображение с поэтапной работой субтрактора фона KNN

2.6 Субтрактор CNT

Алгоритм CNT внес свой вклад в область отслеживания перемещения объектов благодаря модели деформируемого объекта для однократного отслеживания объектов (DPMOST). DPMOST разработана на основе звездообразной модели. Каждый объект моделируется с помощью набора взаимосвязанных частей с общим якорем. Как и во всех звездообразных моделях, общий якорь DPMOST вращается вокруг центра объекта. Звездообразные модели привлекательны тем, что они хорошо справляются с окклюзией (ситуация, в которой два объекта расположены приблизительно на одной линии и один объект, расположенный ближе к виртуальной камере, частично или полностью закрывает видимость другого объекта), поскольку не все части должны присутствовать для успешного обнаружения объекта. DPMOST обрабатывает деформацию объекта особым образом, в результате чего его части предлагаются для соединения и могут соединять внешние части через другой элемент, принадлежащий данной модели. Сильно смещенные части, служащие для моделирования сильных деформаций, могут иметь свою принадлежность к интересующему объекту, усиленную другими соединяющимися частями. На рисунке 2.9 рассмотрены ситуации возможные при работе DPMOST.

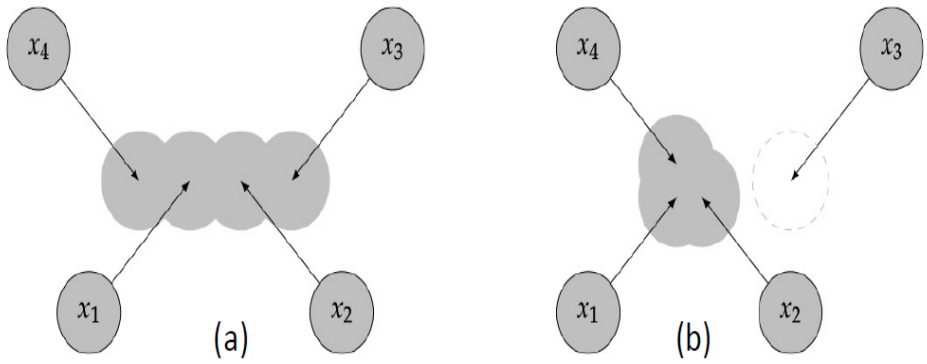


Рисунок 2.9 – схема работы DPMOST

На данном рисунке можно видеть, что под «a» x_3 и x_4 привязаны к одной и той же объектной модели с помощью частей x_1 и x_2, а в свою очередь под «b» x_3 отдалена и не является общей частью с объектной моделью.

Формально DPMOST строит объектную модель из эталонного набора элементов или частей объекта, полученного из образа инициализации (на основе предыдущие кадров) и сопровождающего его ограничивающего прямоугольника, причем эталонный внешний вид нормализован относительно среднего положения в координатах изображения. В рамках модели задается порог деформации объекта (константа) который позволяет до определенной степени точкам не терять связь с объектами (рисунок 2.10).

В таком случае задача состоит в том, чтобы определить набор соответствий между инициализирующим набором кадров и анализируемым кадром. Используемый метод оказывает значительное влияние на успешность алгоритма определения перемещений объектов, так как вся информация, полученная об объекте и его временном поведении, будет основана на наборе соответствий. CNT использует двухэтапный подход с целью повышения надежности согласования. На рисунке 2.16 представлен пример работы субтрактора фона CNT.

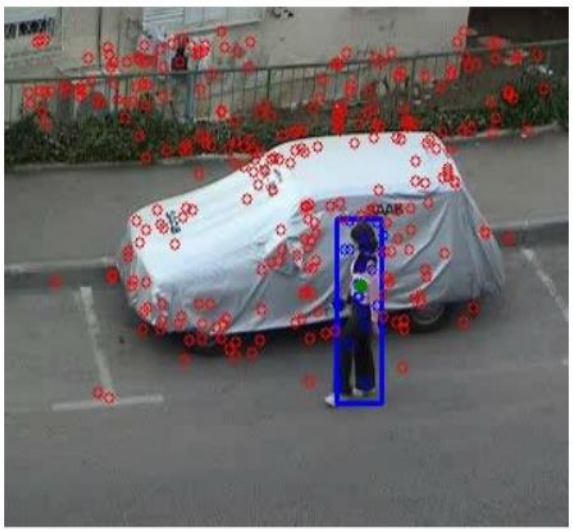


Рисунок 2.10 – Пример работы субтрактора фона CNT

На данном рисунке красными точками являются ключевые точки заднего фона, а синие – динамического объекта. Синей рамкой на изображении обозначен движущийся объект.

Статические соответствия получаются посредством использования меры подобия путем сопоставления дескрипторов между образом инициализации (набор предыдущих кадров) и текущим кадром. Поскольку это требует сравнения каждого вектора дескриптора-кандидата с векторами в эталонном наборе, вычислительная эффективность имеет большое значение. Применение бинарных векторных дескрипторов, в которых не используются числа с плавающей точкой, положительно влияет на производительность рассматриваемого метода. Дескрипторы можно сравнивать, используя оператор XOR для сравнения двух битовых строк одинаковой длины или для измерения расстояния между векторами дескрипторов, а также их можно эффективно вычислять на современных процессорах. Автор CNT подробно останавливается на выборе схемы сопоставления, используемой для сопоставления дескрипторов, и приводит следующую формулу для расчета достоверности соответствий (SNNDR):

$$SNNDR\big(p(x_i)\big) = \begin{cases} NN\big(p(x_i)\big)\ if\ \frac{d(p(x_i),NNp(x_i))}{d(p(x_i),NN_2p(x_i))} < \gamma \\ 0\ otherwise \end{cases} \qquad (2)$$

SNNDR считается надежным методом для определения достоверности соответствий для дескриптора-кандидата $p(x_i)$ и его ближайшего соседа. Данная формула основана на отношении расстояния d между дескриптором-кандидатом $p(x_i)$ и ближайшим соседом $NNp(x_i)$, на расстоянии d до второго ближайшего соседа $NN_2p(x_i)$. Суть, находящаяся за этим отношением, заключается в том, что чем ниже соотношение, которое должно быть меньше порога Y, тем более достоверное соответствие.

Таким образом, принцип действия алгоритма CNT основан на наблюдении за взаимным расположением ключевых точек изображения. Те

места, где изменение взаимного расположения точек не значительное, маркируются субтрактором как фон (рисунок 2.17, пиксели черного цвета). А те места изображения, где взаимное расположение ключевых точек изменилось значительно, маркируются как передний план (рисунок 2.17, пиксели белого цвета).

На рисунке 2.11 слева представлен анализируемый кадр видео ряда, а справа – маска заднего фона сгенерированная субтрактором фона CNT.

Также следует отметить, что субтрактор фона CNT разрабатывался, как более высокопроизводительная замена субтракторам MOG, MOG2 и GMG. Так как алгоритм CNT является относительно новым, то в открытых источниках литературы сложно найти исследования о практическом использовании данного субтрактора.

Рисунок 2.11 – Исходный кадр (слева) и маска заднего фона полученная субтрактором CNT

Как видно из рисунка 2.11, субтрактор хорошо справляется с задачей распознавания перемещений объектов.

В качестве выводов можно отметить следующее, что субтракторы фона относятся к активно развивающейся области алгоритмов анализа видео данных. Положительным моментом является то, что эти алгоритмы продолжают совершенствоваться. В основе рассмотренных алгоритмов лежат разные математические гипотезы о признаках наличия движения объектов в кадре. Конечные выводы при выборе конкретного типа субтрактора можно сделать

только на основе практического эксперимента, с использованием тестовой видеозаписи.

3 Программное обеспечение для выбора субтрактора фона

3.1 Описание разработанного программного обеспечения

Для выбора оптимального субтрактора программа выполняет следующие действия: сначала производится расчёт количества перемещений, определенных с использованием каждого субтрактора, затем производится расчет действительного количества перемещений, определенных с использованием библиотеки Cvlib и обученной нейросети предоставляемой данной библиотекой. После этого производится оценка эффективности работы субтрактора на заданном видеофрагменте, путем нахождения отношения действительного количества перемещений совершенных объектами к количеству перемещений найденных каждым субтрактором.

Также для оценки оптимального субтрактора используется такой показатель как время выполнения алгоритма субтрактора. Время рассчитывается с помощью функции timeit встроенной в базовую функциональность Python.

На высокоуровневом языке программирования общего назначения Python в интегрированной среде разработки PyCharm было создано приложение для автоматизации процесса выбора субтрактора фона при создании систем компьютерного зрения.

Основное назначение приложения – обеспечить разработчиков систем компьютерного зрения приложением, которое могло бы подсказать в выборе наиболее оптимального субтрактора в конкретных условиях работы.

Алгоритм работы с программой следующий:

– разработчики производят видеосъемку с использованием той камеры, которую планируется использовать в составе системы компьютерного

зрения, с того же ракурса и тех же объектов изображения которых планируется отслеживать;

– полученный видеофайл загружается в программу, после чего пользователь запускает выполнение программы, которая выполняет анализ работы каждого субтрактора и на основе оценки эффективности выводит оптимальный субтрактор. При этом выводится статистическая информация по работе субтракторов.

Основные особенности разработанного программного обеспечения:

– поддержка всех форматов видео (mp4, mkv, mov, avi и т.д.) при условии наличия соответствующих кодеков в системе;

– поддержка всех наиболее известных субтракторов фона (GMG, MOG, MOG2, KNN, CNT);

– вывод в программе статистической информации по обработке видео субтрактором (для каждого субтрактора отдельно);

– вывод графическое представление результатов работы каждого субтрактора синхронно с анализируемым кадром видеоряда (для каждого субтрактора в отдельном окне).

3.2 Описание программного кода

В разработанном программном обеспечении используется следующие библиотеки:

– OpenCV – библиотека алгоритмов компьютерного зрения, обработки изображений и численных алгоритмов общего назначения с открытым кодом;

– NumPy – библиотека обеспечивающая поддержку многомерных массивов, высокоуровневых математических функций, предназначенных для работы с многомерными массивами;

– Cvlib – библиотека алгоритмов компьютерного зрения позволяющая использовать алгоритмы с применением обученной нейросети для взаимодействия с видеопотоком.

В строках 1-6 программного кода осуществляется подключение библиотек к проекту. Причем обращение к функциям библиотеки NumPy для краткости будет осуществляться через сочетание np, а для библиотеки cvlib используется сокращение cv.

В строках 8-20 осуществляется создание графического интерфейса, элементов управления программой, задания событий, вызываемые в ответ на нажатие кнопок (рисунок 3.1).

В строке 21 указывается путь до открываемого видеофрагмента.

В строках 25-29 осуществляется инициализация субтракторов заднего фона со стандартными параметрами в следующем порядке: GMG, MOG, MOG2, KNN, CNT.

В строках 32-36 создаются переменные для подсчета количества движения в кадре.

```python
import PySimpleGUI as sg
import cv2
import cvlib as cv
import operator
import timeit
from cvlib.object_detection import draw_bbox

layout = [
    [sg.Text('Выберите видеофрагмент для анализа'), sg.InputText(), sg.FileBrowse('Обзор'),
     ],
    [sg.Output(size=(88, 20))],
    [sg.Submit('Выполнить'), sg.Cancel('Выйти')]
]
window = sg.Window('Selection subtraction', layout)

while True:
    event, values = window.read()
    if event in (None, 'Выйти', 'Cancel'):
        break
    if event == 'Выполнить':
        video_path = values[0] #Путь до видеофрагмента
        cv2.ocl.setUseOpenCL(False)

        #Инициализация субтракторов
        fgbgGMG = cv2.bgsegm.createBackgroundSubtractorGMG(10, .8)
        fgbgMOG = cv2.bgsegm.createBackgroundSubtractorMOG(300)
        fgbgMOG2 = cv2.createBackgroundSubtractorMOG2()
        fgbgKNN = cv2.createBackgroundSubtractorKNN(100, 400, True)
        fgbgCNT = cv2.bgsegm.createBackgroundSubtractorCNT(5, True)

        #Переменные для подсчета количества перемещений объектов
        numberMovementsObjectGMG = []
        numberMovementsObjectMOG = []
        numberMovementsObjectMOG2 = []
        numberMovementsObjectKNN = []
        numberMovementsObjectCNT = []
```

Рисунок 3.1 – Создание графического интерфейса программы

В строке 40 осуществляется считывание видеофайла с заданным именем функцией VideoCapture() из библиотеки OpenCV (рисунок 3.2).

```
39  def DrawCountObjectsGMG():
40      cap = cv2.VideoCapture(video_path)
41      fps = cap.get(cv2.CAP_PROP_FPS)
42      print("GMG: Количество кадров в секунду : {0}".format(fps))
43      while (cap.isOpened):
44
45          ret, frame = cap.read()
46
47          # Если ret true, то ошибок нет с cap.isOpened
48          if not ret:
49              break
50
51          fgmask = fgbgGMG.apply(frame)
52          (contours, hierarchy) = cv2.findContours(fgmask.copy(), cv2.RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
53
54          # Цикл для контуров
55          for c in contours:
56              if cv2.contourArea(c) < 500:
57                  continue
58
59              # Получить ограничивающую рамку из контура
60              (x, y, w, h) = cv2.boundingRect(c)
61              x1 = w / 2
62              y1 = h / 2
63              cx = x + x1
64              cy = y + y1
65              numberMovementsObjectGMG.append([cx, cy])
66
67              # Нарисовать ограничивающую рамку
68              cv2.rectangle(frame, (x, y), (x + w, y + h), (0, 255, 0), 2)
69
70          cv2.imshow('OriginalGMG', frame)
71          cv2.imshow('ForegroundGMG', fgmask)
72
73          cv2.moveWindow('OriginalGMG', 0, 0)
74          cv2.moveWindow('ForegroundGMG', 600, 0)
75
76          if cv2.waitKey(1) & 0xFF == ord("q"):
77              break
78      print('GMG objects', len(numberMovementsObjectGMG))
79      cv2.destroyAllWindows()
```

Рисунок 3.2 – Функция по работе с субтрактором

В строках 41 – 42 переменные для вывода информации о количестве кадров в секунду.

В строке 43 запускается бесконечный цикл While, внутри которого осуществляются все вычисления, связанные с работой субтракторов и выводом результатов на экран. Каждая последующая итерация цикла отвечает за обработку каждого следующего кадра указанного видеофайла.

В строках 48, 49 указано условие, что если следующего кадра не существует (на предыдущей итерации был последний кадр), то осуществляется выход из бесконечного цикла.

В строках 51-52 применяется заданный субтрактор для расчёта маски, затем поиск перемещений с использованием контуров.

В строках 55 -65 осуществляется подсчет количества перемещений.

В строке 68 происходит рисование ограничивающей рамки при перемещении объектов.

В строках 70-77 для каждого субтрактора осуществляется создание своего окна, куда транслируются результат обработки текущего кадра (белые пиксели обозначают движение, черные – задний фон, серые – тень движущегося объекта). Внутри каждого окна для простоты зрительного восприятия выводится названия субтрактора. С помощью клавиши можно «Q» остановить обработку.

В строках 78 – 79 происходит вывод количества перемещений, обнаруженных в кадре данным субтрактором, затем происходит освобождения ресурсов. Таким же образом для каждого субтрактора была создана своя функция для обработки видеофрагмента (рисунок 3.3.).

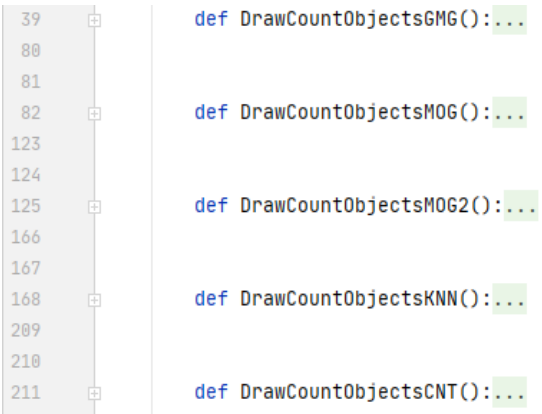
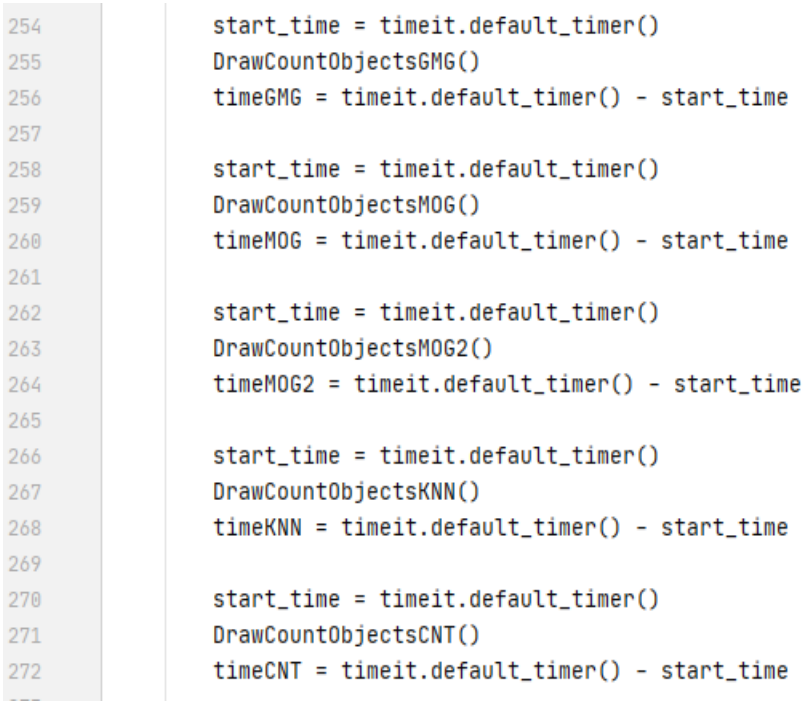


Рисунок 3.3 – Функции для работы с субтракторами

В строках 254 – 272 происходит расчет времени выполнения анализа видеофрагмента с использованием каждого субтрактора (рисунок 3.4).



```
254         start_time = timeit.default_timer()
255         DrawCountObjectsGMG()
256         timeGMG = timeit.default_timer() - start_time
257
258         start_time = timeit.default_timer()
259         DrawCountObjectsMOG()
260         timeMOG = timeit.default_timer() - start_time
261
262         start_time = timeit.default_timer()
263         DrawCountObjectsMOG2()
264         timeMOG2 = timeit.default_timer() - start_time
265
266         start_time = timeit.default_timer()
267         DrawCountObjectsKNN()
268         timeKNN = timeit.default_timer() - start_time
269
270         start_time = timeit.default_timer()
271         DrawCountObjectsCNT()
272         timeCNT = timeit.default_timer() - start_time
```

Рисунок 3.4 – Подсчет количества времени работы функций с субтракторами

В строке 277 определена функция, которая использует функциональность библиотеки Cvlib для нахождения действительного количества перемещений, совершаемых объектами (рисунок 3.5).

В строке 280 осуществляется считывание видео файла с заданным именем функции VideoCapture() из библиотеки OpenCV.

В строках 282-284 идет проверка возможности открытия файла.

В строке 286 переменная для подсчета количества перемещений, выполняемых объектами.

В строке 289 запускается бесконечный цикл While, внутри которого осуществляются все вычисления, связанные с определением количества перемещающихся объектов с использованием функций библиотеки Cvlib, и дальнейшим выводом результатов на экран. Каждая последующая итерация цикла отвечает за обработку каждого следующего кадра указанного видеофайла.

```python
# Функция для обнаружения перемещний объектов с использованием Cvlib
def CountRealMovementsObject():

    # Открыть видеофрагмент
    cap = cv2.VideoCapture(video_path)

    if not cap.isOpened():
        print("Файл не обнаружен")
        exit()

    countMovementsObject = 0

    # Цикл для всех кадров
    while cap.isOpened():

        # Считать кадр
        status, frame = cap.read()

        if not status:
            break

        # Определить перемещения объектов
        bbox, label, conf = cv.detect_common_objects(frame, confidence=0.5, model='yolov3')

        # Переменная для подсчета количества перемещений объектов
        countMovementsObject += len(bbox)

    print('Количество перемещений объектов', countMovementsObject)

    # Освободить ресурсы
    cap.release()
    cv2.destroyAllWindows()

    return countMovementsObject;
```

Рисунок 3.5 – Функция для подсчета действительного количества перемещений с использованием библиотеки Cvlib

В строках 314 – 321 производятся расчеты точности субтракторов (рисунок 3.6).

В строках 324-327 выводится время выполнения работы каждого субтрактора.

В строках 331-341 выводятся значения точности субтракторов в процентном и числовом виде.

```
# Произвести рассчеты точности субтракторов
countMovementsObject = CountRealMovementsObject()
bGMG = len(numberMovementsObjectGMG) / countMovementsObject
bMOG = len(numberMovementsObjectMOG) / countMovementsObject
bMOG2 = len(numberMovementsObjectMOG2) / countMovementsObject
bKNN = len(numberMovementsObjectKNN) / countMovementsObject
bCNT = len(numberMovementsObjectCNT) / countMovementsObject
Result = {'GMG': bGMG, 'MOG': bMOG, 'MOG2' : bMOG2, 'KNN': bKNN, 'CNT' : bCNT}

# Вывести время работы субтракторов
print('Время GMG', timeGMG)
print('Время MOG', timeMOG)
print('Время MOG2', timeMOG2)
print('Время KNN', timeKNN)
print('Время CNT', timeCNT)

# Вывести точность работы субтракторов в процентах
print('Точность субтрактора GMG {percent:.2%}'.format(percent=len(numberMovementsObjectGMG) / countMovementsObject))
print('Точность субтрактора MOG {percent:.2%}'.format(percent=len(numberMovementsObjectMOG) / countMovementsObject))
print('Точность субтрактора MOG2 {percent:.2%}'.format(percent=len(numberMovementsObjectMOG2) / countMovementsObject))
print('Точность субтрактора KNN {percent:.2%}'.format(percent=len(numberMovementsObjectKNN) / countMovementsObject))
print('Точность субтрактора CNT {percent:.2%}'.format(percent=len(numberMovementsObjectCNT) / countMovementsObject))

print("Точность субтрактора GMG", bGMG)
print("Точность субтрактора MOG", bMOG)
print("Точность субтрактора MOG2", bMOG2)
print("Точность субтрактора KNN", bKNN)
print("Точность субтрактора CNT", bCNT)

print('Выбран оптимальный субтрактор для данного видеофрагмента ', max(Result.items(), key=operator.itemgetter(1))[0])
```

Рисунок 3.6 – Вывод точности и скорости работы субтракторов

В конце работы программы на основании полученных значений точности и времени выполнения выводится наиболее оптимальный субтрактор для данного видеофрагмента.

3.3 Пример использования

Рассмотрим пример использования приложения. Например, планируется разработать систему компьютерного зрения, контролирующую перемещение движущихся объектов. Требуется определить какой субтрактор заднего фона больше подходит в рамках решаемой задачи.

Для этого используем фрагмент видео снятого в тех же условиях, в которых планируется использование разрабатываемой системы компьютерного зрения. В данном случае видеозапись велась со скоростью 30 кадров в секунду, общая длина записи 7 секунд.

Для того, чтобы понять, какой субтрактор фона лучше справится с задачей обнаружения перемещений объектов необходимо загрузить в программу отснятый видеофрагмент (рисунок 3.7)

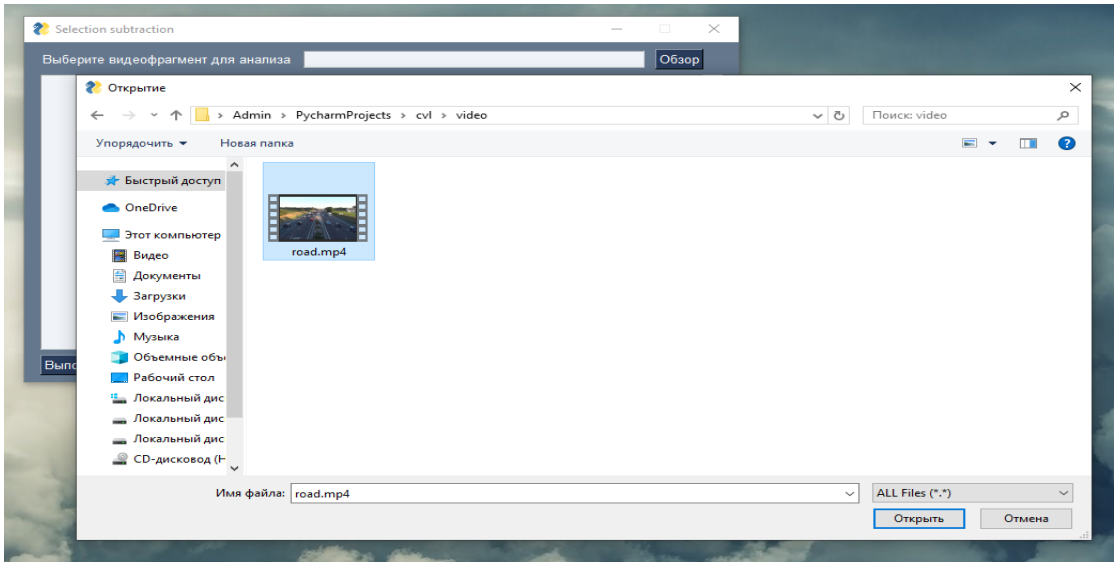


Рисунок 3.7 – Выбор фрагмента видео

После нажатия кнопки «Выполнить» запустится процесс анализа видеофрагмента с использованием каждого субтрактора. При этом будет происходить расчет количества перемещений определенных субтрактором. Так сначала запустится субтрактор GMG (рисунок 3.8).

Рисунок 3.8 – Работа субтрактора GMG

Затем будет запущен субтрактор MOG (рисунок 3.9).

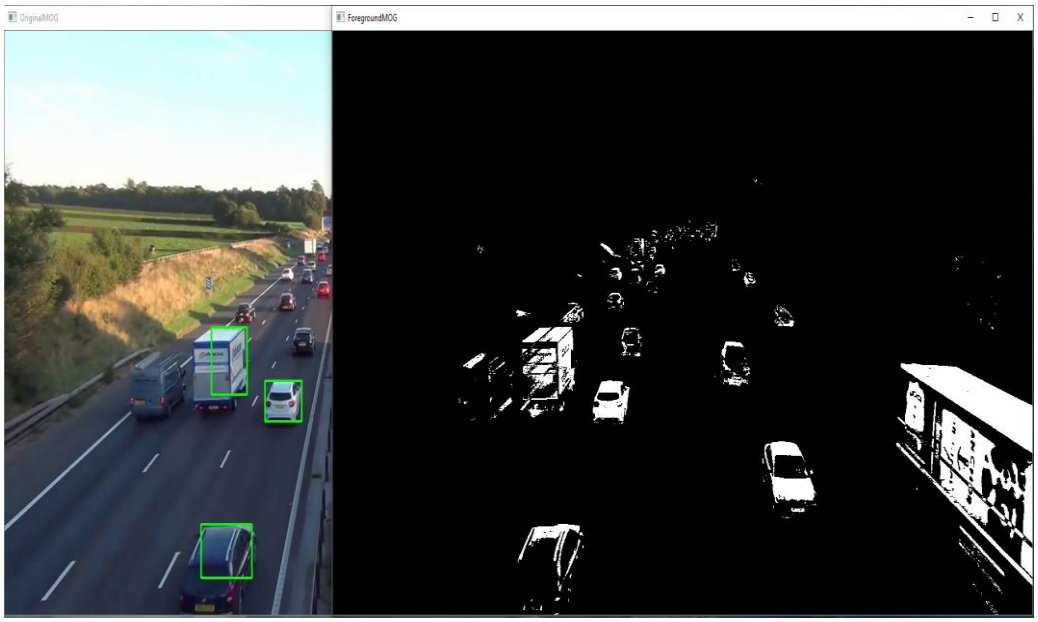


Рисунок 3.9 – Работа субтрактора MOG

После будет выполнен анализ субтрактора MOG2 (рисунок 3.10)

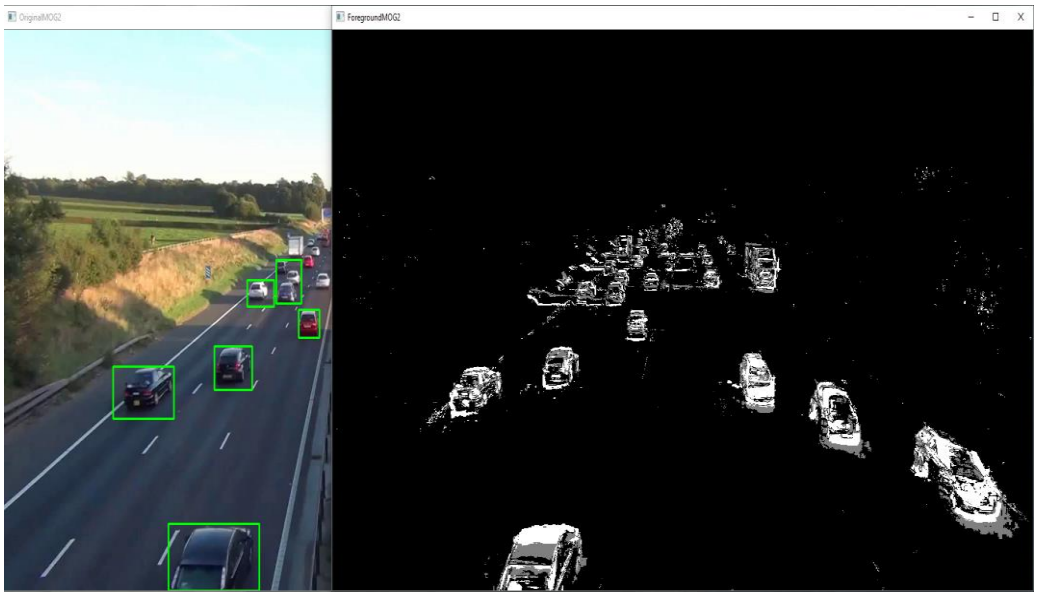


Рисунок 3.10 – Работа субтрактора MOG2

Затем будет выполнен субтрактор KNN (рисунок 3.11).

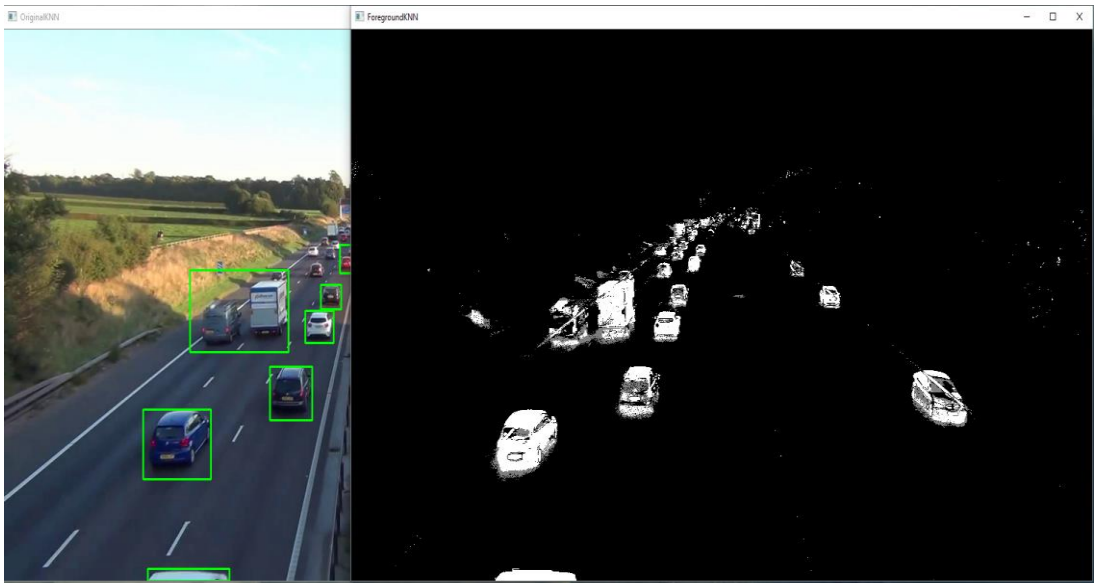


Рисунок 3.11 – Работа субтрактора KNN

И уже после субтрактора KNN будет выполнен субтрактор CNT (рисунок 3.12).

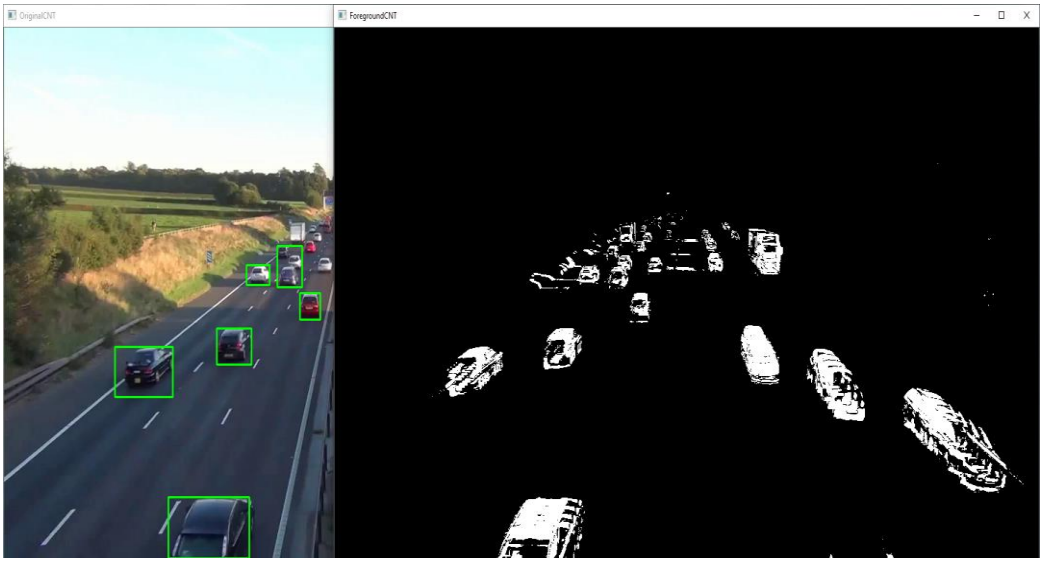


Рисунок 3.12 – Работа субтрактора CNT

После расчета количества перемещений определенных каждым субтрактором, программа запустит функцию из библиотеки Cvlib для расчета действительного количества перемещений. Затем будет произведен расчет точности каждого субтрактора и их времени выполнения, после чего результат отобразится на экране, также будет представлена дополнительная информация по работе каждого конкретного субтрактора и количестве найденных ими перемещений объектов, и также будет выведена информация о количестве кадров в секунду (рисунок 3.13).

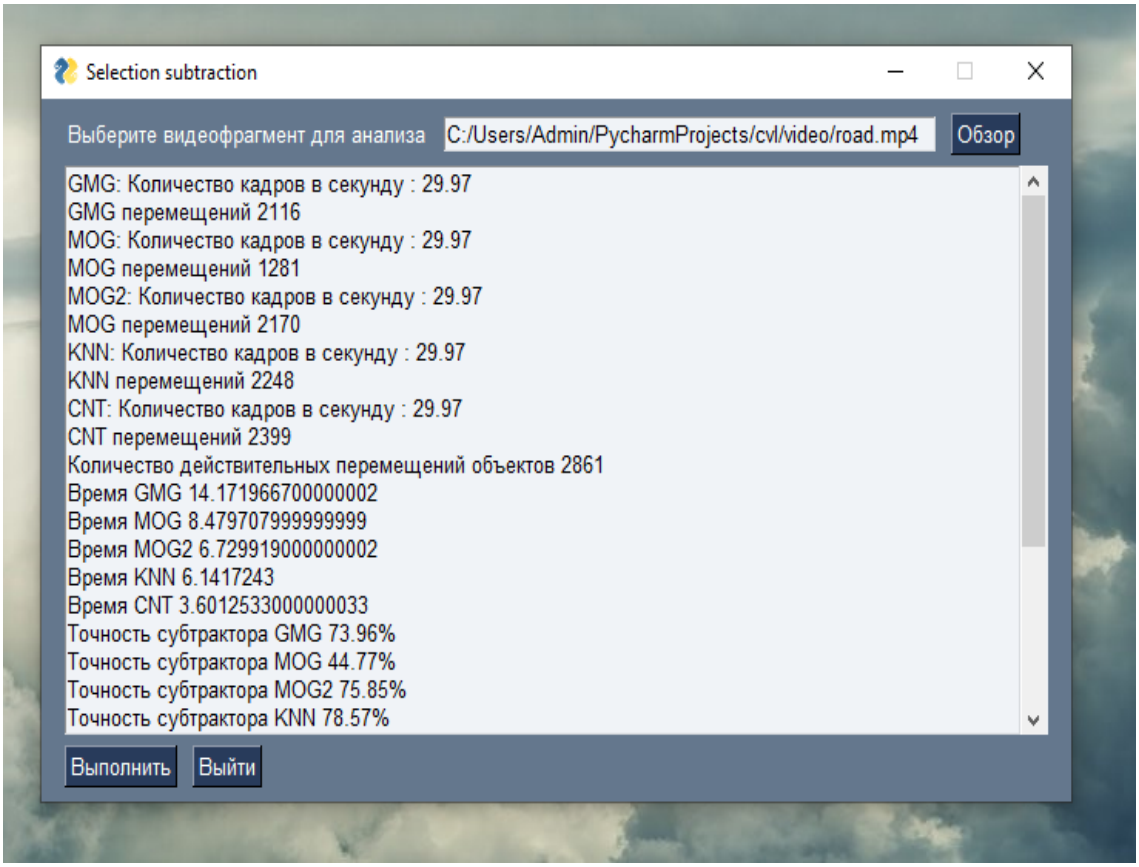


Рисунок 3.13 – Результат работы программы с данными по каждому субтрактору

На основе полученных значений точности и времени будет выбран наиболее оптимальный субтрактор, в данном случае выбран субтрактор CNT (рисунок 3.14).

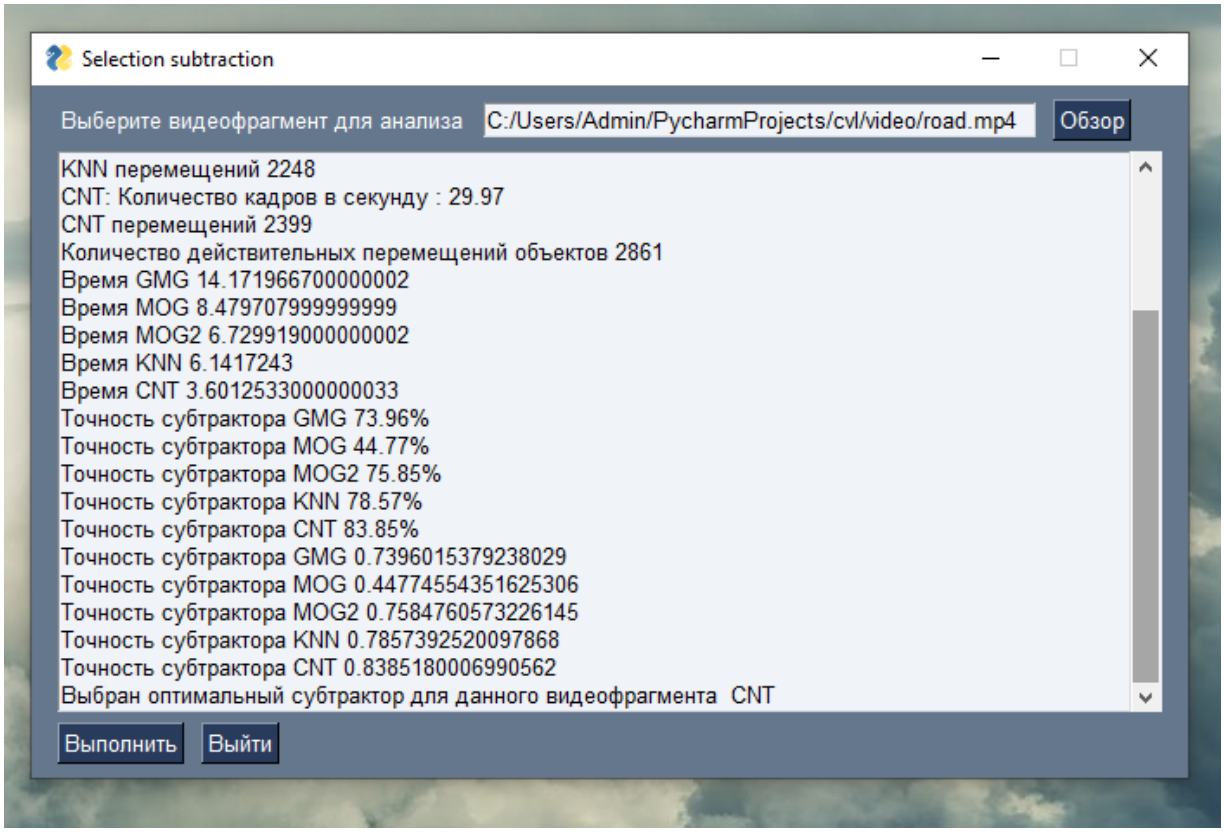


Рисунок 3.14 – Результат выбора оптимального субтрактора

Итак, благодаря использованию нашего приложения был выбран наиболее оптимальный субтрактор для данного видеофрагмента.

Таким образом, можно сделать вывод, что разработанное программное обеспечение справляется с поставленными задачами и позволяет автоматизировать процесс разработки систем компьютерного зрения за счет предоставления программы для выбора оптимального субтрактора фона под поставленные условия видеосъемки.

Заключение

В ходе выполнения работы был проведен анализ бизнес-процессов происходящих при разработке систем компьютерного зрения, которое показало, что разработчикам при создании систем компьютерного зрения часто требуется решать задачу определения перемещений объектов по данным видеоряда. Поставленная задача решается с помощью специального класса алгоритмов – субтракторов фона.

Ввиду того, что человеку сложно спрогнозировать результаты работы субтракторов фона, разработчикам систем компьютерного зрения на этапе тестирования часто приходиться возвращаться на предыдущие этапы разработки, чтобы поменять комбинацию используемых в системе алгоритмов анализа изображений, что является негативным фактором для эффективности работы разработчиков систем компьютерного зрения.

С учетом вышесказанного становится ясно, что для автоматизации процесса разработки систем компьютерного зрения необходима разработка программы, позволяющей выбирать субтрактор фона путем анализа видеоданных, полученных в условиях будущего функционирования системы компьютерного зрения.

Таким образом, для автоматизации процесса выбора субтрактора фона была разработана программа на языке Python для анализа видеофрагмента. Программное обеспечение позволяет осуществлять выбор оптимального субтрактора для заданного видеоряда.

Благодаря внедрению данного программного обеспечения в процесс разработки систем компьютерного зрения экономится время, связанное с этапом выбора субтрактора, что положительно сказывается на эффективности работы разработчиков систем компьютерного зрения. Подробное описание разработанного программного обеспечения представлено в третьей главе бакалаврской работы.

Список используемой литературы

1. Евтушенко, В.Ю. Анализ методов обнаружения объектов /В.Ю Евтушенко, А.Я Номерчук, А.А. Панина, А.И. Хвостовец // Современные технологии, естествознание, педагогика Южный федеральный университет. 2015. – Ростов-на-Дону : Издательство Южный федеральный университет, 2015. – С. 52-55. – Текст : непосредственный.

2. Кулманакова, Е.В. Распознавание и отслеживание людей на видео / Е.В Кулманакова, В.Г. Спицын// Технологии Microsoft в теории и практике программирования Национальный исследовательский Томский политехнический университет. 2012. – Томск : Издатель Национальный исследовательский Томский политехнический университет, 2012. – С. 78-80. – Текст : непосредственный.

3. Локтеев, А.С. Алгоритм выделения объектов на основе вычитания фона / А.С. Локтеев // Новые информационные технологии в научных исследованиях материалы XXII Всероссийской научно-технической конференции студентов, молодых ученых и специалистов. Рязанский государственный радиотехнический университет. 2017. – Рязань : Издатель Рязанский государственный радиотехнический университет, 2017. – С. 219-220. – Текст : непосредственный.

4. Небаба, С.Г. Увеличение скорости трекинга алгоритмом виолы-джонса в системе распознавания личности с помощью метода вычитания фона / С.Г. Небаба, А.А Захарова // Современные проблемы математического моделирования, обработки изображений и параллельных вычислений 2017 (СПММОИИПВ-2017) Труды Международной научной конференции. 2017. – Ростов-на-Дону : Издатель Общество с ограниченной ответственностью "ДГТУ-ПРИНТ", 2017. – С. 187-195. – Текст : непосредственный.

5. Савостин, А.А. Использование метода вычитания фона для обнаружения автотранспорта в видеопотоке / А.А Савостин // Современные проблемы радиотехники и телекоммуникаций "РТ-2015" Материалы 11-ой

международной молодежной научно-технической конференции. Севастопольский государственный университет; под ред. А. А. Савочкина. 2015. – Севастополь : Издатель Федеральное государственное автономное образовательное учреждение высшего образования "Севастопольский государственный университет", 2015. – С. 160. – Текст : непосредственный.

6. Селянкин, В.В. Компьютерное зрение. Анализ и обработка изображений / В.В Селянкин, С. Макаров. – СПб : Издательство «Лань», 2019. – С. 152. – Текст : непосредственный

7. Синдеев, М.С. Отслеживание контура лабораторной мыши в видеопоследовательности на основе метода ветвей и границ / М.С. Синдеев, А.С. Конушин // ГРАФИКОН'2010 Труды конференции. Санкт-Петербургский государственный университет информационных технологий, механики и оптики. 2010. – Москва : Издательство Автономная некоммерческая организация Научное общество "Графикон", 2010. – С. 214-217. – Текст : непосредственный.

8. Снегирева, М.А. Цифровая обработка видеоизображений. Методы вычитания фона / М.А. Снегирева, А.И. Алиферова, С.В. Аксенов // Молодежь и современные информационные технологии Сборник трудов XI Международной научно-практической конференции студентов, аспирантов и молодых ученых. 2013. – Томск : Издатель Национальный исследовательский Томский политехнический университет, 2013. – С. 461-463. – Текст : непосредственный.

9. Чеурин, Е.А. Сравнение методов вычитания фона, построенных на основе смеси гауссиан (MOG) и устойчивых к дрожанию камеры / Я.Е Чеурин, С.В. Машкин // Физика для Пермского края Материалы региональной научно-практической конференции студентов, аспирантов и молодых ученых. Под общей редакцией Н. Н. Картавых; Пермский государственный национальный исследовательский университет. 2019. – Пермь : Издатель Пермский государственный национальный исследовательский университет, 2019. – С. 168-173. – Текст : непосредственный.

10. Шальнов, Е. Алгоритм вычитания фона, основанный на поблочных классификаторах / Е. Шальнов, В. Конолов, В. Конушин // ГРАФИКОН'2011 21-я международная конференция по компьютерной графике и машинному зрению. 2011. – Москва : Издатель Автономная некоммерческая организация Научное общество "Графикон", 2011. – С. 227-230. – Текст : непосредственный.

11. Rangaraju, H.G. Design of Efficient Reversible Parallel Binary Adder/Subtractor / H.G. Rangaraju, U. Venugopal, K.N. Muralidhara, K.B. Raja // International Conference on Advances in Communication, Network, and Computing, Computer Networks and Information Technologies, Second International Conference on Advances in Communication, Network, and Computing, CNC 2011, Bangalore, India, March 10-11, 2011. Proceedings: Part of the Communications in Computer and Information Science book series (CCIS, volume 142). – Berlin : Springer-Verlag Berlin Heidelberg, 2011. – P. 83-87. – Text : direct.

12. Gandhi, S.M. A New Reversible SMG Gate and Its Application for Designing Two's Complement Adder/Subtractor with Overflow Detection Logic for Quantum Computer-Based Systems / S.M. Gandhi, J. Devishree, S.S. Mohan // Computational Intelligence, Cyber Security and Computational Models, Proceedings of ICC3, 2013: Part of the Advances in Intelligent Systems and Computing book series (AISC, volume 246). New Delhi : Springer India, 2014. – P. 259-266. – Text : direct.

13. Govindrao, W.M. Design and Analysis of Quantum Dot Cellular Automata Technology Based Reversible Multifunction Block / W.M. Govindrao, K.P. Dakhole // Proceedings of the International Conference on Data Engineering and Communication Technology, ICDECT 2016, Volume 2: Part of the Advances in Intelligent Systems and Computing book series (AISC, volume 469). – Singapore : Springer Science+Business Media Singapore, 2017. – P. 399-410. – Text : direct.

14. Jena, G. Fast Subtractor Algorithm and Implementation / G. Jena // Advances in Human Factors, Software, and Systems Engineering, Proceedings of the AHFE 2016 International Conference on Human Factors, Software, and Systems Engineering, July 27-31, 2016, Walt Disney World®, Florida, USA: Part of the

Advances in Intelligent Systems and Computing book series (AISC, volume 492). – Cham : Springer International Publishing Switzerland, 2016. – P. 87-92. – Text : direct.

15. Lech, M. Examining Quality of Hand Segmentation Based on Gaussian Mixture Models / M. Lech, P. Dalka, G. Szwoch, A. Czyżewski // International Conference on Multimedia Communications, Services and Security, Multimedia Communications, Services and Security, 7th International Conference, MCSS 2014, Krakow, Poland, June 11-12, 2014. Proceedings: Part of the Communications in Computer and Information Science book series (CCIS, volume 429). – Cham : Springer International Publishing Switzerland, 2014. – P. 111-121. – Text : direct.

16. Raveendran, A. RISC-V Half Precision Floating Point Instruction Set Extensions and Co-processor / A. Raveendran, S. Jean, J. Mervin, D. Vivian, D. Selvakumar // International Symposium on VLSI Design and Test, VLSI Design and Test, 23rd International Symposium, VDAT 2019, Indore, India, July 4–6, 2019, Revised Selected Papers: Part of the Communications in Computer and Information Science book series (CCIS, volume 1066). – Singapore : Springer Nature Singapore Pte Ltd, 2019 – P. 482-495 – Text : direct.

17. Sethi, P. All-Optical Ultrafast Adder/Subtractor and MUX/DEMUX Circuits with Silicon Microring Resonators / P. Sethi, S. Roy // International Workshop on Optical Supercomputing, Optical Supercomputing, 4th International Workshop, OSC 2012, in Memory of H. John Caulfield, Bertinoro, Italy, July 19-21, 2012. Revised Selected Papers: Part of the Lecture Notes in Computer Science book series (LNCS, volume 7715). – Berlin : Springer-Verlag Berlin Heidelberg, 2013 – P. 42-53 – Text : direct.

18. Sun, J. Two-Digit Full Subtractor Logical Operation Based on DNA Strand Displacement / J. Sun, X. Li, C. Huang, G. Cui, Y. Wang // International Conference on Bio-Inspired Computing: Theories and Applications, Bio-inspired Computing – Theories and Applications, 11th International Conference, BIC-TA 2016, Xi'an, China, October 28-30, 2016, Revised Selected Papers, Part I: Part of the Communications in Computer and Information Science book series (CCIS, volume

681). – Singapore : Springer Nature Singapore Pte Ltd, 2016 – P. 21-29 – Text : direct.

19. Wang, Y. One-Bit Full Adder-Full Subtractor Logical Operation Based on DNA Strand Displacement / Y. Wang, X. Li, C. Huang, G. Cui, J. Sun // International Conference on Bio-Inspired Computing: Theories and Applications, Bio-inspired Computing – Theories and Applications, 11th International Conference, BIC-TA 2016, Xi'an, China, October 28-30, 2016, Revised Selected Papers, Part I: Part of the Communications in Computer and Information Science book series (CCIS, volume 681). – Singapore : Springer Nature Singapore Pte Ltd, 2016 – P. 30-38 – Text : direct.

20. Zhao, J. Design and Application of Full-Adder / J. Zhao, L. Wang, Z. Shi // International Conference on Mechatronics and Intelligent Robotics, Recent Developments in Mechatronics and Intelligent Robotics, Proceedings of the International Conference on Mechatronics and Intelligent Robotics (ICMIR2017) - Volume 1: Part of the Advances in Intelligent Systems and Computing book series (AISC, volume 690). – Cham : Springer International Publishing AG, 2018. – P. 425-432. – Text : direct.