МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ

Кафедра компьютерных технологий и систем

**ОБНАРУЖЕНИЕ АНОМАЛИЙ ДВИЖЕНИЯ НА ВИДЕОПОСЛЕДОВАТЕЛЬНОСТИ ПРИ ПОМОЩИ МЕТОДОВ ОБРАБОТКИ ВРЕМЕННЫХ РЯДОВ**

Курсовая работа

Ёды Никиты Дмитриевича

обучающегося 4 курса специальности

“Прикладная математика”

Научный руководитель:

старший преподаватель

Шолтанюк Станислав Витальевич

**ОГЛАВЛЕНИЕ**

[ВВЕДЕНИЕ 3](#_Toc185282888)

[Глава 1](#_Toc185282889) [ОБЗОР ОСНОВНЫХ МЕТОДОВ И АЛГОРИТМОВ АНАЛИЗА ДВИЖЕНИЯ ОБЪЕКТОВ ВИДЕОПОСЛЕДОВАТЕЛЬНОСТЕЙ 4](#_Toc185282890)

[1.1 Оптический поток 4](#_Toc185282891)

[1.2 Трекинг объектов 5](#_Toc185282892)

[1.3 Сегментация объектов 6](#_Toc185282893)

[1.4 Современные методы на основе глубокого обучения 6](#_Toc185282894)

[Глава 2](#_Toc185282895) [ИССЛЕДОВАНИЕ АНОМАЛИЙ ДВИЖЕНИЯ ОБЪЕКТОВ ВИДЕОПОСЛЕДОВАТЕЛЬНОСТИ 8](#_Toc185282896)

[2.1 Определение аномалий 8](#_Toc185282897)

[2.2 Обработка видеопоследовательностей 8](#_Toc185282898)

[2.3 Подходы к выявлению аномалий 9](#_Toc185282899)

[2.4 Методы анализа временных рядов 13](#_Toc185282900)

[2.5 Применение анализа временных рядов для выявления аномалий 15](#_Toc185282901)

[2.6 Классификация аномалий 15](#_Toc185282902)

[2.7 Применение классификации аномалий 16](#_Toc185282903)

[Глава 3](#_Toc185282904) [ПОСТАНОВКА ЗАДАЧИ. ПРИМЕНЕНИЕ МЕТОДОВ И АЛГОРИТМОВ АНАЛИЗА ВРЕМЕННЫХ РЯДОВ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ 18](#_Toc185282905)

[3.1 Постановка задачи 18](#_Toc185282906)

[3.2 Применение методов и алгоритмов анализа временных рядов для решения поставленной задачи 19](#_Toc185282907)

[ЗАКЛЮЧЕНИЕ 27](#_Toc185282908)

[СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ 28](#_Toc185282909)

[ПРИЛОЖЕНИЕ А 29](#_Toc185282910)

ВВЕДЕНИЕ

В современном мире, где объёмы данных растут с каждым днем, анализ и обработка информации становятся ключевыми задачами для различных областей науки и техники. Одной из актуальных задач является обнаружение аномалий в движении объектов на видеопоследовательностях. Аномалии могут сигнализировать о различных событиях, таких как нештатные ситуации, нарушения безопасности или необычное поведение объектов.

Методы обработки временных рядов представляют собой мощный инструмент для анализа динамических процессов. Они позволяют выявлять закономерности, предсказывать поведение системы и обнаруживать отклонения от нормального функционирования. С учетом этого, применение методов временных рядов для анализа видеопоследовательностей открывает новые горизонты в области компьютерного зрения и анализа данных.

Актуальность темы определяется не только ростом интереса к проблемам компьютерного зрения, но и необходимостью обеспечения безопасности в различных сферах, таких как транспорт, охрана, спорт и многие другие. В условиях постоянного увеличения объёма видеоданных становится важным автоматизировать процессы их анализа, что позволяет не только повышать эффективность систем наблюдения, но и оперативно реагировать на потенциальные угрозы.

Цель данной курсовой работы заключается в исследовании методов обнаружения аномалий в движении на видеопоследовательностях с использованием подходов обработки временных рядов.

Для достижения поставленной цели была поставлена задача изучить современные методы анализа движения объектов и выявления аномалий, исследовать алгоритмы обработки временных рядов, разработать подход к выявлению аномалий на основе этих методов.

Настоящее исследование направлено на изучение теоретических основ и практическую реализацию методов анализа временных рядов для решения задач обнаружения аномалий в движении объектов.

Глава 1

ОБЗОР ОСНОВНЫХ МЕТОДОВ И АЛГОРИТМОВ АНАЛИЗА ДВИЖЕНИЯ ОБЪЕКТОВ ВИДЕОПОСЛЕДОВАТЕЛЬНОСТЕЙ

Анализ движения объектов в видеопоследовательностях включает в себя разнообразные методы, каждый из которых имеет свои особенности и области применения. Эти методы могут варьироваться от простых подходов, таких как оптический поток, до более сложных алгоритмов, использующих машинное обучение и глубокие нейронные сети.

**1.1 Оптический поток**

Метод оптического потока основывается на анализе изменения яркости пикселей между последовательными кадрами. Он позволяет оценить движение объектов, вычисляя вектор движения для каждого пикселя. Одним из самых известных алгоритмов для вычисления оптического потока является метод Лукаса-Канаде. Этот алгоритм предполагает, что движение объектов между кадрами невелико и что яркость пикселей остается постоянной.

Уравнение оптического потока записывается как:

,

где (, ) - частные производные яркости изображения *I(x, y, t)* по пространственным координатам (*x*, *y)*, - частная производная яркости по времени *t*, *(u, )* - компоненты вектора оптического потока, то есть скорости изменения положения пикселя по осям (*x*, *y).*

Уравнение яркости описывается следующим образом:

,

где – яркость пикселя, – координаты пикселя, – вектор скорости.

Для вычисления оптического потока с использованием метода Лукаса-Канаде необходимо решить систему уравнений:

,

где () – производные яркости по (*x*, *y)*, – производные яркости по времени. Оптический поток широко используется для отслеживания объектов и выделения их траекторий, а также для анализа динамических сцен.

1.2 Трекинг объектов

Существует несколько популярных алгоритмов трекинга объектов. Один из них — фильтр Калмана, который используется для предсказания положения объектов на основе предыдущих наблюдений. Этот метод хорошо работает для линейного движения и может быть адаптирован для более сложных, нелинейных систем. Обновление состояния в фильтре Калмана задается формулой:

,

где – вектор состояния в момент времени *k*, – матрица перехода состояния, – матрица управления, – вектор управления, – шум процесса.

Другой известный метод — Mean Shift Clustering и его адаптация Continuously Adaptive Mean Shift. Эти алгоритмы используют цветовые гистограммы для отслеживания объектов, что делает их эффективными при изменении масштаба и ориентации объектов. Mean Shift Clustering основывается на итеративном перемещении к более плотным областям в пространстве признаков. Формула для обновления позиции в Mean Shift Clustering:

,

где *K* – ядро, – точки данных, *N* – общее количество точек.

Эти подходы часто применяются в реальном времени и могут быть адаптированы к различным условиям съемки.

1.3 Сегментация объектов

Сегментация является ключевым этапом в анализе движения. Она подразумевает разделение изображения на значимые области, которые могут быть проанализированы отдельно.

Пороговая сегментация — один из самых простых подходов, который основывается на разбиении изображения на основе интенсивности пикселей. Этот метод может быть эффективным в условиях хорошего контраста.

Сегментация на основе кластеризации, например, алгоритм K-means Clustering, группирует пиксели по цвету или текстуре. Обновление центров кластеров задается формулой:

,

где – центр *j*-го кластера, – количество точек в кластере *j*, – набор точек в кластере *j*, – точки данных.

Этот подход позволяет выделять объекты на фоне и может быть использован для анализа динамических сцен. Современные методы сегментации, такие как U-Net Convolutional Network и Mask Region-based Convolutional Neural Network, основаны на глубоких нейронных сетях и обеспечивают высокую точность в выделении объектов. Эти методы особенно эффективны в сложных сценах, где объекты могут перекрываться или иметь схожие цветовые характеристики.

## **1.4 Современные методы на основе глубокого обучения**

С развитием технологий глубокого обучения наибольшую популярность приобрели модели, такие как You Only Look Once. Этот алгоритм позволяет одновременно детектировать и классифицировать объекты в реальном времени, что делает его идеальным для применения в условиях потоковой передачи видео. Функция потерь для You Only Look Once задается следующим образом:

,

где *S* – размер сетки, *B* – количество боксов, – ошибка координат, – ошибка классификации.

Single Shot MultiBox Detector также является эффективным методом, который использует многоуровневую архитектуру для улучшения точности распознавания. Он позволяет детектировать объекты на различных масштабах, что важно для анализа сложных сцен.

Глава 2

ИССЛЕДОВАНИЕ АНОМАЛИЙ ДВИЖЕНИЯ ОБЪЕКТОВ ВИДЕОПОСЛЕДОВАТЕЛЬНОСТИ

Исследование аномалий движения объектов на видеопоследовательностях — это одна из ключевых задач компьютерного зрения и обработки видеоданных. Аномалии, как правило, определяются как события, отклоняющиеся от ожидаемого или нормального поведения в видеопоследовательности.

2.1 Определение аномалий

Аномалия в контексте видеопоследовательностей — это отклонение от нормального шаблона движения, которое может проявляться в:

1. Неправильном перемещении объектов (например, человек, внезапно бегущий в толпе).

2. Нарушении привычного потока движения (машина, едущая по тротуару).

3. Появлении необычных объектов или исчезновении привычных объектов.

Определение аномалии сильно зависит от контекста: то, что считается нормой в одном случае, может быть аномалией в другом.

2.2 Обработка видеопоследовательностей

Для анализа движения необходимо начать с обработки видеоданных, чтобы обеспечить качество и точность дальнейшего анализа. Этот процесс включает несколько ключевых этапов:

­ 1. Устранение шума. Видеоданные могут содержать различные виды помех, такие как цифровые артефакты, шумы камеры или внешние факторы (например, дождь, снег, пыль). Для их устранения используются фильтры сглаживания, например, медианный или гауссовский фильтр:

,

где - исходное изображение, - отфильтрованное изображение, - весовая функция фильтра, *k* – радиус окна фильтра.

2. Стабилизация видео. Для устранения колебаний камеры применяется компенсация движения, которая основана на определении матрицы преобразования между кадрами:

*,*

где *T* - матрица аффинного преобразования, (, *)* – сдвиг по осям, (*a, b, c, d)* - параметры масштабирования и вращения.

3. Преобразование данных. Для упрощения анализа видеоданные разбиваются на последовательности кадров. Пусть последовательность кадров, где – *i*-й кадр. При выборе ключевых кадров часто используется метод на основе разности кадров:

,

где *D* – мера изменения между соседними кадрами.

Выделение ключевых характеристик:

1. Использование оптического потока для определения направления и скорости движения. Формула для оптического потока (1).

2. Применение методик сегментации для выделения движущихся объектов. Формула для сегментации (2).

2.3 Подходы к выявлению аномалий

Существующие подходы к выявлению аномалий можно разделить на три основные группы:

1. Статистические методы.
2. Методы машинного обучения.
3. Методы глубокого обучения.

Статистические методы. Эти методы основаны на анализе данных через распределения и вероятности. Они предполагают, что нормальное поведение может быть описано статистической моделью, а аномалии представляют собой отклонения от этой модели.

1. Методы порогового анализа. Задаются фиксированные пороги для определённых характеристик (например, скорости, размера объекта или направления движения). Если значение превышает порог, событие считается аномалией. Например, если *x ­*– скорость объекта, то:

,

где *T* – пороговое значение.

2. Параметрические модели. Гауссовы модели, которые описывают данные как нормальное распределение. Аномалиями считаются точки, лежащие за пределами определённого доверительного интервала. Если , то аномалии могут быть определены как:

,

где *z* – значение из стандартного нормального распределения.

3. Непараметрические модели. Гистограммы или ядерная оценка плотности, используются для описания распределения данных без предположений о форме распределения. Ядерная оценка плотности может быть представлена как:

,

где - ядерная функция, *h* – ширина окна, а – наблюдаемые данные.

Методы машинного обучения.Машинное обучение предоставляет гибкие инструменты для выявления аномалий через обучение на данных.

1. Обучение с учителем. Предполагает наличие размеченного набора данных, где нормальные и аномальные события уже выделены. Модели классификации, такие как деревья решений, Support Vector Machine или ансамблевые методы (например, Random Forest), могут быть использованы для обучения классификатора. Например, для Support Vector Machine задача формулируется как:

,

где - вектор весов (опорных векторов), *C* - параметр регуляризации, - переменные, представляющие собой штрафы за ошибки в классификации.

При ограничениях:

,

где - метка класса, - вектор признаков для i-го образца, *b* – смещение, которое добавляется к линейной комбинации весов и входных данных, - транспонированный вектор весов .

2. Обучение без учителя. Применяется, когда размеченные данные отсутствуют. Методы кластеризации, такие как K-means Clustering, группируют данные:

,

где – центр кластера , *k* – число кластеров, - наблюдение в данных, - набор объектов, принадлежащих *i*-му кластеру.

3. Методы полуобучения. Основываются на обучении модели на размеченных нормальных данных. Например, алгоритмы одно-классовой Support Vector Machine (One-Class Support Vector Machine) могут быть определены как:

,

где – целевая функция, – вектор весов.

При ограничениях:

,

где - скалярное произведение вектора весов и преобразованного вектора признаков , - переменные, представляющие собой штрафы за ошибки или отклонения.

Методы глубокого обучения. Современные подходы, основанные на глубоких нейронных сетях, позволяют эффективно выявлять аномалии в сложных видеопоследовательностях:

1. Автоэнкодеры. Используются для сжатия и восстановления данных. Если объект или событие плохо восстанавливается, это может указывать на аномалию. Ошибка восстановления может быть определена как:

,

где - восстановленное значение.

2. Generative Adversarial Networks. Обучаются генерировать нормальные данные. Отклонения от созданной модели выявляются как аномалии. Функция потерь может быть представлена как:

,

где *D* – дискриминатор, *G* – генератор.

3. Свёрточные и рекуррентные нейронные сети: Convolutional Neural Network используются для анализа пространственных особенностей, таких как формы или текстуры, а Recurrent Neural Network или Long Short-Term Memory — для временных последовательностей. Для Long Short-Term Memory можно использовать следующие уравнения:

Формула для забвения:

,

где - вес входа для забытия информации, - сигмойдная функция, - матрица весов для забытия, - смещение для забытия, - конкатенация предыдущего состояния и входных данных .

Формула для входных данных:

,

где - вес входа для обновления информации, - матрица весов для входных данных, - смещение для входных данных.

Формула для состояния ячейки:

,

где - кандидат в состояние ячейки.

Обновление состояния ячейки:

,

где - новое состояние ячейки.

Формула для выходного состояния:

,

где - выходные веса, - окончательное выходное состояние.

4. Пространственно-временные модели. Комбинируют Convolutional Neural Network и Recurrent Neural Network для анализа одновременно пространственных и временных данных. Модель может быть представлена как:

,

где *Y* – предсказания, *X* - пространственные данные, *T* - временные данные.

2.4 Методы анализа временных рядов

Для исследования видеопоследовательностей используются как традиционные, так и современные методы анализа временных рядов:

1. Автокорреляционный анализ. Этот метод позволяет выявить временные зависимости между последовательными кадрами, такие как повторяющиеся или циклические паттерны движения.

,

где *N* - общее количество наблюдений, *-* среднее значение временного ряда, – значение временного ряда, *k* – сдвиг.

2. Методы анализа временных рядов, такие как авторегрессионные интегрированные модели и рекуррентные нейронные сети, применяются для предсказания движения объектов на основе их исторических данных. Модель AutoRegressive Integrated Moving Average определяется следующим уравнением:

,

где – временной ряд, B – оператор задержки, и - полиномы авторегрессии и скользящего среднего, – белый шум.

Эти подходы позволяют выявлять аномалии и предсказывать будущие позиции объектов. Например, Recurrent Neural Network хорошо справляются с задачами, связанными с временной зависимостью, и могут использоваться для анализа последовательностей видеоданных.

3. Спектральный анализ. Применяется для выявления частотных характеристик движения, таких как вибрации или периодические движения. Одним из ключевых методов является преобразование Фурье:

,

где - спектр сигналов, *x(n)* – временной ряд, *N* – количество отсчётов

4. Сегментация временных рядов. Этот метод делит временной ряд на отдельные фрагменты для выявления аномалий или изменения динамики. Один из подходов к сегментации — использование алгоритма CUSUM:

,

где - текущее наблюдение, - предполагаемое среднее, *k* - пороговое значение.

Методы машинного и глубокого обучения. Современные алгоритмы машинного обучения позволяют более эффективно анализировать временные ряды, извлекая скрытые закономерности:

1. Рекуррентные нейронные сети (Recurrent Neural Network). Используются для анализа последовательных данных, таких как траектории объектов. LSTM и GRU являются популярными модификациями Recurrent Neural Network, которые справляются с задачами долгосрочной зависимости и прогнозирования движения.

2. Свёрточные сети для временных данных. Convolutional Neural Network могут быть адаптированы для анализа временных последовательностей путем обработки временных окон, выделяя ключевые особенности движения объектов.

3. Пространственно-временные модели. Комбинация Convolutional Neural Network и Recurrent Neural Network используется для изучения как временных, так и пространственных зависимостей в видеопоследовательностях. Эти модели эффективны для анализа сложных динамических сценариев.

2.5 Применение анализа временных рядов для выявления аномалий

Анализ временных рядов позволяет выявлять отклонения от нормального поведения объектов или системы.

1. Детектирование изменений во временной последовательности. Методы, такие как детекторы резких изменений или всплесков, используются для выявления неожиданного изменения скорости или направления движения. Кластеризация может основываться на расстоянии между временными рядами, например, с использованием метрики Dynamic Time Warping, для которой можно использовать следующую формулу:

,

где *d(i, j)* - расстояние между точками *i* и *j* временных рядов.

2. Обнаружение аномальных траекторий. Например, использование кластеризации временных рядов для определения движений, которые отличаются от большинства.

3. Прогнозирование и сравнение. Если прогнозируемое поведение объекта существенно отклоняется от наблюдаемого, это может свидетельствовать об аномалии.

**2.6 Классификация аномалий**

Классификация аномалий по типу объекта:

1. Люди: аномалии в поведении людей могут быть разнообразными и часто выражаются в нарушении стандартных паттернов движения. Примеры включают резкие изменения траектории, ускорения или остановки, которые не соответствуют привычным закономерностям. Например, бег в толпе, перемещение в обратном направлении относительно основного потока людей или внезапные изменения в скорости движения, такие как резкие ускорения без видимой причины. Также могут встречаться отклонения, связанные с нарушениями личного пространства, например, вторжение в чужое пространство в ситуации, когда человек должен двигаться по строгому маршруту. Такие аномалии могут указывать на стрессовое или опасное состояние.

2. Транспортные средства: аномалии в движении транспортных средств могут проявляться как отклонения от стандартных маршрутов, нарушение скорости или поведение, которое не соответствует нормам дорожного движения. Например, движение по встречной полосе, резкие торможения без видимой причины, внезапные повороты или отклонения от заданного пути. Аномальные манёвры могут также включать движение в запрещенных зонах, выезд на тротуары, создание пробок или движение в условиях ограниченной видимости. Эти аномалии также могут быть связаны с неисправностями в транспортном средстве или внешними факторами, такими как плохие дорожные условия или погодные явления.

3. Животные и природные объекты: движение животных или природных объектов в некоторых случаях может быть нестандартным и создавать аномальные ситуации, требующие внимания. Например, необычные повадки животных, такие как бег в опасных районах или появление в местах, где их присутствие нехарактерно. Включение природных объектов, таких как лавины, оползни, торнадо или другие стихийные бедствия, тоже может привести к значительным изменениям в окружающей среде, что необходимо отслеживать для предотвращения аварий или спасательных операций. Такие аномалии могут иметь катастрофические последствия, если их не зафиксировать вовремя. Классификация аномалий по уровням анализа:

1. Уровень траекторий: анализируются отклонения в маршрутах движения объектов. Это может включать неожиданные пересечения траекторий или резкие изменения в их направлении.

2. Потоковое движение: анализируется общая динамика, например, нарушение плотности, скорости или однородности движения толпы, транспортного потока или групп объектов.

3. Индивидуальные объекты: внимание сосредоточено на характеристиках конкретного объекта, таких как изменение формы, размера или характера движения, что указывает на его аномальное поведение.

**2.7** **Применение классификации аномалий**

1. Безопасность и видеонаблюдение: анализ позволяет выявлять подозрительное поведение, например, агрессию или внезапное изменение направления движения в толпе, а также отслеживать транспортные средства, нарушающие правила дорожного движения.

2. Транспорт и логистика: методы используются для анализа траекторий беспилотных автомобилей и предотвращения аварий, а также для контроля нестандартного поведения на транспортных узлах, например, аномального перемещения багажа.

3. Медицина: задачи связаны с выявлением нарушений движения у пациентов, например, при расстройствах походки или внезапных падениях, а также с анализом видеоданных для диагностики двигательной дисфункции.

4. Спорт: эти методы позволяют анализировать движения игроков, выявлять тактические отклонения или ошибки, а также отслеживать траектории мяча для понимания нестандартных действий.

5. Промышленность и робототехника: задачи применяются для обнаружения отклонений в движении промышленных роботов, что может указывать на неисправности, или для мониторинга работы конвейерных линий с целью выявления нарушений в их функционировании.

Глава 3

ПОСТАНОВКА ЗАДАЧИ. ПРИМЕНЕНИЕ МЕТОДОВ И АЛГОРИТМОВ АНАЛИЗА ВРЕМЕННЫХ РЯДОВ ДЛЯ РЕШЕНИЯ ПОСТАВЛЕННОЙ ЗАДАЧИ

3.1 Постановка задачи

Задача заключается в анализе видеопоследовательности, в которой фиксируется движение автомобилей по дороге, для автоматического выявления аномалий в скорости их движения. Под аномалией в данном случае понимается ситуация, когда скорость автомобиля оказывается ниже или выше средней скорости, рассчитанной для всех транспортных средств в текущем видеоматериале.

Основные этапы постановки задачи:

1. Идентификация объектов движения. Видеопоследовательность содержит движущиеся автомобили, которые необходимо автоматически выделить. Для этого используются методы обработки видео, такие как детекция объектов и трекинг.

2. Вычисление параметров движения. Для каждого автомобиля определяется его скорость на основе анализа траекторий и временных характеристик. Вычисляется средняя скорость всех автомобилей за определённый период времени.

3. Определение критерия аномалии. Аномалией считается ситуация, когда скорость конкретного автомобиля значительно ниже средней скорости. Важно установить количественные пороговые значения, например, 80% от средней скорости, чтобы исключить незначительные отклонения.

4. Подготовка данных для визуализации. Видеокадры, на которых автомобили двигаются с аномально низкой скоростью, должны быть выделены и проанализированы для визуального представления и последующего подтверждения аномалий.

Цель задачи:

Выявление участков видеопоследовательности, где движение автомобилей нарушает ожидаемую динамику, с фокусом на низкой скорости, что может сигнализировать о потенциальных проблемах (например, пробки, неисправности автомобилей или другие факторы).

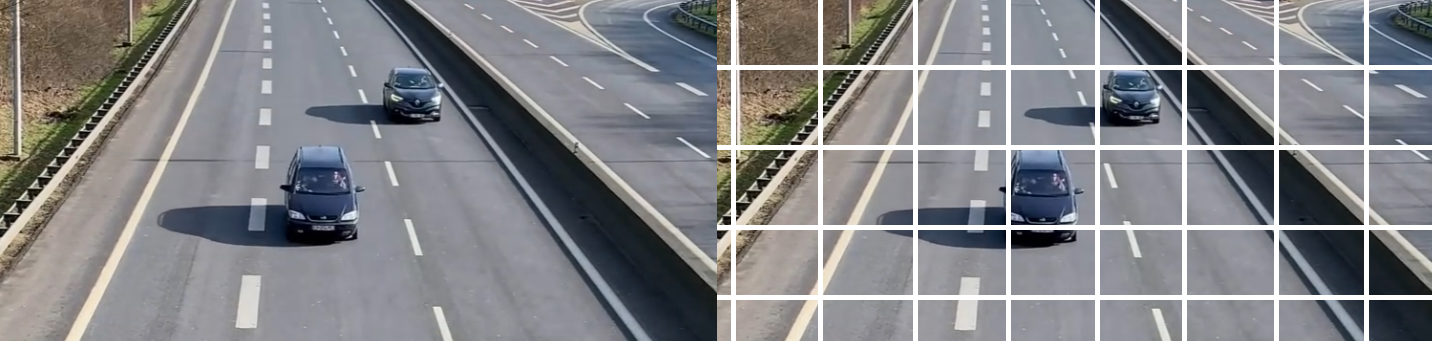
На данном этапе задача формулируется как определение подхода и критериев для выявления таких аномалий, а также подготовка данных для последующего анализа.

3.2 Применение методов и алгоритмов анализа временных рядов для решения поставленной задачи

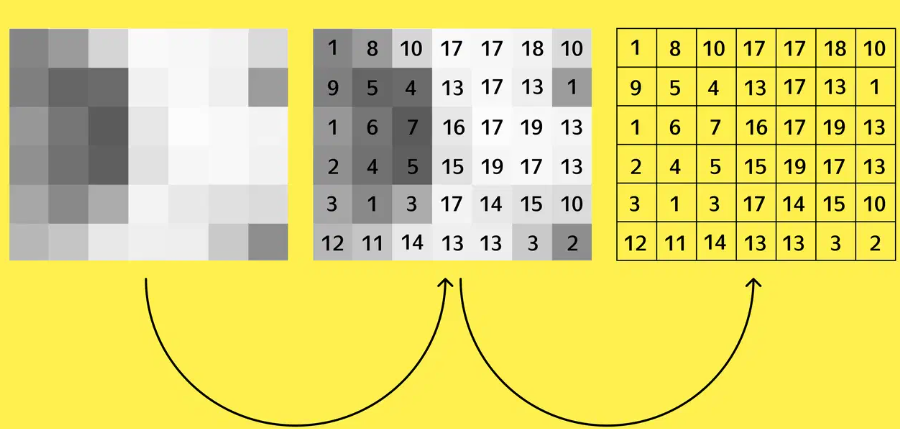
You Only Look Once, версия 8 — это современная модель глубокого обучения, предназначенная для задачи детекции и классификации объектов на изображениях и в видеопоследовательностях. Она позволяет выделять объекты в реальном времени с высокой точностью, что делает её особенно полезной для анализа видеопотоков.

Модель You Only Look Once использует уникальный подход к детекции объектов, который позволяет эффективно анализировать изображения. Основной принцип работы заключается в том, что входное изображение разбивается на несколько областей, что позволяет модели одновременно оценивать множество объектов.

При поступлении изображения модель разбивает его на сетку фиксированного размера, например, 13x13 или 19x19. Каждая ячейка этой сетки отвечает за обнаружение объектов, находящихся в её пределах. Это разделение позволяет модели сосредоточиться на различных участках изображения и эффективно ловить объекты, даже если они находятся близко друг к другу.

Рисунок 1 – Разбиение фото на сетку

Для каждой ячейки сетки модель применяет свёрточные нейронные сети. Cвёрточная нейронная сеть “видит” изображение в особом представлении — в виде трёхмерных массивов чисел или массивов матриц. В RGB цвет закодирован в виде трёх целых чисел от 0 до 255. Каждый пиксель в массиве матриц представлен в виде числа, которое показывает яркость в этом диапазоне [5].

Рисунок 2 – Представление картинки в виде матрицы

Чтобы нейросеть “узнала” машину, нужно проделать с изображением несколько типовых операций на каждом слое.



Рисунок 3 – Фотография машины

Ключевая из этих операций — свёртка. Во время свёртки нейросеть удаляет лишнее и оставляет полезное — то, что поможет проанализировать изображение. Например, линии, края или ровные области. Свёртку можно создавать для каждого признака. Нейросеть будет подбирать их во время распознавания и классификации на каждом свёрточном слое.

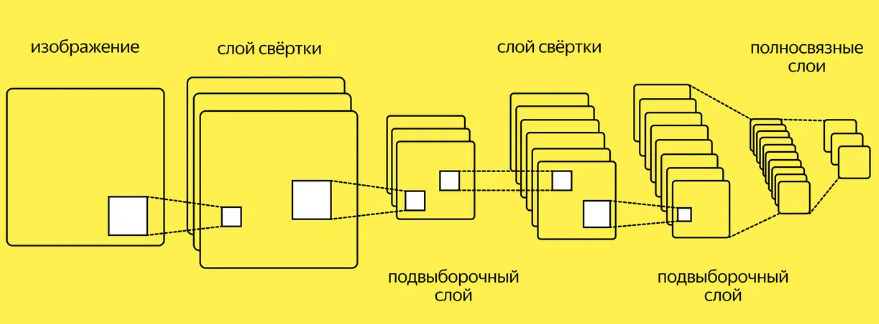


Рисунок 4 – Структура свёрточной нейронной сети

После свёрточного слоя идёт слой пулинга. Из признаков, которые выделил свёрточный слой, выбирает самые важные, а несущественные удаляет. К результату, который получился во время пулинга, можно снова применить свёрточный слой и сделать несколько циклов. Это нужно, чтобы выстроить иерархию признаков: от самых примитивных, например кусочков контура, до более сложных.

Первые слои нейросети анализируют мелкие детали изображения. Размер изображения может быть 2×2 или 3×3 пикселя — по такому маленькому контексту сложно выявить признаки. На следующих слоях появляются объекты посложнее. Чем глубже слой сети, тем более сложные признаки она извлекает. На последних слоях нейросеть выявляет признаки и пытается соотнести их с конкретными изображениями и определить, что именно на них нарисовано.

Чтобы анализировать изображение и выделять признаки, у свёрточного слоя есть фильтры или же ядра. Они представлены как массивы матриц.

Разберём свёртку на примере одного пикселя. По каждому пикселю проходит фильтр свёртки. На картинке выбрали пиксель со значением 2. Сначала нужно перемножить по очереди выбранный пиксель и соседние с ним со значениями в матрице свёртки, а потом всё сложить. После этого на исходной картинке заменить первое значение пикселя на то, что получилось, — 13, а значения соседних пикселей не менять. Эти же действия нужно проделать с каждым пикселем на картинке.

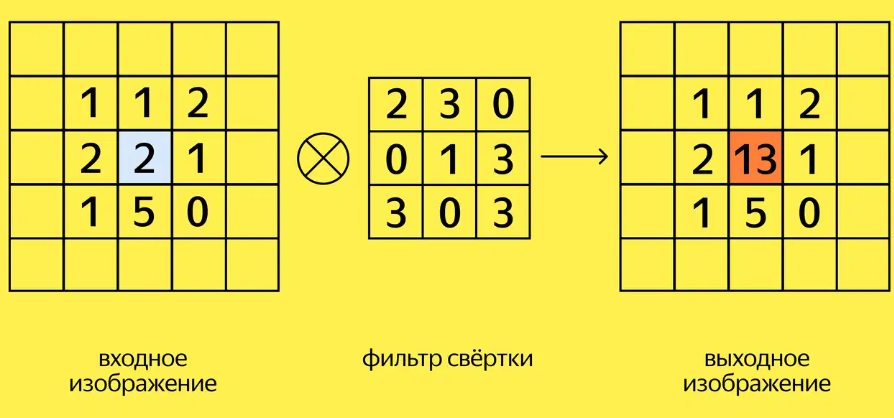


Рисунок 5 - Свёртка на примере одного пикселя

Так выглядит операция свёртки по всему изображению:

1. Фильтр проходит по каждому пикселю изображения, перемножает и суммирует числа своей матрицы и матрицы картинки. На выходе получается новая матрица.

2. Числа полученных матриц суммируются в одну матрицу.

3. К каждому значению матрицы добавляется одинаковое число — значение, на которое переместился фильтр, или шаг свёртки. Шаг равен 1 — фильтр перемещался на один пиксель. Шаг равен 2 — фильтр шагал на 2 пикселя. Финальная матрица — это один канал выходной карты признаков.

4. Все каналы, или матрицы, которые получили после обработки изображения фильтрами, объединяются в один тензор. В итоге получается изображение другого размера и с другим числом каналов.

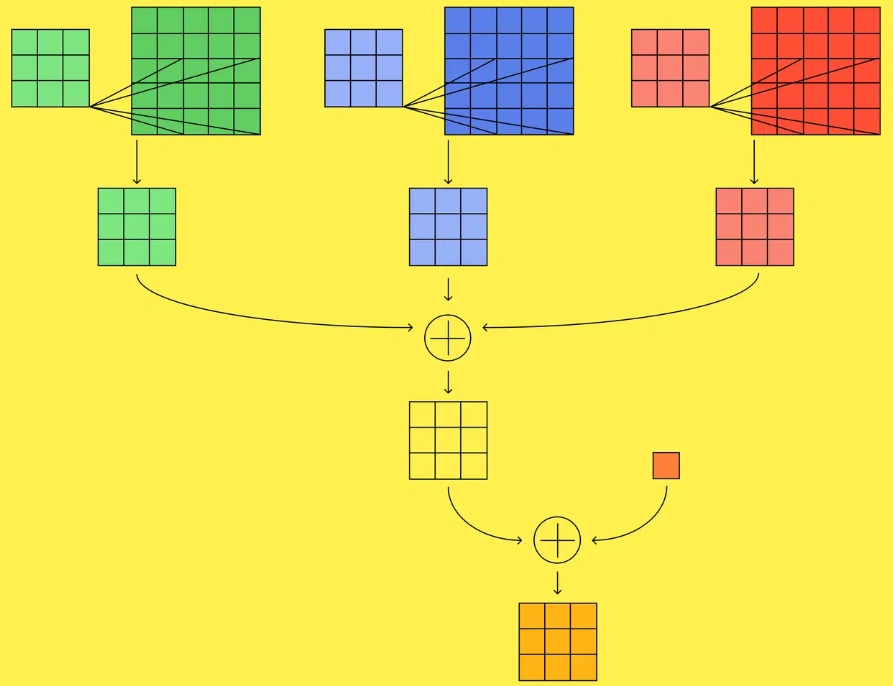


Рисунок 6 – Составление матрицы для RGB

После применения этих фильтров на разных слоях сети формируются карты признаков, представляющие собой матрицы, которые содержат информацию о выявленных характеристиках. Эти матрицы затем используются для классификации объектов, находящихся в пределах каждой ячейки сетки

Модель анализирует каждый кадр видеопоследовательности, выделяя автомобили с помощью рамок (Bounding Boxes).

Рисунок 7 - Детекция автомобиля

Для отслеживания автомобилей между кадрами используется дополнительный алгоритм трекинга, такой как Simple Online Realtime Tracking или DeepSORT. Это позволяет сопоставлять объекты в последовательных кадрах, формируя траектории движения. В основе SORT лежит Калманов фильтр, который используется для предсказания положения объектов на следующем кадре. Это позволяет корректировать положение объектов на основе предыдущих наблюдений.

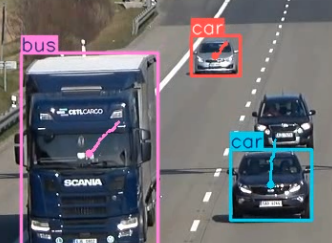


Рисунок 8 - Траектория движения

Для каждого обнаруженного автомобиля фиксируются его координаты в пикселях (центры рамок Bounding Box). Эти данные формируют временные ряды, которые затем используются для расчёта скорости и других параметров движения.

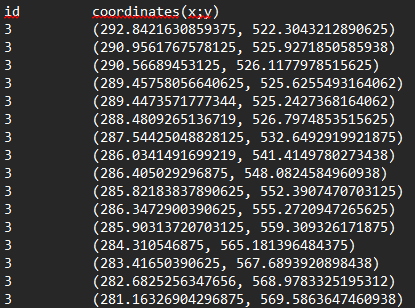


Рисунок 9 - Координата автомобиля

На основе координат и временных меток для каждого кадра вычисляется скорость автомобилей. Скорость рассчитывается с использованием формулы:

,

где ( и ( ​ — координаты объекта в двух последовательных кадрах, — временной интервал между кадрами.

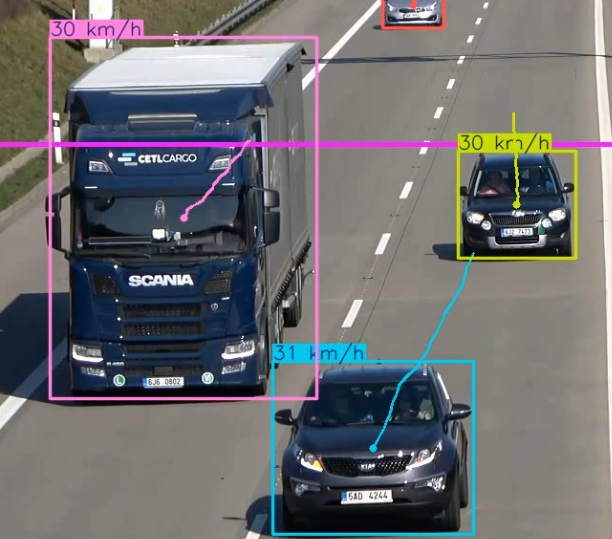


Рисунок 10 - Вычисление скорости автомобиля

На основании скорости объектов, вычисленной с помощью You Only Look Once, осуществляется сравнение с заранее установленным порогом. Например, если скорость автомобиля значительно ниже средней, это фиксируется как аномалия.

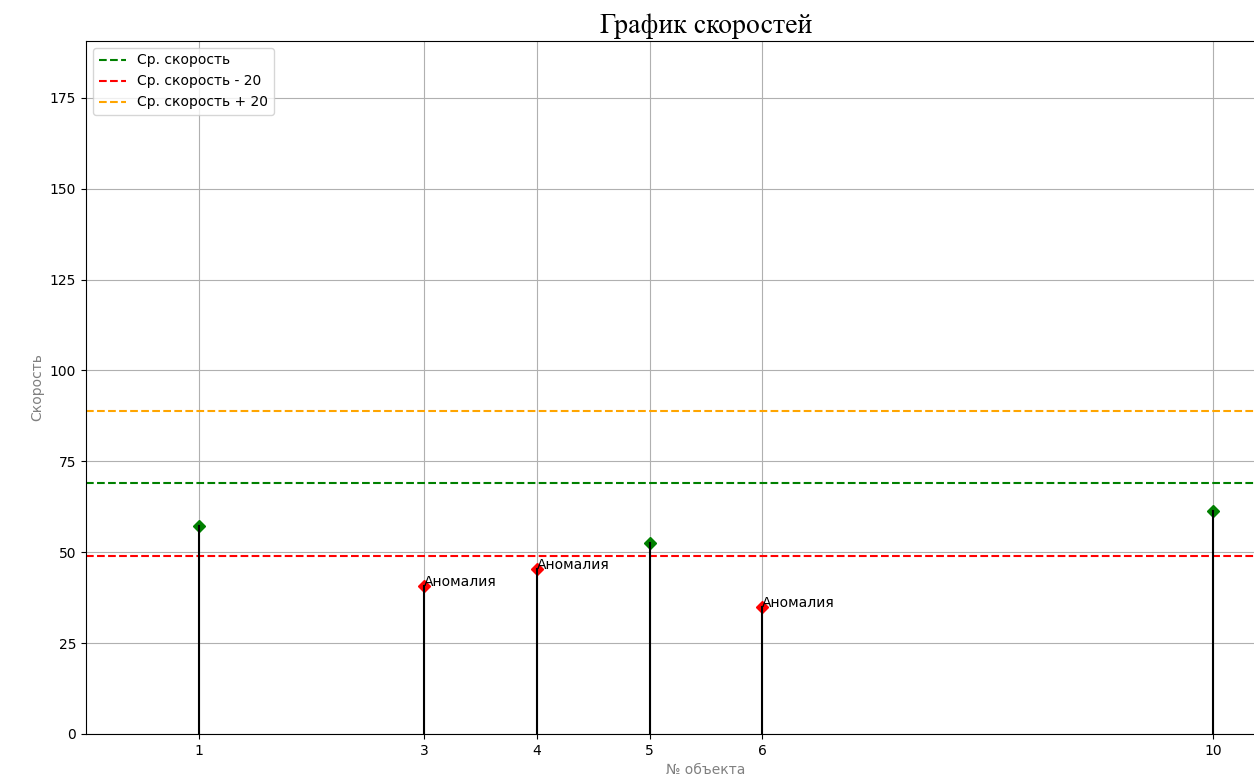


Рисунок 11 - Сравнение скоростей

ЗАКЛЮЧЕНИЕ

Анализ видеопоследовательности, проведенный с использованием модели YOLOv8, продемонстрировал высокую эффективность в задаче автоматического детектирования автомобилей и восстановлении их траекторий. Видеопоток, полученный со статической камеры, был успешно обработан. Модель YOLOv8 обеспечила точное распознавание автомобилей на каждом кадре, что позволило создать полную картину движения на анализируемом участке дороги.

В ходе анализа были вычислены ключевые параметры движения для каждого автомобиля, включая их координаты и скорости. Полученные временные ряды скоростей предоставили возможность глубже оценить динамику движения, что является важным для понимания поведения транспортного потока. Это также дало возможность выявить аномалии, такие как ситуации, когда скорость отдельных автомобилей значительно ниже средней скорости потока. Эти аномалии были автоматически обнаружены и визуализированы на видеокадрах и графиках, что позволяет быстро и наглядно оценить состояние дорожной обстановки.

Использование YOLOv8 в сочетании с алгоритмами трекинга и анализа временных рядов показало свою эффективность в реальном времени. Это позволяет не только выявлять, но и анализировать проблемные участки движения, что является важным для оперативного реагирования. Выявленные аномалии могут указывать на потенциальные проблемы на дороге, такие как пробки, остановки из-за неисправностей или другие факторы, влияющие на безопасность и эффективность движения. Это делает предложенный метод крайне полезным для мониторинга дорожной обстановки и принятия обоснованных оперативных решений.

Эффективность интеграции различных технологий обеспечила высокую точность и производительность анализа, что открывает новые перспективы для применения в системах управления дорожным движением. В будущем этот подход может быть улучшен и адаптирован для различных условий, включая городские и загородные дороги, и может стать основой для разработки интеллектуальных транспортных систем, способных значительно повысить безопасность и эффективность дорожного движения.

# **СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ**

1. Hyndman, R.J. Forecasting: principles and practice [Electronic resource] / R.J. Hyndman, G. Athanasopoulos. – 3rd ed. – Melbourne, Australia : OTexts, 2021. –URL: <https://otexts.com/fpp3/>.

2. Duong, H.-T. Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey / H.-T. Duong, V.-T. Le, V. T. Hoang // Sensors. – 2023. – Vol. 23, Iss. 11. – Article No. 5024. – 25 p. – DOI: [10.3390/s23115024](https://doi.org/10.3390/s23115024).

3. FlowNet 2.0: Evolution of Optical Flow Estimation With Deep Networks / E. Ilg [et al.]. // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – 2017. – P. 2462–2470. – URL: <https://openaccess.thecvf.com/content_cvpr_2017/html/Ilg_FlowNet_2.0_Evolution_CVPR_2017_paper.html>.

4. Интегральный оптический поток и его применение для мониторинга динамических объектов по видеопоследовательности / Ch. Chen [и др.]. // Журнал прикладной спектроскопии. – 2017. – № 84(1). – С. 138–146. – URL: <https://zhps.ejournal.by/jour/article/view/23>.

5. Как свёрточные нейросети имитируют работу мозга. – URL: https://practicum.yandex.ru/blog/svertochnye-neyronnye-seti/

# **ПРИЛОЖЕНИЕ А**

**ЛИСТИНГ ПРОГРАММЫ**

import os

import cv2

import time

import math

import shutil

import matplotlib.pyplot as plt

from ultralytics import YOLO, solutions

# Запуск таймера (узнаем сколько по времени работает наша программа)

start\_time = time.time()

# Путь к main папке

main\_patch = "C:/Users/nikit/Desktop/Diplom"

# Словарь папок, которые будем создавать

directories = {

    "graphic":              os.path.join(main\_patch, "graphic"),

    "final\_video":          os.path.join(main\_patch, "final\_video"),

    "time\_series":          os.path.join(main\_patch, "time\_series"),

    "screenshots":          os.path.join(main\_patch, "screenshots"),

    "screenshots\_anomaly":  os.path.join(main\_patch, "screenshots\_anomaly")

}

*def* delete\_directory(*dir*):

    """Удаляет директории"""

    if os.path.exists(dir) and os.path.isdir(dir):

        shutil.rmtree(dir)

        print(*f*"Папка '{dir}' удалена")

    else:

        print(*f*"Папка '{dir}' не найдена")

# Удаляем диркетории

for dir in directories.values():

    delete\_directory(dir)

print()

*def* create\_directory(*dir*):

    """Создает директорию"""

    if not os.path.exists(dir):

        os.makedirs(dir)

        print(*f*"Папка '{dir}' создана")

    else:

        print(*f*"Папка '{dir}' не найдена")

# Создаём директории

for dir in directories.values():

    create\_directory(dir)

print()

# Коэффициенты для вычисления расстояния

coef\_1 = 10 # people

# coef\_1 = 1 # car\_1

# coef\_1 = 1 # car\_2

# Коэффициенты для обнаружения аномалий

coef\_2 = 2 # people

# coef\_2 = 10 # car\_1

# coef\_2 = 10 # car\_2

# Открытие видеофайла

data\_set = os.path.join(main\_patch, "data\_set/people.mp4")

# data\_set = os.path.join(main\_patch, "data\_set/car\_1.mp4")

# data\_set = os.path.join(main\_patch, "data\_set/car\_2.mp4")

cap = cv2.VideoCapture(data\_set)

# Проверка на сущесвование файла

assert cap.isOpened(), "Ошибка чтения видеофайла"

# Получение параметров видео

w, h, fps = (*int*(cap.get(x)) for x in (

    cv2.CAP\_PROP\_FRAME\_WIDTH,

    cv2.CAP\_PROP\_FRAME\_HEIGHT,

    cv2.CAP\_PROP\_FPS

))

# Загрузка модели

model = YOLO("yolov8n.pt")

names = model.model.names

# Инициализация видеописателя

video\_writer = cv2.VideoWriter(

    os.path.join(directories["final\_video"], "final\_video.avi"),

    cv2.VideoWriter\_fourcc(\*"mp4v"),

    fps,

    (w, h)

)

# Определение горизонтальной линии (розовая линия)

line\_pts = [(0, *int*(h/2)), (*int*(w), *int*(h/2))]

# Инициализация объекта оценки скорости

speed\_obj = solutions.SpeedEstimator(*reg\_pts*=line\_pts, *names*=names, *view\_img*=True)

# Переменные для сохранения скриншотов

count = 0

flag = False

# Сохраняю каждую координату объекта и его время в каждый момент

all\_distance = {}

all\_time = {}

# Прогресс программы

progress = 1

# Ядро скрипта обрабатывает видео кадр за кадром

while cap.isOpened():

    success, im0 = cap.read()

    if not success:

        break

    # Отслеживание объектов и оценка скорости

    tracks = model.track(im0, *persist*=True, *show*=False, *verbose*=False)

    im0 = speed\_obj.estimate\_speed(im0, tracks, all\_time, all\_distance, coef\_1)

    video\_writer.write(im0)

    # Создание скриншотов автомобилей при пересечении линии

    if len(speed\_obj.trkd\_ids) != count and flag:

        for i in range(count, len(speed\_obj.trkd\_ids)):

            cv2.imwrite(os.path.join(directories["screenshots"], *f*"screenshot\_{speed\_obj.trkd\_ids[i]}.png"), im0)

            print(*f*"'screenshot\_{speed\_obj.trkd\_ids[i]}.png' сохранён")

            count += 1

        flag = False

    if len(speed\_obj.trkd\_ids) != count:

        flag = True

    # Сбор данных о расстоянии

    for i in speed\_obj.trk\_history:

        if i not in all\_distance:

            all\_distance[i] = []

        length = len(speed\_obj.trk\_history[i]) - 1

        all\_distance[i].append(speed\_obj.trk\_history[i][length])

    # Сбор данных о времени

    for i in speed\_obj.trk\_pt:

        if i not in all\_time:

            all\_time[i] = []

        all\_time[i].append(speed\_obj.trk\_pt[i])

    print(*f*'Progress: ({progress}/{*int*(cap.get(cv2.CAP\_PROP\_FRAME\_COUNT))})')

    progress += 1

# Освобождение ресурсов

cap.release()

video\_writer.release()

cv2.destroyAllWindows()

print(*f*"\nВидео обработано и сохранено '{directories['final\_video']}'")

# Убирает повторы в словаре all\_distance

for i in all\_distance:

    seen = *set*()

    all\_distance[i] = [j for j in all\_distance[i] if not (j in seen or seen.add(j))]

# Убирает повторы в словаре all\_time

for i in all\_time:

    seen = *set*()

    all\_time[i] = [j for j in all\_time[i] if not (j in seen or seen.add(j))]

*def* distance\_function(*all\_distance*, *i*, *j*, *d*):

    """Находит расстояние от одной точки до другой"""

    x1 = all\_distance[i][j][0]

    y1 = all\_distance[i][j][1]

    x2 = all\_distance[i][j + 1][0]

    y2 = all\_distance[i][j + 1][1]

    d = math.sqrt((x2 - x1)\*\*2 + (y2 - y1)\*\*2)

    return d

# Запись временного ряда в файл

with open(os.path.join(directories["time\_series"], "time\_series.txt"), 'w') as file:

    file.write(*f*'{"id"*:<10*} {"coordinates(x;y)"*:<50*} {"time(s)"*:<25*} {"distance(m)"*:<25*} {"speed(km/h)"*:<25*}\n')

    for i in speed\_obj.trkd\_ids:

        d = 0

        for j in range(0, len(all\_distance[i])):

            # Расстояние

            if j != 0:

                d += distance\_function(all\_distance, i, j - 1, d)

            # Время

            t = all\_time[i][j] - all\_time[i][0]

            # Скорость

            if t != 0:

                u = ((d / coef\_1) / t) \* 3.6

            else:

                u = 0

            # Вывод

            x, y = all\_distance[i][j]

            file.write(*f*'{i*:<10*} {*f*"({x}, {y})"*:<50*} {t*:<25*} {(d / coef\_1)*:<25*} {(u)*:<25*}\n')

        file.write('\n')

print(*f*"Временной ряд сформирован и сохранён '{directories['time\_series']}'")

# Переменные для рисования графика

speed\_list = *list*(speed\_obj.spd.values())

average\_speed = sum(speed\_list) / len(speed\_list)

max\_x = max(speed\_obj.trkd\_ids)

max\_y = max(speed\_list)

# Рисуем сам график

plt.figure(*figsize*=(16, 9))  # Размер графика на весь экран

plt.title('График скоростей', *fontsize*=20, *fontname*='Times New Roman') # Название графика

plt.axis([0, max\_x, 0, max\_y]) # Размерность по осям

plt.xticks(speed\_obj.trkd\_ids) # На оси Х только точки из trkd\_ids

# plt.yticks(speed\_list) # На оси Y только точки из trkd\_ids

plt.xlabel('№ объекта', *color*='gray') # Название оси Х

plt.ylabel('Скорость', *color*='gray') # Название оси Y

plt.grid(True) # Сетка

# Горизонтальные линии

plt.plot([0, max\_x], [average\_speed, average\_speed], *color*='green', *linestyle*='--')

plt.plot([0, max\_x], [average\_speed - coef\_2, average\_speed - coef\_2], *color*='red', *linestyle*='--')

plt.plot([0, max\_x], [average\_speed + coef\_2, average\_speed + coef\_2], *color*='orange', *linestyle*='--')

# Точки на графике

for i in speed\_obj.trkd\_ids:

    if speed\_obj.spd[i] <= average\_speed - coef\_2 or speed\_obj.spd[i] >= average\_speed + coef\_2:

        plt.plot([i], [speed\_obj.spd[i]], *color*='red', *marker*='D')

        plt.plot([i, i], [0, speed\_obj.spd[i]], *color*='black')

        plt.text(i, speed\_obj.spd[i], 'Аномалия')

        # Сохраняем скриншот аномалии

        shutil.copy(os.path.join(directories["screenshots"], *f*"screenshot\_{i}.png"), directories["screenshots\_anomaly"])

    else:

        plt.plot([i], [speed\_obj.spd[i]], *color*='green', *marker*='D')

        plt.plot([i, i], [0, speed\_obj.spd[i]], *color*='black')

plt.legend(['Ср. скорость', *f*'Ср. скорость - {coef\_2}', *f*'Ср. скорость + {coef\_2}'], *loc*=2) # Легенда

plt.savefig(os.path.join(directories["graphic"], "graphic.png")) # Сохранение графика

plt.show() # Показать график

print(*f*"График аномальных явлений построен и сохранён '{directories['graphic']}'")

# Время работы программы

end\_time = time.time()

execution\_time = end\_time - start\_time

print(*f*"Время выполнения программы {execution\_time*:.2f*} секунд")