МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«САРАТОВСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМЕНИ Н. Г. ЧЕРНЫШЕВСКОГО»

Кафедра дискретной математики и информационных технологий

АНАЛИЗ ДИНАМИЧЕСКИХ ИЗОБРАЖЕНИЙ ДЛЯ ВЫЯВЛЕНИЯ АНОМАЛИЙ В ПОВЕДЕНИИ НАБЛЮДАЕМОГО ОБЪЕКТА

АВТОРЕФЕРАТ БАКАЛАВРСКОЙ РАБОТЫ

студента 4 курса 421 группы
направления 09.03.01 — Информатика и вычислительная техника
факультета КНиИТ
Гаджимусаева Рабадана Магомедовича

Научный руководитель
доцент, к. ф.-м. н. _____ А. Д. Панферов

Заведующий кафедрой
к. ф.-м. н., доцент _____ Л. Б. Тяпаев

Саратов 2026

ВВЕДЕНИЕ

В современном мире системы видеонаблюдения стали неотъемлемой частью инфраструктуры безопасности городов, промышленных объектов, транспортных узлов и общественных пространств. Ежесекундно генерируются петабайты видеоданных, однако их эффективный анализ остаётся нетривиальной задачей.

Проблемы безопасности, мониторинга состояния объектов и оперативного реагирования на нештатные ситуации приобретают всё большую важность. Развитие технологий искусственного интеллекта, в частности компьютерного зрения и анализа видео, открывает новые возможности для создания интеллектуальных систем мониторинга.

Актуальность темы определяется тем, что существующие коммерческие решения видеоаналитики либо требуют дорогостоящей инфраструктуры и настройки специалистами, либо ограничены фиксированным набором детектируемых событий и не допускают гибкой настройки через естественный язык. Задача обнаружения аномалий в поведении наблюдаемого объекта сложна: аномалии по определению редки и разнообразны — невозможно заранее предсказать и промаркировать все возможные варианты нештатного поведения. Развитие технологий глубокого обучения (YOLO, ByteTrack) и больших языковых моделей открывает возможность создания доступных систем, настраиваемых через естественно-языковые запросы без специальных технических знаний. При этом акцент делается не на создании новых алгоритмов, а на интеграции существующих готовых ИИ-решений в клиент-серверную архитектуру с мобильным интерфейсом оператора.

Целью данной квалификационной работы является разработка мобильного приложения, обеспечивающего дистанционный контроль за поведением наблюдаемого объекта и оповещение об аномалиях его поведения.

Для достижения поставленной цели необходимо решить следующие задачи:

- изучить современные возможности ИИ по анализу динамических изображений и аномалий в поведении наблюдаемых объектов;
- рассмотреть инструменты, обеспечивающие использование возможностей ИИ-агентов различного вида при решении рассматриваемых задач;
- изучить инструменты разработки мобильных приложений и серверных

решений;

- определить технологический стек для разработки нативного или кроссплатформенного мобильного приложения;
- разработать прототип системы и протестировать его на модельных примерах.

Предметом исследования являются готовые решения и сервисы на основе искусственного интеллекта для выявления аномалий в поведении наблюдаемого объекта, а также архитектурные и программные решения, обеспечивающие интеграцию серверной аналитики с мобильным клиентским приложением.

Объектом исследования выступает процесс обработки динамических изображений, получаемых с камер видеонаблюдения, включающий их последующий анализ с помощью сервисов на основе искусственного интеллекта для детекции объектов и выявления отклонений от нормативного поведения.

Работа состоит из введения, трёх разделов, заключения, списка использованных источников из 26 наименований и четырёх приложений. Работа содержит 3 таблицы и 9 рисунков.

КРАТКОЕ СОДЕРЖАНИЕ РАБОТЫ

В первом разделе рассматриваются теоретические основы и современное состояние технологий анализа динамических изображений.

В подразделе 1.1 уточняется понятие динамического изображения и аномалий поведения. Динамическое изображение представляется как функция $I(x, y, t)$, где x, y — пространственные координаты пикселя, t — момент времени. Разностное изображение $D(t) = |F(t) - F(t-1)|$ позволяет выделить области движения между кадрами; на его основе строятся траектории объектов и вычисляются кинематические характеристики. Рассмотрены три типа аномалий поведения: *точечные* (одно наблюдение резко выбивается из общего распределения, например внезапное резкое ускорение объекта), *контекстуальные* (отклонение, зависящее от контекста окружения, например бег в коридоре университета) и *коллективные* (аномальный паттерн из действий, нормальных по отдельности, например одновременное перемещение группы людей). Также выделены аномалии движения, аномалии внешнего вида и аномалии взаимодействия. Описаны три ключевые проблемы практического обнаружения: редкость и непредсказуемость аномалий, а также наличие шума в данных видеонаблюдения.

В подразделе 1.2 проведён обзор методов анализа видеопотоков. Среди традиционных методов рассмотрены: *оптический поток* — поле векторов смещения пикселей между двумя последовательными кадрами, обладающее высокой чувствительностью к движению, но подверженное шумам и сложное в вычислении; *методы на основе траекторий* — отслеживают перемещение объектов во времени с помощью фильтров Калмана, кластеризуют типичные маршруты и выявляют аномальное движение по отклонению от них.

Среди методов глубокого обучения рассмотрены следующие архитектуры. *Свёрточные нейронные сети* (CNN) применяются как экстракторы признаков и детекторы объектов. Специализированные архитектуры YOLO, SSD, Faster R-CNN выдают прямоугольные ограничивающие рамки вокруг объектов с указанием класса. Главный недостаток стандартных CNN для видео — обработка каждого кадра по отдельности без учёта временных связей. *Рекуррентные сети* (RNN/LSTM) разработаны специально для последовательностей. LSTM решает проблему затухания градиента через механизм трёх вентилей: входного (что запомнить),

забывания (что удалить из памяти) и выходного (что передать дальше). В задачах обнаружения аномалий LSTM получают признаки от CNN и предсказывают следующий набор или классифицируют поведение. *Автокодировщики* обучаются только на нормальных данных — энкодер сжимает кадр до латентного вектора, декодер восстанавливает исходное изображение. Аномалии обнаруживаются по высокой ошибке восстановления: нормальный кадр восстанавливается с малой ошибкой, аномальный — с большой. *GAN* используют генератор и дискриминатор: аномальный кадр генератор не может воспроизвести точно, что фиксируется по ошибке. GAN дают более чёткую реконструкцию, чем автокодировщики, но сложнее в обучении. *Vision Transformers* (ViT) делят видеокадр на патчи и через механизм самовнимания улавливают глобальные пространственно-временные зависимости. Показывают высокую точность при значительной вычислительной сложности — self-attention требует квадратичного числа операций от числа патчей. Для обучения и сравнения алгоритмов используются стандартные наборы данных: UCSD Anomaly Detection Dataset, CUHK Avenue, ShanghaiTech Campus.

В подразделе 1.3 описываются возможности выявления аномалий на современном этапе развития ИИ. Главное отличие нейросетевых методов — способность автоматически обучаться на данных без ручного задания правил. Автокодировщики и GAN работают без размеченных аномалий. Современные модели учитывают не только движение, но и контекст: место, время, окружение. Активно развиваются мультимодальные подходы (VLM): такие системы способны не просто обнаружить аномалию, но и описать её словами, что снижает количество ложных срабатываний и упрощает работу оператора. Рассмотрены и ограничения ИИ-методов: потребность в больших объёмах данных, вычислительная сложность, проблема интерпретируемости и вопросы приватности.

Во втором разделе проведён анализ инструментальных средств разработки системы.

В подразделе 2.1 рассматриваются инструменты и библиотеки для работы с ИИ. Сопоставлены фреймворки PyTorch и TensorFlow. PyTorch выбран как основа библиотеки Ultralytics YOLO благодаря динамическому вычислительному графу, облегчающему отладку. Библиотека OpenCV

предоставляет функции захвата и покадрового декодирования видеофайлов. Семейство детекторов YOLO (You Only Look Once) обеспечивает приемлемый компромисс между качеством и скоростью. Алгоритм Deep SORT объединяет фильтр Калмана с визуальными признаками CNN для трекинга с учётом внешности объекта. ByteTrack демонстрирует высокую точность при низком пороге детекции и проще настраивается; он учитывает все обнаружения, включая низкоуверенные, что обеспечивает устойчивое восстановление треков после окклюзий. Рассмотрен фреймворк Ollama для локального запуска мультимодальных моделей (LLaVA, LLaMA 3.2 Vision, MiniCPM-V, Gemma 3), позволяющий выполнять семантический анализ сцены без передачи данных на внешние серверы.

В подразделе 2.2 анализируются инструменты для создания серверной части. Сопоставлены языки и фреймворки: Python/FastAPI, Python/Django, Python/Flask, Go/Gin, Java/Spring Boot, Node.js/Express. Обоснован выбор FastAPI: асинхронная архитектура (`async/await`) критична для одновременного обслуживания нескольких SSE-соединений, нативная поддержка Server-Sent Events обеспечивает потоковую передачу событий в реальном времени, строгая типизация через Pydantic снижает вероятность ошибок при разборе входных данных. Рассмотрены протоколы обмена данными: REST API, WebSocket и gRPC. Для систем видеоаналитики актуальны инструменты контейнеризации (Docker, Kubernetes) и мониторинга (Prometheus, Grafana).

В подразделе 2.3 рассматриваются инструменты разработки мобильных приложений. Сопоставлены нативные подходы — Android/Kotlin и iOS/Swift — и кроссплатформенные фреймворки: Flutter, React Native, .NET MAUI. Обоснован выбор Flutter: отрисовка интерфейса собственным движком Skia исключает зависимость от нативных UI-компонентов ОС и обеспечивает стабильное поведение на всех платформах с производительностью 60–120 FPS. Описаны инструменты для сетевого взаимодействия (Dio, Retrofit, URLSession), подходы к локальному хранению данных (Room, CoreData, Hive, SharedPreferences) и архитектурный паттерн MVVM, при котором ViewModel хранит состояние и бизнес-логику, а View лишь отображает данные.

В третьем разделе описаны проектирование, реализация и тестирование системы анализа видеоизображения.

В подразделе 3.1 описана архитектура системы. Система построена по

клиент-серверной архитектуре и включает два компонента: серверную часть и мобильное приложение. Серверный компонент (Python 3.10+, FastAPI, Uvicorn) выполняет ресурсоёмкие задачи: покадровую обработку видео нейронной сетью YOLO11, детекцию и отслеживание объектов трекером ByteTrack, определение фактов пересечения зоны контроля или обнаружение движущихся объектов при её отсутствии, кэширование результатов и взаимодействие с языковыми моделями через сервис DeepInfra. Мобильное приложение (Flutter) является тонким клиентом: настройка через естественно-языковые запросы, разметка зоны контроля, воспроизведение видео с привязанными событиями, генерация push-уведомлений. Для передачи событий в реальном времени применяется технология Server-Sent Events (SSE) — однонаправленный поток от сервера к клиенту, не требующий постоянного опроса. Состояние (конфигурация, зона контроля) сохраняется в JSON-файлах и автоматически восстанавливается после перезапуска сервера.

В подразделе 3.2 описана реализация серверной части. Сервер организован в виде четырёх Python-модулей: `server.py` (FastAPI-приложение с API-маршрутами), `main.py` (покадровый цикл обработки видео), `YOLO.py` (класс `AnomalyDetector`) и `analyzer.py` (взаимодействие с языковыми моделями). Сервер предоставляет эндпоинты для управления конфигурацией, видеоматериалами, анализом и кэшем событий. Ресурсоёмкая обработка видео выполняется в дочерних процессах ОС через `subprocess.Popen`, что обходит ограничение GIL Python. Для синхронизации используются примитивы `threading.Lock` и `threading.Event`. Реализован механизм предварительного кэширования событий: процедура `/api/cache/build` обрабатывает все видеофайлы с текущими параметрами и сохраняет результаты в `analysis_cache.json`. Кэш автоматически сбрасывается при изменении конфигурации. Реализован семантический поиск по кэшу — языковой модели передаётся компактное представление событий всех видеофайлов, и она возвращает имена записей, соответствующих запросу пользователя.

В подразделе 3.3 описан модуль обнаружения и отслеживания объектов. Класс `AnomalyDetector` (`YOLO.py`) инкапсулирует работу с библиотекой Ultralytics YOLO. Встроенный трекинг использует алгоритм ByteTrack, основанный на венгерском алгоритме назначения с предсказанием положения фильтром Калмана. Это обеспечивает сохранение идентификаторов `track_id`

при кратковременных окклюзиях — один и тот же объект учитывается как единица независимо от числа кадров, в которых его видит детектор. Система поддерживает два режима задания зоны контроля: двухточечный (`gate-line`) и четырёхточечный (`gate-quad`). Направление пересечения (вход/выход) определяется знаком z-компоненты векторного произведения:

$$s = v_x \cdot u_y - v_y \cdot u_x,$$

где $\vec{v} = P_2 - P_1$ — направляющий вектор зоны, $\vec{u} = C - P_1$ — вектор до центра объекта. Смена знака s однозначно кодирует направление независимо от ориентации линии в кадре. Координаты точек зоны, переданных мобильным клиентом, нормализованы в диапазоне $[0, 1]$ относительно размеров кадра; сервер пересчитывает их в пиксельные по метаданным видеофайла. Проведено сравнительное тестирование трёх конфигураций модели (YOLO11n, YOLO11s, YOLO11m) на наборе из 40 видеофайлов суммарной длительностью около 40 минут. Параметры тестирования: размер входного изображения 640 пикселей, порог уверенности 0,25, устройство — GPU NVIDIA GeForce GTX 1650. Результаты представлены в таблице 1.

Таблица 1 – Сравнение конфигураций YOLO11

Показатель	**Эталон**	**YOLO11n**	**YOLO11s**	**YOLO11m**
Обнаружено событий	12	10	13	11
Полнота (Recall), %	100	83,3	**100,0**	91,7
Точность (Precision), %	—	100,0	92,3	100,0
F1-мера	—	0,909	**0,960**	0,957
Скорость (FPS, GPU)	—	≈ 45	≈ 38	≈ 22

По совокупности точности и вычислительной эффективности выбрана модель YOLO11s: полнота 100%, F1-мера 0,960, скорость 38 FPS превышает стандартную частоту кадров видео (25–30 FPS). YOLO11n пропустила два события из-за низкой уверенности при неблагоприятных условиях съёмки. YOLO11m показала схожую точность, но вдвое меньшую производительность.

В подразделе 3.4 описана интеграция языковых моделей через сервис DeepInfra. В модуле `analyzer.py` реализованы три сценария взаимодействия с LLM: интерпретация текстового запроса пользователя, анализ событий после обработки видеофайла и семантический поиск по кэшу.

Функция `interpret_gate_config_prompt_deepinfra` принимает запрос на естественном языке и возвращает структурированный JSON с тремя полями: `direction` — направление наблюдения (in/out/both), `targets` — список классов отслеживаемых объектов и `needs_zone` — признак необходимости зоны контроля. При `needs_zone = false` этап разметки зоны пропускается автоматически, и система переходит в режим общего обнаружения движущихся объектов. Реализован вспомогательный парсер, извлекающий первый валидный JSON из ответа модели, включая JSON внутри markdown-блоков. Проведено сравнение четырёх языковых моделей на наборе из 30 тестовых запросов. Результаты представлены в таблице 2.

Таблица 2 – Сравнение языковых моделей для задач системы

Модель	**Точность, %**	**Латентность, с**	**JSON-формат**
Gemini 2.0 Flash	**96,7**	**1,8**	Отличный
DeepSeek-V3	93,3	2,4	Хороший
Llama 3.3 70B	70,0	3,1	Нестабильный
Mistral 7B	45,0	1,2	Слабый

Модель Gemini 2.0 Flash показала наилучший результат: 96,7% корректно интерпретированных запросов при среднем времени ответа 1,8 с и стабильном JSON-выводе. Она выбрана основной с идентификатором `google/gemini-2.0-flash-001`.

В подразделе 3.5 описана реализация мобильного приложения на Flutter. Вся логика сосредоточена в `StatefulWidget` — `MyHomePage`. Жизненный цикл управляется перечислением `_AppStage` с пятью состояниями: `prompt` (ввод адреса сервера и запроса), `selectLine` (разметка зоны четырьмя касаниями; пропускается при `needs_zone` = false), `analyzing` (ожидание построения кэша), `analysisDone` (результаты готовы) и `playing` (воспроизведение видео с событиями). При разметке зоны координаты касаний нормализуются в диапазоне $[0, 1]$ и передаются серверу; сервер автоматически исправляет самопересекающиеся четырёхугольники. При воспроизведении видео таймер каждые 500 мс сравнивает текущую позицию плеера с временны́ми метками событий кэша и генерирует push-уведомление через библиотеку `flutter_local_notifications`. Для исключения дублирующих уведомлений используется множество уже отправленных

ключей `"videoName_trackId_eventType"`. Параллельно поддерживается SSE-подписка для получения событий в реальном времени. Приложение реализовано по паттерну MVVM.

В подразделе 3.6 представлены результаты интеграционного тестирования системы. Тестирование проводилось по пяти сценариям на наборе из 40 видеофайлов. Результаты представлены в таблице 3.

Таблица 3 – Результаты интеграционного тестирования

Сценарий	**Эталон**	**Обнаружено**	**Точность, %**
Въезд/выезд автомобилей	8	8	100,0
Проход пешеходов	12	11	91,7
Смешанный трафик	10	9	90,0
Ночная съёмка	5	5	100,0
Высокая плотность	4	3	75,0
Итого	**39**	**36**	**92,3**

Общая точность составила 92,3%. Снижение до 75% в сценарии высокой плотности объясняется частичными перекрытиями и потерей идентификаторов трекером ByteTrack. Задержка push-уведомления от момента события до получения на устройстве — менее 500 мс во всех сценариях. Время построения кэша для 40 видеофайлов — около 34 минут на тестовом оборудовании.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы были решены все поставленные задачи и получены следующие результаты.

Проведён обзор теоретических основ и современного состояния технологий детекции аномалий на видеопотоке. Рассмотрены классические методы (оптический поток, метод траекторий) и современные подходы на основе глубокого обучения (CNN, LSTM, автокодировщики, GAN, Vision Transformer). Показано, что мультимодальные языковые модели открывают новые возможности для интерпретации результатов анализа и взаимодействия с пользователем.

Проанализированы инструментальные средства разработки систем видеоаналитики. Обоснован выбор технологического стека: Python/FastAPI для серверной части, Flutter для мобильного клиента, YOLO11s для детекции объектов, ByteTrack для многообъектного трекинга, DeepInfra API для языковых моделей.

Разработана и реализована клиент-серверная архитектура системы. Серверная часть на Python/FastAPI выполняет детекцию и отслеживание объектов, определение факта и направления пересечения зоны контроля, кэширование результатов и взаимодействие с языковыми моделями через DeepInfra. Мобильное приложение на Flutter обеспечивает настройку через естественно-языковые запросы, разметку зоны контроля, воспроизведение видео с событиями и генерацию push-уведомлений.

Реализован механизм опциональной зоны контроля: языковая модель по тексту запроса определяет необходимость зоны (`needs_zone`), что позволяет использовать систему как для отслеживания пересечений, так и для общего обнаружения движущихся объектов.

Проведено двухэтапное тестирование системы. Сравнительное тестирование трёх конфигураций YOLO11: наилучший результат у YOLO11s — F1-мера 0,960, скорость 38 FPS. Сравнение четырёх языковых моделей: выбрана Gemini 2.0 Flash с точностью интерпретации 96,7% и средним временем ответа 1,8 с. Интеграционное тестирование по пяти сценариям: общая точность 92,3%, задержка push-уведомлений — менее 500 мс.

Практическая значимость состоит в том, что разработанная система автоматизирует видеонаблюдение и делает его доступным для пользователей

без специальных знаний в области компьютерного зрения.

В качестве направлений дальнейшего развития можно выделить: интеграцию мультимодальных языковых моделей для прямого анализа видеокадров без промежуточного YOLO-детектора, расширение типов обнаруживаемых аномалий, добавление поддержки IP-камер в реальном времени и облачного мониторинга.

Основные источники информации:

1 Каспаров И. А. Исследование алгоритмов сегментации видеоданных на основе пирамидальной сети анализа сцены. — Самара: Самарский университет, 2025.

2 Бутакова М. А., Щербань И. В., Мишин Н. А., Белявский Г. И. Два метода оценки оптического потока по видеоряду изображений // Искусственный интеллект и принятие решений. — 2025. — № 1. — С. 115–127.

3 Рока С., Дивакар М. A survey on abnormal behavior detection based intelligence information video surveillance system using optimized machine learning methods // Engineering Applications of Artificial Intelligence. — 2026. — Vol. 166. — Article 113438.

4 Duong H. T. et al. Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey // Sensors. — 2023. — Vol. 23, No. 12.

5 Shukla V. et al. A systematic survey: role of deep learning-based image anomaly detection in industrial inspection contexts // Frontiers in Robotics and AI. — 2025. — Vol. 12.

6 Гриценко Г. Г., Фраленко В. П. Аналитический обзор архитектур, моделей, методов и алгоритмов для локализации и трекинга неригидных объектов // Программные системы: теория и приложения. — 2024. — Т. 15, № 4. — С. 111–151.

7 Wang W. Comparative Survey of Deep Learning Architectures for Video Anomaly Detection: CNNs, Autoencoders, VAEs, RNNs, GANs, and Hybrids // Informatica. — 2025. — Vol. 49, No. 29.

8 Dilek E., Dener M. An overview of transformers for video anomaly detection // Neural Computing and Applications. — 2025. — Vol. 37. — P. 17825–17857.

9 Gao S., Yang P., Guo H. et al. The Evolution of Video Anomaly Detection: A Unified Framework from DNN to MLLM // arXiv. — 2025.