МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Кафедра компьютерных технологий и систем

КАПИТАНОВ
Иван Сергеевич

ОБНАРУЖЕНИЕ И КЛАССИФИКАЦИЯ ВОЗДУШНЫХ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ ТЕХНОЛОГИЙ ИСКУССТВЕННОГО ИНТЕЛЛЕКТА

Дипломная работа

Научный руководитель:
Доцент кафедры технологий программирования
Войтешенко Иосиф Станиславович

Допущена к защите
«____» _____ 2025 г.

Зав. кафедрой компьютерных технологий и систем
кандидат физ.-мат. наук, доктор педагогических наук
профессор В. В. Казаченок

Минск, 2025

РЕФЕРАТ

Дипломная работа, 68 стр., 21 источник, 4 рисунка.

Ключевые слова: обнаружение объектов, летательные аппараты, нейронные сети, YOLO, видеотрансляция, RTSP, стабилизация видео, компьютерное зрение.

Объекты исследования – методы и технологии для обнаружения и распознавания летательных аппаратов в видеопотоке с использованием нейронных сетей, включая системы видеотрансляции и алгоритмы стабилизации изображения.

Цель исследования – разработка и исследование прототипа системы обнаружения летательных аппаратов на основе нейросетевой модели YOLOv11, интегрированной с системой видеотрансляции по протоколу RTSP, а также анализ методов стабилизации видео для повышения качества детекции.

Методы исследования – системный анализ предметной области, сравнительный анализ существующих технологий и протоколов, проектирование архитектуры программного комплекса, разработка программных модулей, экспериментальное тестирование, анализ алгоритмов стабилизации видео.

В результате исследования – разработан прототип системы, включающий модуль видеотрансляции RTSP, серверный модуль анализа с использованием YOLOv11 и клиентский интерфейс. Проанализирована эффективность системы на тестовых данных, исследованы методы стабилизации видео и оценена их применимость.

Области применения – системы безопасности и мониторинга, автоматизированные системы наблюдения, робототехника, анализ воздушного пространства.

ABSTRACT

Thesis, 68 pages, 21 sources, 4 figures.

Keywords: object detection, aerial vehicles, neural networks, YOLO, video streaming, RTSP, video stabilization, computer vision.

Objects of research – methods and technologies for detecting and recognizing aerial vehicles in a video stream using neural networks, including video streaming systems and image stabilization algorithms.

Aim of the study – development and research of a prototype system for detecting aerial vehicles based on the YOLOv11 neural network model, integrated with an RTSP video streaming system, as well as analysis of video stabilization methods to improve detection quality.

Methods of research – system analysis of the subject area, comparative analysis of existing technologies and protocols, software complex architecture design, software module development, experimental testing, analysis of video stabilization algorithms.

Results of the study – a prototype system was developed, including an RTSP video streaming module, a server-side analysis module using YOLOv11, and a client interface. The system's effectiveness was analyzed on test data, video stabilization methods were investigated, and their applicability was assessed.

Fields of application – security and monitoring systems, automated surveillance systems, robotics, airspace analysis.

РЭФЕРАТ

Дыпломная работа, 68 стар., 21 крыніца, 4 малюнкі.

Ключавыя словы: выяўленне аб'ектаў, лятальныя апараты, нейронныя сеткі, YOLO, відэатрансляцыя, RTSP, стабілізацыя відэа, камп'ютарны зрок.

Аб'екты даследавання – метады і тэхналогіі для выяўлення і распазнавання лятальных апаратаў у відэаструмені з выкарыстаннем нейронных сетак, уключаючы сістэмы відэатрансляцыі і алгарытмы стабілізацыі выявы.

Мэта даследавання – распрацоўка і даследаванне прататыпа сістэмы выяўлення лятальных апаратаў на аснове нейрасеткавай мадэлі YOLOv11, інтэграванай з сістэмай відэатрансляцыі па пратаколе RTSP, а таксама аналіз метадаў стабілізацыі відэа для павышэння якасці дэтэкцыі.

Метады даследавання – сістэмны аналіз прадметнай вобласці, параўнальны аналіз існуючых тэхналогій і пратаколаў, праектаванне архітэктуры праграмнага комплексу, распрацоўка праграмных модуляў, эксперыментальнае тэсціраванне, аналіз алгарытмаў стабілізацыі відэа.

У выніку даследавання – распрацаваны прататып сістэмы, які ўключае модуль відэатрансляцыі RTSP, серверны модуль аналізу з выкарыстаннем YOLOv11 і кліенцкі інтэрфейс. Прааналізавана эфектыўнасць сістэмы на тэставых даных, даследаваны метады стабілізацыі відэа і ацэнена іх прымяняльнасць.

Галіны прымянення – сістэмы бяспекі і маніторынгу, аўтаматызаваныя сістэмы назірання, робататэхніка, аналіз паветранай прасторы.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Стремительное развитие беспилотных летательных аппаратов (БПЛА) и их широкое распространение в различных сферах, от логистики и сельского хозяйства до систем безопасности и развлечений, порождает новые вызовы, связанные с необходимостью их своевременного обнаружения и идентификации. Несанкционированное использование БПЛА может представлять угрозу для критически важной инфраструктуры, общественной безопасности и частной жизни. В связи с этим, разработка эффективных систем обнаружения и распознавания летательных аппаратов (ЛА), особенно малоразмерных и низколетящих, становится актуальной научно-технической задачей.

Традиционные методы обнаружения, такие как радиолокационные системы, могут быть неэффективны против малых БПЛА из-за их низкой радиолокационной заметности и малых скоростей. Оптические системы, использующие видеокамеры, в сочетании с современными методами компьютерного зрения и искусственного интеллекта (ИИ), представляют собой перспективное направление для решения этой задачи. В частности, глубокие сверточные нейронные сети (CNN) продемонстрировали выдающиеся результаты в задачах детекции и классификации объектов на изображениях и видео. Модели семейства YOLO (You Only Look Once) зарекомендовали себя как эффективные инструменты для обнаружения объектов в реальном времени, сочетая высокую точность и скорость работы.

Однако эффективность любой системы видеоанализа на базе ИИ напрямую зависит от качества и своевременности поступления видеоданных от камеры к модулю анализа. Это выдвигает на передний план вопросы, связанные с проектированием и реализацией надежной системы видеотрансляции, способной передавать видеопоток с минимальной задержкой и достаточным качеством через сетевую инфраструктуру. Кроме того, качество самого видеосигнала, особенно получаемого с подвижных платформ, может быть подвержено влиянию вибраций и дрожания, что негативно сказывается на точности работы нейросетевых детекторов. Поэтому исследование методов стабилизации видео также является важным аспектом повышения общей производительности системы.

ГЛАВА 1

НЕЙРОННЫЕ СЕТИ В ЗАДАЧЕ ДЕТЕКЦИИ ОБЪЕКТОВ

1.1 Роль нейронных сетей в задаче детекции

Детекция объектов — это одна из ключевых задач в области компьютерного зрения, которая заключается в определении местоположения объектов на изображении (с помощью ограничивающих рамок) и их классификации по заранее заданным категориям. С развитием технологий глубокого обучения нейронные сети стали основным инструментом для решения этой задачи, благодаря их способности автоматически выделять признаки из данных и обеспечивать высокую точность и производительность.

Основные преимущества нейронных сетей в детекции:.

1. Автоматическое извлечение признаков:

   Нейронные сети автоматически выделяют важные признаки объектов, такие как форма, текстура и контуры, что исключает необходимость ручного выбора признаков.

2. Высокая адаптивность:

   Нейронные сети могут быть настроены для работы с различными типами данных и задачами детекции, включая детекцию объектов на изображениях, в видео и в реальном времени.

3. Совмещение классификации и локализации:

   Современные архитектуры позволяют одновременно классифицировать объект и определять его координаты на изображении.

4. Возможность обучения на больших данных:

   Нейронные сети хорошо масштабируются и демонстрируют высокую производительность при наличии больших объемов данных.

1.2 Обзор существующих подходов к детекции объектов

Современные подходы к детекции объектов можно разделить на традиционные методы и методы, основанные на глубоких нейронных сетях.

1.2.1 Традиционные методы детекции объектов:

Традиционные подходы к детекции объектов в основном опираются на методы извлечения признаков и алгоритмы машинного обучения. Они были популярны до появления глубокого обучения, но в современных задачах уступают нейронным сетям.

Этапы традиционной детекции:

1. Извлечение признаков: Для определения объектов использовались такие методы, как:

   - HOG (Histogram of Oriented Gradients): Использовался для описания градиентов изображения и выделения формы объектов. Этот метод был особенно эффективен для детекции людей.
   - SIFT (Scale-Invariant Feature Transform) и SURF (Speeded Up Robust Features): Эти методы использовались для нахождения ключевых точек и их описания.

2. Обнаружение объектов: Для поиска объектов на изображении применялись методы, основанные на разбиении изображения на области, например:

   - Скользящее окно: Метод предполагает перебор всех возможных областей изображения фиксированного размера и их классификацию.
   - Метод опорных векторов (SVM): Применялся для классификации объектов в комбинации с описанными выше признаками..

Несмотря на свою простоту и эффективность на небольших наборах данных, традиционные методы имеют серьёзные ограничения. Они плохо масштабируются, требуют ручного подбора признаков и часто уступают глубоким нейронным сетям в точности.

1.2.2 Методы детекции, основанные на глубоких нейронных сетях:

С развитием глубокого обучения основным инструментом классификации объектов стали нейронные сети. Они вытеснили традиционные методы благодаря своей способности автоматически извлекать признаки. К основным подходам относятся:

1. Одноэтапные методы детекции (Single-Stage Detectors): Эти методы оптимизированы для скорости и часто применяются в реальных приложениях, таких как автономные автомобили и системы видеонаблюдения.

    - YOLO (You Only Look Once): Быстрая модель, которая обрабатывает изображение за один проход, возвращая координаты рамок и классы объектов.
    - SSD (Single Shot MultiBox Detector): Использует многомасштабные карты признаков для детекции объектов разных размеров.

2. Двухэтапные методы детекции (Two-Stage Detectors): Эти методы обычно обеспечивают более высокую точность за счет разделения задачи на этапы выделения областей и последующей их классификации.

    - R-CNN (Region-based Convolutional Neural Networks): Проводит генерацию регионов-кандидатов и затем классифицирует их.
    - Faster R-CNN: Ускоренная версия R-CNN, которая включает региональные предложения как часть модели.

3. Современные подходы:

    - Detr (Detection Transformer): Использует механизм внимания для обработки всего изображения и прямого определения объектов без разделения на этапы.
    - EfficientDet: Комбинирует высокую точность и эффективность, используя архитектуры на основе EfficientNet.

1.3 Сверточные нейронные сети как основной инструмент детекции

Сверточные нейронные сети (CNN) являются основой большинства современных подходов к задаче детекции объектов. Их архитектура, включающая свёрточные слои, пулинги и полностью соединённые слои, позволяет эффективно извлекать пространственные признаки из изображений, что делает их идеальными для анализа визуальных данных. В детекции объектов CNN используются как для выделения признаков, так и для классификации и локализации объектов.

Принципы работы CNN в задаче детекции В задаче детекции сверточные нейронные сети решают две ключевые подзадачи:

- Локализация объектов: Определение координат ограничивающей рамки для каждого объекта.
- Классификация объектов: Присвоение класса объекту внутри рамки.

Эти задачи решаются совместно благодаря особенностям архитектур CNN, которые позволяют одновременно прогнозировать как классы объектов, так и их координаты. Для этого выходные слои сети модифицируются, чтобы предсказывать не только вероятности классов, но и параметры ограничивающих рамок.

Модели YOLO (You Only Look Once), SSD (Single Shot MultiBox Detector), Faster R-CNN, Mask R-CNN, которые упоминались ранее, являются сверточными нейронными сетями. Также существуют Ретина-сети (RetinaNet). RetinaNet использует Focal Loss для обработки несбалансированных данных, где "фоновые"области значительно превосходят количество объектов. Это улучшает качество детекции мелких и слабовыраженных объектов.

Преимущества использования CNN для детекции:

- Автоматическое извлечение признаков: CNN автоматически учатся выделять ключевые признаки, такие как границы, формы и текстуры объектов.
- Обработка сложных сцен: Сверточные сети способны обнаруживать объекты в условиях значительного наложения, сложного фона или слабой освещённости.

- Гибкость архитектур: CNN могут быть адаптированы для различных задач детекции, включая реальное время, многообъектные сцены и мелкие объекты.

- Поддержка мультимасштабного анализа: Использование пирамид признаков (Feature Pyramid Networks, FPN) позволяет детектировать объекты разных размеров в одном изображении.

Несмотря на высокую производительность, сверточные нейронные сети имеют ограничения:

- Выделение ресурсов: CNN требуют значительных вычислительных мощностей, особенно для обработки видео или изображений высокого разрешения.

- Сложности с мелкими объектами: Детекция мелких объектов или объектов на большом расстоянии может быть затруднена из-за потери деталей на ранних уровнях свёртки.

- Сложность аннотации данных: Для обучения требуется большое количество аннотированных данных, что может быть трудоёмким процессом.

1.4 RetinaNet и использование Focal Loss для работы с несбалансированными данными

RetinaNet — это одноэтапная архитектура для детекции объектов, которая была разработана для решения одной из ключевых проблем в задачах детекции: несбалансированности данных. Эта проблема особенно актуальна в условиях, когда изображение содержит большое количество фона и относительно немного объектов. Благодаря использованию функции потерь Focal Loss, RetinaNet достигает высокой точности при сохранении скорости одноэтапных методов.

В большинстве задач детекции данных существует сильный дисбаланс между фоновыми регионами (области, где нет объектов) и регионами, содержащими объекты. Этот дисбаланс приводит к следующим проблемам:

- Доминирование фоновых областей: Во время обучения модель концентрируется на правильной классификации фона, игнорируя области с объектами. Это снижает качество детекции.

- Ошибки в детекции мелких или редких объектов: Мелкие объекты или объекты, встречающиеся реже, могут быть проигнорированы моделью.

RetinaNet решает эту проблему с помощью комбинации следующих подходов:

- Одноэтапный подход:

  RetinaNet является одноэтапной сетью, что означает, что локализация и классификация объектов выполняются за один проход через сеть. Это делает модель быстрее, чем двухэтапные методы, такие как Faster R-CNN.

- Feature Pyramid Network (FPN):

  Для эффективной детекции объектов разных размеров RetinaNet использует FPN, которая создает пирамиду признаков. Это позволяет обрабатывать как крупные, так и мелкие объекты, обеспечивая мультимасштабный анализ.

- Функция потерь Focal Loss:

  Focal Loss специально разработана для обработки несбалансированных данных, что является ключевым новшеством RetinaNet.

1.4.1 Focal Loss: Принцип работы

Focal Loss — это модификация стандартной функции перекрестной энтропии, которая учитывает сложность классификации различных примеров, добавляя весовые коэффициенты. Эта функция специально разработана для борьбы с несбалансированностью данных в задачах детекции.

Функция Focal Loss определяется следующим образом:

$$FL(p_t) = -\alpha_t(1 - p_t)^\gamma \log(p_t),$$

где:

- p_t — вероятность правильной классификации, предсказанная моделью.
- α_t — весовой коэффициент, регулирующий важность редких классов.

- γ — параметр фокусировки, который уменьшает вклад легко классифицируемых примеров.

Основные аспекты работы Focal Loss:

- Примеры, которые классифицируются с высокой уверенностью ($p_t \to 1$), получают меньший вклад в функцию потерь благодаря фактору $(1-p_t)^\gamma$.
- Примеры, которые классифицируются с низкой уверенностью ($p_t \to 0$), получают больший вклад в функцию потерь, что стимулирует модель уделять им больше внимания.

Параметры α_t и γ позволяют гибко настраивать вклад различных примеров:

- γ определяет, насколько сильно подавляются легко классифицируемые примеры. При $\gamma = 0$ Focal Loss превращается в стандартную перекрестную энтропию.
- α_t используется для учёта дисбаланса классов, повышая важность редких классов.

Ограничения RetinaNet

Несмотря на свои преимущества, RetinaNet имеет ряд ограничений, которые могут повлиять на её применение в определённых задачах:

- Выше вычислительные затраты по сравнению с YOLO:

  Хотя RetinaNet быстрее двухэтапных моделей, таких как Faster R-CNN, она уступает YOLO в скорости обработки, что может быть критичным для задач реального времени.

- Чувствительность к настройке гиперпараметров Focal Loss:

  Параметры α_t и γ требуют тщательной настройки для достижения оптимальной производительности. Неправильный выбор этих параметров может привести к ухудшению качества детекции.

- Сложности при детекции в условиях сильного перекрытия объектов:

  Хотя RetinaNet хорошо работает с несбалансированными данными, плотные сцены с перекрывающимися объектами могут представлять для неё сложность, требуя дополнительных механизмов постобработки.

1.5 Faster R-CNN и двухэтапные методы детекции

Faster R-CNN является одной из самых популярных архитектур для решения задачи детекции объектов. Она относится к классу двухэтапных методов детекции, которые разделяют процесс на два основных шага: генерация регионов-кандидатов и последующая классификация и уточнение рамок. Такой подход обеспечивает высокую точность и делает модель универсальной для работы с различными типами данных.

Принципы работы двухэтапных методов

Двухэтапные методы, такие как Faster R-CNN, выполняют детекцию объектов в два последовательных этапа:

1. Генерация регионов-кандидатов (Region Proposal):

   На первом этапе модель идентифицирует области изображения, которые могут содержать объекты. Это осуществляется с помощью специальной подсети, называемой Region Proposal Network (RPN). RPN генерирует ограничивающие рамки (bounding boxes) с высокой вероятностью присутствия объектов.

2. Классификация и уточнение рамок:

   На втором этапе регионы-кандидаты, сгенерированные RPN, передаются в основной классификатор. Этот классификатор определяет классы объектов и уточняет координаты рамок для повышения точности локализации.

Архитектура Faster R-CNN

Faster R-CNN представляет собой улучшенную версию предыдущих двухэтапных моделей, таких как Fast R-CNN. Её ключевые компоненты:

1. Сверточная нейронная сеть для выделения признаков:

   На вход изображения подается сверточная сеть (например, ResNet или VGG), которая извлекает карты признаков. Эти карты признаков служат основой для всех последующих операций.

2. Region Proposal Network (RPN):

   Это отдельная подсеть, которая сканирует карту признаков и предлагает ограничивающие рамки-кандидаты. RPN использует механизм якорей (anchors), которые представляют собой предварительно определённые формы и размеры рамок.

3. ROI Pooling (или ROI Align):

   Этот модуль преобразует области-кандидаты, предложенные RPN, в фиксированный размер, необходимый для классификации. ROI Align, в отличие от старого ROI Pooling, обеспечивает более точное отображение признаков, что повышает точность.

4. Классификатор и регрессор рамок:

   На последнем этапе классификатор предсказывает классы объектов, а регрессор уточняет координаты рамок для их более точного позиционирования.

Преимущества Faster R-CNN

- Высокая точность:

  Использование двухэтапного подхода позволяет добиться превосходных результатов в сложных сценах и при наличии множества объектов.

- Модульность:

  Faster R-CNN легко адаптируется под различные задачи, заменяя базовую сверточную сеть или оптимизируя RPN для специфических данных.

- Универсальность:

  Модель может эффективно работать с различными размерами объектов благодаря использованию якорей разного масштаба.

Ограничения двухэтапных методов

- Высокие вычислительные затраты:

  Двухэтапный подход требует значительных ресурсов, особенно при обработке изображений высокого разрешения.

- Скорость работы:

  Faster R-CNN медленнее одноэтапных моделей, таких как YOLO или RetinaNet, что ограничивает её использование в задачах реального времени.

- Чувствительность к настройке параметров:

  Эффективность модели сильно зависит от точности настройки якорей и других гиперпараметров.

1.6 Примеры успешных решений задач детекции объектов с использованием нейронных сетей

Нейронные сети демонмтрируют выдающиеся результаты в различных задачах детекции объектов.

Детекция летательных объектов

- Дроны и самолёты:

  Системы мониторинга воздушного пространства используют такие архитектуры, как Faster R-CNN и RetinaNet, для точного обнаружения и классификации летательных объектов в реальном времени. Применение Focal Loss помогает эффективно справляться с несбалансированностью данных, когда на фоне преобладает значительное количество пустого пространства.

- Птицы и их маршруты миграции:

  Использование сверточных сетей позволяет точно идентифицировать птиц на изображениях и видео, что важно для предотвращения столкновений с авиацией или изучения экологических процессов.

Медицинская диагностика

- Обнаружение опухолей:

  Faster R-CNN успешно применяется для локализации опухолей на медицинских изображениях, таких как рентгеновские снимки, КТ и МРТ. Высокая точность модели позволяет минимизировать количество ложных срабатываний, что особенно важно для медицинских задач.

- Анализ кожных заболеваний:

  Модели, такие как YOLO и RetinaNet, используются для идентификации различных типов кожных повреждений и болезней, обеспечивая автоматизацию процесса диагностики.

Системы безопасности и видеонаблюдение

- Обнаружение людей:

  Архитектуры YOLO и SSD применяются для отслеживания людей на видео в режиме реального времени. Такие системы востребованы в интеллектуальном видеонаблюдении для охраны объектов.

- Распознавание подозрительных объектов:

  Сети, такие как RetinaNet, используются для детекции потенциально опасных объектов, таких как оружие, на видеозаписях или фотографиях.

Автомобильные технологии

- Обнаружение автомобилей и пешеходов:

  Системы помощи водителям (ADAS) широко используют Faster R-CNN и YOLO для детекции автомобилей, пешеходов и дорожных знаков. Эти модели играют ключевую роль в разработке автономных транспортных средств.

- Анализ дорожных условий:

  Нейронные сети позволяют определять дорожные ямы, полосы движения и препятствия, повышая безопасность водителей.

Анализ изображений в промышленности

- Контроль качества продукции:

  Сверточные нейронные сети успешно применяются для обнаружения дефектов на производственных линиях, например, трещин, царапин или некорректных размеров продукции.

- Мониторинг состояния инфраструктуры:

  Faster R-CNN используется для анализа состояния мостов, зданий и других инженерных объектов на снимках и видео с дронов.

Экологический мониторинг

- Отслеживание животных:

  Нейронные сети помогают обнаруживать и идентифицировать животных в дикой природе, поддерживая программы защиты видов.

- Обнаружение пожаров:

  Системы на базе YOLO используются для детекции очагов возгорания на изображениях и видео, что позволяет оперативно реагировать на угрозы лесных пожаров.

ГЛАВА 2

АНАЛИЗ ТРЕБОВАНИЙ И ПОСТАНОВКА ЗАДАЧИ

2.1 Цель системы обнаружения и классификация воздушных объектов

Центральным элементом подсистемы анализа изображений будет выступать модель глубокого обучения, основанная на архитектурном паттерне YOLO (You Only Look Once). Выбор данного семейства архитектур обусловлен его признанной эффективностью в задачах обнаружения объектов в режиме реального времени. В отличие от двухстадийных детекторов (например, семейства R-CNN), YOLO реализует парадигму однопроходного анализа изображения: нейронная сеть напрямую предсказывает координаты ограничивающих рамок и вероятности принадлежности к классам для всех объектов на изображении за один прямой проход. Это позволяет достигать высокой скорости обработки, измеряемой десятками кадров в секунду даже на относительно доступном аппаратном обеспечении, что является критически важным параметром для задач видеонаблюдения за динамичными сценами. Математически, задача детекции в YOLO формулируется как регрессионная задача. Изображение делится на сетку $S \times S$. Для каждой ячейки сетки предсказывается B ограничивающих рамок и вероятности классов C. Каждая рамка характеризуется 5 параметрами: $(x, y, w, h, conf)$, где (x, y) – координаты центра рамки относительно границ ячейки, (w, h) – ширина и высота рамки относительно всего изображения, $conf$ – уверенность (confidence score), отражающая наличие объекта в рамке и точность предсказания. Уверенность определяется как $conf = P(\text{Object}) \times \text{IOU}_{\text{pred}}^{\text{truth}}$, где $P(\text{Object})$ – вероятность наличия объекта в ячейке, а $\text{IOU}_{\text{pred}}^{\text{truth}}$ (Intersection over Union) – мера пересечения предсказанной рамки с истинной. Для каждой ячейки также предсказываются условные вероятности классов $P(Class_i|\text{Object})$. Итоговый тензор предсказаний имеет размерность $S \times S \times (B \times 5 + C)$.

Успешная интеграция и эффективная работа модели глубокого обучения, подобной YOLO, в составе конечной системы напрямую зависит от качества и своевременности поступления входных данных – видеопотока. Нейронная

сеть требует непрерывной последовательности кадров с заданными параметрами (разрешение, частота кадров), минимизацией задержек и искажений. Следовательно, подсистема получения, передачи и подготовки видеоданных является критически важным инфраструктурным компонентом. Задержки в передаче видео, вариации времени между кадрами, потеря кадров или недостаточная пропускная способность канала связи могут привести к снижению эффективности обнаружения, пропуску быстродвижущихся целей или некорректной оценке их траекторий. Таким образом, целью настоящей работы является создать систему непрерывной видеотрансляции и связать ее с YOLO моделью для обработки видеопотока в реальном времени.

2.2 Функциональные требования к системе видеотрансляции

Функциональные требования определяют ключевые возможности и характеристики, которыми должна обладать разрабатываемая система видеотрансляции для обеспечения корректной и эффективной работы последующего модуля анализа изображений на базе архитектуры YOLO. Эти требования напрямую вытекают из специфики задачи обнаружения воздушных объектов в режиме реального времени и особенностей функционирования современных детекторов глубокого обучения. Несоответствие системы этим требованиям может привести к существенной деградации производительности всей системы обнаружения или к ее полной неработоспособности . Основными функциональными требованиями к системе видеотрансляции являются обеспечение непрерывности потока, передача видео с заданными параметрами разрешения и частоты кадров, а также использование формата кадров, оптимального для дальнейшей обработки.

Непрерывность видеопотока. Система должна обеспечивать непрерывную передачу последовательности видеокадров от камеры к модулю ИИ без пропусков и значительных непредсказуемых пауз. Любые прерывания в потоке данных могут привести к потере информации о динамике сцены, пропуску кратковременных событий или объектов, пересекающих поле зрения камеры за время отсутствия данных. Для задач мониторинга воздушного пространства, где объекты могут двигаться с высокой скоростью, непрерывность является критически важным параметром, гарантирующим целостность наблюдения и возможность построения траекторий объектов.

Параметры видеопотока. Система должна поддерживать передачу видео с разрешением и частотой кадров, соответствующими требованиям модели YOLO и характеру наблюдаемых объектов.

Разрешение (Resolution) определяет пространственную детализацию изображения. Модели семейства YOLO обычно оперируют с входными изображениями фиксированного размера. Система видеотрансляции должна либо получать видеопоток в нативном разрешении, близком к требуемому моделью, либо обеспечивать возможность эффективного масштабирования (resizing) на стороне приема или передачи. Выбор разрешения является компромиссом: более высокое разрешение позволяет обнаруживать более мелкие объекты на больших дистанциях, но увеличивает объем передаваемых данных и вычислительную нагрузку на модуль ИИ.

Частота кадров (Frames Per Second, FPS) определяет временное разрешение видеопотока. Для задач обнаружения динамических объектов, таких как летательные аппараты, требуется достаточно высокая частота кадров для минимизации смазывания изображения и обеспечения возможности надежного трекинга. С другой стороны, FPS напрямую влияет на пропускную способность канала и нагрузку на систему обработки. Минимально допустимая частота кадров f_{min} может быть оценена исходя из максимальной ожидаемой угловой скорости объекта ω_{max} относительно камеры и максимально допустимого углового смещения объекта $\Delta\theta_{max}$ между кадрами, при котором обнаружение/трекинг еще возможны:

$$f_{min} \gtrsim \frac{\omega_{max}}{\Delta\theta_{max}} \tag{2.1}$$

Типичные значения для систем видеонаблюдения реального времени лежат в диапазоне 15-30 FPS, однако для высокоскоростных объектов может потребоваться и более высокая частота. Система видеотрансляции должна гарантировать передачу видеопотока с заданной или согласованной частотой кадров.

Формат передаваемых кадров. Видеокадры должны передаваться в формате, который легко и с минимальными задержками может быть обработан библиотеками компьютерного зрения (например, OpenCV) и фреймворками глубокого обучения (PyTorch, TensorFlow) на стороне ИИ-модуля. Возможны два основных подхода: передача несжатых кадров или использование видеокодеков для сжатия. Передача *несжатых кадров* (например, в формате

BGR или RGB) минимизирует задержку на декодирование на принимающей стороне, так как данные готовы к непосредственной передаче в нейронную сеть. Однако это требует значительно большей пропускной способности канала связи. Пропускная способность BW_{raw} (бит/с) рассчитывается как:

$$BW_{raw} = W \times H \times d \times f \tag{2.2}$$

где W, H – ширина и высота кадра в пикселях, d – глубина цвета в битах на пиксель (например, 24 для BGR), f – частота кадров (FPS). Использование *видеокодеков* (например, H.264, H.265/HEVC) позволяет существенно снизить требования к пропускной способности за счет алгоритмов сжатия с потерями или без потерь . Однако это вносит дополнительные задержки на кодирование на передающей стороне и декодирование на принимающей стороне, а также может привести к появлению артефактов сжатия, потенциально влияющих на точность обнаружения мелких или низкоконтрастных объектов. Выбор конкретного формата и кодека (если используется сжатие) является важным проектным решением, влияющим на баланс между качеством видео, задержкой и требованиями к сети. Система должна поддерживать согласованный формат на всем пути передачи данных.

Таким образом, функциональные требования к системе видеотрансляции определяют ее способность доставлять непрерывный, своевременный и качественно подготовленный видеопоток, параметры которого (разрешение, FPS, формат) оптимизированы для эффективной работы последующего модуля обнаружения воздушных объектов на базе архитектуры YOLO.

2.3 Нефункциональные требования

Помимо функциональных требований, определяющих что система должна делать, не менее важную роль играют нефункциональные требования (НФТ), которые описывают как система должна выполнять свои функции. Эти атрибуты качества, такие как производительность, надежность и эксплуатационные характеристики, являются критически важными для систем, работающих в режиме реального времени, к которым относится разрабатываемая система видеотрансляции для последующей интеграции с модулем ИИ. Несоответствие НФТ может сделать систему практически бесполезной, даже если все функциональные требования формально выполнены. Ключевыми НФТ для данной системы являются минимальная задержка передачи,

высокая надежность и, потенциально, масштабируемость.

Требование к задержке передачи (Latency). Задержка, или латентность, определяется как временной интервал между моментом захвата кадра камерой и моментом его доступности для обработки модулем ИИ на принимающей стороне. Для систем обнаружения быстродвижущихся воздушных объектов, таких как БПЛА, минимизация задержки является первостепенной задачей. Высокая задержка приводит к тому, что система ИИ анализирует устаревшую информацию о положении объекта, что снижает точность сопровождения (трекинга), увеличивает время реакции системы на появление новой цели и может привести к пропуску объектов, быстро пересекающих поле зрения . Модели типа YOLO, хотя и обладают высокой скоростью инференса, не могут компенсировать значительные задержки, вносимые системой доставки данных. Общая сквозная задержка T_{total} складывается из нескольких компонент:

$$T_{total} = T_{capture} + T_{encode} + T_{network} + T_{decode} + T_{buffer} + T_{transfer\_to\_AI} \quad (2.3)$$

где $T_{capture}$ – время захвата кадра, T_{encode} – время кодирования (если используется сжатие), $T_{network}$ – время сетевой передачи, T_{decode} – время декодирования, T_{buffer} – задержка в буфере приема (для компенсации джиттера), $T_{transfer\_to\_AI}$ – время передачи кадра из модуля видеотрансляции в модуль ИИ. Целевое значение для T_{total} в системах реального времени часто стремятся удерживать в пределах 100-300 миллисекунд, однако конкретное значение должно определяться исходя из максимальной ожидаемой скорости объектов и требуемой точности позиционирования. Проектируемая система видеотрансляции должна минимизировать компоненты T_{encode}, $T_{network}$, T_{decode} и T_{buffer} за счет выбора эффективных кодеков, протоколов с низкой задержкой (например, RTP/UDP, WebRTC) и оптимизированной реализации модулей.

Требование к надежности (Reliability). Надежность системы видеотрансляции характеризует ее способность функционировать без сбоев и потерь данных в течение заданного времени. В контексте передачи видеопотока, надежность проявляется в стабильности соединения, отсутствии существенных пропусков кадров и искажений изображения. Потеря даже нескольких последовательных кадров может привести к срыву сопровождения объекта или невозможности его первичного обнаружения. Использование протоколов, не гарантирующих доставку (например, UDP), которое предпочтительно для

минимизации задержки, повышает риск потери пакетов данных в сети. Допустимый уровень потерь пакетов должен быть строго ограничен. Хотя точное значение зависит от используемого кодека и его устойчивости к ошибкам, для качественной видеоаналитики уровень потерь пакетов обычно не должен превышать 0.1-1% . Для повышения надежности при использовании негарантированных протоколов могут применяться методы упреждающей коррекции ошибок или механизмы ограниченных повторных передач, однако они вносят дополнительную задержку и избыточность. Система должна быть спроектирована с учетом баланса между задержкой и надежностью, возможно, с использованием адаптивных механизмов, подстраивающихся под текущее состояние сети. Также важна стабильность работы самих программных модулей системы, минимизация вероятности их "зависания"или аварийного завершения.

Требование к масштабируемости (Scalability). Масштабируемость определяет способность системы наращивать производительность или обслуживать большее количество пользователей/источников при увеличении ресурсов (аппаратных или программных). В контексте данной задачи, масштабируемость может рассматриваться по нескольким измерениям: поддержка большего числа одновременных видеопотоков от разных камер, обработка потоков с более высоким разрешением или частотой кадров, подключение большего числа клиентов (модулей ИИ или систем визуализации). Хотя на этапе преддипломной практики основной фокус может быть на обеспечении работы одного потока с заданными параметрами, архитектура системы должна закладывать потенциальную возможность масштабирования в будущем. Это влияет на выбор архитектурных паттернов (например, использование микросервисов, балансировщиков нагрузки), протоколов и технологий, способных эффективно распределять нагрузку. Например, использование протокола multicast для одновременной доставки одного потока многим получателям может быть более масштабируемым решением, чем множество unicast-соединений. На данном этапе требование к масштабируемости является вторичным по отношению к задержке и надежности, но архитектурные решения должны его учитывать.

В совокупности, нефункциональные требования по задержке, надежности и масштабируемости определяют качественные характеристики разрабатываемой системы видеотрансляции и ее пригодность для интеграции в комплексную систему обнаружения воздушных объектов на базе ИИ, способную

эффективно функционировать в реальных условиях эксплуатации.

2.4 Формальная постановка задачи

На основе проведенного анализа требований к системе обнаружения летательных аппаратов (ЛА) и, в частности, к подсистеме видеотрансляции, формулируется следующая задача дипломной работы.

Пусть задан входной видеопоток $V = \{I_t\}_{t=1}^{N}$, где I_t – кадр видеоизображения в момент времени t, а N – общее количество кадров (в случае потоковой передачи $N \to \infty$). Каждый кадр I_t представляет собой матрицу пикселей размером $W \times H \times C$, где W, H – ширина и высота кадра, C – количество цветовых каналов. Необходимо разработать и исследовать программный комплекс S, способный в режиме, приближенном к реальному времени, выполнять следующие функции:

1. **Прием и обработка видеопотока:** Система S_{stream} должна обеспечивать прием видеопотока V от удаленного источника по сети с использованием протокола RTSP. Принятые кадры I_t должны быть декодированы и подготовлены для дальнейшего анализа. Задержка передачи и декодирования Δt_{stream} должна быть минимизирована.

2. **Обнаружение летательных аппаратов:** Система S_{detect} должна анализировать последовательность кадров $\{I'_t\}$ (где I'_t – кадр после приема и возможной предобработки) с использованием предварительно обученной сверточной нейронной сети семейства YOLO (версии YOLOv11 или аналогичной) для идентификации и локализации объектов класса "летательный аппарат". Для каждого обнаруженного объекта o_j на кадре I'_t система должна возвращать набор параметров $D_{t,j} = (\mathbf{b}_{t,j}, c_{t,j}, s_{t,j})$, где $\mathbf{b}_{t,j}$ – координаты ограничивающей рамки (bounding box), $c_{t,j}$ – класс объекта, и $s_{t,j}$ – уверенность модели в предсказании.

3. **Визуализация результатов:** Система должна предоставлять возможность визуализации исходного видеопотока с наложенными результатами детекции.

Дополнительной исследовательской задачей является анализ влияния методов программной стабилизации видео на качество детекции ЛА. Необходимо реализовать и оценить алгоритм стабилизации, а также исследовать его

вычислительную сложность и применимость в контексте системы реального времени.

Таким образом, требуется создать интегрированную систему, включающую модули видеотрансляции, нейросетевого анализа и визуализации, и провести ее экспериментальное исследование для оценки соответствия поставленным требованиям и определения путей дальнейшего совершенствования.

2.5 Выводы

В настоящей главе был проведен всесторонний анализ требований к системе обнаружения летательных аппаратов с использованием технологий искусственного интеллекта, с особым акцентом на подсистему видеотрансляции, которая является критически важным компонентом для обеспечения своевременной и качественной передачи данных на модуль анализа.

Была сформулирована конечная цель дипломной работы – разработка и исследование прототипа такой системы, способной интегрировать современные подходы к передаче видео и его анализу с помощью нейронных сетей семейства YOLO. На основе анализа были определены ключевые функциональные требования к подсистеме видеотрансляции. К ним относятся: способность захватывать видеопоток с различных источников, кодировать его в эффективный формат, передавать по сети с использованием стандартизированного протокола, обеспечивать возможность подключения клиентов для приема и декодирования потока, а также поддерживать различные разрешения и частоты кадров.

Также были сформулированы нефункциональные требования, играющие важную роль в практической применимости системы. Среди них: обеспечение низкой задержки передачи видео, высокая производительность, надежность и стабильность работы, масштабируемость.

ГЛАВА 3

ПРОЕКТИРОВАНИЕ СИСТЕМЫ ВИДЕОТРАНСЛЯЦИИ И ВЫБОР ТЕХНОЛОГИЙ

3.1 Обзор существующих решений и протоколов для видеотрансляции

Передача видеоданных в реальном времени является фундаментальной задачей для множества приложений, включая системы видеонаблюдения, телеконференцсвязь и, что особенно релевантно для данного исследования, системы компьютерного зрения, предназначенные для анализа динамических сцен, таких как обнаружение и распознавание воздушных объектов. Эффективность таких систем напрямую зависит от характеристик используемого механизма видеотрансляции, прежде всего от задержки (latency), пропускной способности (bandwidth) и надежности доставки данных. Существует несколько устоявшихся протоколов и технологических стеков, предназначенных для решения этой задачи, каждый со своими особенностями, преимуществами и недостатками.

Одним из наиболее распространенных протоколов в системах видеонаблюдения является **Real-Time Streaming Protocol (RTSP)**. RTSP функционирует как протокол управления сеансами связи, позволяя клиенту удаленно управлять потоком медиаданных с сервера. Сами медиаданные обычно передаются с использованием протокола **Real-time Transport Protocol (RTP)**, который работает поверх UDP или TCP. Использование UDP обеспечивает минимальную задержку за счет отсутствия гарантий доставки и механизмов повторной передачи, что критично для приложений реального времени. Однако это может приводить к потере пакетов в сетях с низким качеством связи. Информацию о качестве приема и синхронизации предоставляет протокол **RTCP (RTP Control Protocol)**.

Альтернативой, набирающей популярность в веб-приложениях, является технология **Web Real-Time Communication (WebRTC)**. WebRTC представляет собой набор API и протоколов, позволяющих организовать передачу потоковых данных напрямую между браузерами или другими клиентскими

приложениями (peer-to-peer) без необходимости установки дополнительных плагинов. WebRTC включает механизмы для обхода NAT (Network Address Translation) и файрволов, что упрощает установление соединения. Для передачи медиаданных используется RTP, а для управления сессией — протоколы на основе SDP (Session Description Protocol). WebRTC ориентирован на минимизацию задержки и хорошо подходит для интерактивных приложений. Однако его настройка и управление сигнальным сервером могут быть сложнее по сравнению с RTSP.

Для доставки видеоконтента большому числу пользователей через Интернет часто применяются протоколы потоковой передачи на базе HTTP, такие как **HTTP Live Streaming (HLS)** и **Dynamic Adaptive Streaming over HTTP (MPEG-DASH)** . Эти протоколы разбивают видеопоток на небольшие сегменты (чанки), которые клиент последовательно загружает по HTTP. Такой подход позволяет использовать стандартную веб-инфраструктуру (веб-серверы, CDN) для масштабирования и обеспечивает адаптацию качества видео под текущую пропускную способность сети клиента. Основным недостатком HLS и MPEG-DASH является сравнительно высокая задержка (от нескольких секунд до десятков секунд), обусловленная буферизацией и сегментированием, что делает их менее пригодными для задач, требующих реакции в реальном времени.

Помимо протоколов, важную роль играют программные библиотеки и фреймворки, обеспечивающие захват, обработку и передачу видео. Библиотека **OpenCV (Open Source Computer Vision Library)** является стандартом для задач компьютерного зрения и предоставляет функции для захвата видео с камер и файлов, но её встроенные возможности для сетевой трансляции ограничены. Более мощными инструментами для работы с медиапотоками являются **GStreamer** и **FFmpeg**. GStreamer представляет собой фреймворк на основе графов (пайплайнов), позволяющий гибко конструировать конвейеры обработки медиаданных, включая захват, кодирование, мультиплексирование и передачу по различным протоколам (включая RTSP, RTP, WebRTC). FFmpeg — это универсальный набор библиотек и утилита командной строки для работы с аудио и видео, поддерживающий огромное количество форматов и протоколов. FFmpeg часто используется как бэкенд во многих мультимедийных приложениях и может служить как для создания, так и для приема видеопотоков.

Выбор кодека также имеет существенное значение. Стандарт **H.264 (AVC)**

обеспечивает хорошее соотношение качества и степени сжатия при умеренных вычислительных затратах и широкой аппаратной поддержке. Более современный стандарт **H.265 (HEVC)** позволяет достичь примерно двукратного улучшения эффективности сжатия по сравнению с H.264 при сопоставимом визуальном качестве, но требует значительно больших вычислительных ресурсов для кодирования и декодирования. Эффективность сжатия η можно оценить как отношение исходного битрейта B_{raw} к сжатому B_{coded}:

$$\eta = \frac{B_{raw}}{B_{coded}} \approx \frac{W \times H \times FPS \times D}{B_{coded}} \tag{3.1}$$

где W, H — ширина и высота кадра, FPS — частота кадров, D — глубина цвета (бит/пиксель). Выбор кодека влияет на требуемую пропускную способность сети B_{coded} и вычислительную нагрузку на систему.

Таким образом, выбор конкретного решения для видеотрансляции требует тщательного анализа требований задачи, особенно в контексте задержки, масштабируемости, сложности реализации и совместимости с последующими этапами обработки, такими как анализ с помощью нейронных сетей.

3.2 Сравнение технологий и обоснование выбора

Выбор технологического стека для системы видеотрансляции, предназначенной для последующей интеграции с системой обнаружения объектов на базе нейронных сетей, должен основываться на комплексном анализе рассмотренных протоколов и инструментов в контексте специфических требований задачи. Ключевыми критериями для сравнения являются: задержка передачи видеопотока, надежность доставки данных, вычислительная сложность, гибкость настройки, масштабируемость и совместимость с инструментами компьютерного зрения.

Сравнение протоколов передачи. Протоколы HLS и MPEG-DASH, несмотря на их преимущества в масштабируемости и адаптивности для веб-доставки, характеризуются значительной задержкой (от нескольких секунд), что является критическим недостатком для систем обнаружения объектов в реальном времени. Реакция системы обнаружения должна быть максимально быстрой, особенно при работе с динамичными объектами, такими как летательные аппараты. Следовательно, эти протоколы являются неподходящими для основной цели — обеспечения входных данных для YOLOv11 с мини-

мальной латентностью.

Протоколы RTSP/RTP и WebRTC обеспечивают значительно меньшую задержку. WebRTC предлагает современные механизмы для P2P-соединений и обхода сетевых ограничений, что привлекательно для интерактивных веб-приложений. Однако его реализация, особенно управление сигнализацией и поддержка серверной части для не-браузерных клиентов (как система анализа на базе YOLO), может быть более сложной. RTSP, в свою очередь, является зрелым и широко распространенным протоколом в системах видеонаблюдения и IP-камерах. Он обеспечивает низкую задержку при передаче через RTP/UDP и относительно прост в реализации как на стороне сервера (источника потока), так и на стороне клиента (системы анализа). Потенциальная проблема потери пакетов при использовании UDP может быть частично смягчена применением RTP поверх TCP, хотя это несколько увеличит задержку, но обеспечит гарантированную доставку. Для сценария, где один или несколько источников видео передают поток на выделенный сервер анализа, RTSP представляется более прагматичным выбором из-за баланса между низкой задержкой и простотой интеграции.

Сравнение программных библиотек и фреймворков. Библиотека OpenCV незаменима для задач обработки изображений и интеграции с нейронными сетями, такими как YOLO. Она предоставляет удобный интерфейс для захвата кадров с различных источников. Однако ее возможности по созданию полноценного сервера потоковой передачи ограничены. FFmpeg является мощным инструментом для кодирования, декодирования и трансляции, но его использование в виде библиотеки требует значительных усилий по интеграции. GStreamer предлагает гибкую модульную архитектуру на основе конвейеров (pipelines), которая позволяет легко комбинировать элементы для захвата, обработки (включая аппаратное ускорение кодирования/декодирования при наличии), и трансляции по различным протоколам. GStreamer имеет готовые плагины для работы с RTSP и RTP, а также может взаимодействовать с OpenCV для включения этапов предварительной обработки видео перед передачей. Гибкость GStreamer в построении конвейеров обработки медиаданных делает его предпочтительным выбором для создания кастомизированной системы видеотрансляции, способной эффективно подготавливать поток для системы анализа.

Выбор видеокодека. Выбор между H.264 и H.265 зависит от баланса между требуемой пропускной способностью и доступными вычислительны-

ми ресурсами. Для системы реального времени, где на том же оборудовании может выполняться ресурсоемкая задача анализа YOLOv11, дополнительная нагрузка от кодирования H.265 может оказаться чрезмерной. Хотя H.265 позволяет снизить битрейт примерно на 30-50% по сравнению с H.264 при сопоставимом качестве, что выражается формулой:

$$B_{coded,H265}(Q) \approx (0.5 \dots 0.7) \times B_{coded,H264}(Q) \tag{3.2}$$

где Q — некоторый уровень качества. Однако, вычислительная сложность кодирования C_{enc} для H.265 может быть в несколько раз выше: $C_{enc,H265} \gg C_{enc,H264}$. Учитывая широкую аппаратную поддержку H.264, в том числе для ускоренного кодирования (например, через NVENC для GPU NVIDIA или VA-API/VDPAU для других платформ), и его меньшую вычислительную сложность, H.264 представляется более безопасным выбором на начальном этапе. Это позволит минимизировать нагрузку на систему и обеспечить стабильную работу как подсистемы трансляции, так и подсистемы анализа. Переход на H.265 может рассматриваться как дальнейшая оптимизация при наличии достаточных вычислительных мощностей или специализированного аппаратного ускорения.

Обоснование выбора. На основании проведенного сравнительного анализа для реализации системы видеотрансляции в рамках данной работы выбирается следующий технологический стек:

- **Протокол передачи:** RTSP с использованием RTP поверх UDP для минимизации задержки. Возможность переключения на RTP поверх TCP рассматривается как резервная для сетей с высоким уровнем потерь пакетов.

- **Фреймворк для работы с медиапотоками:** GStreamer. Его гибкая конвейерная архитектура позволяет эффективно реализовать захват видео с источника, кодирование и организацию RTSP-сервера. Предусмотрена возможность интеграции с OpenCV на этапе захвата или предварительной обработки при необходимости.

- **Видеокодек:** H.264 (AVC). Данный кодек обеспечивает приемлемое качество сжатия при умеренных вычислительных затратах и имеет широкую поддержку, включая аппаратное ускорение, что критически важно

для снижения нагрузки на CPU и обеспечения ресурсов для работы нейросетевой модели YOLOv11.

Данный выбор обеспечивает компромисс между критически важной низкой задержкой, необходимой для системы обнаружения объектов в реальном времени, надежностью передачи в контролируемой сетевой среде (сервер анализа и источники видео), гибкостью реализации и умеренной вычислительной нагрузкой, совместимой с параллельной работой системы ИИ-анализа.

3.3 Проектирование архитектуры системы

Проектирование архитектуры системы, где источником видеопотока выступает мобильное приложение, разработанное на фреймворке Flutter, требует учета специфики мобильных платформ и ограничений, связанных с обработкой медиаданных в реальном времени. В отличие от сценария с фиксированными IP-камерами, здесь мобильное устройство само отвечает за захват, кодирование и инициирование передачи потока. Конечной целью остается предоставление видеоданных системе анализа на базе YOLOv11 с минимальной задержкой. В связи с этим предлагается трехуровневая архитектура: Мобильный издатель (Flutter App) $\rightarrow$ RTSP-сервер (ретранслятор/брокер) $\rightarrow$ Клиент-анализатор (YOLOv11).

Компонент мобильного издателя (Flutter Application). Данный компонент, функционирующий на мобильном устройстве (Android или iOS), является первоисточником видеоданных. Его ключевые задачи включают:

1. **Захват видео:** Получение доступа к камере устройства и захват последовательности видеокадров. Стандартным решением в экосистеме Flutter для этого является использование официального плагина **camera**, который предоставляет Dart API для управления камерой и получения превью-потока изображений. Этот плагин взаимодействует с нативными API камеры.

2. **Кодирование видео:** Преобразование сырых видеокадров (часто в формате YUV или BGRA) в сжатый формат H.264. Поскольку Dart не имеет встроенных эффективных видеокодеров, эта задача должна быть делегирована нативному коду платформы через механизм Platform Channels или с использованием специализированного плагина, который инкапсу-

лирует нативные библиотеки кодирования. Критически важным является использование аппаратного ускорения кодирования (MediaCodec на Android, VideoToolbox на iOS) для минимизации нагрузки на CPU мобильного устройства и обеспечения работы в реальном времени. Вычислительная сложность программного кодирования H.264 на мобильном процессоре может привести к неприемлемо высоким задержкам и чрезмерному энергопотреблению.

3. **Пакетизация RTP:** Упаковка закодированного H.264 потока в RTP-пакеты согласно RFC 6184. Этот процесс также выполняется на нативном уровне, часто в связке с кодировщиком. Каждый пакет должен содержать корректные временные метки и порядковые номера.

4. **Публикация по RTSP:** Установление соединения с RTSP-сервером и инициирование сессии для публикации потока. В отличие от стандартного клиента, запрашивающего поток (методы DESCRIBE, SETUP, PLAY), издатель использует методы `ANNOUNCE` и `RECORD` для объявления и начала передачи медиаданных на сервер. Реализация логики RTSP-публикации также требует нативного кода или использования плагина, предоставляющего такую функциональность, например, обертки над библиотекой FFmpeg.

Таким образом, Flutter-приложение выступает в роли "умного"источника, инкапсулируя логику захвата и первичной обработки/передачи, но существенно полагаясь на нативные возможности платформы для ресурсоемких операций кодирования и сетевого взаимодействия на уровне протоколов RTP/RTSP.

Компонент RTSP-сервера (Брокер/Ретранслятор). Промежуточный RTSP-сервер выполняет роль центрального узла, принимающего опубликованный поток от мобильного приложения и предоставляющего его одному или нескольким клиентам-потребителям, включая систему анализа YOLOv11. Его основные функции:

1. **Прием публикуемых потоков:** Сервер должен поддерживать RTSP-методы **ANNOUNCE** и **RECORD**, позволяющие клиентам (Flutter-приложению) инициировать передачу медиаданных на сервер.

2. **Управление сессиями:** Обработка подключений как от издателей, так и от потребителей, управление медиа-сессиями.

3. **Ретрансляция RTP-потока:** Прием RTP-пакетов от издателя и их пересылка всем подписанным потребителям без перекодирования (для минимизации задержки и нагрузки).

В качестве реализации RTSP-сервера может быть использован GStreamer с библиотекой gst-rtsp-server, настроенной для работы в режиме приема публикуемых потоков, либо специализированные медиа-серверы, поддерживающие RTSP-ingest. Использование GStreamer позволяет сохранить единообразие технологического стека, если другие компоненты системы также его используют. Сервер вносит минимальную дополнительную задержку T_{server_relay}, связанную с сетевой передачей и буферизацией внутри сервера.

Компонент клиента-анализатора (Система с YOLOv11). Этот компонент является потребителем видеопотока и реализует стандартную логику RTSP-клиента. Он подключается к RTSP-серверу, запрашивает поток, принимает RTP-пакеты, депакетизирует их, декодирует H.264 поток (желательно с аппаратным ускорением на платформе, где работает анализатор) и передает полученные кадры на вход нейронной сети YOLOv11 для анализа. Для реализации клиента могут использоваться библиотеки OpenCV (с бэкендом GStreamer или FFmpeg), непосредственно GStreamer или FFmpeg API.

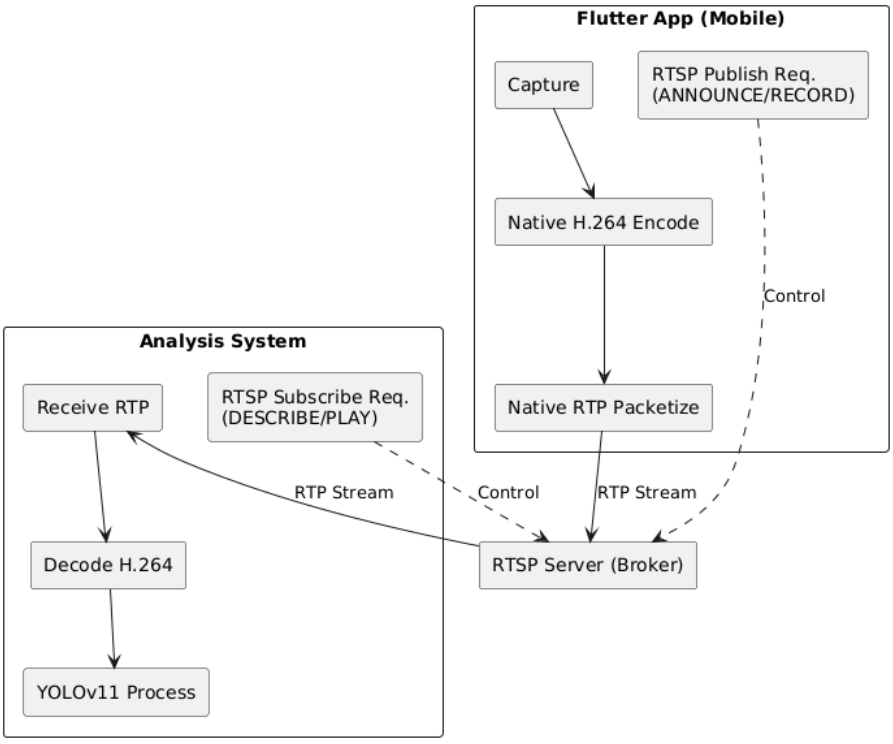

Рисунок 3.1 Общая схема взаимодействия

Выбор данной трехуровневой архитектуры, несмотря на повышенную слож-

ность реализации мобильного издателя и необходимость развертывания промежуточного сервера, оправдан возможностью использования мобильного устройства как гибкого источника видео и стандартизированным доступом к потоку для различных потребителей (не только YOLOv11, но и, например, систем мониторинга) через RTSP-сервер.

3.4 Выводы

В рамках настоящей главы были рассмотрены ключевые аспекты проектирования системы видеотрансляции, предназначенной для интеграции с системой обнаружения летательных аппаратов на базе нейронных сетей. Основное внимание было уделено анализу существующих решений и протоколов, их сравнительной оценке и обоснованию выбора технологического стека, наиболее полно отвечающего сформулированным функциональным и нефункциональным требованиям.

Проведенный обзор существующих протоколов видеотрансляции, включая RTSP, WebRTC, HLS и MPEG-DASH, а также программных библиотек, таких как OpenCV, GStreamer и FFmpeg, позволил выявить их сильные и слабые стороны в контексте задачи обеспечения низколатентной и надежной доставки видеоданных для анализа в реальном времени.

На основе сравнительного анализа и с учетом требований к минимальной задержке, простоте интеграции и совместимости с инструментами компьютерного зрения, для реализации системы видеотрансляции был выбран протокол **RTSP (Real-Time Streaming Protocol)** совместно с протоколом **RTP (Real-time Transport Protocol)** для передачи медиаданных, преимущественно поверх UDP. В качестве основного видеокодека был выбран стандарт **H.264 (AVC)** благодаря его оптимальному соотношению степени сжатия, качества изображения, широкой аппаратной поддержке и умеренным вычислительным требованиям, что особенно важно для снижения нагрузки на систему при параллельной работе с нейросетевым модулем. Для реализации серверной части, отвечающей за формирование и трансляцию RTSP-потока, было предложено использовать фреймворк **GStreamer** из-за его гибкости в построении конвейеров обработки медиаданных и наличия готовых компонентов для работы с RTSP.

Таким образом, в результате выполнения работ по данной главе была спроектирована архитектура системы видеотрансляции, которая обеспечива-

ет необходимые условия для последующей интеграции с модулем ИИ-анализа. Выбранные технологии и архитектурные решения направлены на достижение баланса между производительностью, надежностью, низкой задержкой и относительной простотой реализации в рамках дипломной работы. Полученные проектные решения являются основой для практической реализации и тестирования системы, описанных в последующих главах.

ГЛАВА 4

МЕТОДЫ СТАБИЛИЗАЦИИ ВИДЕОПОТОКА ДЛЯ УЛУЧШЕНИЯ ДЕТЕКЦИИ

4.1 Необходимость стабилизации видео для систем компьютерного зрения

Эффективность систем компьютерного зрения, предназначенных для анализа динамических сцен, таких как обнаружение и распознавание летательных аппаратов, в значительной степени зависит от качества входного видеопотока. Одним из существенных факторов, негативно влияющих на производительность алгоритмов анализа, является нежелательное движение камеры, проявляющееся в виде дрожания, вибраций или резких смещений. Такие артефакты особенно характерны для видео, получаемого с мобильных платформ, включая беспилотные летательные аппараты, ручные камеры или камеры, установленные на транспортных средствах. Устранение или минимизация этих нежелательных движений посредством стабилизации видеопотока становится критически важной задачей для обеспечения высокой точности и надежности систем детекции.

Нестабильность видеоряда приводит к ряду проблем для алгоритмов компьютерного зрения, особенно для тех, что основаны на глубоких нейронных сетях, таких как семейство моделей YOLO. Во-первых, дрожание вызывает размытие контуров объектов (motion blur), что затрудняет точную локализацию и классификацию. Визуальные признаки, используемые нейронной сетью для идентификации объекта, могут искажаться или становиться менее различимыми. Это особенно критично для малоразмерных или удаленных объектов, которые и без того представляют сложную задачу для детекции. Если объект занимает лишь небольшую область на кадре, его смещение даже на несколько пикселей из-за вибрации камеры может привести к тому, что детектор либо пропустит объект, либо неверно определит его границы.

Во-вторых, нежелательные движения камеры создают ложное меж кадровое смещение пикселей, которое не соответствует истинному движению объектов в сцене. Многие продвинутые алгоритмы детекции и сопровождения

объектов используют информацию о движении, полученную путем анализа разницы между последовательными кадрами. Например, оптический поток, который оценивает векторное поле смещений пикселей, будет искажен наличием вибраций камеры. Это может привести к ошибкам в оценке траектории движения объектов, ложным срабатываниям системы трекинга или, наоборот, к потере сопровождения реальных объектов. Влияние дрожания можно смоделировать как аддитивный шум к истинному движению пикселя p_t в момент времени t:

$$p'_t = p_t + \mathbf{M}_{cam}(t) \cdot p_t + n_t \tag{4.1}$$

где p'_t — наблюдаемое положение пикселя, $\mathbf{M}_{cam}(t)$ — матрица преобразования, описывающая нежелательное движение камеры (например, аффинное или проективное), и n_t — прочий шум. Задача стабилизации заключается в оценке $\mathbf{M}_{cam}(t)$ и применении обратного преобразования $\mathbf{M}_{cam}^{-1}(t)$ для компенсации этого движения.

В-третьих, высокочастотные вибрации могут приводить к увеличению вычислительной нагрузки на последующие этапы обработки. Алгоритмы, основанные на поиске соответствий между кадрами или на выделении ключевых точек, могут тратить значительные ресурсы на обработку ложных движений, вызванных дрожанием камеры, вместо того чтобы фокусироваться на анализе значимых изменений в сцене . Это снижает общую производительность системы в реальном времени.

Для систем обнаружения ЛА, где объекты могут быть быстрыми, малоразмерными и часто наблюдаются на фоне сложного или динамичного заднего плана (например, облака, земля с высоты), стабилизация видеопотока приобретает особое значение. Устранение дрожания позволяет:

- Повысить отношение сигнал/шум для визуальных признаков объекта, облегчая их выделение нейронной сетью.
- Улучшить точность определения границ (bounding box) детектируемых объектов.
- Снизить вероятность ложных срабатываний, вызванных случайными смещениями фона.
- Обеспечить более стабильную работу алгоритмов сопровождения (трекинга) объектов между кадрами.

- Потенциально снизить требования к сложности или глубине нейронной сети, так как ей придется работать с более "чистыми"данными.

Таким образом, предварительная стабилизация видео является важным этапом предварительной обработки, способным существенно повысить общую эффективность и устойчивость систем компьютерного зрения, ориентированных на детекцию и анализ объектов в условиях неидеальной съемки. Игнорирование этой проблемы может привести к значительному снижению производительности всей системы, несмотря на использование современных и мощных алгоритмов детекции.

4.2 Обзор существующих методов стабилизации видео

Проблема стабилизации видеоизображения, то есть компенсации нежелательных движений камеры, решается различными методами, которые можно условно разделить на три основные категории: механическая стабилизация, оптическая стабилизация изображения (OIS) и программная (или электронная) стабилизация изображения (EIS). Каждая из этих категорий обладает своими особенностями, преимуществами и недостатками, что определяет их применимость в различных сценариях и для разных типов систем.

4.2.1 Механическая стабилизация

Механическая стабилизация достигается за счет использования внешних устройств, физически изолирующих камеру от вибраций и резких движений платформы, на которой она установлена. Наиболее известными примерами таких устройств являются стедикамы и гироскопические подвесы (gimbals). Стедикамы представляют собой систему противовесов и шарниров, которые позволяют оператору плавно перемещать камеру, минимизируя передачу дрожания рук. Гироподвесы, часто используемые на БПЛА и профессиональном съемочном оборудовании, оснащены несколькими (обычно двумя или тремя) моторизованными осями и инерциальными измерительными блоками (IMU), включающими гироскопы и акселерометры. IMU детектирует угловые скорости и ускорения камеры, а система управления в реальном времени активирует моторы для компенсации этих движений, удерживая камеру в стабильном положении относительно земли или заданного направления.

Преимущества механической стабилизации очевидны: она работает непосредственно с физическим движением камеры, предотвращая попадание размытых изображений на сенсор. Это обеспечивает наиболее высокое качество стабилизации, особенно при значительных и низкочастотных колебаниях. Отсутствует необходимость в обрезке кадра (кроппинге) или искажении изображения, что характерно для некоторых программных методов. Качество результирующего видео, как правило, выше, так как исходный сигнал не подвергается цифровой обработке с целью стабилизации.

Недостатки механической стабилизации связаны с ее физической природой. Такие системы увеличивают габариты, вес и стоимость съемочного оборудования. Они требуют дополнительного энергопотребления для работы моторов и электроники. Гироподвесы могут быть чувствительны к сильным ударам или экстремальным ускорениям, выходящим за пределы их компенсационных возможностей. Кроме того, механические системы не могут полностью устранить высокочастотные вибрации малой амплитуды, которые все же могут проходить на сенсор.

4.2.2 Оптическая стабилизация изображения (OIS)

Оптическая стабилизация изображения (OIS) представляет собой технологию, встроенную непосредственно в объектив камеры или в механизм крепления сенсора. Принцип OIS заключается в активном смещении группы линз внутри объектива или самого сенсора для компенсации дрожания камер. Подобно гироподвесам, OIS-системы используют миниатюрные гироскопические датчики для определения нежелательных угловых смещений камеры. На основе данных с датчиков система управления генерирует сигналы для актуаторов (например, электромагнитов или пьезоэлектрических двигателей), которые с высокой скоростью перемещают корректирующий элемент в направлении, противоположном дрожанию. Таким образом, проекция изображения на сенсор остается стабильной, даже если корпус камеры испытывает небольшие колебания. Эффективность OIS E_{OIS} может быть выражена как отношение амплитуды углового смещения камеры θ_{cam} к остаточной амплитуде смещения изображения на сенсоре θ_{img}:

$$E_{OIS} = \frac{\theta_{cam}}{\theta_{img}} \tag{4.2}$$

Типичные значения E_{OIS} позволяют компенсировать дрожание в несколько угловых градусов.

Преимущества OIS заключаются в том, что стабилизация происходит до того, как свет попадает на сенсор, что предотвращает размытие изображения и сохраняет его полное разрешение без необходимости цифровой обработки или кроппинга. OIS системы компактны и могут быть интегрированы даже в небольшие камеры, такие как камеры смартфонов. Они эффективны против дрожания рук и низкочастотных вибраций.

Недостатки OIS также существуют. Диапазон компенсации движений у OIS обычно меньше, чем у продвинутых механических гироподвесов. OIS менее эффективна против очень быстрых и резких движений или сильных вибраций, характерных, например, для спортивной съемки или установки на вибрирующих платформах. Стоимость камер с качественной OIS выше, чем у камер без нее. Кроме того, OIS может вносить некоторые оптические артефакты при работе на пределе своих возможностей и не способна компенсировать поступательные смещения камеры, а только угловые.

4.2.3 Программная (электронная) стабилизация изображения (EIS)

Программная, или электронная, стабилизация изображения (EIS) достигается путем цифровой обработки уже захваченных видеокадров. В отличие от механической и оптической стабилизации, EIS не предотвращает попадание размытого изображения на сенсор, а пытается компенсировать смещения между кадрами постфактум. Общий принцип работы EIS включает три основных этапа:

1. **Оценка глобального движения (Global Motion Estimation):** Анализируется последовательность кадров для определения параметров нежелательного движения камеры между ними. Это может включать оценку смещения, вращения, масштабирования и, в более сложных моделях, проективных искажений.

2. **Фильтрация движения (Motion Filtering/Smoothing):** Оцененные параметры движения сглаживаются для отделения намеренного движения камеры (например, панорамирование) от нежелательного дрожания.

3. **Компенсация движения (Motion Compensation):** К текущему кадру применяется геометрическое преобразование (например, сдвиг, поворот, масштабирование), обратное оцененному нежелательному движению, чтобы выровнять его относительно предыдущих или усредненных кадров.

Преимущества EIS заключаются в отсутствии необходимости в дополнительном механическом или оптическом оборудовании, что делает этот метод наиболее дешевым и легко реализуемым, особенно если вычислительные ресурсы доступны. EIS может быть очень эффективной против высокочастотных вибраций и способна компенсировать как угловые, так и поступательные смещения. Современные алгоритмы EIS могут обеспечивать качество стабилизации, сопоставимое с OIS в определенных условиях.

Недостатки EIS являются прямым следствием того, что она работает с уже захваченным изображением. Если исходный кадр размыт из-за движения камеры во время экспозиции (motion blur), EIS не сможет устранить это размытие, а лишь стабилизирует уже размытое изображение. Для компенсации движения EIS часто требует обрезки краев кадра (кроппинг), так как после выравнивания по краям могут появляться пустые области. Это приводит к потере части поля зрения и эффективного разрешения. Степень кроппинга C_{EIS} зависит от амплитуды компенсируемого движения. Более сложные алгоритмы EIS могут требовать значительных вычислительных ресурсов, что может быть проблемой для систем реального времени или устройств с ограниченной мощностью. Также, при неточной оценке движения или чрезмерной компенсации, EIS может вносить в видео нежелательные артефакты, такие как эффект "плавающего"или "желейного"изображения (jello effect).

Таким образом, каждый из рассмотренных подходов к стабилизации видео имеет свою область применения. Для задач, где требуется максимальное качество и есть возможность использовать дополнительное оборудование, предпочтительна механическая стабилизация. OIS является хорошим компромиссом для компактных устройств, обеспечивая хорошее качество без значительного увеличения габаритов. EIS предлагает гибкое и дешевое решение, но с потенциальными компромиссами по качеству и потере разрешения, что требует тщательного выбора и настройки алгоритмов, особенно для систем компьютерного зрения.

4.3 Алгоритмы программной стабилизации видео

Программная стабилизация изображения (EIS) основывается на цифровой обработке последовательности видеокадров с целью компенсации нежелательных движений камеры. В основе большинства EIS-алгоритмов лежит трехэтапный процесс: оценка глобального движения камеры между кадрами, фильтрация этого движения для выделения нежелательной составляющей и, наконец, компенсация этого движения путем применения обратного геометрического преобразования к текущему кадру. Различные алгоритмы EIS отличаются, прежде всего, методами, используемыми на этапе оценки движения, а также моделями движения и техниками его фильтрации.

4.3.1 Методы, основанные на выделении и отслеживании ключевых точек

Одним из наиболее распространенных подходов к оценке глобального движения камеры является использование методов, основанных на выделении и отслеживании особых (ключевых) точек или признаков (features) на изображении. Предполагается, что большинство этих признаков принадлежат статическому фону, и их смещение между кадрами отражает движение камеры. Популярные дескрипторы признаков, такие как SIFT (Scale-Invariant Feature Transform), SURF (Speeded Up Robust Features), или ORB (Oriented FAST and Rotated BRIEF), используются для обнаружения устойчивых к масштабированию, вращению и изменению освещенности точек на изображении. После обнаружения признаков на двух последовательных кадрах, I_t и I_{t-1}, производится их сопоставление (matching) для нахождения соответствующих пар. Пусть $\mathcal{P}_t = \{p_{i,t}\}$ - множество ключевых точек на кадре I_t, а $\mathcal{P}_{t-1} = \{p_{i,t-1}\}$ - соответствующее множество точек на кадре I_{t-1}. Задача состоит в нахождении геометрического преобразования $\mathbf{M}_t$, которое наилучшим образом отображает точки из $\mathcal{P}_{t-1}$ в $\mathcal{P}_t$:

$$p_{i,t} \approx \mathbf{M}_t \cdot p_{i,t-1} \tag{4.3}$$

В качестве модели движения $\mathbf{M}_t$ часто используются аффинное преобразование (6 параметров), проективное преобразование (гомография, 8 параметров) или более простые модели, такие как евклидово преобразование (смещение и вращение, 3 параметра. Параметры преобразования находятся путем ми-

нимизации ошибки перепроецирования для всех сопоставленных пар точек, часто с использованием робастных методов, таких как RANSAC (Random Sample Consensus), для исключения выбросов (неверно сопоставленных точек или точек, принадлежащих движущимся объектам).

Преимущества методов на основе признаков заключаются в их относительной устойчивости к изменениям освещения. Они способны оценивать сложные модели движения, включая перспективные искажения.

Недостатки включают высокую вычислительную сложность этапов детектирования и описания признаков (особенно для SIFT и SURF), что может быть критично для систем реального времени. Качество оценки движения сильно зависит от наличия достаточного количества текстуры на изображении; на однородных или слабо текстурированных участках (например, небо) количество надежных признаков может быть невелико. Алгоритмы также чувствительны к наличию большого числа независимых движущихся объектов в сцене, которые могут быть ошибочно приняты за часть глобального движения камеры.

4.3.2 Методы, основанные на глобальном преобразовании (прямые методы)

Альтернативный подход к оценке движения, известный как прямые методы, не требует явного выделения и сопоставления признаков. Вместо этого, параметры модели движения оцениваются непосредственно путем минимизации некоторой метрики расхождения между интенсивностями пикселей в последовательных кадрах после применения предполагаемого преобразования. Например, если используется модель яркостной константности, предполагается, что интенсивность пикселя $I(x, y)$ остается неизменной после смещения: $I(x, y, t) \approx I(x - u, y - v, t - 1)$, где (u, v) – компоненты вектора смещения. Для более сложных моделей движения, таких как аффинное преобразование, задаваемое матрицей $\mathbf{A}$ и вектором смещения $\mathbf{b}$, задача сводится к минимизации функции ошибки вида:

$$E(\mathbf{A}, \mathbf{b}) = \sum_{(x,y)\in\Omega} \left(I_t(x, y) - I_{t-1}(\mathbf{A} \begin{bmatrix} x \\ y \end{bmatrix} + \mathbf{b}) \right)^2 \tag{4.4}$$

где Ω – область изображения. Минимизация обычно выполняется итерационными методами, такими как алгоритм Лукаса-Канаде или его вариации,

часто с использованием иерархического (многомасштабного) подхода для повышения устойчивости и скорости сходимости.

Преимущества прямых методов заключаются в том, что они используют всю информацию об интенсивностях пикселей, а не только информацию из окрестностей ключевых точек, что может приводить к более точной оценке движения на слабо текстурированных участках. Они могут быть вычислительно менее затратными, чем методы на основе признаков, если не используются сложные итерационные схемы.

Недостатки прямых методов включают их чувствительность к изменениям освещения между кадрами, так как предположение о яркостной константности нарушается. Они также могут плохо работать при больших смещениях между кадрами, если не используется иерархический подход. Устойчивость к выбросам ниже, чем у методов, использующих RANSAC.

4.3.3 Методы, использующие фильтрацию движения

Независимо от способа оценки глобального движения $\mathbf{M}_t$ между кадрами, полученная траектория движения камеры $\mathcal{T} = \{\mathbf{M}_1, \mathbf{M}_2, \ldots, \mathbf{M}_N\}$ обычно содержит как намеренное движение оператора (например, панорамирование, наклон), так и нежелательное высокочастотное дрожание. Задача фильтрации движения — разделить эти два компонента и сгенерировать новую, сглаженную траекторию $\mathcal{T}' = \{\mathbf{M}'_1, \mathbf{M}'_2, \ldots, \mathbf{M}'_N\}$, которая сохраняет намеренное движение, но удаляет дрожание. Одним из распространенных подходов является использование фильтра нижних частот (ФНЧ) на параметрах траектории движения. Например, если движение описывается параметрами смещения (dx_t, dy_t) и углом поворота $d\theta_t$, то к этим временным рядам можно применить скользящее среднее, медианный фильтр или более сложные фильтры, такие как фильтр Гаусса. Более продвинутым методом является использование фильтра Калмана . Фильтр Калмана — это рекурсивный алгоритм, который оценивает состояние динамической системы по зашумленным измерениям. В контексте стабилизации, состоянием системы могут быть параметры сглаженного движения камеры, а измерениями — параметры движения, оцененные на предыдущем этапе. Фильтр Калмана позволяет моделировать динамику намеренного движения и предсказывать его, одновременно фильтруя высокочастотный шум (дрожание). Модель системы для фильтра

Калмана может быть представлена в виде:

$$\mathbf{x}_k = \mathbf{F}_k \mathbf{x}_{k-1} + \mathbf{B}_k \mathbf{u}_k + \mathbf{w}_k \quad \text{(уравнение состояния)} \tag{4.5}$$

$$\mathbf{z}_k = \mathbf{H}_k \mathbf{x}_k + \mathbf{v}_k \quad \text{(уравнение измерения)} \tag{4.6}$$

где $\mathbf{x}_k$ — вектор состояния в момент k, $\mathbf{F}_k$ — матрица перехода состояния, $\mathbf{B}_k$ — матрица управления, $\mathbf{u}_k$ — вектор управления, $\mathbf{w}_k$ — шум процесса, $\mathbf{z}_k$ — вектор измерений, $\mathbf{H}_k$ — матрица наблюдения, $\mathbf{v}_k$ — шум измерения.

Преимущества использования фильтра Калмана заключаются в его способности оптимально оценивать состояние системы и адаптироваться к изменяющейся динамике движения. Он хорошо справляется с разделением плавного намеренного движения и случайного дрожания.

Недостатки — необходимость точного задания модели системы (матриц $\mathbf{F}, \mathbf{H}$) и ковариационных матриц шумов процесса и измерений ($\mathbf{Q}, \mathbf{R}$), что может быть нетривиальной задачей. Некорректная настройка фильтра может привести к чрезмерному сглаживанию (потере резкости намеренных движений) или недостаточному подавлению дрожания. После получения сглаженной траектории $\mathcal{T}'$, компенсационное преобразование для каждого кадра I_t вычисляется как $\mathbf{C}_t = \mathbf{M}'_t \cdot \mathbf{M}_t^{-1}$ и применяется к кадру для его стабилизации. Выбор конкретного алгоритма или комбинации алгоритмов на каждом этапе EIS зависит от требований к качеству стабилизации, вычислительным ресурсам и характеристикам входного видео.

4.4 Реализация алгоритма программной стабилизации видео на C++ с использованием OpenCV

В рамках данного исследования была предпринята попытка реализации алгоритма программной стабилизации видеопотока с использованием языка программирования C++ и библиотеки компьютерного зрения OpenCV. Выбранный подход базируется на методах отслеживания ключевых точек и последующей фильтрации траектории движения камеры для генерации компенсирующих преобразований. Алгоритм состоит из нескольких ключевых этапов, реализованных в виде последовательности операций над видеокадрами.

Первым этапом является оценка меж кадрового движения. Для этого извлекается текущий кадр I_t и предыдущий кадр I_{t-1} из видеопотока. Оба кад-

ра преобразуются в полутоновое представление для упрощения дальнейшей обработки и снижения вычислительной нагрузки. На предыдущем кадре I_{t-1} производится поиск ключевых точек с помощью функции goodFeaturesToTrack. Данная функция реализует алгоритм Ши-Томаси, который находит углы на изображении, являющиеся хорошими кандидатами для отслеживания. Для повышения устойчивости и фокусировки на значимых областях, поиск точек может быть ограничен определенной областью интереса (ROI) и/или предварен применением детектора границ Кэнни для выделения контуров, на которых чаще всего располагаются информативные признаки. В реализованном коде используется детектор Кэнни перед goodFeaturesToTrack, а также задается маска ROI для ограничения области поиска.

Найденные на предыдущем кадре ключевые точки $\mathcal{P}_{t-1}$ отслеживаются на текущем кадре I_t с использованием пирамидального алгоритма Лукаса-Канаде для вычисления оптического потока. Этот метод оценивает смещение каждой ключевой точки, предполагая яркостную константность и локальную гладкость поля смещений. В результате для каждой успешно отслеженной точки из $\mathcal{P}_{t-1}$ получается ее новое положение $\mathcal{P}_t$ на текущем кадре. Точки, для которых отслеживание не удалось (например, из-за окклюзии или выхода за пределы кадра), отфильтровываются на основе статуса, возвращаемого функцией.

На основе наборов соответствующих точек $(\mathcal{P}_{t-1}, \mathcal{P}_t)$ вычисляется матрица жесткого (евклидова) преобразования $\mathbf{T}_t$, описывающая глобальное движение камеры от кадра I_{t-1} к кадру I_t. Для этого используется функция estimateRigidTransform. Данное преобразование включает смещения по осям dx_t, dy_t и угол поворота da_t:

$$\mathbf{T}_t = \begin{bmatrix} \cos(da_t) & -\sin(da_t) & dx_t \\ \sin(da_t) & \cos(da_t) & dy_t \end{bmatrix} \tag{4.7}$$

Параметры (dx_t, dy_t, da_t) сохраняются для каждого перехода между кадрами. В случае, если преобразование не может быть надежно оценено (например, из-за малого количества отслеженных точек), используется преобразование с предыдущего шага.

Далее, на основе последовательности межкадровых преобразований:

$$\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_{N-1} \tag{4.8}$$

Вычисляется накопленная траектория движения камеры. Для каждого кадра k параметры накопленного смещения (X_k, Y_k) и угла A_k рассчитываются как сумма соответствующих параметров от первого кадра до k-го: $X_k = \sum_{i=1}^{k} dx_i$, $Y_k = \sum_{i=1}^{k} dy_i$, $A_k = \sum_{i=1}^{k} da_i$. Эта исходная траектория (X_k, Y_k, A_k) затем подвергается сглаживанию. В реализованном коде для сглаживания применяется фильтр скользящего среднего с заданным радиусом окна. Для каждой точки k сглаженной траектории (X'_k, Y'_k, A'_k) ее значения вычисляются как среднее арифметическое значений исходной траектории в окне $[k - radius, k + radius]$.

После получения сглаженной траектории вычисляются параметры корректирующего преобразования для каждого кадра. Для i-го кадра разница между сглаженной и исходной траекторией ($\Delta X_i = X'_i - X_i, \Delta Y_i = Y'_i - Y_i, \Delta A_i = A'_i - A_i$) добавляется к исходному межкадровому преобразованию (dx_i, dy_i, da_i), формируя новое, "сглаженное"межкадровое преобразование (dx'_i, dy'_i, da'_i). Из этих параметров формируется итоговая матрица преобразования $\mathbf{T}'_i$, которая применяется к исходному i-му кадру с помощью аффинного преобразования warpAffine для получения стабилизированного кадра.

Наконец, для минимизации артефактов в виде черных полей по краям стабилизированного изображения, которые возникают из-за компенсационных смещений и поворотов, применяется небольшой масштабирующий зум с помощью функции warpAffine с матрицей масштабирования. В представленном коде этот этап реализован в функции fixBorder, которая применяет масштабирование с коэффициентом 1.04.

Данный подход к программной стабилизации, основанный на отслеживании признаков Ши-Томаси, оценке евклидова преобразования и сглаживании траектории скользящим средним, представляет собой классическую схему EIS, реализованную с использованием стандартных функций библиотеки OpenCV.

4.5 Оценка вычислительной сложности реализованного алгоритма стабилизации и его применимости в системе реального времени

Анализ вычислительной сложности реализованного алгоритма программной стабилизации видео является ключевым для определения его пригодности к использованию в системах, требующих обработки видеопотока в реальном времени, особенно в связке с ресурсоемкими задачами, такими как детекция объектов с помощью нейронных сетей YOLO. Сложность предложенного алгоритма складывается из нескольких этапов, каждый из которых вносит свой вклад в общее время обработки кадра.

Наиболее вычислительно затратными операциями в цикле обработки каждого кадра являются:

1. **Детектирование ключевых точек (goodFeaturesToTrack с предобработкой Canny):** Сложность детектора границ Кэнни зависит от размера изображения и включает операции свертки, вычисления градиентов и пороговой обработки. Детектор Ши-Томаси также требует вычисления производных и анализа собственных значений матрицы градиентов в окрестности каждого пикселя. Суммарная сложность этих операций может быть оценена как $O(W \cdot H)$, где W, H – ширина и высота кадра, хотя константа пропорциональности может быть значительной.

2. **Вычисление оптического потока (calcOpticalFlowPyrLK):** Пирамидальный алгоритм Лукаса-Канаде является итеративным и применяется на нескольких уровнях пирамиды изображений. Его сложность зависит от количества отслеживаемых точек N_p, размера окна поиска и числа уровней пирамиды L. Приблизительно сложность можно оценить как $O(N_p \cdot K^2 \cdot L)$, где K – размер окна. В худшем случае, когда N_p пропорционально $W \cdot H$, этот этап также может быть очень ресурсоемким.

3. **Оценка матрицы преобразования (estimateRigidTransform):** Если используется метод наименьших квадратов после RANSAC, то сложность RANSAC зависит от числа итераций N_{iter} и числа точек k, выбираемых на каждой итерации. В худшем случае это может быть $O(N_{iter} \cdot N_p)$.

4. **Аффинное преобразование кадра (warpAffine):** Эта операция требует интерполяции значений пикселей для каждого пикселя выходного

изображения, что приводит к сложности порядка $O(W \cdot H)$.

Операции сглаживания траектории выполняются один раз после обработки всех меж кадровых преобразований и их сложность $O(N_{frames} \cdot radius)$ распределяется на весь видеоряд, поэтому на покадровую обработку в реальном времени влияют меньше, чем перечисленные выше этапы. Однако сам этап накопления трансформаций и последующего их применения требует двукратного прохода по видеоданным (или буферизации значительного числа кадров/трансформаций), что вносит системную задержку, равную как минимум длине окна сглаживания.

Суммируя, покадровая вычислительная нагрузка алгоритма является значительной. На стандартных CPU, без специализированных аппаратных ускорителей, обработка видео с разрешением Full HD (1920x1080) или даже HD (1280x720) с использованием данного набора функций OpenCV вряд ли сможет достичь производительности, необходимой для реального времени (25-30 кадров в секунду), особенно если на том же вычислительном узле будет запущен модуль нейросетевой модели YOLO. Модели YOLO сами по себе требуют значительных вычислительных ресурсов, особенно на CPU. Параллельное выполнение столь ресурсоемкого алгоритма стабилизации и детектора YOLO на одном CPU приведет к существенному падению общей производительности системы, увеличению задержек и невозможности обработки потока в реальном времени.

Применение масштабирования в функции fixBorder с фиксированным коэффициентом 1.04 является простым способом скрыть черные поля, но это приводит к постоянной потере части информации по краям кадра, независимо от реальной амплитуды дрожания. Адаптивный кроппинг мог бы быть более эффективным, но и более сложным в реализации.

Учитывая вышеизложенное, реализованный алгоритм программной стабилизации, несмотря на свою концептуальную ясность и использование стандартных инструментов, является слишком вычислительно трудозатратным для интеграции в систему обнаружения ЛА на базе YOLO, предназначенную для работы в реальном времени, если не предполагается использование мощных выделенных GPU для обеих задач или существенная оптимизация и перенос части вычислений на аппаратные ускорители. Для практического применения в целевой системе потребовались бы либо значительно более легковесные алгоритмы EIS, либо использование аппаратных методов ста-

билизации (OIS или механической), либо применение данного алгоритма в режиме оффлайн-обработки видео.

4.6 Выводы

В рамках настоящей главы было проведено исследование методов стабилизации видеопотока с целью повышения качества входных данных для систем компьютерного зрения, в частности, для задачи обнаружения летательных аппаратов. Анализ показал, что нестабильность видеоряда, вызванная дрожанием или вибрациями камеры, существенно снижает эффективность алгоритмов детекции, приводя к размытию объектов, ошибкам в оценке движения и увеличению числа ложных срабатываний.

Были рассмотрены три основных класса методов стабилизации: механический, оптический (OIS) и программный (EIS). Механическая стабилизация, реализуемая с помощью гироподвесов, обеспечивает наилучшее качество за счет физической компенсации движений, однако сопряжена с увеличением габаритов, веса и стоимости системы. Оптическая стабилизация, интегрированная в объектив или сенсор, является эффективным компромиссом для компактных устройств, но имеет ограниченный диапазон компенсации. Программная стабилизация, основанная на цифровой обработке кадров, не требует дополнительного оборудования, но ее эффективность зависит от сложности применяемых алгоритмов и может приводить к потере разрешения из-за кроппинга или вносить артефакты при некорректной работе.

В ходе практической части исследования был реализован алгоритм программной стабилизации видео на языке C++ с использованием библиотеки OpenCV.

Однако ключевым выводом, сделанным по результатам анализа и практической реализации, является значительная вычислительная сложность большинства эффективных алгоритмов программной стабилизации, особенно тех, что основаны на поиске и сопоставлении большого числа признаков. При текущем уровне реализации, разработанный алгоритм стабилизации требует существенных вычислительных ресурсов, что делает его интеграцию в систему обнаружения летательных аппаратов в режиме реального времени, работающую параллельно с ресурсоемкой нейросетевой моделью YOLO, проблематичной без применения специализированного аппаратного ускорения или значительной алгоритмической оптимизации.

ГЛАВА 5
РЕАЛИЗАЦИЯ И ТЕСТИРОВАНИЕ ИНТЕГРИРОВАННОЙ СИСТЕМЫ

В настоящей главе описывается процесс практической реализации и экспериментального исследования интегрированной системы обнаружения летательных аппаратов. Система объединяет разработанный модуль видеотрансляции по протоколу RTSP, серверную часть для анализа видеопотока с использованием предварительно обученной модели YOLOv11 и простейший пользовательский интерфейс для захвата видеопотока. Особое внимание уделяется методике тестирования и анализу полученных результатов.

5.1 Описание среды разработки и используемых инструментов

Разработка и тестирование прототипа интегрированной системы обнаружения летательных аппаратов проводились на персональном компьютере под управлением операционной системы Windows 10. Аппаратная конфигурация включала: центральный процессор AMD Ryzen 7 7700, 32 ГБ оперативной памяти DDR5 и графический ускоритель NVIDIA GeForce RTX 4070 Ti SUPER с 16 ГБ видеопамяти. Данная конфигурация была выбрана для обеспечения достаточной вычислительной мощности как для обработки видеопотоков, так и для выполнения ресурсоемких операций нейросетевого инференса модели YOLOv11 в режиме, приближенном к реальному времени.

В качестве основного языка программирования для серверной части, отвечающей за прием видеопотока и его анализ с помощью нейронной сети, был выбран **Python 3.11**. Широкий набор библиотек для компьютерного зрения, машинного обучения и сетевого взаимодействия делает Python предпочтительным выбором для быстрого прототипирования подобных систем. Ключевыми библиотеками, использованными в серверной части, являются:

- **OpenCV (cv2)** (версия 4.11): для приема и декодирования RTSP-потока, предварительной обработки кадров (изменение размера, нормализация) и визуализации результатов детекции (наложение ограничивающих рамок и меток классов).

- **PyTorch** (версия 2.5.1): фреймворк глубокого обучения, использованный для загрузки предварительно обученной модели YOLOv11 и выполнения инференса на графическом процессоре с использованием технологии CUDA.

- Библиотеки для работы с конфигурационными файлами и логирования.

Для разработки пользовательского интерфейса (UI), предназначенного для отображения обрабатываемого видеопотока и результатов детекции, был использован кроссплатформенный фреймворк **Flutter** (версия 3.29.0) на языке программирования **Dart**. Выбор Flutter обусловлен его способностью создавать нативные приложения для различных платформ из единой кодовой базы и наличием плагинов для работы с видео.

Учитывая сложность организации реальной трансляции видео с ЛА в лабораторных условиях, для имитации RTSP-видеопотока был использован медиасервер **MediaMTX**. MediaMTX представляет собой готовый к использованию, высокопроизводительный RTSP/RTMP/HLS сервер, позволяющий публиковать видеофайлы или потоки с других источников в виде RTSP-трансляции. Это позволило тестировать систему на разнообразных видеозаписях из открытых источников.

Разработка велась в интегрированной среде разработки **Visual Studio Code** с использованием соответствующих расширений для Python и Flutter. Система контроля версий – **Git**. Для управления зависимостями Python использовался менеджер пакетов `pip` и виртуальные окружения (`venv`).

5.2 Описание данных

Для обучения и тестирования моделей был использован размеченный с использованием аннотаций, содержащих координаты ограничивающих прямоугольников (bounding boxes), публичный датасет, состоящий из изображений, разделённых на три класса:

- Самолёт("AirPlane")

- Дрон("Drone")

- Вертолеты("Helicopter")

Также в датасете присутствуют пустые изображения, на которых отсутствуют интересующие объекты (самолёты, дроны или вертолеты). К ним относятся фотографии чистого неба, облаков, горизонта и других природных пейзажей. Этои изображения важны для минимизации ложных срабатываний модели, так как она должна правильно различать фон и целевые объекты.

5.3 Обучение модели и анализ результатов

В рамках реализации системы распознования беспилотных летательных объектов была обучена модель YOLOv11, которая является одной из самых современных архитектур семейства YOLO (You Only Look Once). Эта модель разработана специально для задач, требующих высокой скорости детекции объектов при сохранении высокой точности, что делает её идеальным выбором для работы в реальном времени.

Модель показала высокую производительность на тестовых данных и смогла обрабатывать изображения практически без задержеки, о чем свидетельствует рисунок 5.1.

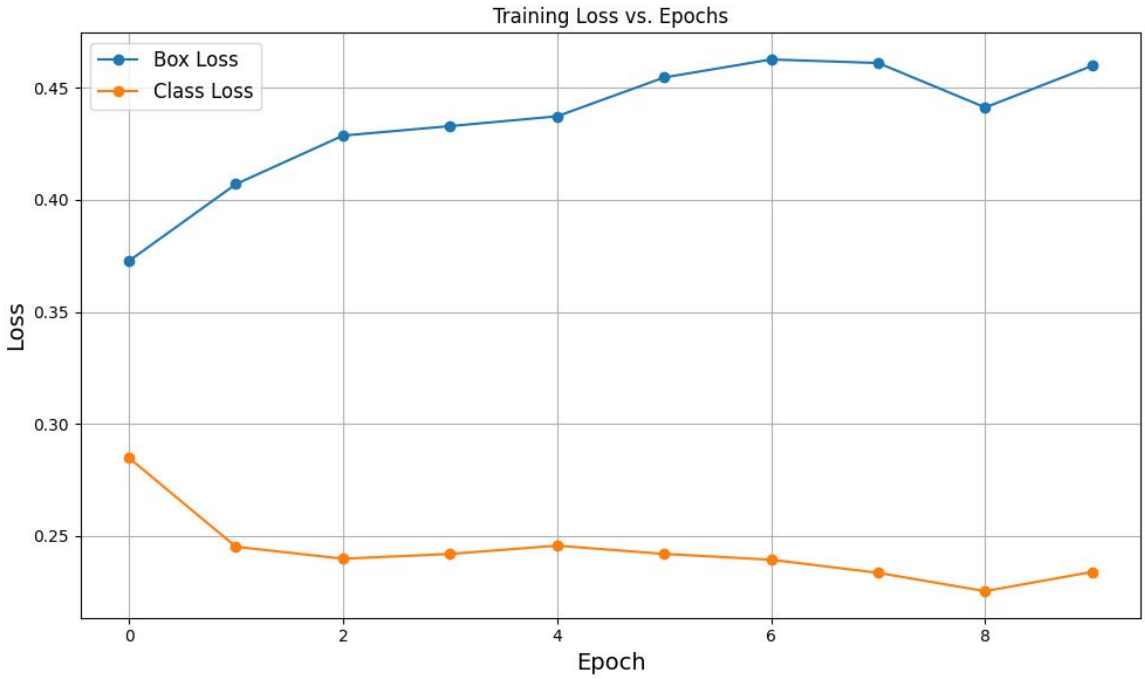

Рисунок 5.1 График потерь для YOLOv11

5.3.1 Precision-Confidence Curve (зависимость точности предсказаний от уровня уверенности модели)

График Precision-Confidence Curve показывает, что модель демонстрирует хорошую производительность в целом, особенно для классов "AirPlane"и "Helicopter". Однако класс "Drone"выделяется как проблемный. Это может быть связано с внешней схожестью объектов класса "Drone"с объектами других классов и с их размером. Это может потребовать дополнительных улучшений, к примеру добавления данных для этого класса.

При уровне уверенности 0.81 модель достигает максимальной точности для всех классов, что может быть оптимальным значением для выбора порога уверенности при использовании этой модели на практике.

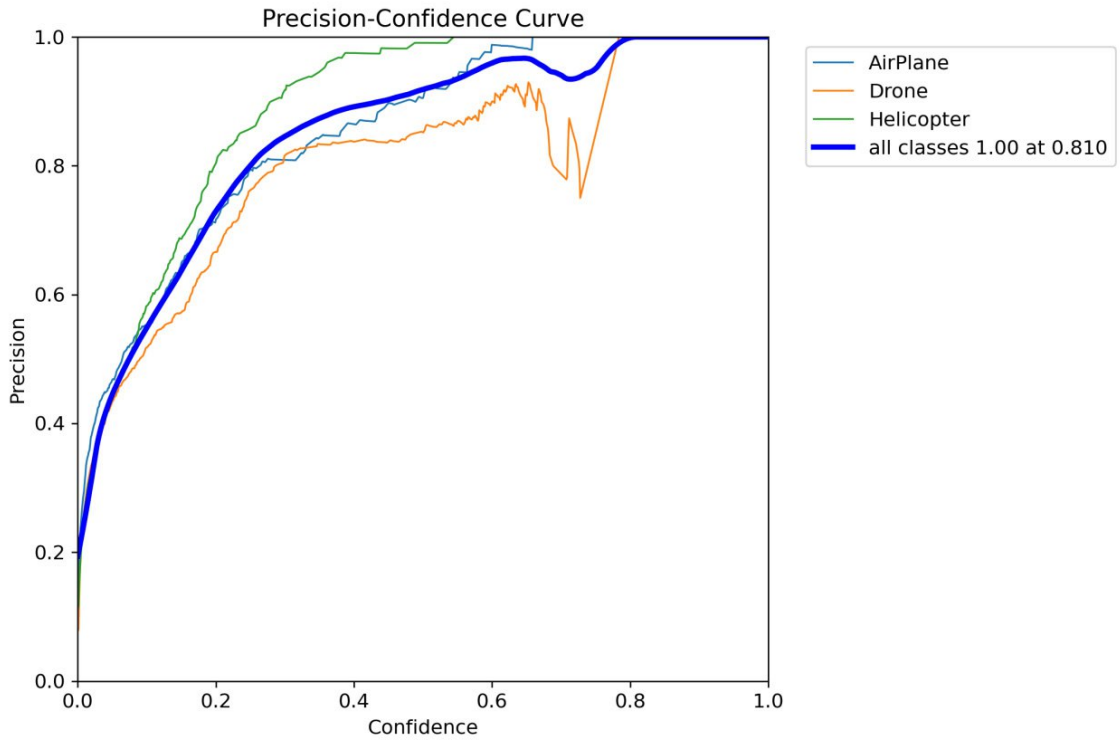

Рисунок 5.2 Precision-Confidence Curve

5.3.2 Precision-Recall Curve (зависимость точности от доли правильно предсказанных объектов среди всех истинных объектов)

График Precision-Recall Curve показывает, что модель демонстрирует высокую эффективность, с усреднённой точностью 0.896. Для всех классов точность резко падает при высоких значениях полноты (близких к 1.0). Это

связано с тем, что для достижения высокой полноты модель начинает захватывать больше объектов, включая ложные срабатывания. Наилучшая производительность наблюдается для классов "Helicopter"и "AirPlane". Однако класс "Drone"остаётся проблемным, со значительными потерями точности на высоких значениях полноты.

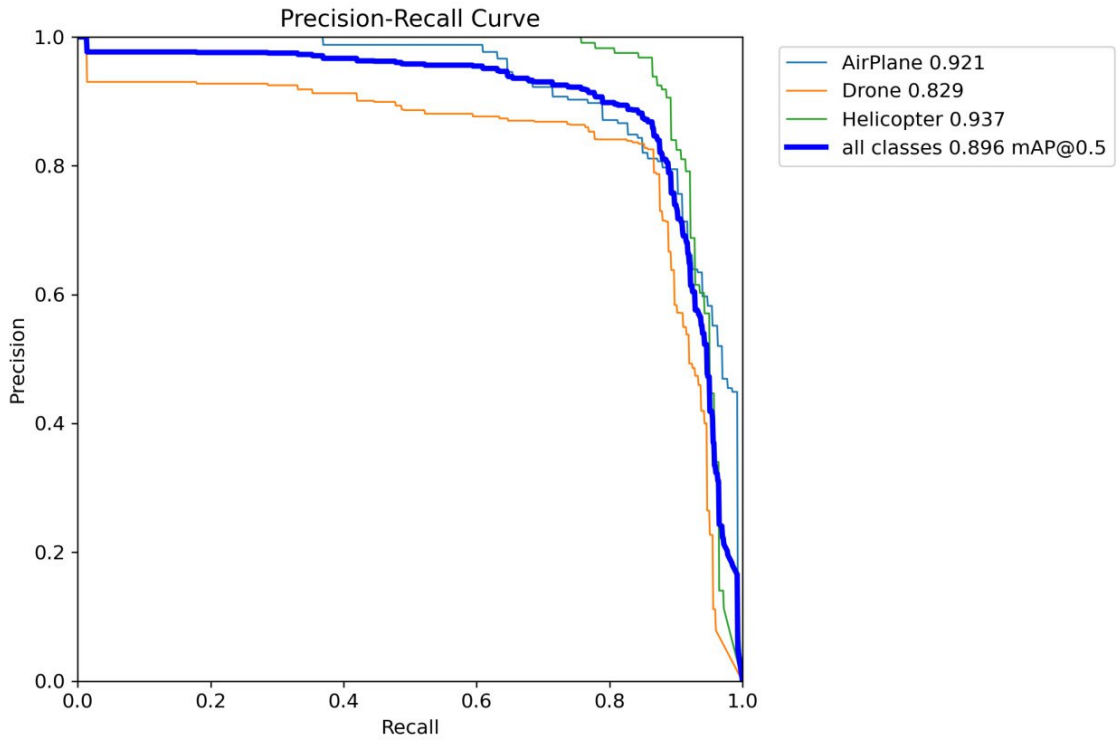

Рисунок 5.3 Recall-Confidence Curve

5.4 Реализация модуля сервера видеотрансляции и анализа на базе RTSP и YOLOv11

Серверная часть интегрированной системы выполняет две ключевые функции: прием и обработку видеопотока, транслируемого по протоколу RTSP, и последующий анализ полученных кадров с помощью нейросетевой модели YOLOv11 для детекции летательных аппаратов. Данный модуль был реализован на языке Python с активным использованием библиотеки OpenCV для работы с видео и фреймворка PyTorch для выполнения нейросетевого инференса.

Процесс работы серверного модуля можно разделить на следующие этапы:

1. **Инициализация и подключение к RTSP-потоку:** При запуске сервер инициализирует объект VideoCapture с URL-адресом RTSP-потока,

предоставляемого медиасервером MediaMTX. Осуществляется попытка подключения к потоку и проверка его доступности. В случае неудачи предпринимаются повторные попытки подключения с определенным интервалом.

2. **Захват и декодирование кадров:** В цикле происходит чтение кадров из видеопотока с помощью метода read(). Полученные кадры представляют собой матрицы NumPy, содержащие пиксельные данные изображения. Скорость захвата кадров (FPS) определяется параметрами исходного потока и производительностью системы.

3. **Предварительная обработка кадра:** Перед подачей на вход нейронной сети каждый кадр подвергается необходимой предварительной обработке. Это включает изменение размера кадра до разрешения, ожидаемого моделью YOLOv11 (640x640 пикселей) и нормализацию значений пикселей (деление на 255.0 для приведения к диапазону [0, 1]). Эти операции выполняются с использованием функций OpenCV.

4. **Инференс модели YOLOv11:** Предварительно обработанный кадр преобразуется в тензор PyTorch и передается на вход загруженной модели YOLOv11, работающей в режиме инференса. Для ускорения вычислений инференс выполняется на графическом процессоре NVIDIA RTX 4070 Ti SUPER с использованием CUDA. Модель возвращает набор предсказаний, который обычно включает координаты ограничивающих рамок (bounding boxes) обнаруженных объектов, уверенность (confidence score) для каждой рамки и класс объекта.

5. **Постобработка результатов детекции:** Выходные данные модели подвергаются постобработке, включающей применение порога уверенности (confidence threshold) для отсеивания предсказаний с низкой уверенностью и использование алгоритма подавления немаксимумов (Non-Maximum Suppression, NMS) для устранения дублирующихся рамок, относящихся к одному и тому же объекту. В результате остаются только наиболее релевантные детекции.

Для обеспечения возможности обработки потока в реальном времени или с минимальной задержкой, особое внимание уделялось оптимизации каждого этапа, особенно инференса нейросети. Использование аппаратного ускорения

GPU является здесь ключевым фактором. Архитектура сервера спроектирована таким образом, чтобы быть модульной, позволяя в будущем легко заменять модель детекции или добавлять новые этапы обработки.

5.5 Реализация клиентского модуля для захвата и передачи видеопотока на базе Flutter

В рамках данной дипломной работы, для обеспечения гибкости источника видеосигнала и возможности использования мобильных устройств в качестве камер, был разработан клиентский модуль на фреймворке Flutter. Основная задача данного модуля — захват видео с камеры устройства, его кодирование и последующая трансляция по протоколу RTSP на медиасервер. Реализация клиентского издателя (publisher) включает следующие ключевые аспекты:

1. **Захват видео с камеры устройства:** Используется официальный плагин camera для Flutter, который предоставляет API для доступа к камере устройства, выбора разрешения и частоты кадров. Полученные сырые видеокадры становятся основой для дальнейшей обработки.

2. **Кодирование видео в формат H.264:** Поскольку Dart не имеет встроенных эффективных видеокодеров, эта задача делегируется нативному коду платформы (Android/iOS) через механизм Platform Channels. Критически важным является использование аппаратного ускорения для минимизации нагрузки на CPU мобильного устройства и обеспечения работы в реальном времени.

3. **Пакетизация в RTP и формирование RTSP-потока:** Закодированный H.264 поток пакетизируется в RTP-пакеты. Далее, для публикации по RTSP, клиент реализует логику RTSP-издателя, используя методы ANNOUNCE и RECORD для установления сессии с RTSP-сервером и передачи RTP-пакетов

4. **Пользовательский интерфейс:** Реализован минимальный UI для запуска/остановки трансляции и указания URL-адреса RTSP-сервера для публикации.

5.6 Экспериментальное исследование и тестирование системы

Экспериментальное исследование и тестирование разработанной системы обнаружения летательных аппаратов проводилось с целью оценки ее работоспособности, производительности и точности детекции в условиях, имитирующих реальные. Тестирование включало проверку как отдельных модулей (видеотрансляция, нейросетевой анализ), так и интегрированной системы в целом.

Подготовка тестовых видеоданных и среды тестирования. Ввиду отсутствия возможности проведения натурных испытаний с реальными ЛА и организации прямой трансляции в контролируемых условиях, для тестирования использовался набор видеозаписей из открытых источников. Были подобраны видео различного качества, с разными типами ЛА (квадрокоптеры, самолеты), в различных погодных условиях и на разном фоне. Эти видеофайлы публиковались через медиасервер MediaMTX в виде RTSP-потоков с разрешением 1920x1080 (Full HD) и частотой 25-30 кадров в секунду. Серверная часть запускалась на описанном ранее тестовом стенде.

Тестирование модуля видеотрансляции и приема потока. На данном этапе проверялась стабильность приема RTSP-потока Python-сервером с использованием OpenCV. Оценивалась задержка между публикацией потока на MediaMTX и его получением сервером. В большинстве случаев задержка была минимальной (менее 1 секунды), что приемлемо для задач реального времени.

Тестирование модуля детекции YOLOv11. Основное внимание было уделено оценке качества и скорости работы модели YOLOv11. Тестирование проводилось на различных видеопоследовательностях.

Качество детекции: На видеозаписях с хорошей видимостью и относительно крупными объектами ЛА система демонстрировала удовлетворительные результаты, корректно обнаруживая и классифицируя большинство летательных аппаратов с высокой степенью уверенности. Однако на видео с малоразмерными, удаленными объектами, объектами на сложном фоне или в условиях плохой освещенности наблюдались пропуски детекции и ложные срабатывания, когда за ЛА принимались другие объекты. Средняя точность на доступном наборе тестовых видео не измерялась количественно из-за отсутствия точной разметки для всех видео, оценка носила преимущественно

качественный характер.

Скорость детекции: На графическом процессоре NVIDIA RTX 4070 Ti SUPER серверная часть, включая захват кадра, предобработку, инференс YOLOv11 и постобработку, достигала средней скорости обработки около 30-45 кадров в секунду для входного потока Full HD, что позволяет обрабатывать видео в реальном времени. При этом сам инференс YOLOv11 занимал порядка 15-25 мс на кадр.

Анализ влияния стабилизации видео (офлайн). Дополнительно было исследовано влияние предварительной программной стабилизации на качество детекции. Тестовые видео, содержащие заметное дрожание, были стабилизированы офлайн с использованием разработанного ранее алгоритма на C++ и OpenCV. Затем стабилизированные видео подавались на вход системы детекции. Было отмечено, что на стабилизированных видео количество ложных срабатываний несколько уменьшилось, а контуры объектов стали более четкими, что потенциально улучшало локализацию. Однако, как и ожидалось, процесс стабилизации с использованием методов на основе отслеживания признаков оказался ресурсоемким и не позволил интегрировать его в контур реального времени без существенной оптимизации или использования специализированного аппаратного ускорения. Кроме того, артефакты стабилизации, такие как черные обрезанные края, вносили свои особенности. Для полноценного использования преимуществ стабилизации, нейросетевую модель YOLOv11 целесообразно было бы дообучить на датасете, включающем как оригинальные, так и стабилизированные кадры с характерными артефактами обрезки.

5.7 Выводы по результатам тестирования и пути улучшения.

Реализованный прототип системы продемонстрировал базовую функциональность по обнаружению ЛА в видеопотоке. Удовлетворительная производительность была достигнута за счет использования GPU для нейросетевого инференса. Основными направлениями для улучшения системы являются:

- **Повышение устойчивости модели YOLOv11:** Дообучение модели на более репрезентативном и разнообразном датасете, включающем малоразмерные объекты, сложные фоны, различные погодные условия, а

также аугментация данных, имитирующая различные виды искажений и стабилизированные кадры.

- **Разработка системы сигнализации:** Интеграция модуля оповещения о детекции ЛА, например, через отправку уведомлений в Telegram-бот или через другой канал связи, что повысит практическую ценность системы.

- **Оптимизация или выбор другого алгоритма стабилизации:** Для работы в реальном времени требуется либо существенная оптимизация реализованного алгоритма стабилизации, либо использование более легковесных методов, либо применение аппаратных решений для стабилизации.

- **Разработка более функционального UI:** Расширение возможностей клиентского приложения, например, для настройки параметров детекции.

Несмотря на выявленные ограничения, полученные результаты подтверждают перспективность использования выбранного стека технологий для создания систем обнаружения ЛА.

ЗАКЛЮЧЕНИЕ

В ходе выполнения настоящей дипломной работы была успешно решена задача исследования и разработки прототипа системы обнаружения летательных аппаратов с использованием технологий искусственного интеллекта и современных методов передачи видеоданных. Были достигнуты все поставленные цели и выполнены основные задачи исследования.

В рамках работы был проведен детальный анализ требований к подобным системам, что позволило сформулировать как функциональные, так и нефункциональные критерии для разрабатываемого прототипа. На основе этого анализа был выполнен обзор существующих технологий и протоколов для видеотрансляции в реальном времени.

Значительное внимание было уделено теоретическому исследованию методов детекции объектов на основе глубоких нейронных сетей. Был проведен обзор современных подходов, с особым акцентом на семейство моделей YOLO, в частности, на предполагаемую к использованию версию YOLOv11. Рассмотрены принципы работы данных моделей, их архитектурные особенности и метрики оценки качества.

Отдельным важным аспектом исследования стала проблема стабилизации видеопотока, поскольку качество входных данных напрямую влияет на эффективность работы нейросетевых детекторов, особенно при работе с видео с подвижных платформ. Был проведен обзор существующих методов стабилизации – механических, оптических и программных – с анализом их преимуществ и недостатков. В рамках практической части был реализован и исследован алгоритм программной стабилизации видео на языке C++ с использованием библиотеки OpenCV. Анализ показал, что, хотя программная стабилизация способна улучшить качество видео для последующего анализа, ее вычислительная сложность может быть существенным ограничением для применения в системах реального времени без значительной оптимизации или аппаратной поддержки.

Практическая реализация интегрированной системы включала разработку серверной части на Python для приема RTSP-потока, его анализа с помощью предварительно обученной модели YOLOv11, и клиентского модуля на Flutter, который в одном из исследуемых сценариев выступал в качестве источника видеопотока, транслируя его по RTSP. Для имитации реальных

условий и тестирования на разнообразных видеоданных использовался медиасервер MediaMTX.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Гонсалес, Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс ; пер. с англ. – 3-е изд. – Москва : Техносфера, 2012. – 1104 с. – ISBN 978-5-94836-327-8.

2. Форсайт, Д. А. Компьютерное зрение. Современный подход / Д. А. Форсайт, Ж. Понс ; пер. с англ. – Москва : Вильямс, 2004. – 928 с. – ISBN 5-8459-0542-7.

3. Battiato, S. Image/Video Stabilization / S. Battiato, S. Curti, M. La Cascia, M. Tortora, F. Scordato // Digital Signal Processing for Multimedia Systems / Ed. by M. Vetterli, J. Kovačević. – Boca Raton : CRC Press, 2007. – P. 325–348. – ISBN 978-0849373568.

4. Bay, H. SURF: Speeded Up Robust Features / H. Bay, A. Ess, T. Tuytelaars, L. Van Gool // Computer Vision and Image Understanding. – 2008. – Vol. 110, № 3. – P. 346–359. – DOI: 10.1016/j.cviu.2007.09.014.

5. Bochkovskiy, A. YOLOv4: Optimal Speed and Accuracy of Object Detection / A. Bochkovskiy, C.-Y. Wang, H.-Y. M. Liao // arXiv preprint arXiv:2004.10934. – 2020. – 20 p. – URL: `https://arxiv.org/abs/2004.10934` (дата обращения: 21.03.2025).

6. Bradski, G. The OpenCV Library / G. Bradski // Dr. Dobb's Journal of Software Tools. – 2000. – Vol. 25, № 11. – P. 120–125.

7. Cheng, K.-Y. Real-time video stabilization for a wide-angle camera on a COTS-based UAV / K.-Y. Cheng, C.-S. Chen, C.-C. Lin // IEEE Transactions on Instrumentation and Measurement. – 2007. – Vol. 56, № 5. – P. 1631–1640. – DOI: 10.1109/TIM.2007.904567.

8. Flutter: Build apps for any screen [Электронный ресурс]. – URL: `https://flutter.dev/` (дата обращения: 08.04.2025).

9. GStreamer: Open Source Multimedia Framework [Электронный ресурс]. – URL: `https://gstreamer.freedesktop.org/` (дата обращения: 15.04.2025).

10. Johnston, A. B. WebRTC: APIs and RTCWEB Protocols of the HTML5 Real-Time Web / A. B. Johnston, D. C. Burnett. – [Place of publication not identified] : Digital Codex LLC, 2013. – 328 p. – ISBN 978-0985081626.

11. Kalman, R. E. A New Approach to Linear Filtering and Prediction Problems / R. E. Kalman // Transactions of the ASME – Journal of Basic Engineering. – 1960. – Vol. 82, Series D, № 1. – P. 35–45. – DOI: 10.1115/1.3662552.

12. Lee, K. Challenges in Visual Detection of Small Unmanned Aerial Vehicles: A Review / K. Lee, S. Kim // IEEE Access. – 2021. – Vol. 9. – P. 75310–75325. – DOI: 10.1109/ACCESS.2021.3081518.

13. Lowe, D. G. Distinctive image features from scale-invariant keypoints / D. G. Lowe // International Journal of Computer Vision. – 2004. – Vol. 60, № 2. – P. 91–110. – DOI: 10.1023/B:VISI.0000029664.99615.94.

14. MediaMTX (formerly rtsp-simple-server) [Электронный ресурс]. – URL: `https://github.com/bluenviron/mediamtx` (дата обращения: 20.03.2025).

15. Morimoto, C. Fast electronic digital image stabilization / C. Morimoto, R. Chellappa // Proceedings of 1998 International Conference on Pattern Recognition (ICPR), Brisbane, QLD, Australia, 16-20 August 1998. – IEEE, 1998. – Vol. 1. – P. 284–288. – DOI: 10.1109/ICPR.1998.711138.

16. Neubeck, A. Efficient non-maximum suppression / A. Neubeck, L. Van Gool // Proceedings of the 18th International Conference on Pattern Recognition (ICPR), Hong Kong, China, 20-24 August 2006. – IEEE, 2006. – Vol. 3. – P. 850–855. – DOI: 10.1109/ICPR.2006.485.

17. Redmon, J. You Only Look Once: Unified, Real-Time Object Detection / J. Redmon, S. Divvala, R. Girshick, A. Farhadi // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27-30 June 2016. – IEEE, 2016. – P. 779–788. – DOI: 10.1109/CVPR.2016.91.

18. Rublee, E. ORB: An efficient alternative to SIFT or SURF / E. Rublee, V. Rabaud, K. Konolige, G. Bradski // Proceedings of the IEEE International Conference on Computer Vision (ICCV), Barcelona, Spain, 6-13 November 2011. – IEEE, 2011. – P. 2564–2571. – DOI: 10.1109/ICCV.2011.6126544.

19. Schulzrinne, H. Real Time Streaming Protocol (RTSP) / H. Schulzrinne, A. Rao, R. Lanphier. – RFC 2326. – Internet Engineering Task Force, 1998. – 80 p. – URL: https://www.rfc-editor.org/info/rfc2326 (дата обращения: 21.03.2025).

20. Ultralytics YOLO [Электронный ресурс]. – URL: https://docs.ultralytics.com/ (дата обращения: 21.03.2025).

21. Wang, Y.-K. RTP Payload Format for H.264 Video / Y.-K. Wang, R. Even, T. Kristensen, R. Frost. – RFC 6184. – Internet Engineering Task Force, 2011. – 78 p. – URL: https://www.rfc-editor.org/info/rfc6184 (дата обращения: 21.03.2025).

22. Wiegand, T. Overview of the H.264/AVC video coding standard / T. Wiegand, G. J. Sullivan, G. Bjontegaard, A. Luthra // IEEE Transactions on Circuits and Systems for Video Technology. – 2003. – Vol. 13, № 7. – P. 560–576. – DOI: 10.1109/TCSVT.2003.815165.