МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра компьютерных технологий и систем

Филиппович Валерий Александрович

ДЕТЕКЦИЯ МЕЛКИХ ОБЪЕКТОВ НА ИЗОБРАЖЕНИЯХ ДРОНОВ

Дипломная работа

Научный руководитель:
Заведующий кафедрой информационных систем управления, доктор технических наук, доцент
А. М. Недзьведь

Допущена к защите
«____»_____20___г.
Заведующий кафедрой компьютерных технологий и систем
доктор педагогических наук, кандидат физико-математических наук, профессор В.В. Казачёнок

Минск, 2025

ОГЛАВЛЕНИЕ

РЕФЕРАТ

Дипломная работа, 53 страницы, 17 рисунков, 1 таблица, 12 источников.

Ключевые слова: ДЕТЕКЦИЯ ОБЪЕКТОВ, МЕЛКИЕ ОБЪЕКТЫ, DEEPSORT, СВЁРТОЧНАЯ СЕТЬ.

Объект исследования: изображения объектов, полученные с дронов.

Предмет исследования: методы детекции мелких объектов на изображениях дронов в статике и в движении.

Цель исследования: разработка и реализация алгоритма детекции мелких объектов.

Методы исследования: изучение существующих решений для детекции; разработка, реализация и тестирование алгоритма детекции мелких объектов.

Полученные результаты и их новизна: разработана и реализована модель детекции объектов в движении и в статике. Модель способна работать в различных условиях, включая движение камеры и маленький размер объектов

Достоверность материалов и результатов дипломной работы: все выводы в рамках данной работы опираются на факты, представленные в проверенных источниках, которые сопровождаются ссылками на соответствующую литературу. Результаты получены при помощи выполнения программного кода и могут быть воспроизведены.

Область возможного практического применения: видеонаблюдение, автономное вождение, анализ движения.

РЭФЕРАТ

Дыпломная праца, 53 старонкі, 17 малюнкаў, 1 табліца, 12 крыніц.

Ключавыя словы: ДЭТЭКЦЫЯ АБ’КТАУ, МАЛЕНЬКІЯ АБ’ЕКТЫ, DEEPSORT, КАНВАЛЮЦЫЙНАЯ СЕТКА.

Аб'ект даследавання: выявы аб'ектаў, атрыманыя з дронов.

Прадмет даследавання: метады дэтэкцыі дробных аб'ектаў на малюнках дронов ў статыцы і ў руху.

Мэта даследавання: распрацоўка і рэалізацыя алгарытму дэтэкцыі дробных аб'ектаў.

Метады даследавання: вывучэнне існуючых рашэнняў для дэтэкцыі; распрацоўка, рэалізацыя і тэставанне алгарытму дэтэкцыі дробных аб'ектаў.

Атрыманыя вынікі і іх навізна: распрацавана і рэалізавана мадэль дэтэкцыі аб'ектаў у руху і ў статыцы. Мадэль здольная працаваць у розных умовах, уключаючы рух камеры і маленькі памер аб'ектаў.

Дакладнасць матэрыялаў і вынікаў дыпломнай працы: усе высновы ў рамках дадзенай працы абапіраюцца на факты, прадстаўленыя ў праверaных крыніцах, якія суправаджаюцца спасылкамі на адпаведную літаратуру.

Вобласць магчымага практычнага прымянення: відэаназіранне, аўтаномнае кіраванне, аналіз руху.

ANNOTATION

Diploma work, 53 *pages*, 17 *figures*, 1 *table*, 12 *sources*.

Keywords: OBJECT DETECTION, SMALL OBJECTS, DEEPSORT, CONVOLUTIONAL NETWORK.

The object of the research: images of objects obtained from drones.

The subject of the research: methods of detecting small objects in drone images in static and in motion.

The aim of the research: to develop and implement an algorithm for detecting small objects.

Research methods: study of existing detection solutions; development, implementation and testing of an algorithm for detecting small objects. The results obtained and their novelty: a model for detecting objects in motion and in statics has been developed and implemented. The model is capable of operating in a variety of conditions, including camera movement and small object size.

The results of the work and their novelty: a model for detecting objects in motion and in statics has been developed and implemented. The model is capable of operating in a variety of conditions, including camera movement and small object size.

Authenticity of the materials and results of the diploma work: all conclusions in the framework of this work are based on facts presented in verified sources, which are accompanied by references to relevant literature. The results are obtained by executing the program code and can be reproduced.

Recommendations on the usage: video surveillance, autonomous driving, motion analysis.

ВВЕДЕНИЕ

Детекция объектов на последовательности изображений является активной исследовательской областью в компьютерном зрении и обработке изображений. Он играет важную роль во множестве приложений, таких как видеонаблюдение, автономное вождение, анализ движения и многих других. Основной задачей детекции является нахождение, отслеживание и последовательное определение положения и движения объектов на изображении или последовательности изображений.

Одной из ключевых сложностей детекции объектов является изменчивость объектов во времени и присутствие различных вызывающих факторов, таких как их размер, форма, освещение, появление и исчезновение, а также перекрытие другими объектами. В связи с этим детекция требует разработки эффективных алгоритмов и методов, которые способны справиться с этими вызовами и обеспечить точную и надежную детекцию и отслеживание объектов на изображении и их последовательности.

Данная работа посвящена исследованию и разработке методов и алгоритмов детекции объектов на изображениях, сделанных дронами на разных высотах. Она охватывает важные аспекты, такие как точность детекции, скорость обработки данных, адаптация к различным условиям освещения и фоновым шумам. Акцент делается на технологиях глубокого обучения, в частности, на нейронных сетях, как одном из самых перспективных подходов к решению задачи детекции объектов.

Целью настоящего проекта является исследование и разработка систем детекции объектов на изображениях, позволяющих эффективно определять местоположение различных элементов изображения и сопоставлять их между кадрами.

Результаты полученной системы для детекции могут быть полезны при решении таких задач, как: определение уровня загруженности дорог и пешеходных улиц, наблюдение за конкретными участками городской и сельской местности для выявления различных видов правонарушений.

ГЛАВА 1
АКТУАЛЬНОСТЬ И ОБЗОР МЕТОДОВ

1.1 Актуальность детекции объектов

Детекция объектов имеет огромную актуальность и широкий спектр применения в современном мире. С развитием компьютерного зрения, машинного обучения и автоматизации, детекция стала неотъемлемой частью многих отраслей, обеспечивая точное определение местоположения и анализ движения объектов.

Детекция объектов продолжает развиваться и улучшаться с развитием новых технологий и методов. Внедрение и использование искусственного интеллекта, глубокого обучения и компьютерного зрения позволяют достичь более точного и надежного трекинга. Более компактные и мощные устройства, такие как датчики и камеры, позволяют проводить трекинг в реальном времени и в различных условиях.

Однако существуют и некоторые проблемы, связанные с детекцией объектов, такие как сложность обработки большого объема данных, учет окружающей среды и возможность ложных срабатываний. Тем не менее, с постоянным развитием технологий и исследований, эти проблемы становятся все более преодолимыми.

Основным источником данных для решения задачи трекинга являются изображения и видео. С развитием технологий требуется все больше данных для решения разнообразных задач, в следствие чего появляется все больше наборов данных. Самыми популярными наборами данных для решения задач трекинга в настоящее время являются:

1) MOT (Multiple Object Tracking) Challenge: Это серия датасетов соревнований по детекции объектов, которая проводится ежегодно. Она включает в себя несколько датасетов, включающих видео с различными сценами и объектами, а также разметку, содержащую информацию о положении и идентификации объектов. Датасеты MOT Challenge широко используются в академических исследованиях и являются стандартом для оценки алгоритмов детекции. В рамках данной работы будет использован именно этот набор данных, а именно его версии MOT17 и MOT20.

2) KITTI: Этот датасет был создан для задачи трекинга объектов в автономных транспортных средствах. Он включает в себя видеофрагменты снятые с автомобиля, а также разметку, содержащую информацию о

положении и классификации объектов (например, автомобили, пешеходы, велосипедисты и т.д.), а также о трассировке объектов во времени.

3) COCO (Common Objects in Context): Этот датасет был создан для задачи обнаружения объектов, но также может быть использован для детекции. Он содержит большое количество изображений с разнообразными объектами, размеченных с указанием классов объектов и ограничивающих рамок вокруг них. Хотя COCO не является специфическим датасетом для трекинга, его можно использовать для создания алгоритмов детекции подвижных на основе обнаружения объектов.

4) UAVDT (Unmanned Aerial Vehicle Detection and Tracking): Этот датасет был создан специально для детекции объектов в видеопотоках, полученных с беспилотных летательных аппаратов (БПЛА). Он содержит видеофрагменты, снятые с БПЛА, с различными сценами и объектами, такими как автомобили, пешеходы, велосипедисты и другие. Датасет включает в себя разметку, содержащую информацию о положении и идентификации объектов.

1.2 Методы детекции объектов в статике

Первые беспилотные летательные аппараты появились еще в 1920-х годах, однако, до 2000-х годов они не были предназначены для детекции объектов. Одними из первых БПЛА, оборудованных камерами и предназначенными для задач детекции объектов были американские БПЛА «AeroVironment RQ-11 Raven» и “Insitu ScanEagle”. Однако, кроме того, что разрешение камер того времени не позволяло проводить качественную детекцию объектов, использовались данные модели исключительно в военной сфере. Первые дроны, предназначенные для применения в сферах, отличных от военной, начали появляться в 2010-х годах, когда в США было официально разрешено использование небольших пользовательских беспилотников. С того времени развитие как самих БПЛА, так и систем автоматизации для дронов, идет очень быстро.

Детекция объектов является одним из направлений автоматизации систем обработки изображений, полученных с дронов. Первые системы детекции объектов на изображениях появились в 1960-х годах, почти сразу после появления компьюерного зрения. Одной из самых ранних известных моделей детекции объектов является “Rapid Object Recognition Engine (RORE)”, основанная на технологии сопоставления с шаблоном, которая является крайне не эффективной. Далее, для повышения качества и эффективнности распознавания образов, решили делать анализ формы и геометрические преобразования:

Анализ формы: в 1980-х годах был достигнут значительный прогресс в технике анализа формы, такой как преобразование Хофа, которое позволило компьютерам распознавать и интерпретировать геометрические фигуры на изображениях.

Геометрические преобразования: Исследователи разработали алгоритмы для выполнения геометрических преобразований, включая масштабирование, поворот и перевод изображений, что расширило возможности выравнивания и сравнения визуальных данных.

Следующим этапом развития систем детекции было извлечение признаков и дескрипторы:

Извлечение признаков: В 1990-х годах исследователи разработали алгоритмы извлечения отличительных признаков из изображений, такие как Scale-Invariant Feature Transform (SIFT) и Speeded-Up Robust Features (SURF). Эти методы позволили надежно обнаруживать и сопоставлять признаки.

Локальные дескрипторы: Введение локальных дескрипторов, таких как Histogram of Oriented Gradients (HOG) и Local Binary Patterns (LBP), позволило представлять и сравнивать области изображений, заложив основу для распознавания объектов.

В 1980-х широкую популярность приобрели нейронные сети, которые изменили подходы в компьютерном зрении. Одним из первых, кто применил нейронные сети в задачах компьютерного зрения, был Ян Лекун, создавший первую сверточную нейронную сеть LeNet, предназначенную для задач классификации. Позже, достижения классификационных сверточных нейронных сетей были применены для решения задач детекции объектов.

Первыми моделями для детекции объектов, которые актуальны в настоящее время, были двух стадийные детекторы. Свое название они получили благодаря работе в 2 стадии, первая из которых – генерация регионов интересов (RoI), вторая – классификация и локализация объектов. Всего существует 3 основных двухстадийных детектора, каждый из которых улучшает эффективность предыдущего:

- R-CNN (Region Based Convolutional Neural Network) [7]. Была разработана в ноябре 2013 года. Первая модель, архитектура которой актуальна в задачах детекции объектов на сегодняшний день. Алгоритм, лежащий в основе создания интересующих нас регионов - выборочный поиск(Selective search). Данный алгоритм является крайне не эффективным, так как, хоть и значительно сокращает количество стартовых предсказаний по сравнению с методом скользящего окна, все еще генерирует большое количество начальных RoI, после чего нейронная сеть обрабатывает каждый

предсказанный регион по отдельности. Графическая схема R-CNN приведена на рисунке 1.1.

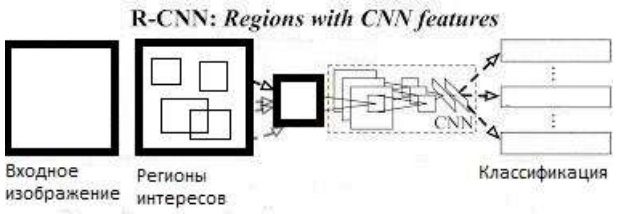

Рисунок 1.1 – Архитектура R-CNN, CNN – сверточная сеть

- Fast R-CNN [6]. Была разработана в апреле 2015 года. Она улучшала производительность R-CNN, так как сперва пропускала через себя все изображение, а затем строила RoI. Однако, алгоритм построения RoI остался прежним – Selective search. Графическая схема Fast R-CNN приведена на рисунке 1.2.

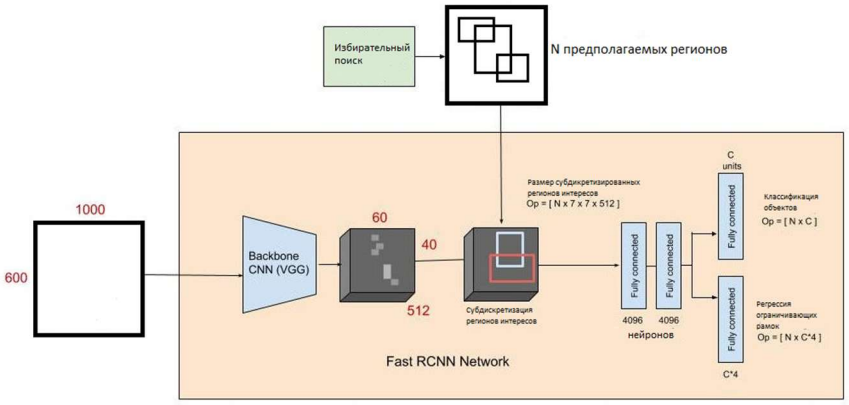

Рисунок 1.2 - Архитектура Fast R-CNN

- Faster R-CNN [10]. Была разработана в июне 2015 года. Так же, как и Fast-RCNN сперва пропускает изображение через нейронную сеть, однако сильно изменился алгоритм построения RoI. Вместо selective search используют отдельную нейронную сеть для построения RoI. Это значительно уменьшает их количество и увеличивает качество. Графическая схема Faster R-CNN приведена на рисунке 1.3.

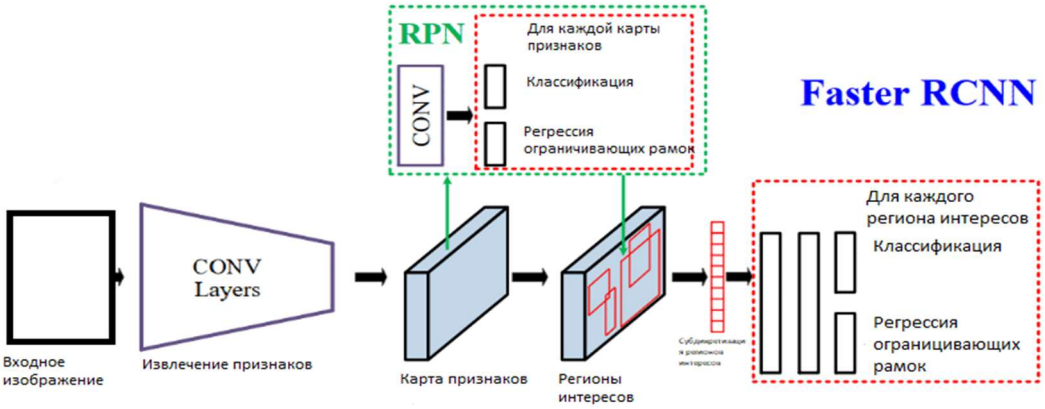

Рисунок 1.3 – Архитектура Faster R-CNN, conv layers – сверточные слои

Несмотря на то, что двухстадийные детекторы показывают хорошие результаты и по качественным, и по временным показателям, детекторы стремительно развивались. Так появились одностадийные детекторы, основными представителями которых являются SSD(single shot detector), YOLO (you only look ones).

Перед описанием данных моделей введем понятие принципа подавления немаксимумов (NMS). Данная техника была придумана для решения проблемы неоднозначности детекции, так как модель может генерировать несколько очень схожих предсказаний для одного объекта. NMS имеет параметр – коэффициент IoU(intersection over union). Затем, среди всех предсказаний, имеющих между собой IoU выше заданного, выбирается самое вероятное. Остальные предсказания удаляются.

- SSD [9]. Была разработана в 2016 году. Основным преимуществом стала скорость работы, которые была увеличена засчет избавления от генерации RoI. Вместо этого, сразу после прохождения через сверточную сеть, генерируются предсказания, после чего используется подавление немаксимумов. Графическая схема SSD приведена на рисунке 1.4
- YOLO [1]. На данный момент существует 8 версий YOLO, каждая из которых улучшает предыдущую. Основным преимуществом YOLO является скорость детекции. Последняя версия(YOLOv8) была создана в мае 2023 года и является новейшей моделью в области детекции изображений. Графическая схема YOLOv3 приведена на рисунке 1.5.

Сравнивая 2 типа моделей (двухстадийные и одностадийный) можно заметить, что качество детекции на них отличается не сильно, однако скорость обработки изображений значительно выше у вторых. На рисунках 1.6 и 1.7 графики, демонстрирующие качество и скорость детекции различных моделей.

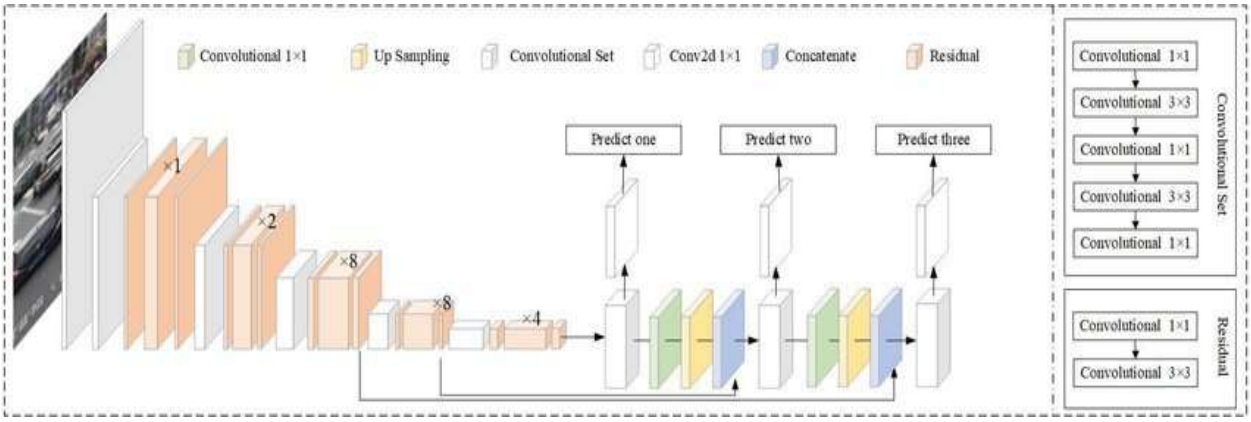

Рисунок 1.4 – Архитектура YOLO v3

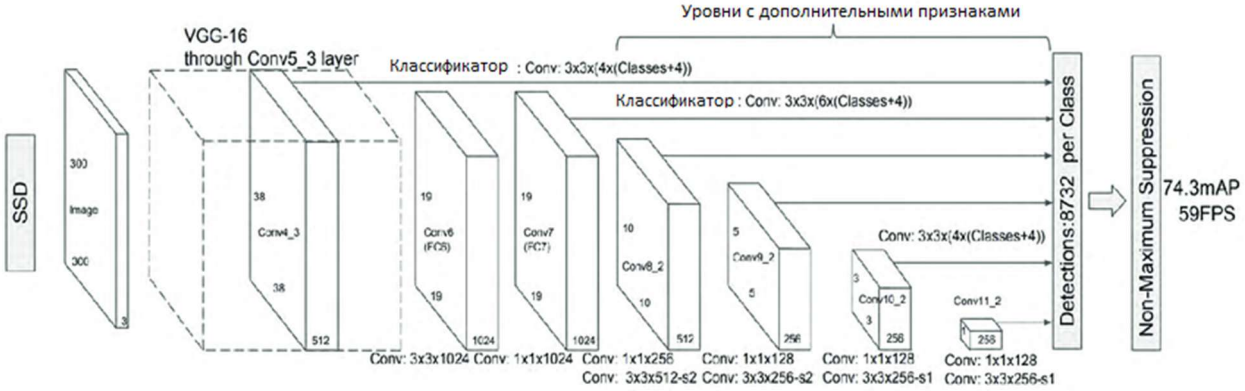

Рисунок 1.5 – Архитектура SSD

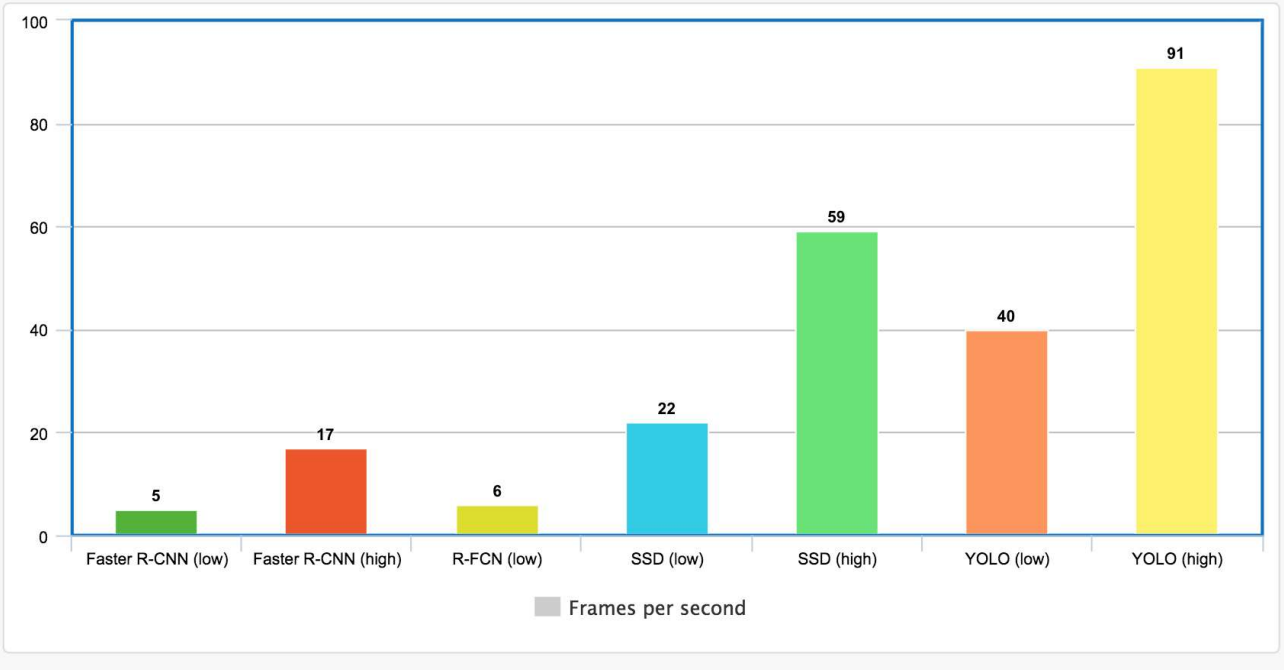

Рисунок 1.6 – FPS для двухстадийных (Fatser-RCNN) и одностадийных моделей (SSD, YOLO), FPS – кадры в секунду, low – низкий, high - высокий

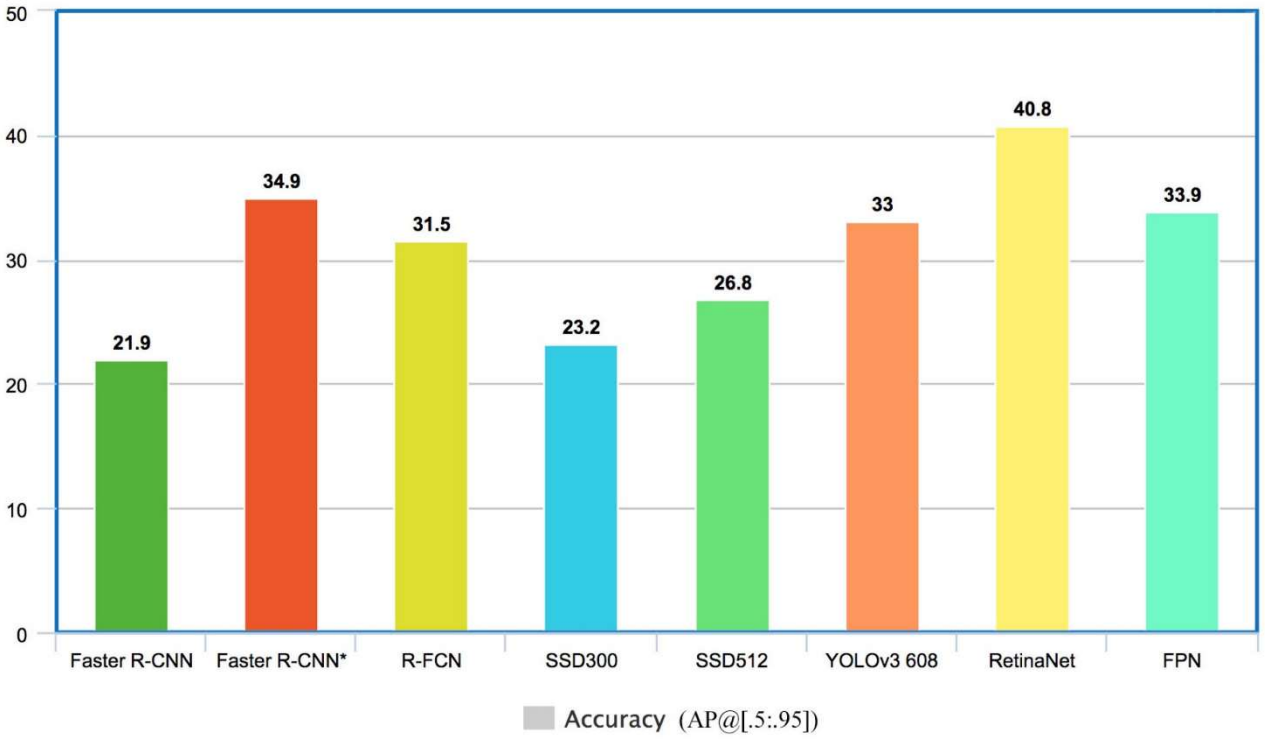

Рисунок 1.7 – mAP для двухстадийных (Fatser-RCNN, RetinaNet, FPN) и одностадийных моделей (SSD, YOLO) на PASCAL-VOC 2007

Таким образом, на данный момент существует множество готовых моделей, направленных на качественную и быструю обработку изображений, в том числе детекцию объектов на них. Однако, несмотря это, не все задачи

хорошо решаются существующим набором технологий. Например, для достижения высокой точности детекции малых относительно размера изображения объектов, необходимо применить некоторые преобразования данных или моделей. Данная необходимость возникает из за того, что вышеупомянутые модели, как правило, обучаются на одном из двух наборов данных: PASCAL-VOC 2007 или MS COCO, в которых изображения имеют нормальный масштаб. Так появлется необходимость улучшения существующих моделей для увеличения качества предсказываемых результатов.

1.3 Методы детекции объектов в движении

Традиционные методы [2] детекции объектов в движении (трекинга) играют важную роль в компьютерном зрении и широко применяются в различных реальных приложениях для обнаружения и отслеживания объектов. Эти методы основаны на таких техниках, как вычитание фона (background subtraction), оптический поток (optical flow) и дифференцирование кадров, и позволяют выполнять задачу отслеживания объектов. Несмотря на то, что более современные методы на основе глубокого обучения стали популярными в последние годы, традиционные подходы по-прежнему актуальны и служат основой для многих алгоритмов отслеживания. Рассмотрим подробнее эти традиционные методы.

Вычитание фона является одним из основных методов, используемых в отслеживании объектов. Идея вычитания фона заключается в моделировании статической фоновой сцены и затем выявлении движущихся объектов как переднего плана. Сначала создается модель фона путем записи сцены без объектов. Затем каждый кадр сравнивается с этой моделью фона для обнаружения изменений. Пиксели, значительно отклоняющиеся от модели фона, считаются пикселями переднего плана, указывая на наличие объектов. Однако методы вычитания фона чувствительны к изменениям в условиях освещения и требуют тщательной настройки параметров.

Оптический поток - это еще один традиционный метод, используемый для отслеживания объектов. Он сосредоточен на движении пикселей между последовательными кадрами. Алгоритмы оптического потока оценивают поле скорости изображения, анализируя перемещение интенсивностей пикселей. Отслеживая движение пикселей, можно определить траекторию объекта в течение времени. Было предложено множество алгоритмов оптического потока, включая методы Лукаса-Канаде, Хорна-Шунка и Фарнебека. Оптическое потоковое отслеживание может обрабатывать как жесткое, так и не жесткое движение объектов. Однако оно может столкнуться с трудностями

в случае наложений и быстрого движения, что может привести к неправильной оценке потока.

Дифференцирование кадров - это простой, но эффективный метод отслеживания объектов. Он заключается в вычитании последовательных кадров для выявления различий, вызванных движущимися объектами. Путем установки порога для полученного разностного изображения можно выделить области с значительными изменениями как потенциальные объекты. Дифференцирование кадров является вычислительно эффективным и может обнаруживать движущиеся объекты в режиме реального времени. Однако он чувствителен к шуму и изменениям условий освещения, что может приводить к ложным срабатываниям или пропускам. Для устранения этих проблем часто применяются такие техники, как морфологические операции и фильтрация шума.

Хотя вычитание фона, оптический поток и дифференцирование кадров являются эффективными методами отслеживания объектов, они имеют определенные ограничения. Эти методы уязвимы к наложениям объектов друг на друга, резким изменениям освещенности и сложным сценариям с несколькими объектами. Кроме того, они могут испытывать трудности при длительном отслеживании из-за смещения или невозможности восстановления после сбоев отслеживания. Для решения этих проблем было предложено множество гибридных подходов, объединяющих традиционные методы с более современными техниками, такими как фильтры Калмана, алгоритмы ассоциации данных между кадрами и нейронные сети.

Традиционные методы отслеживания объектов доказали свою полезность в реальных приложениях. Однако они хороши для достаточно простых ситуаций, например, для отслеживания одного объекта при статической камере. Также большинство классических алгоритмов трекинга чувствительны к изменению освещенности.

С развитием нейронных сетей, на смену классическим алгоритмам компьютерного зрения пришли методы, использующие глубокое обучение. Особо популярным стал метод, основанный на детекции объектов.

Отслеживание, основанное на детекции - это подход к отслеживанию объектов, который включает в себя обнаружение объектов в каждом кадре видео или последовательности изображений, а затем сопоставление обнаруженных объектов между кадрами для отслеживания объектов с течением времени. Вот как обычно работает процесс отслеживания с помощью обнаружения:

1) Обнаружение объектов: на первом этапе алгоритм обнаружения

объектов применяется к каждому кадру видео или последовательности изображений, чтобы идентифицировать и локализовать интересующие объекты. Для этой цели обычно используются алгоритмы обнаружения объектов, такие как Faster R-CNN, YOLO или SSD. Эти алгоритмы обычно выводят ограничивающие рамки вокруг обнаруженных объектов вместе с их метками классов.

2) Сопоставление обнаруженных объектов: после того как объекты обнаружены в каждом кадре, следующим шагом является сопоставление обнаруженных объектов в разных кадрах для определения траекторий объектов. Это делается путем сопоставления обнаруженных объектов в последовательных кадрах на основе различных критериев, таких как пространственная близость, сходство внешнего вида или информация о движении. Для этого процесса сопоставления могут использоваться различные алгоритмы отслеживания объектов, такие как фильтр Калмана, венгерский алгоритм или методы сопоставления данных, такие как IOU (пересечение через объединение) или методы, основанные на корреляции.

3) Сопровождение трека: по мере просмотра видео в каждом кадре появляются новые записи, а существующие треки обновляются. Этап обслуживания трека включает в себя назначение новых записей существующим трекам или, при необходимости, создание новых треков. Этот процесс помогает справиться с такими проблемами, как окклюзия, изменение внешнего вида объекта или временное исчезновение объекта.

4) Проверка и повторная идентификация отслеживаемых объектов: в некоторых случаях выполняются дополнительные действия для проверки и повторной идентификации отслеживаемых объектов. Эти шаги могут включать использование алгоритмов, основанных на внешнем виде, или методов глубокого обучения для сравнения обнаруженных объектов с галереей известных объектов, или использование дополнительной информации, такой как размер объекта или скорость.

1.4 Вывод

Таким образом, были рассмотрены актуальность и методы детекции объектов. Как было показано, в настоящее время, с развитием технологий дроны играют всё большую роль в жизни общества, в связи с чем происходит активное развитие и расширение задач, решаемых обработкой изображений с дронов.

В рамках данной работы будет рассмотрена задача детекции объектов, которые могут быть как статичными, так и находиться в движении.

Большинство современных решений для данной задачи используют глубокое обучение, однако возникают ситуации, когда методы глубокого обучения уступают классическим алгоритмам компьютерного зрения. Одной из таких ситуаций является детекция объектов на высотах, больших 70 метров, когда объекты теряют свои визуальные признаки.

Большинство методов глубокого обучения для задачи детекции статически объектов подразделяются на одностадийные и двухстадийные, однако в последнее время двухстадийные встречаются все реже.

Детекция движущихся объектов в настоящее время имеет множество решений, однако самое популярное использует двухэтапный алгоритм: сначала обнаруживаются объекты в статике, а затем происходит их сопоставление между кадрами.

ГЛАВА 2
ТЕОРЕТИЧЕСКИЕ ОСНОВЫ МОДЕЛИ ТРЕКИНГА

2.1 Детекция статических объектов

YOLOv8 (You Only Look Once version 8) [1] – одна из новейших одностадийных моделей, используемая для задач детекции объектов. YOLOv8 представляет собой улучшенную версию своих предшественников и сочетает в себе высокую скорость работы с высокой точностью обнаружения. YOLOv8 основана на архитектуре сверточных нейронных сетей и использует несколько усовершенствований для повышения качества детекции. Рассмотрим работу YOLOv8 более детально:

Первым этапом обработки изображения является прохождение картинки через сверточную нейронную сеть, отвечающую за создание карты признаков (Feature Maps). В YOLOv8 используется улучшенный «позвоночник» (backbone), например, CSPDarknet или другие легковесные сети, которые обеспечивают баланс между точностью и скоростью. Входное изображение масштабируется до фиксированного размера перед подачей в сеть.

Второй этап – генерация предсказанных объектов. YOLOv8 использует метод анкер-фри (anchor-free), что отличает его от предыдущих версий. Это позволяет модели предсказывать координаты ограничивающих рамок без использования заранее заданных якорей, что улучшает гибкость детекции и повышает точность на различных наборах данных.

Третий этап – предсказание классов и координат объектов. В отличие от двухстадийных моделей, таких как Faster R-CNN, YOLOv8 выполняет предсказание классов и координат ограничивающих рамок в один шаг. В конце нейронная сеть выдаёт вероятности принадлежности объектов к определенным классам и их координаты.

Этап обучения модели ничем не отличается от обучения полносвязных или сверточных нейронных сетей, кроме 1 элемента. Можно по-разному выбирать последовательность обучения RPN и Fast RCNN, однако результаты различаются не сильно, поэтому будет рассмотрен лишь случай когда вся сеть обучается как единое целое. Также стоит отметить, что для детекционных нейронных сетей функция потерь, в силу наличия 2ух независимых предикторов, имеет специфическую функцию потерь, определяемую по формулам 2.1, 2.2:

$$L(\{p_i\},\{t_i\}) = \frac{\gamma_{box}}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \frac{\gamma_{cls}}{N_{reg}} \sum_i p_i L_{reg}(t_i, t_i^*) +$$

$$+ \sum_i L_{fl}(p_i, p_i^*) \quad (2.1)$$

$$L_{reg}(t_i, t_i^*) = R(t_i - t_i^*) = Smooth_{L_1}(x) = f(x) = \begin{cases} 0.5x^2, & |x| < 1 \\ |x| - 0.5, \text{иначе} \end{cases} \quad (2.2)$$

, где i – номер якорной рамки в батче, N_{cls} – общее количество классов, p_i – вероятность того, якорная рамка содержит объект, p_i^* - 1 если в рамке есть объект, о иначе, $L_{cls}(p_i, p_i^*)$ - бинарная кросс-энтропия, γ – гиперпараметры, контролирующие вклад каждой из функций потерь в итоговую функцию потерь, t_i – вектор с четырьмя координатами , содержащий информацию о местоположении объекта, t_i^* - истинное положение объекта, связанного с положительным якорем. L_{fl} – фокальная функция потерь, введенная для обработки неравномерного распределения классов.

Очевидно, что для обучения данной модели с нуля потребуется огромное количество времени, поэтому зачастую используют трансферное обучение (transfer learning). Трансферное обучение позволяет использовать уже готовые решения для новых задач, модифицируя лишь последние слои. Это значительно сокращает временные затраты, а результат может оставаться тем же. Так, в своей задаче я использовал YOLOv8, обученный на MS COCO. Также стоит отметить, что Faster RCNN как правило, также является результатом трансферного обучения, в котором «позвоночник» сети уже имеет готовые веса, а затем YOLOv8 дообучивается на некотором наборе данных, например, MS COCO. В своей модели я имею следующее: «Позвоночник» сети – ResNet50, обучена на ImageNet. Затем YOLOv8 дообучена на MS COCO.

Однако, данная модель плохо применима для задачи детекции изображений с дронов. Объекты изображений из набора данных COCO имеют сильно отличающиеся от объектов, располагающихся на изображениях дронов характеристики, во-первых, классы объектов, во-вторых их масштаб. В то время как объекты из COCO занимают, в среднем, 60% от общей высоты изображения, объекты, полученные с дронов, занимают примерно 1%.

Для повышения точности предсказаний можно применить трансферное обучение на нужном наборе данных.

Крупнейшим набором данных, предоставляющим данных про изображения с дронов, является VisDrone. Он содержит в себе 288 видео,

10209 изображений, из которых 6400 – тренировочные, 548 – валидационные и 1 580 – тестовые.

Построим модель для дообучения. Исходной моделью будет YOLOv8, обученная на MS COCO. Сохраним все слои и их веса, за исключением последнего – полносвязной нейронной сети.

Новая полносвязная нейронная сеть будет иметь структуру, практически идентичную той, которая была в исходной версии. Единственным изменением будет количество выходных нейронов – в то время, как MS COCO содержит информацию об объектах 91 класса, VisDrone имеет 12 классов. Поэтому количество выходных нейронов в новой модели – 12.

Но, как было сказано ранее, одна из основных проблем в обработке изображений с дронов – малый масштаб объектов на них. Эту проблему частично можно решить изменением изображением таким образом, что масштаб объекта относительно размера изображения увеличится.

2.2 Трекинг на основе оценки движения объекта

Как уже было сказано ранее, одна из самых популярных технологий трекинга – трекинг, основанный на детекции. DeepSORT [12] является одной из реализаций этой концепции. DeepSORT – расширение алгоритма SORT [4]. SORT сопоставляет объекты между кадрами только на основании оценки движения объекта, однако данный алгоритм будет неустойчив при движении камеры. DeepSORT, в свою очередь, использует также оценку внешнего вида объекта, что делает его более устойчивым к движению камеры, а также привносит способность продолжать трекинг объектов, пропадающих из кадра на короткий промежуток времени, который является гиперпараметром модели. Так как DeepSORT – расширение SORT, сперва будет рассмотрен именно второй.

В алгоритме SORT основное внимание уделяется эффективной ассоциации объектов для слежения в реальном времени. В этом контексте качество обнаружения определяется как ключевой фактор, влияющий на производительность трекера, и примечательно, что изменение детектора может улучшить трекинг до 18,9%. Несмотря на использование простых комбинаций хорошо известных техник, таких как фильтр Калмана и венгерский алгоритм в качестве этапов трекинга, предложенный подход достигает точности, сравнимой с современными трекерами, работающими в реальном времени. Более того, благодаря простоте метода, обновление трекера происходит с частотой 260 Гц, что в более чем 20 раз быстрее, чем у других современных трекеров. SORT можно логически разбить на несколько этапов.

Первый из них – обнаружение объектов. Как уже было сказано ранее, детектор является ключевой компонентой в алгоритмах трекинга на основе детекции. Так как обнаружение объектов одинаково для одного изображения и для последовательности, детектор можно выбирать исходя из результатов, полученных ранее. Один из лучших детекторов на данный момент – YOLOv8, достигающий высоких качественных и временных показателей.

Второй – оценка движения объектов. В данной части описывается модель объекта, то есть его представление и модель движения, используемые для передачи идентификатора в следующий кадр. Перемещения между кадрами аппрроксимируются между кадрами с помощью линейной модели с постоянной скоростью, которая не зависит от других объектов и движения камеры. Состояние каждой цели моделируется следующим образом: $x = [u,v,s,r,u',v',s']^T$, где u и v представляют положение центра объекта по осям X и Y соответственно в пикселях, а s и r представляют масштаб (площадь) и соотношение сторон ограничивающего прямоугольника объекта соответственно. Когда ограничивающий прямоугольник связывается с объектом, он используется для обновления состояния цели, где компоненты скорости оптимально определяются с использованием фильтра Калмана, описанной на рисунке 2.1. Если для трека не было найдено ограничивающей рамки, его состояние предсказывается согласно обычной линейной модели.

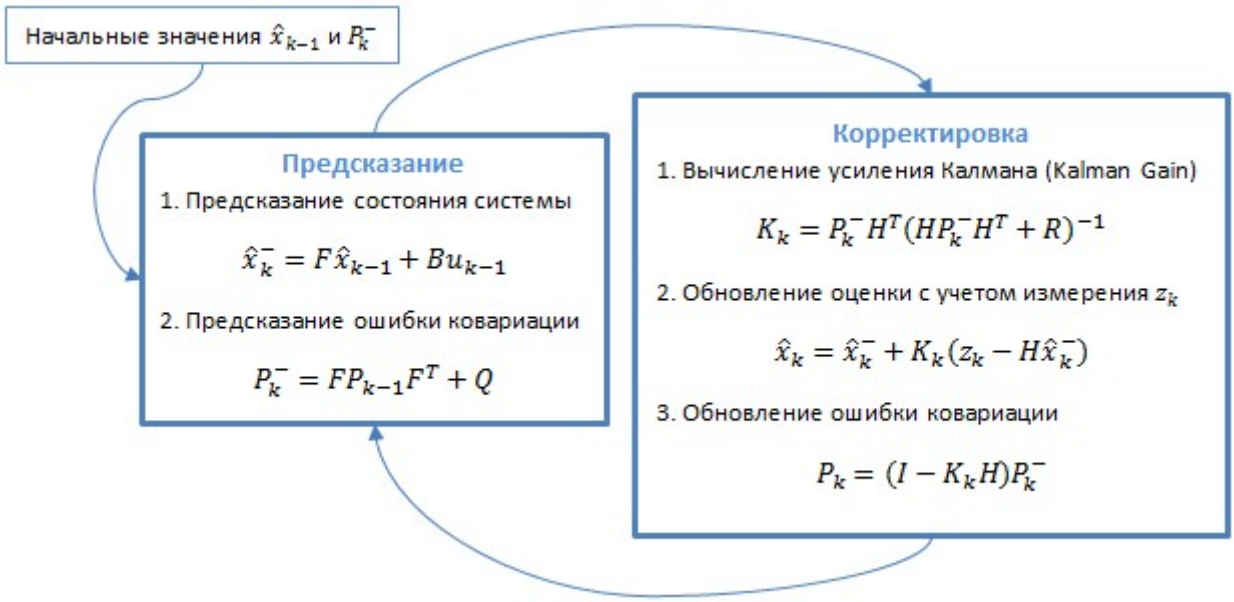

Рисунок 2.1 – алгоритм работы фильтра Калмана. Здесь $\widehat{x_{k-1}}$ – начальное местоположение объекта, а P_k^- - начальная матрица ошибок ковариации.

Третий этап – сопоставление идентификаторов между кадрами (также это часто называют ассоциацией данных). При присваивании ограничивающих рамок существующим трекам оценивается состояние

ограничивающего прямоугольника каждой цели путем предсказания ее нового местоположения в текущем кадре. Затем вычисляется матрица стоимости присваивания как расстояние IOU между каждым обнаружением и всеми предсказанными ограничивающими прямоугольниками. Назначение решается оптимально с использованием венгерского алгоритма. Кроме того, устанавливается минимальное значение IOU_{min}, чтобы отклонять назначения, где значение меньше, чем IOU_{min}.

Также подробнее опишем алгоритм создания и удаления треков.

Когда новые объекты появляются в кадре, новые треки должны быть созданы. Новый трек создается каждый раз, когда находится ограничивающая рамка, не имеющая ассоциации ни с одним из треков. Она инициализируется нулевой скоростью и большой дисперсией, чтобы подчеркнуть неуверенность предсказания. Более того, каждый новый трек проходит испытательный срок, который обычно составляет несколько кадров, в течение которого данному треку должны сопоставляться T_{lost}граничивающие рамки. Треки удаляются, если на протяжении T_{lost} кадров не было найдено соответствующей треку ограничивающей рамки.

Таким образом, из изложенного выше алгоритма видно, что SORT является вычислительно эффективным алгоритмом, позволяющим обрабатывать информацию в режиме реального времени, что является одним из главных его достоинств. Однако, он имеет и несколько недостатков:

1) Алгоритм неустойчив к движениям камеры. Одно из условий использования фильтра Калмана – статичность камеры. В случае нарушения данного условия результаты трекинга сильно ухудшаются.

2) Выход объектов за границы изображений. В случае выхода объекта из зоны видимости камеры, всякая информация о нем теряется, а значит, в случае возвращения объекта в кадр ему будет назначен новый идентификатор.

3) Необратимость ошибочной идентификации. В случае неправильной идентификации объекта, возврат к верной идентификации невозможен

2.3 Модификация DeepSORT для детекции мелких объектов

DeepSORT решает некоторые проблемы, присущие классическому SORT. Во-первых, в случае потери объекта, возможна его дальнейшая корректная идентификация. Это происходит благодаря тому, что DeepSORT хранит информацию о внешнем виде объекта на протяжении нескольких кадров, и в случае повторного появления в кадре способен присвоить ему тот идентификатор, который был до потери.

Во-вторых, DeepSORT уменьшает количество переключений идентификаторов. Переключение идентификатора происходит всякий раз, когда одному и тому же истинному объекту трекер присваивает различные идентификаторы. Главной причиной данной проблемы являются продолжительные окклюзии, в течение которых предсказание модели, основанной на движении, становится все более неточным. Уменьшение количества переключений также происходит благодаря информации о внешнем виде объекта. Рассмотрим работу DeepSORT подробнее.

Обработка трека и оценка состояния**:**

Система обработки трека и фильтра Калмана в основном идентична оригинальному алгоритму SORT. Предполагается, что трекинг ведется на видео с подвижной камерой, причем ее параметры не известны. Каждый трек представлен восьмимерным вектором (u, v, γ, h, x, y, γ', h'), который содержит положение центра ограничивающего прямоугольника (u,v), соотношение сторон γ, высоту h и скорости в координатах изображения. Используется стандартный фильтр Калмана с постоянной скоростью движения и линейную модель наблюдения, где берутся координаты (u, v, γ, h) в качестве непосредственных наблюдений за состоянием объекта. Для каждого трека k подсчитывается количество кадров, прошедших с момента последнего сопоставления данного трека некоторой ограничивающей рамке. Этот счетчик увеличивается во время прогнозирования фильтра Калмана и сбрасывается на 0, когда трек был связан с ограничивающей рамкой. Треки, возраст которых превышает заданный максимум, считаются покинувшими сцену и удаляются из набора треков. Для каждого обнаружения (детекции), которое не может быть связано с существующим треком, выдвигаются новые гипотезы о треке. Эти новые треки классифицируются как предварительные в течение первых нескольких кадров, чаще всего 3. В течение этого времени ожидается успешное сопоставление результатов измерений на каждом временном шаге. Треки, которые не были успешно сопоставлены с результатами измерений в течение первых трех кадров, удаляются.

Сопоставление измерений и треков.

Традиционным способом определения связи между предсказанными состояниями Калмана и вновь полученными измерениями является построение задачи о назначении, которая может быть решена с помощью венгерского алгоритма. В этой постановке задачи объединяется информация о движении и внешнем виде с помощью комбинации двух подходящих метрик. Для включения информации о движении используется квадратичное расстояние Махаланобиса между предсказанными состояниями Калмана и вновь полученными измерениями, которое рассчитывается по формуле 2.1:

$$d_{i,j}^{1} = (d_i - y_i)^T S_i^{-1} (d_i - y_i) \quad (2.1)$$

В данной формуле обозначается проекция распределения i-го трека в пространство измерений с помощью (y_i, S_i) и k-ый ограничивающий прямоугольник с помощью d_k. Расстояние Махаланобиса учитывает неопределенность оценки состояния, измеряя, на сколько стандартных отклонений измерений далеко от математического ожидания расположения трека. Кроме того, используя этот показатель, можно исключить маловероятные связи, установив пороговое значение расстояния Махаланобиса на уровне 95% доверительного интервала, рассчитанного на основе обратного распределения Хи – квадрат. Это решение обозначается показателем, рассчитываемого по формуле (2.2).

$$b_{i,j}^{1} = I[d^1(i,j) \leq t^1] \quad (2.2)$$

Данное значение равно 1, если связь между i-м треком и j-м обнаружением допустима. Для нашего четырехмерного пространства измерений соответствующий порог Махаланобиса равен 9,4877.

В то время как расстояние Махаланобиса является подходящей метрикой при низкой неопределенности движения, в нашей постановке задачи прогнозируемое распределение состояний, полученное с помощью фильтра Калмана, дает лишь приблизительную оценку местоположения объекта. В частности, неучтенное движение камеры может привести к быстрому смещению плоскости изображения, что делает расстояние Махаланобиса довольно неточной метрикой для отслеживания при возникновении окклюзий. Поэтому DeepSORT использует вторую метрику. Для каждого ограничивающего прямоугольника d_k описывается внешний вид с помощью вектора $k, \|r_k\| = 1$. Затем, хранится множество векторов R_k, которые являются $L_k = 100$ последними векторами каждого трека. Затем используется вторая метрика для подсчета минимального косинусного расстояний между измерениями и треками, рассчитываемого по формуле (2.3):

$$d_{i,j}^{2} = \min\{1 - r_j^R r_k^i \mid r_k^i \in R_i\} \quad (2.3)$$

Снова вводится бинарная переменная, определяемая по формуле (2.4), для отклонения невозможных согласно данной метрики назначений:

$$b_{i,j}^2 = I[d^2(i,j) \leq t^2] \quad (2.4)$$

Мы находим подходящее пороговое значение для этого показателя в отдельном наборе обучающих данных. В сочетании оба показателя дополняют друг друга, отвечая за различные аспекты задачи назначения. С одной стороны, расстояние Махаланобиса предоставляет информацию о возможном местоположении объекта на основе движения, что особенно полезно для краткосрочных прогнозов. С другой стороны, косинусное расстояние учитывает информацию о внешнем виде, которая особенно полезна для назначения идентификатора после длительных окклюзий, когда движение. Для решения задачи назначения объединяются обе метрики, используя взвешенную сумму, определяемую формулой (2.5):

$$c_{i,j} = \lambda d_{i,j}^{(1)} + (1 - \lambda) d_{i,j}^{(2)} \quad (2.5)$$

Назначение согласно данной метрике возможно, если оно возможно согласно обоим метрикам, что по можно описать формулой (2.6).

$$b_{i,j} = \prod_{m=1}^{2} b_{i,j}^{(m)} \quad (2.6)$$

Влияние каждого показателя на общую стоимость связи можно контролировать с помощью гиперпараметра λ. В ходе экспериментов было обнаружено, что установка $\lambda = 0$ является разумным выбором при значительном движении камеры. Тем не менее, расстояние Махаланобиса по-прежнему используется для игнорирования назначений, маловероятных согласно фильтру Калмана.

Каскад назначений

Вместо решения проблемы взаимосвязи измерений и отслеживания в глобальной задаче назначения вводится каскад назначений, который решает ряд подзадач. Чтобы обосновать этот подход, рассмотрим следующую ситуацию: когда объект закрыт в течение длительного периода времени, последующие предсказания фильтра Калмана увеличивают неопределенность, связанную с местоположением объект. Следовательно, масса вероятностей распределяется в пространстве состояний и вероятность наблюдения становится невыраженной. Теоретически, метрика ассоциации должна учитывать этот разброс массы вероятности за счет увеличения расстояния от измерения до отслеживания. Парадоксально, но когда два трека конкурируют за одно и то же обнаружение, расстояние Махаланобиса способствует большей неопределенности, поскольку оно эффективно уменьшает расстояние в

стандартных отклонениях любого обнаружения по отношению к прогнозируемому среднему значению трека.

Это нежелательное поведение, поскольку оно может привести к увеличению потерь треков и их нестабильности. Поэтому вводится каскад сопоставлений, который отдает приоритет наиболее часто встречающимся объектам, чтобы закодировать наше представление о разбросе вероятностей в вероятности ассоциации. Алгоритм каскада сопоставлений следующий:

Входные данные: множество индексов треков $\mathrm{T} = \{1, \dots, N\}$ и множество индексов ограничивающих прямоугольников $\mathrm{D} = \{1, \dots, M\}$, максимальный возраст A_{max}

1: Рассчитываем матрицу стоимостей C = $\left[c_{i,j}\right]$ используя формулу 2.5

2: Рассчитываем матрицу фильтрации B = $\left[b_{i,j}\right]$ используя формулу 2.6

3: Инициализируем множество сочетаний $S \leftarrow \emptyset$

4: Инициализируем множество детекций, которым в соответствие не было поставлено трека: $U \leftarrow \emptyset$.

5: Для всех $n \in \{1, \dots, A_{max}\}$:

6: Выбираем треки согласно возрасту $T_n \leftarrow \{i \in T \mid a_i = n\}$

7: $x_{i,j} \leftarrow \min \_cost\_matching(C, T_n, U)$

8: $S \leftarrow S \cup \{(i,j) \mid b_{i,j} * x_{i,j} > 0\}$

9: $U \leftarrow U \{j \mid \sum_i b_{i,j} * x_{i,j} > 0\}$

В данном алгоритме описан каскад сопоставления, результатом выполнения которого станут сформированные множества M, U. В качестве входных данных предоставляется набор индексов отслеживания T и обнаружения D, а также максимальный возраст A_{max}. В строках 1 и 2 вычисляется матрица стоимости сопоставлений и матрица допустимых ассоциаций. Затем перебираются треки с возрастом k, чтобы решить задачу линейного назначения треков. В строке 6 выбираются подмножество треков T_n, которые не были связаны с измерением в последних n кадрах. В строке 7 решается задача линейного назначения между треками из τ_nи не назначенными измерениями U. В строках 8 и 9 обновляется множество назначенных и не назначенных измерений (детекций), которые возвращаются после завершения работы алгоритма в строке 11. Стоит отметить, что этот каскад совпадений отдает приоритет трекам меньшего возраста, т.е. следам, которые были обнаружены позже. На заключительном этапе сопоставления

проводятся назначения на основании IOU, как предложено в оригинальном алгоритме SORT на множестве неподтвержденных и не назначенных треков возраста 1. Это помогает учитывать внезапные изменения внешнего вида, например, из-за частичного перекрытия статичной геометрией сцены, а также повысить устойчивость к ошибочной инициализации.

Извлечение признаков

Как уже было сказано ранее, сеть для извлечения признаков должна быть вычислительно эффективной. Одной из таких сетей является MobileNetV2 [8]. Также для данной задачи часто применяется сеть resnet , однако время ее работы при невысоких вычислительных ресурсах на множестве изображений слишком велико.

MobileNetV2 - это сверточная нейронная сеть, разработанная компанией Google, предназначенная для эффективного выполнения задач компьютерного зрения на мобильных и встроенных устройствах с ограниченными вычислительными ресурсами.

Основная идея MobileNetV2 заключается в создании легкой и быстрой модели, которая достигает хорошего качества классификации изображений и других задач компьютерного зрения. Она применяет несколько техник для достижения этой цели.

Первая - Depthwise Separable Convolution (раздельная свертка по глубине): MobileNetV2 использует раздельные свертки, которые разделяют операцию свертки на две отдельные операции: свертку по каждому каналу входных данных (depthwise convolution) и свертку для объединения выходов (pointwise convolution). Это позволяет значительно снизить количество параметров и вычислительную сложность модели.

Вторая - Inverted Residuals with Linear Bottlenecks (инвертированные остаточные блоки с линейными бутылочными горлышками): MobileNetV2 использует остаточные блоки, которые позволяют эффективно передавать информацию между слоями и улучшают качество предсказаний. Они состоят из сверточных слоев, точечных сверток и применения функций активации.

Третья - Width Multiplier (множитель ширины): MobileNetV2 включает параметр множителя ширины, который позволяет настраивать ширину модели. Множитель ширины контролирует количество фильтров в каждом слое, что позволяет увеличивать или уменьшать количество параметров и вычислительную сложность модели.

Таким образом, разработана и реализована пошаговая модель для трекинга подвижных объектов. Она будет представлять из себя трехфазовый алгоритм. Сперва происходит детекция объектов и извлечение их признаков, затем происходит сопоставление предсказаний фильтра Калмана и

ограничивающих рамок для назначений идентификаторов в данном кадре. В конце происходит обновление фильтра Калмана для передачи треков в следующий кадр. Схематически алгоритм можно описать при помощи рисунка 2.2.

Рисунок 2.2 – алгоритм работы DeepSORT

Однако классический алгоритм DeepSORT был разработан для детекции крупномасштабных объектов, в то время как изображения с дронов являются мелкомасштабными. Одними из техник, позволяющих улучшить точность предсказаний, являются технологии Slicing aided hyper inference (SAHI) и Slicing aided fine tuning [3].

Slicing aided hyper inference – технология преобразования изображения перед его прохождением через нейронную сеть (inference), позволяющая увеличить относительный размер объектов. Каждое изображение нарезается на более мелкие изображения размера $n \times m$, которые могут накладываться друг на друга (техника overlapping patches), причем общая их площадь составляет S. S,n,m являются гиперпараметрами. Затем эти нарезанные изображения увеличиваются, их ширина становится от 800 до 1333 пикселей, сохраняются соотношения сторон. Множество картинок, полученных из одной исходной, прогоняются через сеть, после чего все предсказания наносятся на одно изображение и проводится процесс подавления немаксимумов с учетом коэффициента IoU (Intersection over union), являющегося еще одним гиперпараметром.

Slicing aided fine tuning – процесс дообучения сети на изображениях полученных из исходных путем описанным выше. Однако, данные преобразования увеличивают количество изображений и их размер, что сильно увеличивает время обучения.

Также зачастую используется комбинация SAHI с традиционным подходом, то есть сперва прогоняется целое изображение, затем нарезанные описанным выше алгоритмом части. Все результаты наносятся на одно изображение, и к этому применяется принцип подавления немаксимумов. Это часто называют full inference. В таблице 1 представлены результаты применения различных комбинаций описанных мною технологий на наборе данных MOT17, полученных на одностадийных нейросетевых детекторах. В таблице 1 SF, SAHI, FI, and PO соответствуют следующим терминам: slicing aided fine-tuning, slicing aided hyper inference, full image inference, and overlapping patches, соответственно. $AP_{50}, AP_{50}s, AP_{50}m, AP_{50}l$ отвечают соотвественно за точность на всех объектах при IoU = 0,5, точность на мелких, средних и больших объектах.

2.4 Оценка движения камеры

Данный алгоритм имеет большой недостаток - он предполагает статичность камеры. Во время работы фильтра Калмана, при предсказании следующего состояния трека, учитывается только движение объекта. Камера при этом считается статичной. Предположим, что предполагаемый центр

объекта в следующем кадре – (x, y). Однако, при движении камеры, истинный центр объекта при условии точности фильтра Калмана – $(x + x_1, y + y_1)$, где (x_1, y_1) – сдвиг точки (x, y) согласно модели движения камеры.

Таблица 1 – результаты применения техник работы с маленькими объектами

Setup	AP_{50}	$AP_{50}s$	$AP_{50}m$	$AP_{50}l$
FCOS+FI	25.8	14.2	39.6	45.1
FCOS+SAHI+PO	29.0	18.9	41.5	46.4
FCOS+SAHI+FI+PO	31.0	19.8	44.6	49.0
FCOS+SF+SAHI+PO	38.1	25.7	54.8	56.9
FCOS+SF+SAHI+FI+PO	**38.5**	**25.9**	**55.4**	**59.8**
VFNet+FI	28.8	16.8	44.0	47.5
VFNet+SAHI+PO	32.0	21.4	45.8	45.5
VFNet+SAHI+FI+PO	33.9	22.4	49.1	49.4
VFNet+SF+SAHI+PO	41.9	**29.7**	58.8	60.6
VFNet+SF+SAHI+FI+PO	**42.2**	**29.6**	**59.2**	**63.3**
TOOD+FI	29.4	18.1	44.1	50.0
TOOD+SAHI	31.9	22.6	44.0	45.2
TOOD+SAHI+PO	32.5	22.8	45.2	43.6
TOOD+SAHI+FI	34.6	23.8	48.5	53.1
TOOD+SAHI+FI+PO	34.7	23.8	48.9	50.3
TOOD+SF+FI	36.8	24.4	53.8	**66.4**
TOOD+SF+SAHI	42.5	31.6	58.0	61.1
TOOD+SF+SAHI+PO	43.1	**31.7**	59.0	60.2
TOOD+SF+SAHI+FI	43.4	**31.7**	59.6	65.6
TOOD+SF+SAHI+FI+PO	**43.5**	**31.7**	**59.8**	65.4

Это приводит сразу к 2 проблемам:

1) Модель фильтра Калмана. При обновлении состояний треков, фильтр Калмана будет учитывать движение камеры как движение объекта, что приводит, во первых, к неточности предсказаний местоположения объекта в следующем кадре, а во вторых, к более шумной модели фильтра Калмана.

2) Сопоставление треков и детекций. Здесь, в следствие сдвига, будет неточным подсчет метрики IOU, что приведет к утери или переключению треков.

Особенно заметны эти проблемы становятся при хаотичном и резком движении камеры.

Для решения данной проблемы можно использовать можно использовать несколько решений, однако ниже будут приведены 2 самых эффективных решения.

Первый из них – оптический поток, реализованный методом Лукаса-Канада [2]. Основная идея метода заключается в предположении, что движение в небольшом окне вокруг каждой точки изображения можно аппроксимировать как постоянное, что делает задачу оценки движения более устойчивой к шуму и численно решаемой. Оптический поток, в контексте метода Лукаса-Канаде, представляет собой двумерное векторное поле, в котором каждый вектор указывает направление и величину перемещения точки изображения между двумя временными моментами. Предполагается, что яркость точки изображения остается постоянной между кадрами — это ключевое предположение, известное как постоянность интенсивности. Иначе говоря, если некая точка в первом кадре имеет определённую яркость, то в следующем кадре та же самая точка, сместившаяся вследствие движения, сохранит ту же яркость. Эта идея приводит к основному уравнению оптического потока (2.7):

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0 \qquad (2.7)$$

Однако это уравнение само по себе недостаточно для определения двух компонент движения, потому что в нем содержится только одно уравнение с двумя неизвестными — горизонтальной и вертикальной компонентами потока. Чтобы справиться с этим, метод Лукаса-Канаде делает ещё одно допущение: движение в окрестности некоторой точки (обычно квадратного окна $n \times m$ пикселей) одинаково для всех пикселей в этом окне.

Математически мы получаем систему уравнений из $n \times m$ уравнений, которая не имеет решений, поэтому решений ищется методом наименьших квадратов со следующей функцией потерь (2.8):

$$E(V_x, V_y) = \sum_{i,j} g(i,j) \times (\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t})^2 \qquad (2.8)$$

Однако у метода есть и ограничения. Он чувствителен к быстрым движениям, особенно если те вызывают смещения, превышающие размеры окна. Также его точность зависит от текстурности сцены — на однородных участках, где градиенты малы, он работает хуже или может давать неоднозначные результаты. Для повышения надёжности часто применяются дополнительные методы, такие как маски уверенности, которые указывают, насколько достоверным является результат для каждой точки.

Одним из достоинств метода является его вычислительная лёгкость: поскольку он оперирует малыми окнами и использует простые линейные

вычисления, его легко реализовать на аппаратном уровне, включая графические процессоры.

Метод Лукаса-Канаде часто используется в задачах детекции движущихся объектов, однако в данной работе он будет использован для детекции движения камеры. Этого можно достичь, если зафиксировать ключевые точки на заднем фоне объектов. То есть будет отслеживаться не сам объект, а движение сцены. Для того, чтобы найти ключевые точки на сцене, необходимо использовать детектор ключевых точек, например метод Ши-Томаси [11].

Этот метод приобрёл популярность благодаря своей высокой эффективности и стойкости к шуму, а также тому, что он естественным образом сочетается с алгоритмами отслеживания, в частности с методом Лукаса-Канаде, с которым его часто используют совместно.

Основная задача детектора углов заключается в том, чтобы найти такие точки на изображении, которые легко распознать и отследить при изменениях сцены, освещения или при небольших движениях камеры. Визуально, угол — это точка, в которой резко меняется направление границы или происходит пересечение двух границ. С математической точки зрения это соответствует таким точкам, где интенсивность изображения изменяется сильно и в разных направлениях. Например, если взять окно вокруг какой-либо точки изображения и начать сдвигать это окно в разные стороны, то угол — это такое место, где небольшое смещение приводит к значительному изменению содержимого окна.

Shi и Tomasi предложили определять углы, анализируя матрицу градиентов, определяемую формулой (2.9), изображения в некотором окне W.

$$M = \sum_{(x,y)\in W} W(u,v)\begin{pmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{pmatrix} \tag{2.9}$$

$$W(u,v) = \begin{cases} 1, \text{если } (u,v) \in W \\ 0. \text{иначе} \end{cases} \tag{2.10}$$

Угол характеризуется большими изменениями функции *E(x,y)* по всем возможным направлениям *(x,y)*, что эквивалентно большим по модулю собственным значениям матрицы *M*. Расположение собственных значений приведено на рисунке 2.3.

Поскольку напрямую считать собственные значения является трудоёмкой задачей, Харрисом и Стефеном была предложена мера отклика, определяемая формулой (2.11).

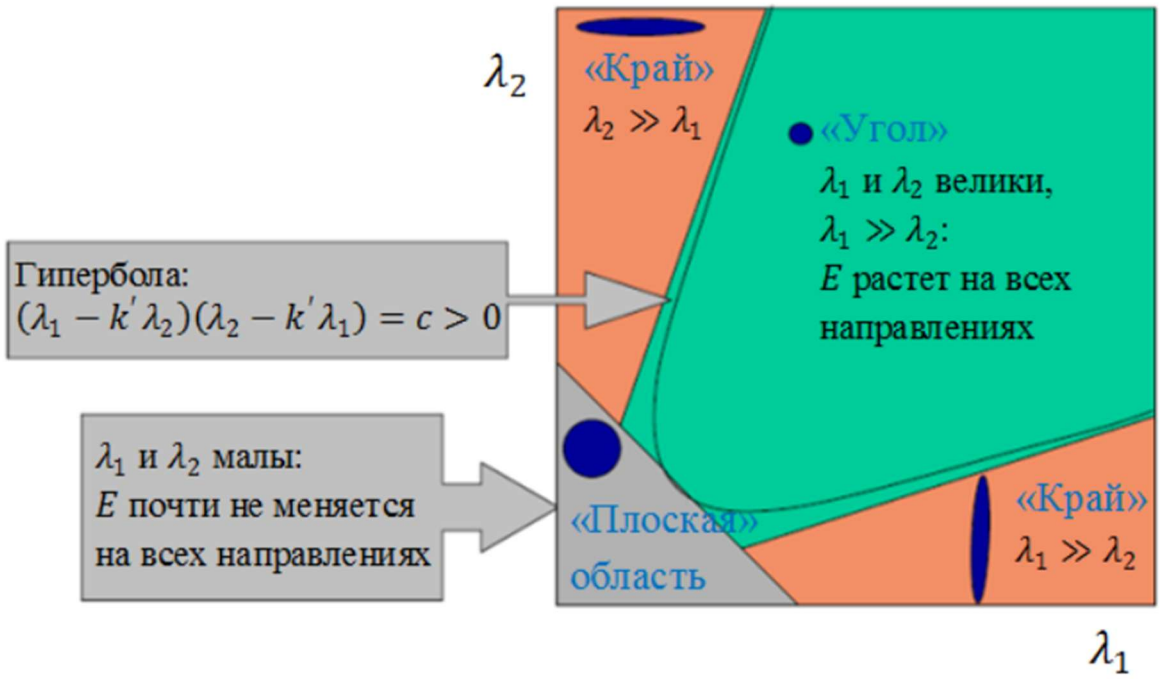

Рисунок 2.3 – расположение собственных значений

$$R = \min(\lambda_1, \lambda_2) \qquad (2.11)$$

Таким образом, значение R положительно для угловых особых точек. Затем производится отсечение точек по найденному порогу R (т.е. те точки, у которых значение R меньше некоторого порога, исключаются из рассмотрения). Далее находятся локальные максимумы функции отклика (non-maximal suppression) по окрестности заданного радиуса и выбираются в качестве уголковых особых точек..

Таким образом, Shi-Tomasi Corner Detector работает следующим образом: для каждой точки изображения вычисляется матрица градиентов, находятся её собственные значения, и затем принимается решение, является ли точка углом, на основе минимального собственного значения. В результате алгоритм выделяет точки, которые обладают высокой локальной изменчивостью изображения, что делает их хорошими кандидатами для отслеживания.

Одним из главных преимуществ этого метода является его надёжность. В отличие от детектора Харриса, который может ошибочно выделять "ложные" углы из-за особенностей своей функции оценки, метод Shi-Tomasi предлагает более строгий и точный критерий. Это особенно важно в задачах трекинга, где необходимо отслеживать те же самые точки на множестве кадров. Именно поэтому этот детектор часто используется как предварительный шаг в методе Лукаса-Канаде: сначала с его помощью находят хорошие точки для отслеживания, а затем вычисляют оптический поток для этих точек.

Кроме того, благодаря своей простоте и эффективности, детектор Ши-Томаси отлично подходит для реального времени.

Еще одним быстрым и качественным решением является коэффициент корреляции между кадрами.

Предположим, что у нас есть пара изображений $I_r(x),\ I_w(y)$, где первое изображение является истинным, а второе — искажённым изображением. Здесь $x = [x_1,\ x_2]^T$, $y = [y_1,\ y_2]^T$ обозначают координаты пикселей на изображении, а $I(x)$ – яркость пикселя x. В дальнейшем $I(x)$ будем называть профилем изображения. Также предполагается, что дано множество координат $T = \{x_k,\ k = 1,...,K\}$ в истинном изображении, которое называется целевой областью. Задача выравнивания изображения заключается в нахождении соответствующего множества координат в искажённом изображении. Конечно, нас не интересуют произвольные соответствия, а только те, которые имеют определённую структуру и могут быть смоделированы с помощью определённого векторного отображения (2.12):

$$y = \varphi(x;\ p) \tag{2.12}$$

Где, в общем случае, $p = [p_1, \dots, p_n]^T, n = 1,2, \dots, N$ — вектор неизвестных параметров. В большинстве случаев p представляет собой матрицу размера 2×3.

Предполагая, что модель трансформации задана (и при условии выполнения предположения о постоянстве яркости), задача выравнивания сводится к нахождению параметров p, таких что:

$$I_r(x) = I_w\big(\varphi(x;\ p)\big), \forall x \in T \tag{2.13}$$

Чтобы получить единственное решение, необходимо, чтобы число N неизвестных параметров не превышало число K целевых координат. На практике обычно выполняется условие $N \ll K$, что говорит о том, что

уравнение (2.13) представляет собой переопределённую систему (нелинейных) уравнений.

Большинство существующих алгоритмов пытаются вычислить вектор параметров p путём минимизации разницы или несоответствия между двумя изображениями. Несоответствие выражается через целевую функцию E(p), которая включает l_p-норму разницы профилей двух изображений. Поскольку в реальных приложениях, из-за различий в углах обзора и/или различных условий освещения, предположение о постоянстве яркости нарушается, необходимо включить дополнительное преобразование яркости, $\Psi(\mathrm{I}, \alpha)$, которое учитывает изменения яркости и параметризовано вектором неизвестных параметров α. Формула типичной задачи оптимизации имеет вид (2.14):

$$\min_{p,a} E(p,\alpha) \quad = \min_{p,a} \sum_{x \in T} |I_r(\mathrm{x}) - \Psi(I_w(\varphi(\mathrm{x};\ \mathrm{p})),\alpha)|^{\mathrm{p}} \qquad (2.14)$$

Следует упомянуть, что задачи оптимизации вида (2.9) часто плохо обусловлены, и обычно необходимо накладывать дополнительные условия регуляризации (гладкости), чтобы получить приемлемое решение.

Решение задачи оптимизации не является простой задачей из-за нелинейности. Вычислительная сложность и качество оценки существующих схем зависят от выбора нормы l_p и моделей, использующихся для искажения и преобразования яркости. Далее будет использована норма p = 2, то есть Евклидова норма.

При преобразовании координат с помощью функции искажения (2.12) координаты x_k целевой области T отображаются в координаты $\mathrm{y}_k(p) = \varphi(\mathrm{x}_k;\ \mathrm{p}), \mathrm{k} = 1, \ldots, \mathrm{K}$. Определим вектор яркостей эталонного изображения как i_r и соответствующий вектор интенсивностей искажённого изображения $i_w(p)$ как $i_r = [I_r(x_1), \ldots, I_r(x_k)]$, $i_w(p) = [I_w(y_1(p)), \ldots, [I_w(y_k(p))]$. Также введем обозначения $\widehat{i_r}$ и $\widehat{i_w}$ – соответствующие вектора с нулевым средним. Тогда мы предлагаем следующий критерий (2.15) для количественной оценки качества преобразования с параметрами p:

$$ECC\ (p) = \left\| \frac{\widehat{i_r}}{\|\widehat{i_r}\|} - \frac{\widehat{i_w(p)}}{\|\widehat{i_w(p)}\|} \right\| \qquad (2.15)$$

Данная формула предполагает, что мера будет инвариантна к любым изменениям яркости.

После некоторых преобразований формулы (2.16), мы получим формулу (2.11) коэффициента корреляции:

$$ECC\ (p) = \left\| \frac{\widehat{i_r}}{\|\widehat{i_r}\|} \frac{\widehat{i_w(p)}}{\|\widehat{i_w(p)}\|} \right\| \qquad (2.16)$$

Оптимизацию данной метрики можно производить, например, градиентными методами.

В зависимости от типа преобразования изображения, который выбирается вручную, формула для преобразования истинного изображения в искаженное может иметь различный вид, однако в обобщенном виде можно записать по формуле (2.17):

$$(x', y') = P * (x, y, 1) \tag{2.17}$$

Где P - матрица параметров, полученных после оптимизации (2.16).

2.5 Продвинутые алгоритмы извлечения признаков

DeepSORT использует сверточную сеть в качестве извлекателя признаков с комбинированной функцией потерь, вычисляемой по формуле (2.18):

$$Loss = L_{Tri} + L_{ID} \tag{2.18}$$

L_{Tri} – функция потерь «триплет», рассчитываемая в (2.19)

$$L_{Tri} = \left[d_p - d_n + a\right]_+ \tag{2.19}$$

Здесь d_pи d_n расстояния между признаками объектов того же класса некоторого другого класса соответственно, a – отступ, z_+ - max(z, 0).

L_{ID} – функция потерь классификации идентификаторов.

Важно, что в (2.18) обе составляющие функции потерь высчитываются из одного вектора признаков.

Для улучшения качества работы алгоритма DeepSORT было предложено множество модификаций [5].

Случайное повреждение изображений

В задачах реидентификации объекты часто оказываются перекрытыми другими объектами (происходят окклюзии). Для решения данной проблемы и для улучшения обобщающей способности сетей используют особый метод аугментации – случайное повреждение объекта. Данный способ случайным

образом маскирует какой-либо прямоугольник исходного изображения. Размер прямоугольника является гиперпараметером.

Применение батч-нормализации

На рисунке 2.3 показано, что ID Loss создает несколько гиперплоскостей, разделяющих пространство векторов на разные подпространства. Таким образом, косинусное расстояние более подходит, чем евклидово расстояние, для модели, оптимизированной с использованием ID Loss. Однако функция потерь «триплет» вычисляется с использованием евклидового расстояния и усиливает компактность внутри класса и разделимость между классами в евклидовом пространстве. Если использовать обе функции потерь для одновременной оптимизации пространства признаков, их цели могут оказаться противоречивыми. Во время обучения возможна ситуация, когда одна функция потерь уменьшается, а другая колеблется или даже увеличивается. В конечном итоге Triplet Loss может повлиять на четкие границы решений ID Loss, а ID Loss может снизить внутриклассовую компактность Triplet Loss. Распределение признаков при комбинации данных функций потерь можно увидеть на рисунке 2.3. Таким образом, прямое объединение этих двух функций потерь может повысить производительность, но это не оптимальное решение.

Батч-нормализация помогает избежать переобучения и улучшает производительность исходной модели. Также слой батч-нормализации может сглаживать распределение признаков в пространстве векторов. Для ID Loss слой батч-нормализации усиливает компактность внутри класса. Он также может улучшить значение этой метрики, так как признаки, близкие к аффинному центру, не имеют четких границ решений и трудны для различения. Тем не менее, такой слой увеличивает радиус кластеров внутриклассовых признаков для Triplet Loss, что ухудшает разделимость.

Для решения этой проблемы используется специальная структура, изображенная на рисунке 2.4.

Данная структура добавляет слой батч-нормализации после признаков и перед классификационными полно связанными слоями. Слои батч-нормализации и полно связанный слой инициализируются с использованием метода инициализации Кайминга. Признак до слоя BN обозначается как f_t. Мы пропускаем f_t через слой BN, чтобы получить признак f_i. На этапе обучения f_t и f_iиспользуются для вычисления Triplet Loss и ID

Loss соответственно. На рисунке 5 показано, что f_t не только сохраняет компактное распределение, но и приобретает знания от ID Loss. Под влиянием слоя BN и ID Loss распределение f_i становится «головастикообразным». По сравнению с ID+Triplet loss, f_i имеет четкие границы решений из-за более слабого влияния Triplet Loss.

На этапе выполнения мы выбираем f_i для выполнения задачи ReID.

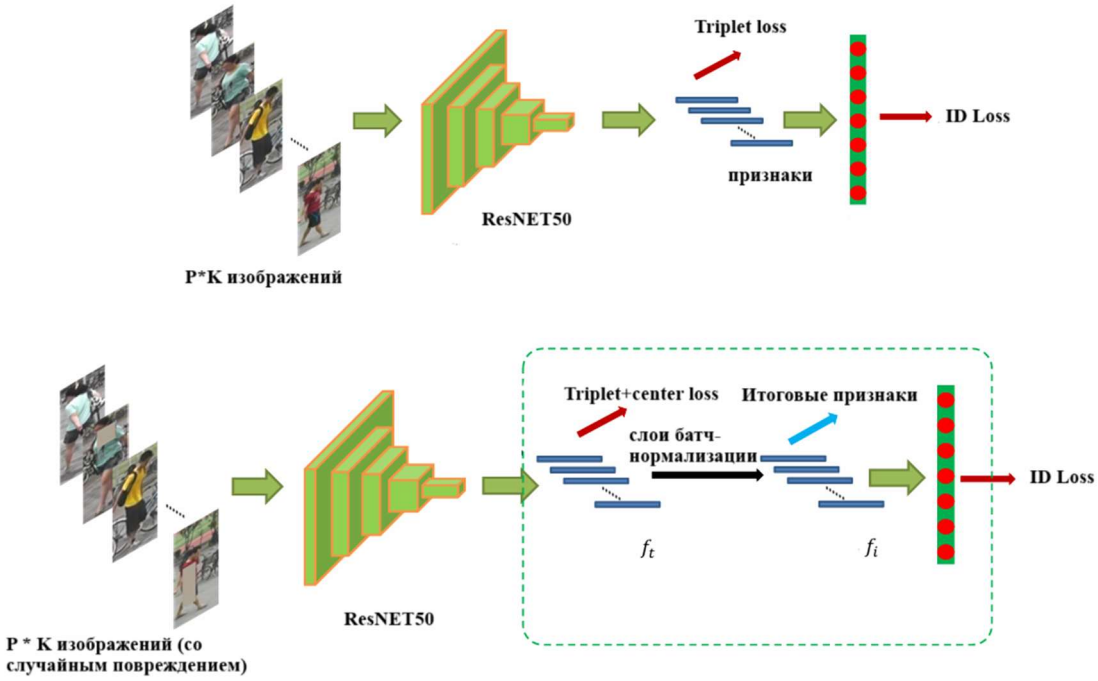

Рисунок 2.4 – сверху – старый метод обучения модели для задачи реидентификации, ниже – новый метод

Изменение функции потерь

Задача реидентификации обычно рассматривается как задача поиска/ранжирования. Такие метрики, как CMC-кривая и mAP, определяются результатами ранжирования, но игнорируют кластеризацию. Однако для некоторых приложений ReID, таких как задача трекинга, важным шагом является определение порога расстояния для разделения положительных и отрицательных объектов.

Классическая триплетная функция потерь учитывает только разницу между d_p и d_n, игнорируя их абсолютные значения. Например, если $d_p = 0.3$ и $d_n = 0.5$, триплетная функция потерь равна 0.1. В другом случае, если $d_p = 1.3$ и $d_n = 1.5$, триплетная функция потерь также равна 0.1. Таким образом, триплетная функция потерь определяется двумя случайно выбранными

идентификаторами объекта. Так, обучение на $d_p < d_n$ происходит только для одного экземпляра. Кроме того, внутриклассовая компактность игнорируется.

Чтобы компенсировать недостатки триплетной функции потерь, будет использоваться еще одна функция потерь, называемая center loss. Она обучает признаки быть ближе к центру своего класса. Функция потерь center loss формулируется следующим образом (2.20):

$$L_c = \frac{1}{2}\sum_{j=1}^{B}\left\|f_{t_j} - c_{y_j}\right\| \quad (2.20)$$

Где $\|x\|$ – евклидова норма, B – размер батча, c_{y_j} – центры векторов класаа y_j

Эта формулировка хорошо характеризует внутриклассовые вариации. Минимизация center loss улучшает внутриклассовую компактность. Наша итоговая функция потерь выглядит следующим образом (2.21):

$$L = L_{ID} + L_{Tri} + \beta * L_C \quad (2.21)$$

Где β – параметр, контролирующий вклад center loss.

Таким образом, как видно из рисунка 2.5, при данной функции потерь лучшим представлением объекта для нашей задачи является вектор f_i

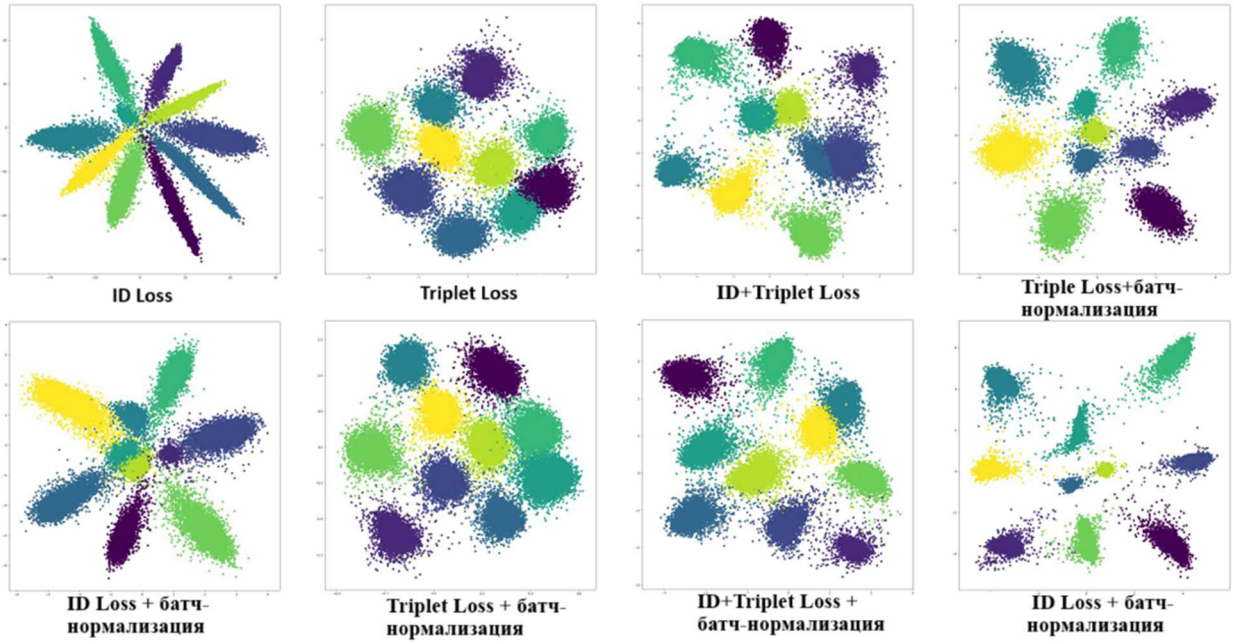

Рисунок 2.5 – 2D визуализация распределения признаков в пространстве векторов при различных структурах и функциях потерь на наборе данных

Сглаживание представлений внешнего вида объектов

Хотя механизм хранения признаков в DeepSORT может сохранять информацию на длительном временном интервале, он чувствителен к шуму в детекциях. Чтобы решить эту проблему, мы заменяем механизм хранения признаков на стратегию обновления признаков, которая обновляет состояние внешнего вида e_i^t для i-го треклета на кадре t с использованием экспоненциального скользящего среднего (EMA) следующим образом (2.22):

$$e_i^t = \alpha\, e_i^{t-1} + (1-\alpha) f_i^t \tag{2.22}$$

Где f_i^t — это вектор внешнего вида соответствующей ограничивающей рамки, а β - коэффициент инерции. Стратегия обновления с помощью EMA использует информацию об изменениях признаков между кадрами и способна подавлять шум в детекциях. Эксперименты показывают, что это не только повышает качество сопоставления, но и снижает временные затраты.

2.6 Детекция объектов в движении на больших высотах

Несмотря на современность и эффективность современных методов компьютерного зрения, основанного на нейронных сетях, они подходят не для всех задач. Методы, основанные на глубоком обучении, работают хорошо в

Рисунок 2.6 – изображение, полученное с дрона, находящегося на высоте ~300 метров

ситуациях, когда объекты имеют выраженные прихнаки, которые нейронные сети могут распознать. Однако когда масштаб объектов, подлежащих детекции, очень мал (рисунок 2.6), нейронные сети не находят необходимых признаков объектов.

В случае с мелкомасштабными объектами, классические методы компьютерного зрения будут показывать лучшие результаты.

Здесь отслеживание объектов основано на идее дифференцирования кадров. В основе этого подхода лежит идея, что если объект на сцене перемещается, то его положение будет отличаться от кадра к кадру. Следовательно, если вычесть один кадр из другого, можно выделить те области изображения, которые изменились, а это, как правило, и есть движущиеся объекты.

В самом простом случае дифференцирование кадров заключается в вычислении разности яркости пикселей между двумя последовательными кадрами видео. Если яркость пикселя изменилась значительно, это означает, что в данном месте произошло некоторое движение. В противном случае, если разность мала или равна нулю, можно предположить, что в этом месте сцена осталась неизменной. После вычисления разности обычно применяется пороговая фильтрация: все значения, меньшие заданного порога, обнуляются, а остальные интерпретируются как признаки движения. Таким образом, результатом является бинарное изображение, где белым отмечены области движения, а чёрным — фон.

Однако в реальных условиях всё несколько сложнее. Освещение сцены может меняться даже при отсутствии движения, и это тоже приведёт к изменению яркости пикселей. Кроме того, цифровой шум камеры, особенно при низком освещении, также может вызывать ложные срабатывания. Поэтому дифференцирование кадров часто сопровождается дополнительной обработкой, такой как сглаживание (например, гауссовым фильтром), морфологические операции (эрозия и дилатация), а также применением адаптивных порогов, которые учитывают локальные особенности изображения. Все эти меры направлены на уменьшение количества ложных срабатываний и выделение только значимого движения.

Также важно учитывать, что в простом дифференцировании участвуют только два кадра — текущий и предыдущий. Это делает метод очень чувствительным к быстрым, кратковременным изменениям, но также может приводить к потере информации о медленном, плавном движении. Чтобы компенсировать этот недостаток, иногда используют трёхкадровую дифференциацию: берут три последовательных кадра и вычисляют логическое пересечение движений между первым и вторым, а также вторым и третьим кадрами. Это позволяет лучше отфильтровывать шум и более точно

определять настоящие движения, однако для задачи детекции объектов на изображениях с дронов можно ограничиться двумя кадрами.

Однако у метода есть и очевидные ограничения. Он не способен различать типы движения, не может отслеживать конкретные объекты и страдает от высокой чувствительности к изменениям освещения. Также этот метод не будет работать при движениях камеры – в этом случае кроме действительно движущихся объектов, градиенты будут иметь и другие объекты в кадре.

Чтобы обработать данную ситуацию, можно использовать оптический поток с определением ключевых точек методом Ши-Томаси.

Применение дифференцирования кадров с оптическим потока для данной задачи оправдано в ситуациях, которые не предполагают резкого изменения условий освещения.

На рисунке 2.7 изображена схема работы алгоритма детекции с использованием оптического потока и дифференцирования кадров.

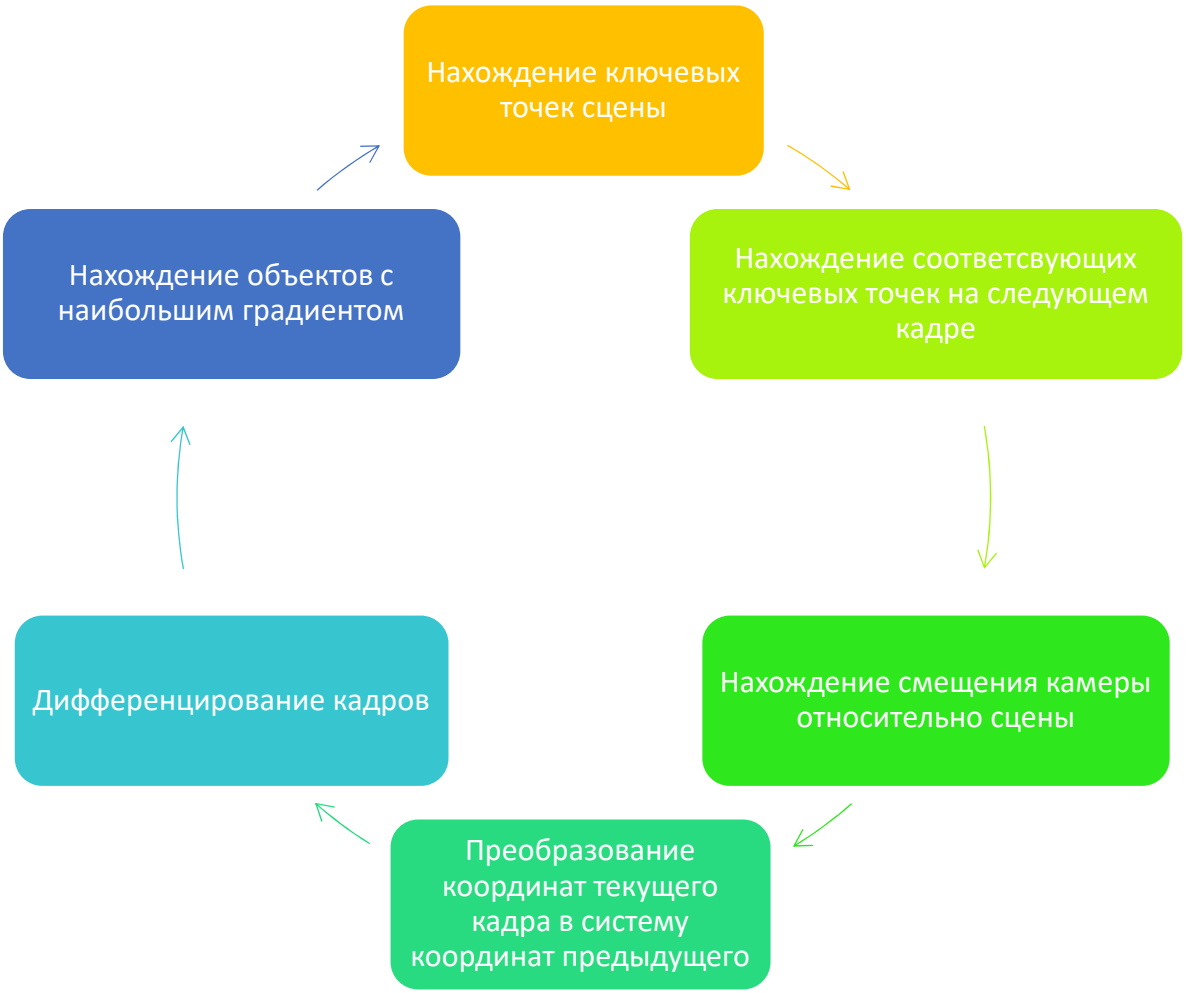

Рисунок 2.7 - схема работы алгоритма детекции с использованием оптического потока и дифференцирования кадров

2.7 Вывод

Таким образом, в данной главе были рассмотрены:

1. Методы детекции статических объектов, основанные на глубоком обучении. Среди них лучшей моделью оказалась YOLOv8, дообученная на собственном наборе данных
2. Методы детекции подвижных объектов на умеренных высотых (10 – 75 метров). В данном случае лучший результат показывают двухстадийные детекторы – на первом этапе определяются объекты в статике, а на втором происходит сопоставление объектов между кадрами.
3. Методы детекции подвижных объектов на больших высотах (больше 75 метров). В данном случае лучший результат показывают детекторы, основанные на алгоритмах классического компьютерного зрения – оптический поток и дифференцирование кадров.

Однако в связи с тем, что для реализации методов детекции подвижных объектов на больших высотах требуются дополнительные специализированные наборы данных, которые не были доступны в рамках текущего исследования. Следовательно, было принято решение сосредоточиться на методах, которые могут быть реализованы с использованием уже доступных данных, что позволило получить более точные результаты

ГЛАВА 3
ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

3.1 Выбор программных средств детекции мелких объектов

Для выбора средств разработки алгоритма были использованы 2 главных критерия: функциональные возможности средств и скорость их работы.

Так, для всего функционала был выбран язык программирования python – он является лидирующим с сфере как глубокого обучения, так и компьютерного зрения, особенно последние несколько лет, после появления оптимизированных библиотек.

Для разработки моделей глубокого обучения была выбрана библиотека pytorch. Это было обусловлено двумя главными причинами:

Во-первых, библиотека полностью написана на языке C++, что сильно ускоряет процесс выполнения кода, и, соответственно, приводит к ускорению всей системы.

Во-вторых, по сравнению со своим главным конкурентом в этой области – tensorflow, она имеет более широкий функционал. Также большинство вспомогательных библиотек написаны при помощи pytorch, и его использование облегчит интеграцию с данными библиотеками.

Для реализации алгоритма детекции в статике была использована библиотека ultralitics, так как, во-первых, она имеет множество моделей детекции, в том числе YOLOv8, а во-вторых, имеет открытый исходный код, что причиной активной поддержки данной библиотеки сообществом. Благодаря открытости данной библиотеки, множество моделей имеют свои предобученные на большом количестве данных версии в библиотеке hugginface, которая также была использована в данной работе.

Также для обработки изображений была использована библиотека OpenCV. Данная библиотека также имеет реализацию на C++, что делает ее предпочтительной для задач, где важна скорость обработки изображений. Кроме того, эта библиотека имеет реализации большинства популярных алгоритмов классического компьютерного зрения, такие как определение ключевых точек, вычисление оптического потока, вычисление коэффициента корреляции между изображениями, и многие другие необходимые функции.

3.2 Реализация алгоритма трекинга на языке python

Так как модифицированный алгоритм DeepSORT представляет собой многошаговый алгоритм трекинга объектов, для разных частей работы программы были подобраны свои инструменты.

1) Детекция объетов. Экспериментально было выведено, что лучшей архитектурой, справляющейся с данной задачей, является YOLOv8. В языке python ее реализацию можно найти в библиотеке ultralitics. Так как обучение данной модели – очень затратный процесс, веса также были взяты с hugginfaces, представляющего собой большой набор различных моделей.

2) Извлечение признаков из обнаруженных объектов. Как уже было упомянуто ранее, для данной задачи лучшим решением является глубокое обучение. Проблемой данного подхода является временная сложность работы нейронных сетей, особенно когда объектов много. Данная проблема частично решается благодаря использованию неглубоких сетей. В настоящее время создано множество архитектур, специализированных для задачи реидентификации. Все они, как правило, имеют не большое количество скрытых слоев. Одной из самых популярных библиотек, предоставляющих реализации этих сетей в python, является torchreid. Как видно из названия, библиотека основана на фреймворке pytorch. Также важно подобрать подходящие для поставленной цели веса.

3) Сопоставление треков. Данная часть реализована при помощи двух библиотек – filterpy, в которой реализован фильтр Калмана и scipy – для каскада ассоциаций.

4) Оценка движения камеры. Для нахождения параметров P преобразования изображения была использована библиотека OpenCV, в которой есть реализация ECC – функция cv.findTransformECC. Данная функция принимает на вход исходное изображение, преобразованное изображение и тип движения камеры, который может принимать следующие значения:

1. Трансляция: Первое изображение можно сдвинуть (перевести) на (x, y), чтобы получить второе изображение. Есть только два параметра x и y, которые нам нужно оценить.

2. Евклидово преобразование: Первое изображение является повернутой и сдвинутой версией второго изображения. Есть три параметра — x, y и угол. Когда квадрат подвергается евклидову преобразованию, его размер не меняется, параллельные линии остаются параллельными, а прямые углы остаются неизменными после преобразован

3. Аффинное преобразование: Аффинное преобразование представляет собой комбинацию вращения, перемещения (сдвига), масштабирования. Это преобразование имеет шесть параметров. Когда квадрат подвергается аффинному преобразованию, параллельные линии остаются параллельными, но линии, сходящиеся под прямым углом, больше не остаются ортогональными.

4. Гомография: Все преобразования, описанные выше, являются 2D преобразованиями. Они не учитывают 3D-эффекты. С другой стороны, гомографическое преобразование может учитывать некоторые 3D-эффекты. Это преобразование имеет 8 параметров. Квадрат, преобразованный с помощью гомографии, может превратиться в любой четырехугольник.

Важнейшей частью разработки практически любого алгоритма, основанного на нейронных сетях, является подбор гиперпараметров. Являясь многостадийными трекерами, SORT и DeepSORT имеют большое количество гиперпараметров. В то же время, так как DeepSORT является улучшением SORT, практически все параметры, актуальные для SORT, актуальны и для DeepSORT. Более того, так как SORT является составной частью DeepSORT, гиперпараметры, оптимальные для первого, будут оптимальны и для второго, то есть для подбора общих параметров достаточно протестировать их для SORT. Были подобраны следующие гиперпараметры:

- NMS_iou = 0.7. Это значение iou-метрики, при котором принцип подавления немаксимумов будет удалять одну из ограничивающих рамок.
- orig = True. Этот параметр отвечает за то, что при прогнозировании новых состоянй фильтра Калмана будут учитываться только новые детекции, то есть объекты, передаваемые в следующий кадр только фильтром Калмана, учтены не будут. Это значительно уменьшает количество ложноположительных результатов, и, как следствие, переключение идентификаторов.
- iou_threshold = 0.3. Это минимальное значение метрики iou, при котором может произойти назначение.

Гиперпараметры, существующие только у DeepSORT:

- max_age=50. Данный параметр позволяет хранить информацию об объекте на протяжении 50 кадров, повышая вероятность вернуть идентификатор в случае ошибочного присваивания.
- gating_only_position = True. Благодаря этому, фильтр Калмана предсказывает исключительно позицию объекта для своих предсказаний, не учитывая размеры объекта.
- max_cosine_distance = 0.07. Данный гиперпараметр отвечает за отклонение невозможных согласно формуле (2.4) назначений.

- embedeer - mobilenetv2_x1_4 – разновидность сети mobilenetv2 с множителем ширины 1.4. Это означает, что количество скрытых сверточных слоев будет в 1.4 раза больше чем в обычной версии.

Гиперпараметры, введенные в рамках данной работы:

- embeeder_wts – веса нейронной сети для извлечения признаков. Были выбраны веса модели, обученной на ImageNet.
- λ - сглаживающий фактор при подсчете матрицы стоимостей. DeepSORT устанавливает $\lambda = 1$, однако в ходе экспериментов было выяснено, что $\lambda = 0.98$ будет оптимальнее в рамках данной работы.
- α - сглаживающий фактор при подсчете вектора внешнего вида объекта. Оптимальным значением является $\alpha = 0.9$
- $\beta -$ вклад center loss в итоговую функцию потерь. Устанавливается равным 0.0005.

3.3 Тестирование полученной модели

Модель была протестирована на изображениях и последовательностях изображений с дронов, сделанных на высоте 5-15 метров при различных условиях. Ниже, на рисунках 3.1, 3.2, 3.3, приведены примеры детекции объектов на последовательности изображений.

Ниже приведены лишь несколько примеров результатов тестирования. В ходе экспериментов были сделаны следующие выводы:

- Модель практически идеально работает в ситуациях со статичной или плавно движущейся камерой, при любом количестве объектов при условии, что они не перекрывают друг друга.

- Модель работает лучше своих предшественников в ситуациях с резким движением камеры и полном перекрытии объектов, однако все еще может назначать неправильные идентификаторы.

- Модель плохо детектирует объекты при нахождении дрона на высоте больше 50 метров. Это связано с тем, что при таких высотах масштаб объектов становится слишком мал даже для модифицированной детекции.

- Модель, в отличие от классического алгоритма SORT, не применима в задачах реального времени.

3.4 Вывод

Таким образом, в данной главе были приведены технические детали реализации алгоритма и анализ его работы. Несмотря на то, что модель имеет свои недостатки, качество ее работы превосходит качество работы предшественников в большинстве ситуаций, вызывающих трудности

детекции. Также стоит отметить, что для данной задачи тяжело разработать безошибочный алгоритм, так как ее решение может оказаться затруднительным даже для человека.

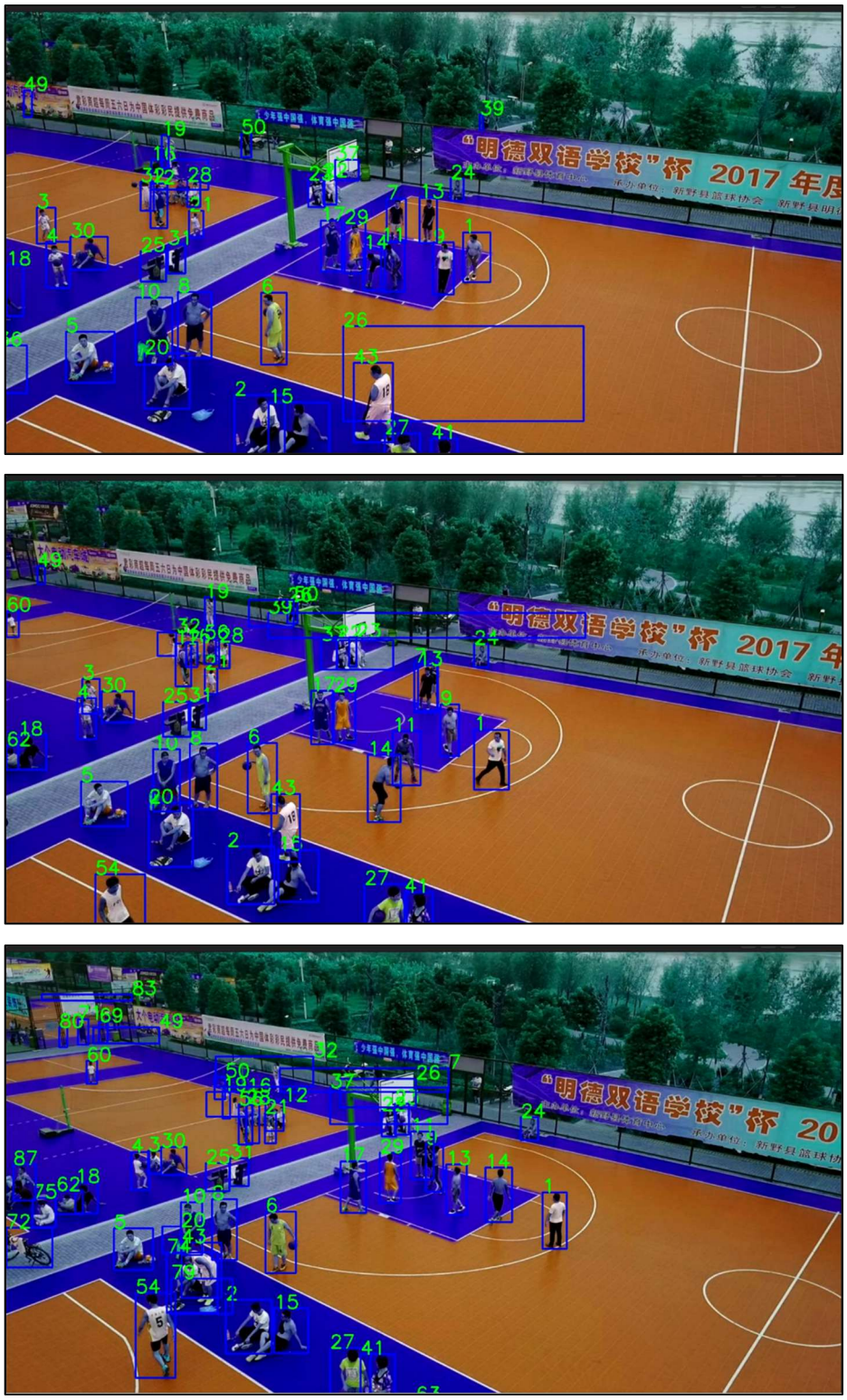

Рисунок 3.1 – последовательность изображений с плавными движениями камеры и большим количеством объектов, которые могут закрывать друг друга.

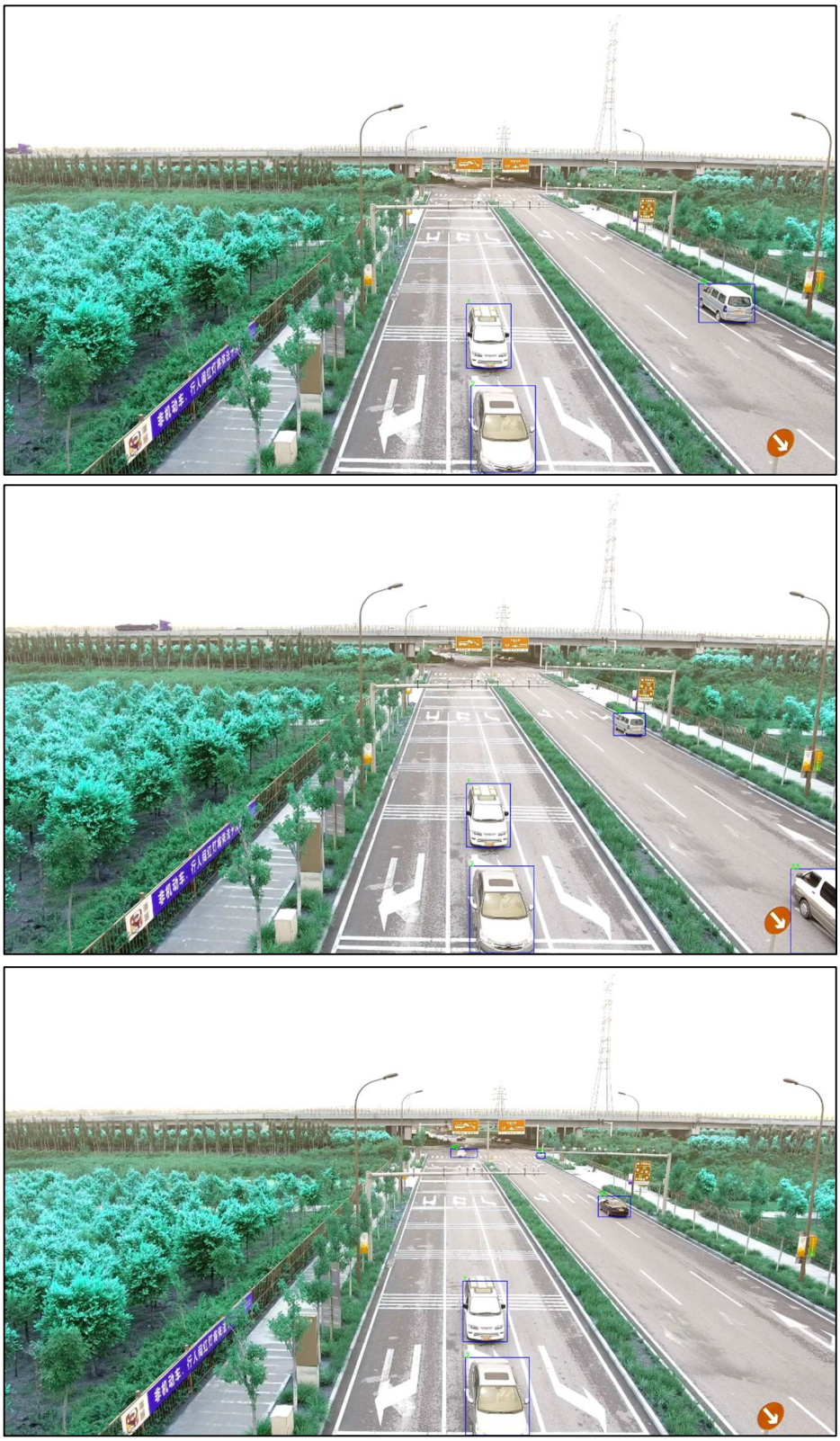

Рисунок 3.2 – последовательность изображений с практически статичной камерой и малым количеством объектов.

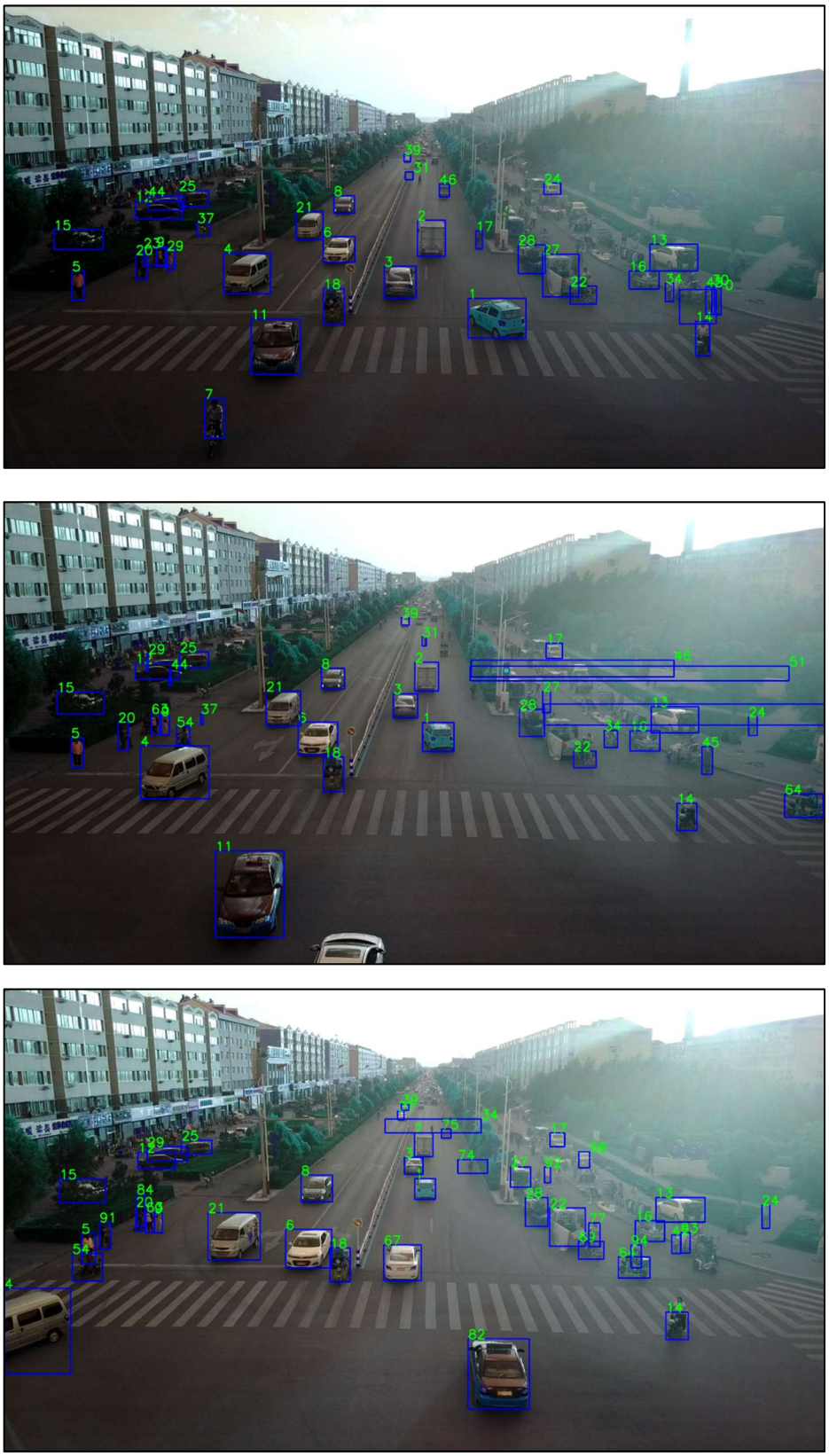

Рисунок 3.3 – последовательность изображений с резким движением камеры между 1 и 2 кадром и большим количеством объектов, не перекрывающих друг друга.

ЗАКЛЮЧЕНИЕ

Таким образом, задача детекции объектов на изображениях дронов была решена при помощи алгоритма, основанного на детекции статических объектов и развивающего идею метода DeepSORT. В отличие от классического алгоритма DeepSORT, приведенный в рамках данной работы алгоритм использует более точную модель извлечения признаков, оценивает параметры движения камеры и использует модель детектирования, адаптированную под малые объекты.

Разработанная модель демонстрирует достаточно качественные предсказания для последовательности изображений из набора данных VisDrone. По сравнению со своими предшественниками, разработанный алгоритм улучшает качество идентификации объектов, однако главным ее недостатком является время работы, не позволяющее применять этот алгоритм в задачах трекинга в реальном времени.

Полученная в ходе выполнения данной работы система может быть использована для множества целей, включая наблюдение за объектами, определения уровня загруженности улиц и дорог, контроль за подчинением правилам дорожного движения.

СПИСОК ЛИТЕРАТУРЫ

1. Интернет-портал Ultralytics [Электронный ресурс]. – Режим доступа: https://docs.ultralytics.com/. – Дата доступа: 21.03.2024.
2. Клетте, Р. Компьютерное зрение. Теория и алгоритмы / пер. с англ. А.А. Слинкин. – М.: ДМК Пресс, 2019. – 506 с.
3. Akyon, F. Slicing Aided Hyper Inference and Fine-Tuning for Small Object Detection / Akyon, F., Altinuc, S., Temizel, A. // 2022 IEEE International Conference on Image Processing (ICIP). – Bordeaux, France, 2022. – P. 966–970.
4. Bewley, A. Simple online and realtime tracking / Bewley, A. [et al.] // 2016 IEEE International Conference on Image Processing (ICIP). – Phoenix, USA, 2016. – P. 3464–3468.
5. Du, Y. StrongSORT: Make DeepSORT Great Again / Du, Y. [et al.] // IEEE Transactions on Multimedia. – 2023. – Vol. 25. – P. 8725–8737.
6. Girshick, R. Fast R-CNN / Girshick, R. // Proceedings of the IEEE International Conference on Computer Vision (ICCV). – Santiago, Chile, 2015. – P. 1440–1448.
7. Girshick, R Feature Hierarchies for Accurate Object Detection and Semantic Segmentation / Girshick, R., Donahue, J., Darrell, T., & Malik, J. Rich // Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). – Columbus, USA, 2014. – P. 580–587.
8. Howard, A.G. Mobilenets: Efficient convolutional neural networks for mobile vision applications. – arXiv preprint arXiv:1704.04861, 2017.
9. Liu, W. Ssd: Single shot multibox detector / Liu, W. [et al.] // Computer Vision–ECCV 2016: 14th European Conference – Amsterdam, The Netherlands, 2016, – P. 21–37.
10. Ren, S. Feature detection with automatic scale selection / Ren, S. [et al.] // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2015. – Vol. 39. – P. 1137–1149.
11. Shi, J. Good features to track / Shi, J. Tomasi, C. // 1994 Proceedings of IEEE Conference on Computer Vision and Pattern Recognition. – Seattle, USA, 1994. – P. 593–600.
12. Wojke, N. Simple online and realtime tracking with a deep association metric / Wojke, N., Bewley, A., Paulus, D // 2017 IEEE International Conference on Image Processing (ICIP). – Beijing, China, 2017. – P. 3645–3649.