Федеральное государственное автономное образовательное учреждение высшего профессионального образования
«Национальный исследовательский университет «Высшая школа экономики»

Московский институт электроники и математики им. А.Н. Тихонова

Прикладная математика, Информатика и вычислительная техника, Компьютерная безопасность, Системный анализ и математические технологии
Бакалавриат, Специалитет, Магистратура

О Т Ч Е Т
по проектной работе

Разработка системы распознавания (ML) и интеллектуального контроля ручных операций в промышленном производстве

Выполнили студенты:
Солдатов Алексей Валерьевич, группа МСМТ233
Гриднева Екатерина Владимировна, группа СКБ221
Ляпунова Софья Александровна, группа БИВ214
Мазиков Ярослав Андреевич, группа БПМ201
Какурин Василий Владимирович, группа БПМ203

_____
(подпись)

Руководитель проекта:
Минченков Виктор Олегович
Руководитель направления:
Сергеев Антон Валерьевич

_____ _____
(оценка) *(подпись)*

_____
(дата)

Москва 2024

Аннотация

Данная работа посвящена разработке интеллектуальной системы контроля ручных операций в промышленном производстве с использованием методов глубинного обучения. Цель работы – повышение качества работы системы и внедрение в нее новых сценариев сборки. Исследовательский стенд, использованный для тестирования системы, представляет собой автоматизированное рабочее место, включающее компьютер, монитор, проектор и промышленные камеры Basler. Стенд позволяет в режиме реального времени контролировать правильность выполнения операций, выявлять типовые ошибки и нарушения, такие как неправильная последовательность действий, ошибки в выборе компонентов и нарушение правил техники безопасности. Использование данной системы на реальном производстве помогает снизить процент брака, увеличить производительность труда и уменьшить число несчастных случаев.

В процессе разработки системы был использован язык программирования Python. Были изучены и успешно применены алгоритмы трекинга объектов, новая модель детекции YOLO v8 и библиотека mediapipe. Для повышения качества работы приложения были собраны и размечены новые наборы данных и внедрены алгоритмы трекинга объектов. С целью повышения скорости работы системы и грамотного использования ресурсов CPU и GPU были исследованы методы параллельных и конкурентных вычислений. Итоговая улучшенная версия приложения контроля ручных операций была успешно запущена и протестирована на стенде в лаборатории и готова к использованию на реальных предприятиях.

Вклад в работу каждого из участников

Для решения задачи контроля ручных операций в промышленном производстве необходим комплексный подход. Студентами была разработана новая версия приложения для контроля сборочного процесса. Основные задачи, которые были решены в ходе данного проекта, следующие:

- внедрение алгоритмов трекинга объектов для повышения качества детекции;
- написание приложения контроля за сборкой FPV-дрона;
- параллелизация приложения для достижения оптимального потребления вычислительных ресурсов приложением и увеличения FPS в режиме работы в реальном времени;
- съемка и разметка новых наборов данных для повышения качества детекции мелких деталей;
- изучение и обучение новых моделей детекции, отслеживание посторонних объектов.

За изучение и внедрение алгоритмов трекинга отвечал Мазиков Ярослав (главы 2-4). За написание приложения и разработку сценариев сборочного процесса FPV-дрона приложения отвечал Солдатов Алексей (глава 5), за параллелизацию приложения контроля сборки – Ляпунова Софья (глава 6), за съёмку и разметку новых наборов данных – Гриднева Екатерина (глава 7), за изучение и обучение новых моделей детекции и отслеживание посторонних объектов – Какурин Василий (глава 8). Главы 1 и 9 являются общими.

1 Введение

Компьютерное зрение прочно вошло в повседневную жизнь и широко применяется в промышленности. В последние годы наблюдается стремление к полной автоматизации производственных процессов и замене людей роботами. Однако для многих компаний такие преобразования остаются финансово неприемлемыми из-за высоких затрат на оборудование и продолжительного периода окупаемости. Тем не менее, производственные предприятия продолжают стремиться к улучшению качества продукции, уменьшению числа дефектов и обеспечению безопасности труда.

Одним из эффективных подходов к решению этих задач является внедрение интеллектуальных систем контроля ручных операций. Эти системы значительно снижают влияние человеческого фактора в сборочных процессах, что способствует повышению общей эффективности и надежности производства.

Создание и внедрение системы интеллектуального контроля ручных операций, основанной на алгоритмах трекинга и методах глубокого обучения, является значительным шагом к автоматизации и оптимизации производственных процессов. Разработанное приложение демонстрирует высокий потенциал для коммерческого использования и промышленного внедрения, что подтверждается успешными испытаниями на проектном стенде и готовностью к проверкам в реальных производственных условиях.

1.1 Цель работы

Цель данной работы — разработка системы интеллектуального контроля ручных операций в промышленном производстве с использованием методов глубинного обучения и алгоритмов трекинга объектов. Разрабатываемое приложение обеспечивает контроль за действиями оператора сборщика в режиме реального времени и оперативно уведомляет его об ошибках. Такое приложение может быть адаптировано для контроля за сборкой различных комплексных изделий, что позволяет снизить процент брака, увеличить производительность труда и уменьшить число несчастных случаев на производстве.

Внедрение алгоритмов трекинга в совокупности с моделями детекции объектов на изображении позволяет эффективнее решить такие задачи как:

- отслеживание положений объектов в рабочей области;
- повышение стабильности работы моделей детекции;
- повышение точности определения положений объектов моделями детекции;
- восстановление пропущенных детекций;
- снижение количества ложных срабатываний моделей детекции;
- реидентификация объектов.

1.2 Актуальность

Автоматизация и интеллектуальный контроль процессов ручной сборки в промышленности становятся все более актуальными в условиях современной экономики. Внедрение таких систем позволяет не только повысить качество и надежность производственных процессов, но и значительно сократить затраты на обучение персонала и снизить риски, связанные с человеческим фактором. Особую значимость это приобретает на предприятиях оборонно-промышленного комплекса (ОПК), где требования к точности и последовательности операций особенно высоки.

На сегодняшний день существуют конкретные примеры внедрения подобных систем на производственных предприятиях. Так, система компании «Мотив» обеспечивает контроль за оператором в процессе сборки турбокомпрессора, отслеживая правильность выполнения 14 операций и выявляя 4 типовых вида ошибок [1]. Пример работы системы компании «Мотив» представлен на Рис. 1.

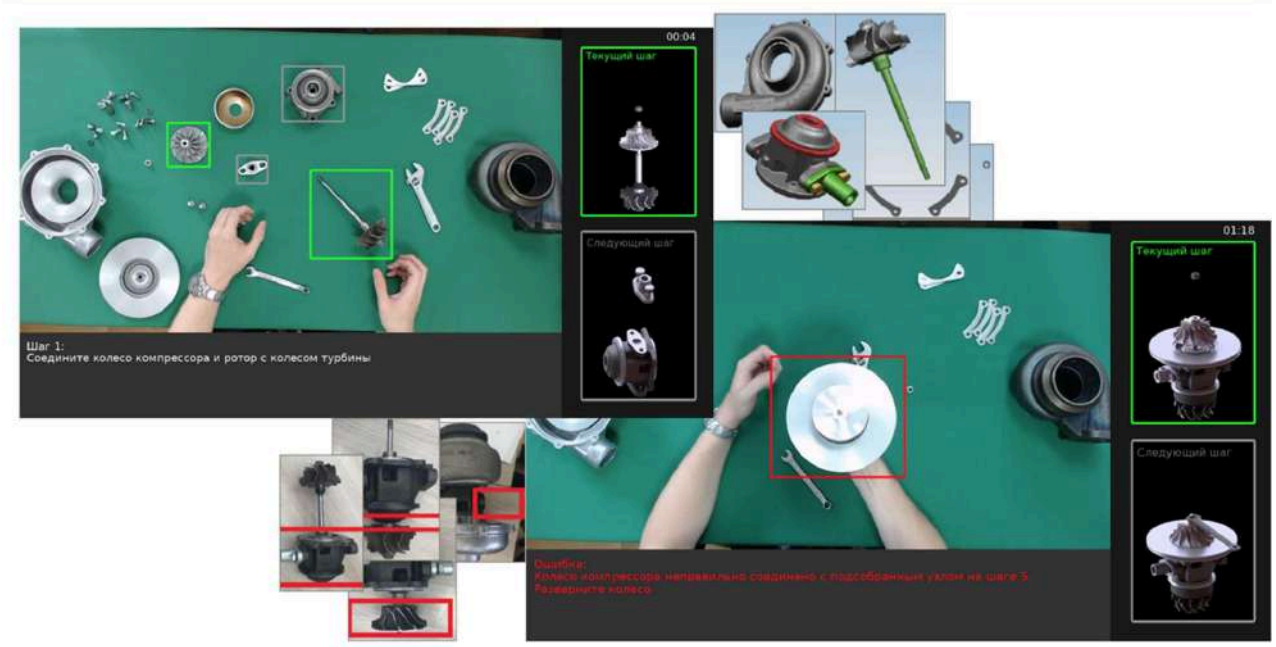

Рис. 1. Система контроля ручных операций компании «Мотив»

1.3 Описание стенда

Исследовательский стенд представляет собой автоматизированное рабочее место оператора, выполняющего сборку и разборку многокомпонентных изделий. Стенд включает в себя металлический каркас, столешницу, компьютер, монитор, проектор, две камеры Basler, две веб-камеры Logitech, мышь и клавиатуру. Основной задачей стенда является обеспечение контроля правильности выполнения регламентированных операций, выявление типовых ошибок и нарушений, таких как неправильная последовательность действий, ошибки в выборе компонентов и инструментов, а также нарушения техники безопасности. Ключевые требования к стенду включают:

- обеспечение работы в реальном времени с минимальной задержкой;
- обработка мелких и визуально схожих деталей размером до 10 мм;
- возможность одновременного контроля до 20 объектов в зоне сборки;
- хорошее освещение зоны сборки;
- фиксированная область контроля (рабочего стола).

На Рис.2 представлен исследовательский стенд контроля ручных операций.

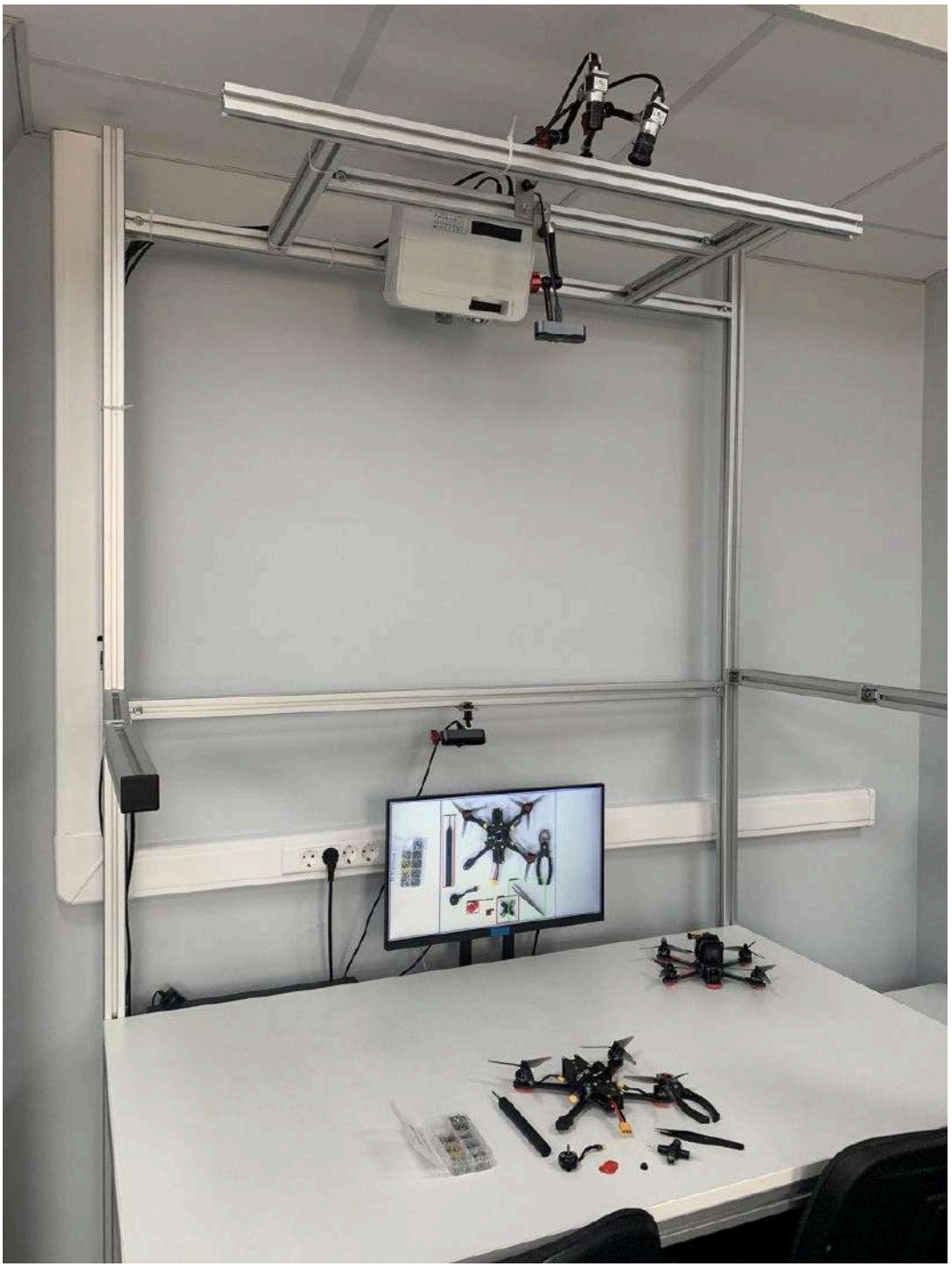

Рис. 2. Исследовательский стенд контроля ручных операций

Для создания новой версии приложения, ориентированной на сборку FPV-дронов, разработаны и интегрированы алгоритмы трекинга объектов. Эти алгоритмы позволяют не только детектировать объекты, но и отслеживать их перемещения, что особенно важно в условиях динамического производства. В рамках данной работы были разработаны сценарии, включающие перенос винтов, исправление ошибок детекции, отслеживание деталей и контроль сборки шасси и моторов. Внедрение этих сценариев позволило значительно повысить

эффективность и точность системы, делая её пригодной для решения сложных задач ручной сборки в промышленном производстве.

2 Алгоритмы трекинга объектов

В настоящее время алгоритмы трекинга объектов занимают ключевое место в широком спектре прикладных областей, включая системы видеонаблюдения, автономное управление транспортными средствами и робототехнику. Данные алгоритмы обеспечивают возможность автоматического отслеживания движения объектов в пространственно-временных координатах, что является критически важным для обработки и анализа данных в реальном времени. С учетом растущих объемов информации и требований к оперативности обработки данных, актуальность развития и совершенствования алгоритмов трекинга постоянно возрастает. В данном разделе предложен детальный обзор существующих алгоритмов трекинга, включая их классификацию, механизмы функционирования, а также оценка преимуществ и ограничений каждого подхода. Особое внимание уделено анализу различных методологий и технологий, применяемых в этой области, что позволяет оценить текущее состояние научных исследований и перспективы дальнейшего развития данной отрасли.

2.1 Обзор предыдущих исследований

В самом начале развития технологий отслеживания, в середине 20 века, алгоритмы трекинга были реализованы с помощью аналоговых систем, представляющих из себя сложные компьютерные комплексы. Подобные громоздкие устройства использовали передовые на тот момент датчики и радары для обнаружения различных объектов. Примером подобного рода радара может служить индикаторная электронно-лучевая трубка кругового обзора, используемая для визуализации знакографической, телевизионной и радиолокационной информации в разнообразных системах и аппаратах. Во время Второй Мировой Войны подобные устройства использовались для построения оптически спроецированных изображений плана местности и отслеживания положений целевых объектов через равные промежутки времени. Другим примером системы аналогового трекинга

может служить компьютер управления огнем Mark 37, который по сути являлся электромеханическим компьютером, связанным с орудийными установками и предсказывающим движения цели для расчета решения по управлению огнем.

Ближе к концу 20 и в начале 21 веков алгоритмы трекинга значительно эволюционировали и перешли от аналоговых систем к цифровым, что позволило значительно увеличить их возможности и устранить различные ограничения, присущие аналоговым системам. Если не учитывать факт использования алгоритмов отслеживания объектов на видео в военно-гражданских системах, то стоит отметить весьма быстрое развитие систем видеонаблюдения, использующих все более и более развитые технологии компьютерного зрения и применяемых в основном в повседневных или производственных целях. Многие отдельные исследователи и даже целые компании с увеличением вычислительных мощностей и развитием нейросетевых технологий поспешили внести свой вклад в развитие технологий компьютерного зрения для решения задач автономного вождения [2], взаимодействия человека и робота [3], мониторинга толпы [4-5] и других. Считается, что первое появление рабочего решения задачи трекинга объектов на наборе последовательных изображений при использовании цифровых технологий (лишь компьютер и камера) отмечено 2010 годом, когда свет увидела статья о применении алгоритма трекинга MOSSE [6], основанном на корреляционных фильтрах. С тех пор в течение последнего десятилетия алгоритмы трекинга лишь становились изощренней, сложнее и распространеннее.

В зависимости от поставленных целей и доступных технических средств появляется несколько возможных направлений для решения задачи трекинга объектов, а именно трекинг единственного объекта в кадре [7], трекинг нескольких объектов одновременно [8-9], пространственный 3D трекинг [10] или же сегментация объекта на видео [11-12]. Очевидно, что трекинг единственного объекта является наиболее простым для решения. Суть данного метода состоит в отслеживании конкретного размеченного объекта в видеопоследовательности кадров как показано на Рис.3.

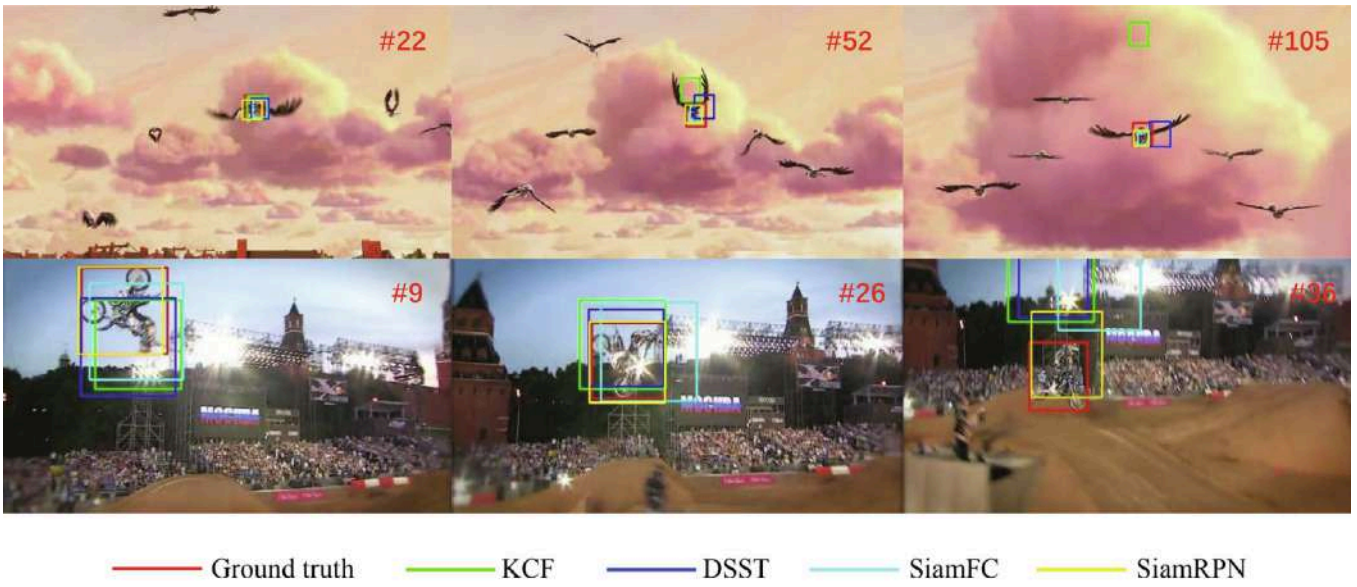

Рис. 3. Изображение показывает Ground Truth разметку в видеопоследовательности, а также предсказанные результаты KCF [13], DSST [14], SiamFC [7] и SiamRPN [15]

Алгоритмы трекинга единственного объекта в кадре в основном базируются на 4 этапах обработки и настройки:

1) Считывание всех изображений из видеопотока и их характеристик, получаемых вручную или с помощью сверточных алгоритмов и нейросетей;

2) Предсказание нескольких возможных ограничивающих зон для объекта (Bounding Box). Классическими методами для генерации предсказаний являются многочастичные фильтры [16] и скользящие окна [17-18];

3) Инициализация модели для трекинга и выбор наилучших областей интереса (Regions of Interest);

4) Своевременное обновление данных модели для подстраивания под окружающие и технические условия при использовании алгоритма в дальнейшем.

В большинстве случаев сценарии, для которых используются алгоритмы отслеживания, сложны и должны учитывать множество факторов, таких как затенение, деформация объекта, сходство с другими объектами, изменение масштаба, вариации освещения, низкое разрешение и высокая скорость движения. Эти факторы создают множество проблем для точного отслеживания отдельных объектов. В рамках задачи трекинга единственного объекта на изображении выделяется несколько основных направлений и подходов к решению:

- Подходы, основанные на обнаружении, которые занимаются поиском объектов на каждом кадре, а затем связывают эти обнаружения между последовательными кадрами (например, DeepSORT [19]);
- Методы, основанные на предсказании, которые используют фильтры, такие как фильтр Калмана [20] или многочастичные фильтры [16] для предсказания положения объекта на основе его предыдущих состояний;
- Подходы, применяющие нейронные сети для изучения и трекинга объектов без явного выделения признаков (например, SiamFC [7]).

2.2 Описание алгоритмов

В данной работе будут подробно рассматриваться только современные методы трекинга объектов на видео с целью их дальнейшего применения в разработке системы интеллектуального контроля. Рассмотрим подробнее самые известные вариации алгоритмов трекинга, включая как классические методы (их в данной работе лишь упомянем), основанные на корреляционных фильтрах методы и нейросетевые подходы (сверточные, рекуррентные и «сиамские» модели).

2.2.1 Классические алгоритмы трекинга

Основными методами классического трекинга объектов являются методы оптического потока (optical flow), методы фильтров и ядерные методы (kernel-based methods) [21]. Методы оптического потока вычисляют тенденцию движения объекта путем нахождения соответствующих пикселей между двумя кадрами, тем самым получая информацию для осуществления трекинга. В 1981 году Бертольд Хорн и Брайан Шанк [22] объединили двумерное поле скоростей с признаками градаций серого для создания уравнения ограничения оптического потока и предложили метод вычисления оптического потока. Брюс Д. Лукас и Такео Канаде [23] предположили, что оптический поток постоянен в окрестности одного пикселя, и решили основные уравнения оптического потока для каждого пикселя в окрестности. Оптический поток, полученный путем объединения нескольких пикселей, является более точным. Для решения проблемы, связанной с неспособностью метода оптического потока обрабатывать быстро движущиеся объекты, Жан-Ив Буге [24] предложил

пирамидальный алгоритм оптического потока, который масштабирует изображение в пирамидальную форму, а затем решает его слой за слоем для получения оптического потока исходного изображения.

Задача трекинга объектов обрабатывает непрерывные кадры, и полное использование взаимосвязи между кадрами оказывает большое влияние на результаты трекинга. Питер С. Мейбек [25] предложил фильтры Калмана для прогнозирования текущего состояния на основе предыдущего состояния и последующего корректирования результата прогноза на основе наблюдательной информации. Расширенные фильтры Калмана [26] выполняют разложение нелинейной модели в ряд Тейлора первого порядка для получения аппроксимированной линейной модели, а затем используют фильтры Калмана для оценки состояния. Симон Жюлье [27] предложил несмещенные фильтры Калмана для решения проблемы больших вычислений и линейных ошибок в расширенных фильтрах Калмана. Катя Нуммиаро и Люк Ван Гуль [28] предложили многочастичные фильтры для работы в условиях неопределенной модели.

Дорин Комanисиу [29] внедрил метод MeanShift в задачи трекинга объектов и предложил алгоритм трекинга на основе ядра. Алгоритм извлекает цветовую гистограмму начального объекта и текущего кандидата, и итеративно корректирует вектор MeanShift так, чтобы он указывал на область интереса, которая имеет наибольшее сходство с исходным объектом. Гэри Рост Брэдски [30] добавил механизм изменения масштаба к алгоритму на основе MeanShift, а также использование цветовых гистограмм в пространстве HSV и обновление шаблона значительно улучшили производительность.

Хоть классические алгоритмы трекинга объектов и имеют долгую и насыщенную историю своего совершенствования, но для решения требуемых в данной работе задач они не подходят, ведь они обладают рядом существенных ограничений, которые препятствуют их использованию в современных приложениях. Во-первых, они часто не справляются с высокими скоростями объектов и сложными динамическими сценами, где могут происходить частые окклюзии и резкие изменения освещения. Во-вторых, эти алгоритмы обычно

требуют значительных вычислительных ресурсов для обработки каждого кадра, что может быть неприемлемо для работы в режиме реального времени. В-третьих, классические методы недостаточно гибки и не могут эффективно адаптироваться к разнообразным и сложным паттернам движения, что приводит к снижению точности и надежности трекинга в реальных условиях. Эти ограничения делают современные методы на основе корреляционных фильтров или глубокого обучения более предпочтительными для сложных задач трекинга объектов, которые предполагается решать в данной работе.

2.2.2 Алгоритмы, основанные на корреляционных фильтрах

Корреляционные фильтры были впервые применены для обработки сигналов. Основной принцип заключается в вычислении корреляции между двумя сигналами. Для адаптации к различным задачам были предложены многие инновационные методы корреляционных фильтров, такие как синтетические дискриминантные функции (SDF) [31] и минимальная средняя корреляционная энергия (MACE) [32]. Из-за ограничения обучающих выборок высота пика этих фильтров остается неизменной. Абхиджит Махаланобис [33] улучшил MACE, устранив жесткие ограничения, и предложил UMACE. Дэвид С. Болме [34] предложил среднее синтетических точных фильтров (ASEF), которое задает отфильтрованный выход для всего изображения, а не только пик во время обучения. Этот метод оказался более точным для локализации объектов по сравнению с предыдущими методами.

Самым ранним алгоритмом трекинга, использующим корреляционные фильтры, является MOSSE [6]. Среди самых популярных алгоритмов трекинга, основанных на подобных фильтрах, в данной работе будут рассматриваться следующие: MOSSE [6], KCF [13], DeepSRDCF [35] и AutoTrack [36].

Разберем подробнее математическую базу, лежащую в основе алгоритмов, перечисленных ранее. Начнем с алгоритма MOSSE [6], название которого расшифровывается как Minimum Output Sum of Squared Error. Модель для данного подхода инициализируется с первого кадра, создавая фильтр для отслеживания объекта. Для каждого нового кадра фильтр применяется для вычисления корреляционной карты, на которой находится пик, указывающий на положение

объекта. Координаты пика используются для обновления координат bounding box вокруг отслеживаемого объекта и отрисовки полученных данных трекинга для одного кадра из видеопотока. В конце работы одного цикла алгоритма происходит обновление фильтра на основе текущего кадра. Во время обновления фильтра производится минимизация суммы квадратов ошибки, что обеспечивает устойчивость к изменениям освещения, масштаба, поворотов и частичной окклюзии объекта. MOSSE-фильтры обучаются на одном кадре и динамически адаптируются по мере изменения внешнего вида отслеживаемого объекта. Этот подход позволяет поддерживать высокую скорость обработки (за счет очень простой схемы алгоритма) и вполне достаточную для простых задач точность трекинга, что дает возможность использовать данный метод для работы в режиме реального времени.

Следующий алгоритм для разбора в данной работе это KCF (Kernelized Correlation Filter) [13]. Метод инициализируется с первого кадра, создавая модель объекта с использованием циклических сдвигов изображения и преобразования Фурье для вычисления корреляций. Для каждого нового кадра фильтр KCF так же, как и предыдущий метод, применяется для вычисления корреляционной карты, указывающей на положение объекта. Координаты пика корреляции обновляют bounding box вокруг объекта. Модель объекта обновляется с учетом текущего кадра для адаптации к изменениям, к тому же в алгоритме используется регуляризация для повышения устойчивости к шуму и искажениям. Данные этапы циклически повторяются для каждого кадра, обеспечивая непрерывное отслеживание объекта в реальном времени. В отличие от предыдущего алгоритма MOSSE данный подход активно использует свойства циркулярных матриц, а также за счет применения преобразования Фурье и уменьшению граничных эффектов появляется возможность производить достаточно быстрые вычисления без значительного понижения точности трекинга, которая из-за проведения дополнительных вычислений (в сравнении с MOSSE) становится заметно выше. Из этого следует, что KCF также является подходящим для приложений реального времени.

Для разбора следующего алгоритма трекинга DeepSRDCF [35] следует кратко ознакомиться с предшественником упомянутого подхода - методом DCF, активно

использующимся в DCFNet [37]. DCF (Discriminative Correlation Filters) использует свойства циклической корреляции для обучения и применения классификатора в режиме скользящего окна. Этот подход позволяет эффективно учить и применять фильтр корреляции к входным изображениям, что делает его схожим с сверткой в сверточных нейронных сетях. DCF обучается путем минимизации ошибки между прогнозируемым и фактическим выходом, что обеспечивает высокую точность и устойчивость к изменениям объекта и фона. Алгоритм DeepSRDCF (Spatially Regularized Discriminative Correlation Filters) улучшает стандартные DCF, используя глубокие сверточные признаки и пространственную регуляризацию. Он получает новый кадр, извлекает сверточные признаки с использованием обученной нейронной сети и применяет фильтр SRDCF [38] для создания карты откликов, которая указывает на положение объекта. Алгоритм обновляет bounding box вокруг объекта и адаптирует модель с учетом текущих данных и фоновых изменений. Пространственная регуляризация снижает влияние граничных эффектов, улучшая точность трекинга, однако из-за сложных математических вычислений скорость такого метода заметно снижается, поэтому данный алгоритм не получится использовать в системах контроля ручных операций в реальном времени.

Последний алгоритм AutoTrack [36], основанный на корреляционных фильтрах, является наиболее современным алгоритмом, созданным в 2020 году. Алгоритм AutoTrack использует автоматическую пространственно-временную регуляризацию для повышения производительности визуального трекинга на беспилотных летательных аппаратах (БПЛА) - это была изначальная цель его разработки, однако его можно применять и в других задачах. Инициализация построенной модели алгоритма начинается с получения нового кадра и вычисления карты откликов для определения местоположения объекта. Локальные и глобальные вариации на карте помогают адаптировать обновление модели объекта. Находится пик откликов, указывающий на новое положение объекта, и обновляются координаты bounding box. Алгоритм выдает обновленные координаты для отрисовки объекта. Процесс повторяется для каждого кадра, обеспечивая непрерывное отслеживание в реальном времени. В отличие от рассмотренных ранее методов, в

AutoTrack реализована пространственно-временная регуляризация, которая обеспечивает высочайшую точность и адаптивность среди всех более ранних алгоритмов трекинга, однако, как и в случае с DeepSRDCF, подобного рода сложные вычисления значительно снижают скорость работы алгоритма, хотя в отличие от DeepSRDCF этот подход можно полноценно назвать достаточно быстрым для использования в режиме реального времени.

2.2.3 Глубинные алгоритмы трекинга

Глубинное обучение помогает решать многие задачи, связанные с компьютерным зрением. В задачах отслеживания объектов глубинные нейронные сети (DNN) внесли выдающийся вклад в извлечение признаков и прогнозировании местоположения. В настоящее время для отслеживания объектов используются такие нейронные сети, как автоэнкодер (AE), сверточные нейронные сети (CNN) и рекуррентные нейронные сети (RNN).

Автоэнкодер, входные данные которого согласуются с ожидаемыми выходными данными, состоит из кодера и декодера, и сеть обучается за счет потери входных и выходных данных в процессе их кодирования и декодирования [39-40]. Сверточная нейронная сеть (CNN) содержит сверточные слои, объединяющие (pooling) слои и полносвязные слои. На протяжении длительного времени нейросетевые алгоритмы совершенствовались и такие модели, как ZF Net [49], VGGNet [41], GoogLe Net [42], ResNet [43] и Dance Net [44], продолжали внедрять инновации в области сверточного ядра, сетевой структуры и сетевых уровней, что постоянно повышало производительность сетей. Алгоритмы отслеживания объектов применяются к видео последовательностям, но CNN не может использовать данные о положении объектов интереса на соседних кадрах, что приводит к потере информации. Поэтому для отслеживания объектов применяется рекуррентная нейронная сеть (RNN), поскольку она больше подходит для задач последовательности. В исследовании [45] предложили LSTM для устранения проблемы исчезновения градиента в RNN, а через какое-то время те же ученые улучшили сеть, чтобы сделать ее широко используемой. Кроме того, существует множество вариантов LSTM, таких как GRU [46] и DLSTM [47].

В данном разделе будут рассматриваться наиболее популярные глубинные нейронные сети («сиамские»), использующиеся для решения задачи трекинга, а именно SiamRPN [15], SiamBAN [48] и SiamRPN++ [49]. В данной работе рассматриваются только «сиамские» нейронные сети для решения задач трекинга, потому что такие нейронные сети специально разработаны для сравнения двух входных изображений, что идеально подходит для задачи трекинга, где нужно сопоставить объект в текущем кадре с шаблоном из первого кадра. К тому же, для достижения глобальной цели по созданию системы контроля ручных операций на реальном производстве требуются технологии и программы, способные работать в режиме реального времени, и «сиамские» нейронные сети лучше справляются с этой задачей, чем рекуррентные или сложные сверточные нейронные сети.

Разберем подробнее математические методы и идеи, лежащие в основе алгоритмов, перечисленных выше. Начнем разбор глубинных моделей с SiamRPN (Siamese Region Proposal Network), которая объединяет в себе «сиамскую» нейронную сеть для извлечения признаков и региональную предложенную сеть (RPN) для генерации предложений. «Сиамская» подмодель включает две ветви с общими параметрами, которые обрабатывают шаблонное изображение и текущее изображение (текущий кадр в видеопотоке) соответственно. RPN также состоит из двух ветвей: одна для классификации (разделение на фон и объект), другая для регрессии (уточнение координат bounding box). Во время инференса (процесс работы нейросети), после инициализации на первом кадре, шаблонная ветвь фиксируется, и только детекционная ветвь используется для поиска объекта в новых кадрах. Такой подход помогает поддерживать высокую скорость работы модели и ее точность. Алгоритм выполняет локальную одноразовую детекцию, сравнивая текущий кадр с шаблоном и обновляя bounding box объекта на протяжении всего цикла работы. На Рис. 4 представлена примерная схема модели SiamRPN.

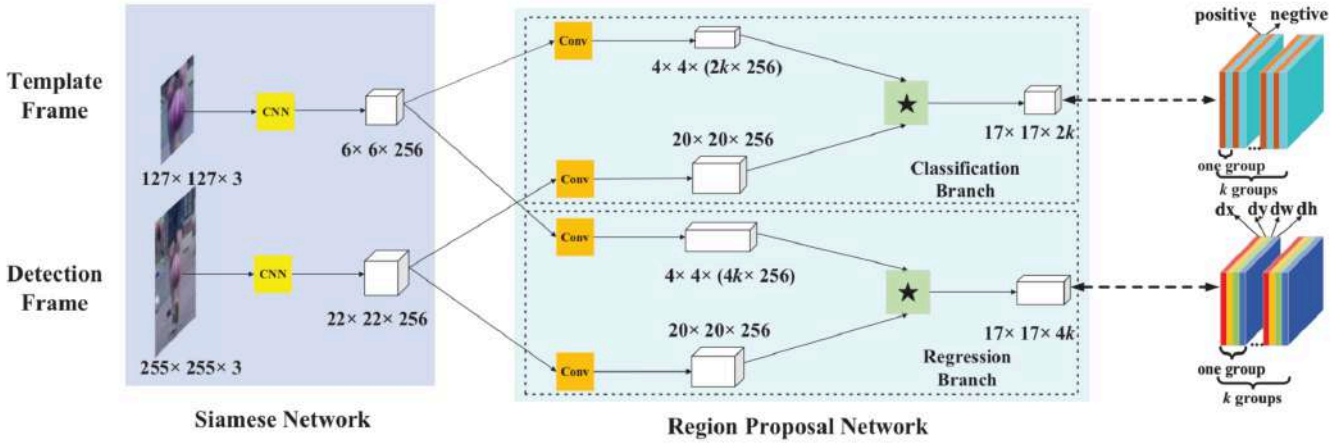

Рис. 4. Общая схема работы модели SiamRPN [15]

Через 1 год после создания модели SiamRPN в открытом доступе появилась ее усовершенствованная версия SiamRPN++ [49]. По сравнению со своим предшественником новая версия модели использует более глубокие нейронные сети, такие как ResNet, для извлечения признаков (в старой версии использовались более простые сети). Также SiamRPN++ использует пространственно-осведомленную стратегию выборки, которая устраняет проблему строгой трансляционной инвариантности, что было ограничением в SiamRPN. Более того, в новой версии модели используется агрегация признаков из нескольких слоев сети (слоистая агрегация), что улучшает точность и надежность трекинга, особенно в сложных условиях. Еще одним значимым отличием от предыдущей модели является скорость и стабильность процесса обучения SiamRPN++, ведь она обладает меньшим числом параметров из-за глубинно-разделяемой корреляционной структуры. Хотя новую модель и можно отнести к нейросетям, работающим в режиме реального времени, ее скорость обработки кадров заметно ниже, чем у SiamRPN, следовательно в задачах с простым трекингом и требованием получать высокий FPS лучше использовать именно SiamRPN. Общая схема работы модели SiamRPN++ представлена на Рис. 5.

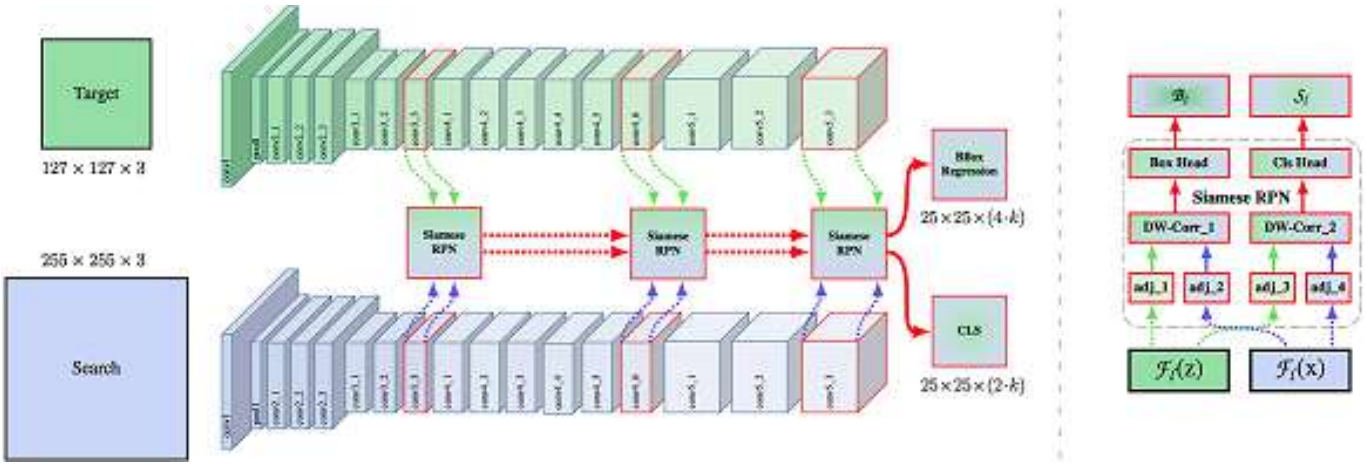

Рис. 5. Общая схема работы модели SiamRPN++ [49]

Последней рассматриваемой глубинной моделью будет SiamBAN (Siamese Box Adaptive Network) [48]. Данный метод решает задачу визуального трекинга, преобразовывая ее в задачу параллельной классификации и регрессии. Вместо использования многомасштабного поиска или заранее определённых якорных боксов для оценки размера и пропорций объекта, SiamBAN напрямую классифицирует объекты и регрессирует bounding boxes в единой сети на основе полносвязной сверточной сети (FCN). Это упрощает процесс, устраняет необходимость в гиперпараметрах для кандидатов (bounding box candidates) и делает трекер более гибким и универсальным. Модель SiamBAN состоит из сиамской сети и нескольких адаптивных головок для боксов (box adaptive heads). Сиамская сеть, основанная на ResNet-50, использует два параллельных пути для обработки шаблона и текущего кадра, извлекая признаки из обоих изображений. Адаптивные головки включают модули классификации и регрессии, которые выполняют классификацию переднего плана и фона, а также предсказание границ боксов. Модель обучается и инференсирует end-to-end, что повышает её эффективность и точность. Благодаря отсутствию якорных боксов и эффективной адаптации к изменяющимся условиям, SiamBAN показывает лучшую производительность на различных наборах данных и в различных сценариях трекинга среди всех 3 рассмотренных моделей глубинного обучения. Общая схема работы SiamBAN представлена на Рис. 6.

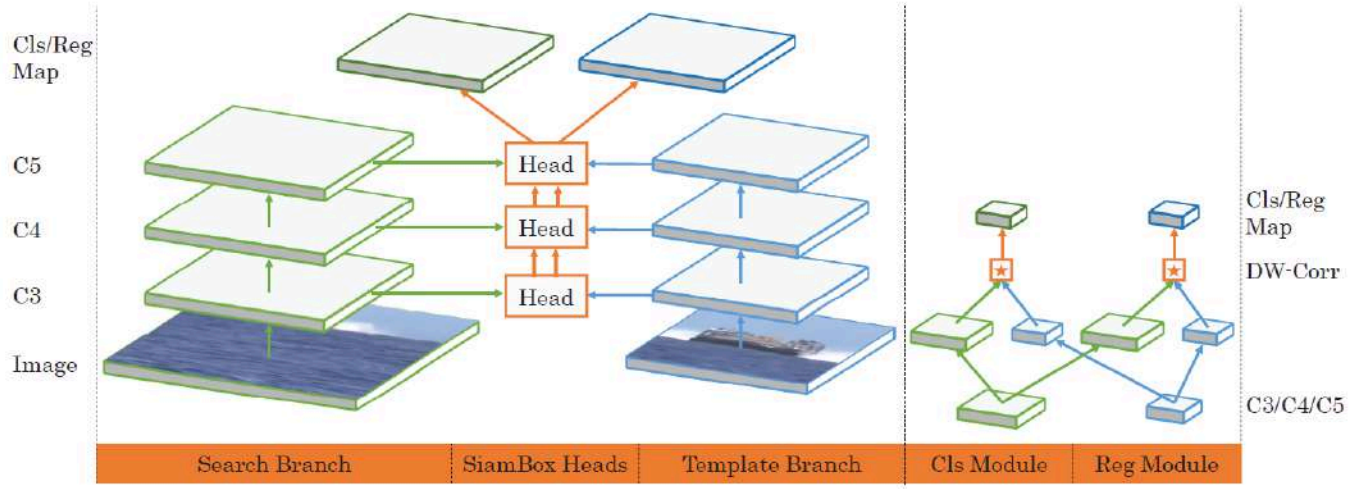

Рис. 6. Общая схема работы модели SiamBAN [48]

2.3 Оценка алгоритмов

Перед применением различных подходов и моделей трекинга в реальных задачах компьютерного зрения следует оценить качество их работы, а также эффективность их применения в зависимости от технических возможностей и внешних условий. Качество работы и эффективность алгоритмов трекинга объектов оценивается по ряду ключевых метрик, включая точность, вычислительную сложность, устойчивость к изменениям условий и другие. В этом разделе представлен сравнительный анализ различных алгоритмов трекинга, описаны возможные ограничения и рекомендации по применению в реальных задачах.

2.3.1 Метрики качества трекинга

Одной из самых важных метрик качества алгоритмов трекинга является точность (pixel-based precision) [50], которая показывает долю правильно детектированных объектов среди всех детектированных. Она помогает оценить, насколько часто алгоритм делает правильные предсказания относительно отслеживаемых объектов. То есть, метрика precision отражает способность алгоритма избегать ложных срабатываний. Чем выше значение precision, тем меньше ложных предсказаний генерирует алгоритм, что важно для надежного трекинга объектов в реальном времени.

Другой полезной метрикой трекинга является Overlap Score (или Region Overlap) [51] используется для оценки точности трекинга объектов путем измерения

степени пересечения предсказанной области (bounding box) с истинной областью объекта (ground truth). В других задачах компьютерного зрения (например, детекция или классификация) эту метрику также могут называть Intersection over Union (IoU) или мерой Жаккара. Overlap Score рассчитывается как отношение площади пересечения предсказанной области и истинной области к площади их объединения. Чем выше значение метрики, тем точнее работает алгоритм трекинга, указывая на лучшее совпадение предсказанных и истинных границ объекта.

Для оценки того, насколько устойчивым является алгоритм трекинга, используется метрика трекинга под названием robustness [52]. Данная метрика качества показывает, насколько часто и насколько критично алгоритм терпит неудачи при отслеживании объекта. Она измеряет количество раз, когда алгоритм теряет объект и требует повторной инициализации, тем самым отражая его надежность в различных условиях и сценариях. Robustness важно учитывать, поскольку она демонстрирует способность алгоритма справляться с различными трудностями, такими как изменения освещения, частичная окклюзия объекта или быстрое движение. Более высокая устойчивость означает, что алгоритм более надежен и стабилен в реальных условиях при решении различных задач трекинга.

Другой полезной метрикой для оценки надежности алгоритма трекинга является failure rate [53], ведь она напрямую измеряет частоту неудач в работе алгоритма трекинга. Неудача происходит, когда алгоритм теряет объект и не может его восстановить в пределах допустимого количества кадров. Чем ниже значение failure rate, тем более устойчив и надежен алгоритм в условиях реального мира. Failure rate показывает, как часто алгоритм сбивается и нуждается в повторной инициализации, что критически важно для задач трекинга в реальном времени, где стабильность и непрерывность трекинга являются приоритетными.

Доля верных ответов модели (accuracy) [52] в качестве метрики также очень часто применяется для оценки моделей трекинга так как она оценивает, насколько точно алгоритм определяет положение объекта в кадре. Чем выше значение accuracy, тем лучше алгоритм справляется с задачей трекинга, демонстрируя высокую точность в определении границ объекта. В целом accuracy помогает понять,

насколько точно алгоритм может отслеживать объект на протяжении всего видеопотока, что важно для оценки общей эффективности трекинга.

Все описанные ранее метрики в основном предполагали оценку качества работы модели в случае отслеживания единственного объекта в кадре [7], однако в большинстве своем задачи трекинга предполагают отслеживание нескольких объектов в кадре [8-9], поэтому в данной работе будут использоваться метрики для оценки качества работы моделей в случае с трекингом множества объектов на видео. Одной из таких метрик является HOTA (Higher Order Tracking Accuracy) [54]. Метрика объединяет в себе три ключевых аспекта: точность детекции, точность ассоциации и точность локализации объектов. HOTA предоставляет сбалансированную оценку, позволяя выявить и сравнить ошибки в детекции, ассоциации и локализации объектов, что делает её более комплексной и информативной по сравнению с предыдущими метриками. HOTA вычисляется как геометрическое среднее между оценками детекции и ассоциации, что обеспечивает равное внимание обеим компонентам, и среднее значение берется по различным порогам локализации, что позволяет учитывать точность локализации.

Последними используемыми в данной работе метриками являются MOTA (Multiple Object Tracking Accuracy) и MOTP (Multiple Object Tracking Precision) [55]. MOTA оценивает общую точность трекинга, учитывая три типа ошибок: пропуски (misses), ложные срабатывания (false positives) и ошибки соответствия (mismatches). Она показывает, насколько хорошо алгоритм справляется с отслеживанием всех объектов, минимизируя количество ошибок. MOTP измеряет точность предсказанных позиций объектов, оценивая среднюю ошибку локализации между предсказанными и истинными положениями объектов. Эта метрика показывает, насколько точно алгоритм определяет местоположение объектов в кадре.

2.3.2 Сравнительный анализ

Для оценки скорости и эффективности работы алгоритмов MOSSE, KCF, DeepSRDCF, AutoTrack, SiamRPN, SiamRPN++ и SiamBAN используются метрики, упомянутые ранее, а также оценка скорости обработки кадров моделью (Frames Per Second, FPS). В таблицах 1-2 приведены численные результаты тестов всех

алгоритмов на существующих наборах данных OTB2015 [56] и VOT2016 [57], которые активно используются для проверки и оценки алгоритмов трекинга в различных IT соревнованиях и на конференциях.

Таблица 1. Метрики качества алгоритмов трекинга на наборе данных OTB2015

Алгоритм	Precision	Accuracy	Overlap Score	Failure Rate	Robustness	HOTA	MOTA	MOTP	FPS	Real Time
MOSSE	0.414	0.31	0.25	0.32	0.51	0.35	0.33	0.4	118	Yes
KCF	0.696	0.477	0.42	0.25	0.59	0.51	0.52	0.55	73	Yes
DeepSRDCF	0.851	0.635	0.58	0.14	0.75	0.65	0.61	0.7	2	No
AutoTrack	0.91	0.691	0.65	0.11	0.82	0.7	0.68	0.75	38	Yes
SiamRPN	0.85	0.636	0.61	0.12	0.78	0.72	0.73	0.77	42	Yes
SiamRPN++	0.914	0.676	0.68	0.08	0.82	0.75	0.75	0.79	57	Yes
SiamBAN	0.932	0.713	0.72	0.07	0.85	0.78	0.77	0.83	73	Yes

Таблица 2. Метрики качества алгоритмов трекинга на наборе данных VOT2016

Алгоритм	Precision	Accuracy	Overlap Score	Failure Rate	Robustness	HOTA	MOTA	MOTP	FPS	Real Time
MOSSE	0.35	0.31	0.27	0.34	0.48	0.34	0.32	0.38	118	Yes
KCF	0.69	0.48	0.46	0.26	0.62	0.52	0.53	0.54	73	Yes
DeepSRDCF	0.85	0.64	0.61	0.14	0.77	0.66	0.61	0.71	2	No
AutoTrack	0.89	0.68	0.66	0.09	0.81	0.71	0.69	0.76	38	Yes
SiamRPN	0.87	0.66	0.63	0.11	0.8	0.73	0.71	0.78	42	Yes
SiamRPN++	0.92	0.7	0.67	0.07	0.83	0.76	0.74	0.81	57	Yes
SiamBAN	0.94	0.72	0.71	0.06	0.87	0.79	0.77	0.84	73	Yes

Различия в значениях метрик среди сравниваемых алгоритмов связаны с их архитектурными особенностями и применяемыми внутри математическими методами. MOSSE демонстрирует исключительную скорость обработки, что объясняется использованием эффективных фильтров корреляции и обработки в пространстве Фурье. Однако его точность уступает другим методам из-за ограничений в адаптации к сложным изменениям объекта. KCF значительно улучшает Precision и Accuracy, но все еще не достигает высокой Robustness и Overlap

Score. KCF использует ядровые трекеры на основе корреляционных фильтров, что обеспечивает высокую скорость при приемлемой точности. DeepSRDCF, несмотря на более низкую скорость, обеспечивает высокую точность благодаря использованию дискриминационных корреляционных фильтров с пространственными регуляризациями. AutoTrack достигает высоких значений всех метрик, особенно Precision и Robustness, при умеренной скорости. SiamRPN показывает высокие значения Precision и Accuracy с хорошей скоростью, но умеренные значения Robustness и Failure Rate. SiamRPN++ улучшает Precision, Accuracy и Robustness, сохраняя хорошую скорость, но имеет немного выше Failure Rate по сравнению с SiamBAN. SiamBAN достигает высочайших значений Precision, Accuracy и Robustness с отличной скоростью, но сложнее в реализации и, соответственно, обучении.

2.3.3 Ограничения

Каждый из рассматриваемых алгоритмов имеет свои преимущества и недостатки, которые ограничивают их применение в различных сценариях. Выявление ограничений каждого алгоритма и возможных путей их устранения является важной задачей перед непосредственным их применением в различных реальных задачах.

MOSSE имеет низкие значения метрик качества, таких как Precision, Accuracy и Overlap Score, что ограничивает его точность и надежность в отслеживании объектов. Кроме того, алгоритм характеризуется высокой Failure Rate, что делает его менее устойчивым к изменениям внешнего вида объекта и требует частых повторных инициализаций. Его простота реализации и высокая скорость работы не компенсируют недостатков в условиях сложных и динамичных сценариев трекинга. К тому же, из-за его низких значений метрик качества и высокой Failure Rate, он будет испытывать трудности с отслеживанием объектов в условиях недостаточного освещения.

KCF показывает значительное улучшение по сравнению с MOSSE в Precision и Accuracy, однако его Robustness и Overlap Score остаются на среднем уровне. Умеренная скорость (FPS) ограничивает его применение в задачах реального

времени с высокими требованиями к точности и стабильности. Алгоритм также испытывает трудности с обработкой масштабных изменений и окклюзий, что снижает его эффективность в разнообразных условиях. Так же, как и MOSSE, KCF может столкнуться с проблемами при попытке трекинга объектов в условиях малой освещенности.

DeepSRDCF демонстрирует высокие значения Precision, Accuracy и Robustness, но его основное ограничение — низкая скорость работы. Высокие вычислительные затраты делают его неприменимым для задач реального времени без мощного оборудования. Этот алгоритм требует значительных ресурсов, что может ограничивать его использование в системах с ограниченными вычислительными возможностями. В отличие от предыдущих рассмотренных алгоритмов, DeepSRDCF способен достаточно хорошо отслеживать объект в условиях с плохим освещением, ведь в нем используются глубокие сверточные признаки, что позволяет лучше справляться с изменениями освещения и условиями плохой видимости. То есть, его устойчивость к изменениям освещения выше по сравнению с более простыми алгоритмами.

AutoTrack достигает высоких значений всех метрик, особенно Precision и Robustness, но при этом имеет умеренную скорость работы. Настройка и адаптация алгоритма могут быть сложными, что требует дополнительных усилий для его эффективного использования. Также возможны проблемы с отслеживанием высоко динамичных объектов, что может ограничивать его применение в некоторых сценариях. AutoTrack может также испытывать трудности в условиях плохого освещения.

SiamRPN показывает высокие значения Precision и Accuracy, однако его Robustness и Failure Rate остаются умеренными. Этот алгоритм может сталкиваться с трудностями в сложных сценариях, требующих высокой точности и устойчивости. Потребность в тонкой настройке для различных объектов ограничивает его универсальность и требует дополнительных усилий для адаптации. Из-за простоты используемых моделей для экстракции признаков данный подход будет показывать плохие результаты в условиях плохой освещенности.

SiamRPN++ улучшает показатели Precision, Accuracy и Robustness по сравнению с SiamRPN, однако имеет несколько выше Failure Rate по сравнению с SiamBAN. Сложность реализации и высокие вычислительные затраты делают его требовательным к ресурсам, что может ограничивать его использование в системах с ограниченными вычислительными возможностями. В сравнении с предыдущей версией модели, SiamRPN++ способен более эффективно справляться с условиями плохого освещения, извлекая признаки на различных уровнях глубины сети, что делает его менее чувствительным к изменениям освещения.

SiamBAN достигает высочайших значений Precision, Accuracy и Robustness, однако его сложная реализация и настройка требуют значительных усилий. Высокие требования к вычислительным ресурсам делают его менее доступным для использования в условиях с ограниченными ресурсами. Возможны трудности при интеграции в существующие системы, что может требовать дополнительной адаптации и настройки. Благодаря своей сложной архитектуре и использованию слоистой агрегации признаков, SiamBAN демонстрирует высокую устойчивость к изменениям освещения. Этот алгоритм может более точно отслеживать объекты даже в условиях низкой освещенности, что делает его одним из наиболее устойчивых к таким условиям среди рассматриваемых.

2.3.4 Рекомендации по применению

Приведем краткие рекомендации по использованию определенных ранее алгоритмов в конкретных приложениях или условиях.

MOSSE не рекомендуется использовать в условиях плохой освещенности и возможных окклюзий из-за его низких значений метрик качества и высокого Failure Rate. Однако алгоритм подходит для простых задач трекинга в хорошо освещенных условиях без значительных изменений внешнего вида объекта, особенно когда вычислительные ресурсы ограничены.

KCF показывает умеренные результаты в условиях плохой освещенности и окклюзий, но не подходит для сильно изменяющихся условий. Он требует умеренных вычислительных ресурсов и может использоваться в сценариях с умеренными изменениями освещения и объектами, которые не подвержены

значительным окклюзиям.

DeepSRDCF хорошо справляется с изменениями освещенности и окклюзиями благодаря использованию глубоких признаков, но требует высоких вычислительных ресурсов, поэтому использование данного метода в системах, предназначенных для решения задач в режиме реального времени, не рекомендуется. Этот алгоритм рекомендуется для сложных задач трекинга в изменяющихся условиях и при наличии окклюзий.

AutoTrack эффективен в условиях плохой освещенности и окклюзий благодаря автоматической адаптации, но требует умеренных вычислительных ресурсов. Он рекомендуется для разнообразных сценариев, включая динамичные условия и умеренные изменения освещенности, при наличии достаточных вычислительных ресурсов.

SiamRPN хорошо справляется с умеренными изменениями освещения и окклюзиями, но не идеален для сильно изменяющихся условий. Он требует умеренных вычислительных ресурсов и подходит для задач, требующих быстрой и точной обработки, в условиях умеренного освещения и окклюзий.

SiamRPN++ отлично справляется с изменениями освещения и окклюзиями благодаря улучшенной архитектуре, но требует высоких вычислительных ресурсов. Этот алгоритм идеально подходит для сложных задач трекинга в условиях изменяющегося освещения и значительных окклюзий, при наличии мощных вычислительных систем.

SiamBAN демонстрирует высокую устойчивость к изменениям освещения и окклюзиям благодаря сложной архитектуре, но требует очень высоких вычислительных ресурсов. Он рекомендуется для наиболее требовательных задач трекинга, где важны высокая точность и устойчивость в условиях изменяющегося освещения и окклюзий, особенно в системах с наивысшими вычислительными возможностями.

2.4 Экспериментальная проверка

В этом разделе представлен набор экспериментов с алгоритмами трекинга объектов: MOSSE, KCF, DeepSRDCF, AutoTrack, SiamRPN, SiamRPN++ и SiamBAN.

Эксперименты проводились на различных примерах видеопоследовательностей и включают числовые данные и визуальные примеры, иллюстрирующие сильные и слабые стороны каждого алгоритма. В качестве наборов данных для проверки использовались видео, отснятые на текущей версии стенда для контроля ручных операций для реальных деталей, используемых при сборке составных частей FPV-дрона. Для каждого алгоритма измерялись метрики: Precision, Accuracy, Overlap Score, Robustness, Failure Rate, HOTA, MOTA, MOTP и FPS. Все алгоритмы тестировались в условиях изменений освещения, окклюзии и быстрой смены скорости движения объектов в кадре. В Таблице 3 представлены результаты оценки работы алгоритмов, а на Рис.7-9 отчетливо видно, как в каждой из трудных ситуаций ведут себя рассмотренные алгоритмы.

Таблица 3. Метрики качества для алгоритмов на кастомных наборах данных

Алгоритм	Precision	Accuracy	Overlap Score	Failure Rate	Robustness	HOTA	MOTA	MOTP	FPS
MOSSE	0.45	0.4	0.3	0.2	0.55	0.4	0.38	0.42	600
MOSSE (окклюзии)	0.3	0.25	0.2	0.4	0.4	0.25	0.3	0.3	600
MOSSE (плохое освещение)	0.35	0.3	0.25	0.35	0.45	0.3	0.35	0.35	600
KCF	0.7	0.6	0.5	0.15	0.65	0.55	0.6	0.5	170
KCF (окклюзии)	0.5	0.45	0.35	0.25	0.5	0.4	0.45	0.4	170
KCF (плохое освещение)	0.55	0.5	0.4	0.2	0.55	0.45	0.5	0.45	170
DeepSRDCF	0.85	0.8	0.7	0.1	0.8	0.75	0.7	0.75	20
DeepSRDCF (окклюзии)	0.7	0.65	0.55	0.2	0.7	0.6	0.65	0.6	20
DeepSRDCF (плохое освещение)	0.75	0.7	0.6	0.15	0.75	0.65	0.7	0.65	20
AutoTrack	0.9	0.85	0.75	0.08	0.85	0.78	0.75	0.78	25
AutoTrack (окклюзии)	0.75	0.7	0.6	0.18	0.75	0.65	0.7	0.65	25
AutoTrack (плохое освещение)	0.8	0.75	0.65	0.12	0.8	0.7	0.75	0.7	25

SiamRPN	0.88	0.8	0.7	0.12	0.8	0.75	0.7	0.75	80
SiamRPN (окклюзии)	0.7	0.65	0.55	0.22	0.7	0.6	0.65	0.6	80
SiamRPN (плохое освещение)	0.75	0.7	0.6	0.18	0.75	0.65	0.7	0.65	80
SiamRPN++	0.92	0.85	0.78	0.06	0.88	0.82	0.8	0.82	150
SiamRPN++ (окклюзии)	0.8	0.75	0.65	0.12	0.8	0.7	0.75	0.7	150
SiamRPN++ (плохое освещение)	0.85	0.8	0.7	0.1	0.85	0.75	0.8	0.75	150
SiamBAN	0.95	0.9	0.82	0.05	0.9	0.85	0.85	0.85	180
SiamBAN (окклюзии)	0.85	0.8	0.7	0.1	0.85	0.75	0.8	0.75	180
SiamBAN (плохое освещение)	0.9	0.85	0.75	0.08	0.88	0.8	0.85	0.8	180

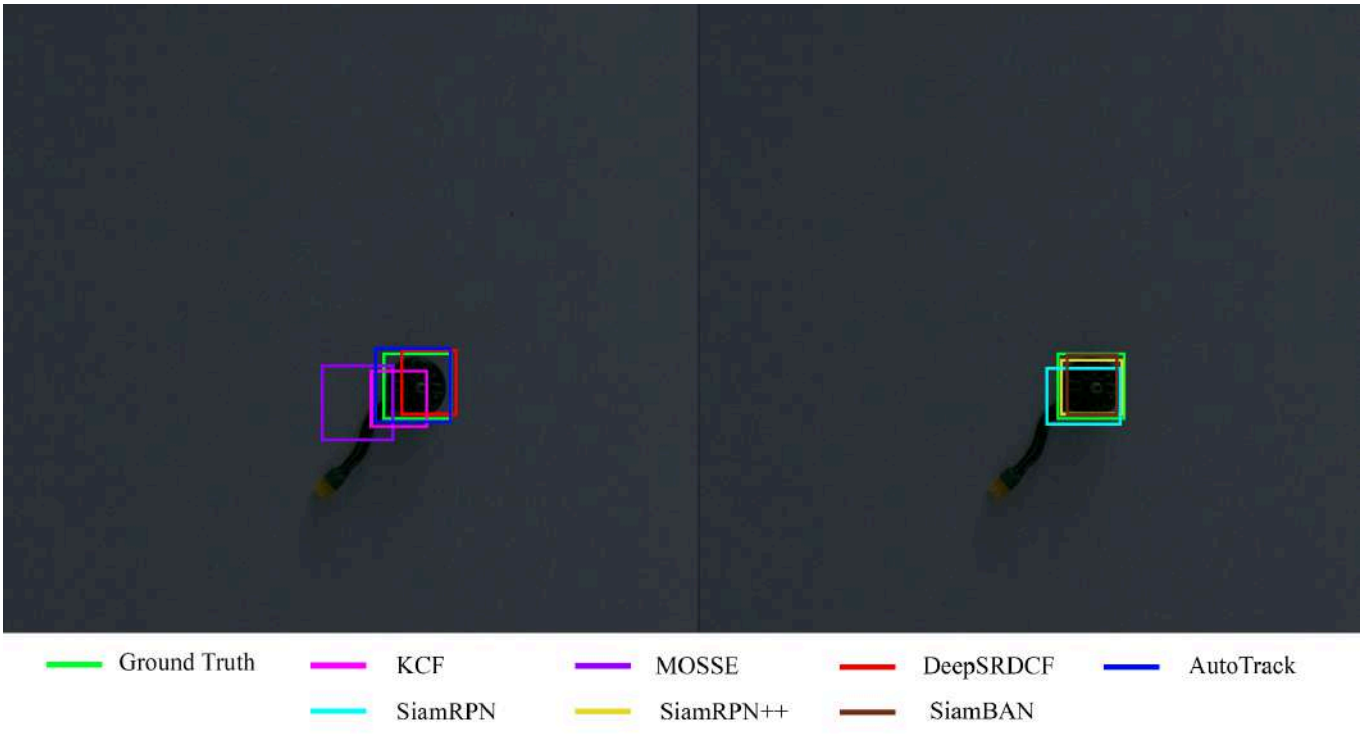

Рис. 7. Пример одного из видеокадров в эксперименте по трекингу мотора при изменении освещения. Слева - алгоритмы, основанные на корреляционных фильтрах. Справа - глубинные модели.

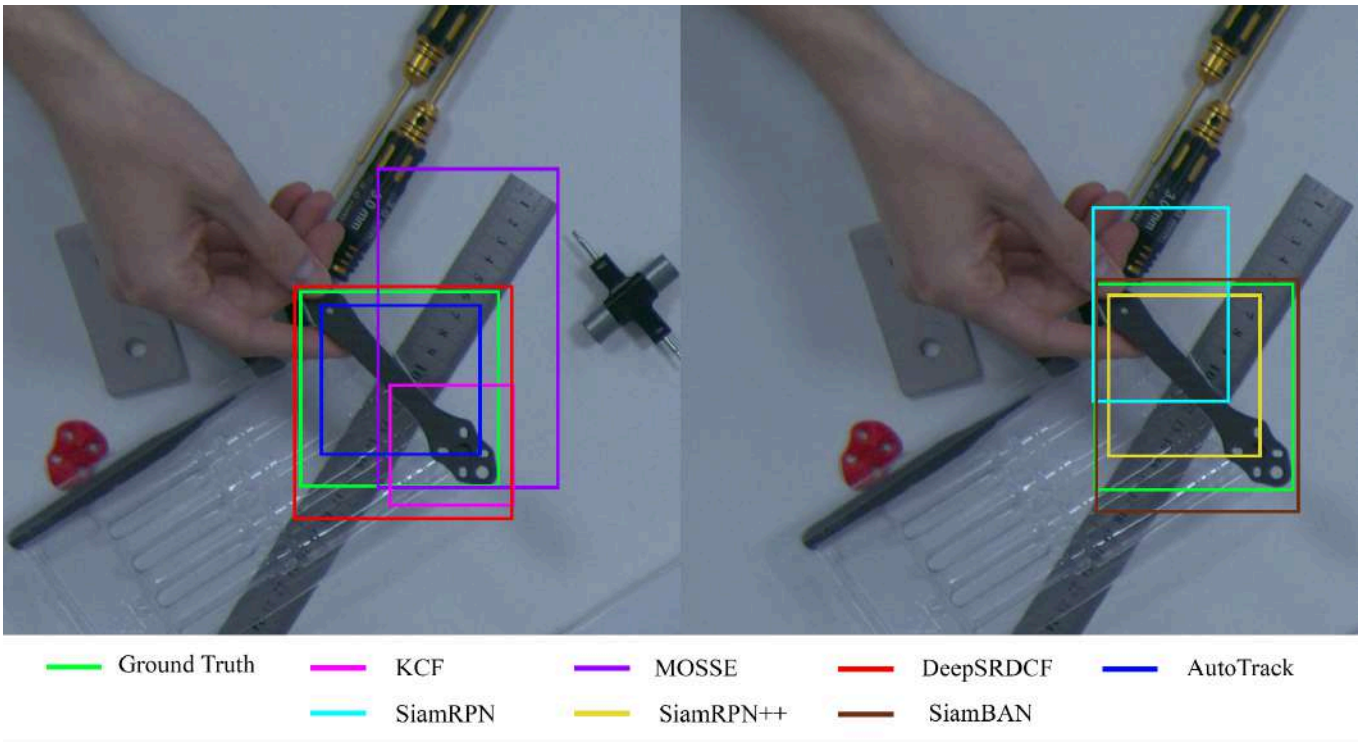

Рис. 8. Пример одного из видеокадров в эксперименте по трекингу луча при окклюзии. Слева - алгоритмы, основанные на корреляционных фильтрах. Справа - глубинные модели.

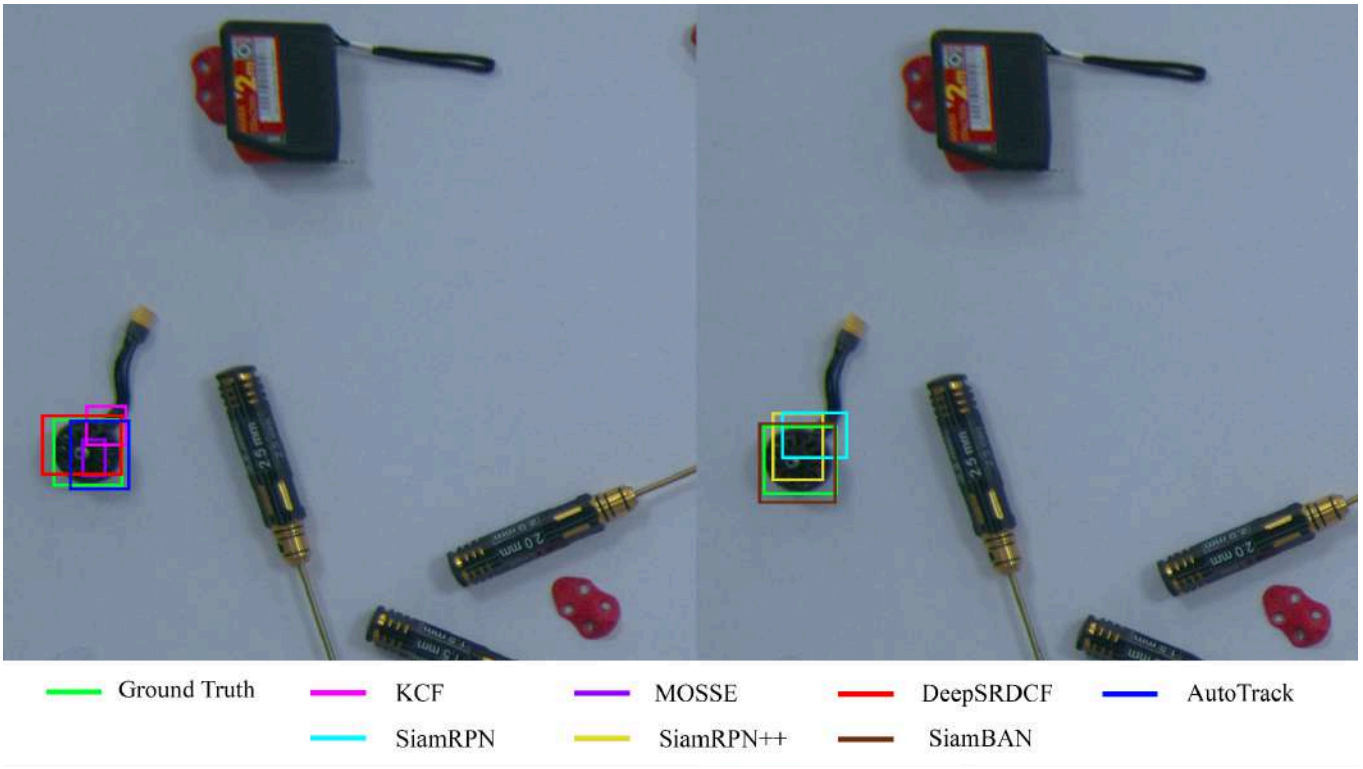

Рис. 9. Пример одного из видеокадров в эксперименте по трекингу мотора при нормальных условиях. Слева - алгоритмы, основанные на корреляционных фильтрах. Справа - глубинные модели.

Экспериментальная проверка подтвердила, что почти все ограничения и рекомендации по применению, собранные благодаря теоретическому анализу статей и иных источников информации для алгоритмов трекинга и приведенные немногим ранее в тексте данной работы, являются корректными. Каждый из алгоритмов имеет

свои сильные и слабые стороны. Выбор конкретного алгоритма должен основываться на специфических требованиях задачи, таких как требуемая точность, скорость обработки и условия эксплуатации.

2.5 Выводы по разделу

В данном разделе были проведены описание, анализ и экспериментальная проверка семи алгоритмов трекинга: MOSSE, KCF, DeepSRDCF, AutoTrack, SiamRPN, SiamRPN++ и SiamBAN. Результаты показали, что более сложные алгоритмы, такие как SiamRPN++ и SiamBAN, демонстрируют высокую точность и устойчивость к изменениям освещенности и окклюзиям, однако требуют значительных вычислительных ресурсов. Простые алгоритмы, такие как MOSSE и KCF, обеспечивают высокую скорость обработки, но имеют ограничения в условиях изменяющегося освещения и частых окклюзий. Таким образом, выбор алгоритма трекинга должен основываться на специфических требованиях задачи, балансируя между точностью, устойчивостью и вычислительными ресурсами.

3 Повышение качества работы моделей детекции

Помимо стандартного способа применения алгоритмов трекинга объектов для отрисовки bounding box и пути движения (трека) объектов интереса на последовательности кадров из видео существуют другие способы использования данных моделей. Алгоритмы трекинга объектов в некоторых особенных задачах могут играть ключевую роль в улучшении качества детекции, что особенно важно в динамически меняющихся и сложных условиях. В данном разделе будет рассмотрено применение алгоритмов трекинга для повышения качества детекции объектов. Основная цель заключается в том, чтобы изучить существующие решения, протестировать их в реальных условиях, выбрать наиболее эффективные варианты и внедрить их в действующее приложение для контроля ручных операций, использующее модели детекции. В рамках этой задачи будут проведены экспериментальные проверки, чтобы определить сильные и слабые стороны различных алгоритмов трекинга (будем брать рассмотренные в предыдущем разделе модели), а также подведены итоги по выбору конкретного алгоритма для условий

работы разрабатываемой системы контроля ручных операций.

3.1 Проблемы моделей детекции и возможные решения

Модели детекции, несмотря на значительные успехи в определении положения объектов на изображении в последние годы, все же имеют ряд недостатков, которые могут негативно влиять на их эффективность при решении реальных задач. Рассмотрим основные из этих недостатков и обсудим, как алгоритмы трекинга могут помочь их решить. Также будут рассмотрены несколько известных подходов к повышению качества, точности и надежности работы моделей детекции в сложных условиях с помощью методов отслеживания объектов на видеокадрах.

3.1.1 Пропуски в детекции

Одной из самых распространенных проблем, возникающих во время работы моделей детекции, являются так называемые пропуски в детекции. Пропуски в детекции возникают, когда модель не распознает объект на кадре, даже если он там присутствует. Это может быть вызвано различными факторами, такими как плохое освещение, окклюзия, быстрое движение объектов или недостаточная разрешающая способность модели.

Фактор плохого освещения влияет на способность модели извлекать признаки из данных изображения, ведь любое изображение можно представить математически как набор чисел и для особенно сильно затененных кадров все числа, отвечающие за интенсивность цвета, близки к нулю. Такие маленькие числа зачастую приводят к тому, что математические модели не способны их должным образом обработать, так как при подстановке в формулы почти нуля на выходе тоже получается нуль. Учитывая данный факт, легко можно предположить, что выходные числовые признаки так же будут очень малы и на их основе модель не сможет сделать качественное предположение о положении объекта в кадре.

Фактор окклюзии предполагает наличие препятствия в виде постороннего объекта перед камерой. Данный объект может полностью или частично, на долгое время или же кратковременно скрыть объект интереса из зоны видимости камеры, что затрудняет его детекцию. Факторы быстрого движения или недостаточной

разрешающей способности, при которых объект "размывается" на кадрах из видео потока, также препятствуют точному определению положения объекта на изображении, ведь входные данные для модели перестают обладать точной информацией о положении, форме и даже цвете границ и других частей объекта (вся эта информация подвергается слишком серьезной "деформации").

Алгоритмы трекинга же могут заполнять пробелы в детекции, продолжая отслеживать объект, даже если он временно не был распознан детектором. Трекинг использует информацию из предыдущих кадров для предсказания местоположения объекта в следующих кадрах, что позволяет поддерживать непрерывность детекции и снижать количество пропущенных объектов. Для реализации данного решения требуется непрерывно отслеживать все объекты интереса в видеопотоке, однако при наличии их большого количества общая производительность программы может сильно снизиться, поэтому в таких случаях нужно корректно выбирать подходящий под условия задачи подход.

3.1.2 Нестабильность детекции

Нестабильность детекции проявляется в "дрожании" окружающих прямоугольников (bounding box), когда их положения изменяются между кадрами даже при небольших движениях объекта или отсутствии его движения. Это может быть результатом неустойчивости модели к шуму или мелким изменениям в изображении.

Алгоритмы трекинга могут сглаживать траектории объектов, используя информацию из нескольких кадров для уменьшения нестабильности. Это помогает создать более плавные и устойчивые траектории, улучшая качество и точность детекции. Самым примитивным способом решения данной проблемы является сравнение окружающих объект прямоугольников (bounding box) на соседних кадрах и вычисление меры Жаккара (IoU), которая отвечает за степень их пересечения. В случае высокого значения коэффициента IoU можно сделать предположение о том, что объект не сильно изменил свое положение по сравнению с предыдущим кадром, следовательно нет необходимости как либо изменять форму и положение окружающего объект прямоугольника, ведь это действие в обычных обстоятельствах

привело бы к "дрожанию" прямоугольника на видео.

3.1.3 Ложные срабатывания

Ложные срабатывания возникают, когда модель ошибочно идентифицирует объекты, которых на самом деле нет. Это может быть вызвано шумом в данных, сложными фонами или объектами, которые выглядят похожими на целевые.

Алгоритмы трекинга могут эффективно решить проблему ложных срабатываний благодаря использованию временной информации для фильтрации недостоверных детекций. Когда алгоритм трекинга обрабатывает видеопоток, он анализирует последовательность кадров, а не только один кадр. Это позволяет ему учитывать движение объектов и их поведение во времени. Если модель детекции идентифицирует объект на одном кадре, алгоритм трекинга проверяет его наличие на следующих кадрах. Если предполагаемый объект не появляется на следующих кадрах, трекер может классифицировать эту детекцию как ложное срабатывание и исключить ее из дальнейшего рассмотрения и обработки. Если же объект детектируется последовательно на нескольких кадрах, алгоритм с высокой степенью уверенности будет считать его настоящим. Такой подход позволяет существенно снизить количество ошибочных детекций.

3.1.4 Изменения масштаба и ракурса

Объекты могут изменять свой размер и положение в кадре, что затрудняет их детекцию. Изменение масштаба происходит, когда объект приближается или удаляется от камеры, а изменение ракурса может затруднить его распознавание из-за изменений видимых характеристик.

Алгоритмы трекинга, решающие проблему изменения масштаба, используют различные подходы. Один из ключевых методов — это адаптивное масштабирование области интереса (bounding box) в соответствии с движением и размером объекта. Когда объект приближается к камере и увеличивается в размере, трекер масштабирует bounding box, чтобы захватить всю область объекта. Аналогично, если объект удаляется и становится меньше, трекер уменьшает размер bounding box. Некоторые продвинутые трекеры, такие как SiamRPN++ и SiamBAN, используют

глубокие нейронные сети для извлечения сложных признаков, которые помогают моделировать изменения масштаба. Эти алгоритмы обучены на больших наборах данных, содержащих разнообразные примеры объектов в различных масштабах и ракурсах. Они способны предсказывать и адаптировать bounding box на основе извлеченных признаков и временной информации из последовательности кадров.

Алгоритмы трекинга могут также использовать предсказательные модели, такие как фильтр Калмана или многочастичные фильтры, для предсказания будущего положения объекта. Эти модели учитывают текущую скорость и направление движения объекта, а также его размер, что позволяет более точно адаптировать bounding box при изменении ракурса. Например, если объект изменяет свою ориентацию в кадре, алгоритм трекинга использует предыдущие кадры для предсказания его нового положения и формы. Адаптивные трекеры могут корректировать bounding box, чтобы он оставался максимально близким к контурам объекта, даже если его видимые характеристики изменяются. Это обеспечивает устойчивое отслеживание объекта несмотря на изменения ракурса, сохраняя высокую точность детекции и трекинга.

3.1.5 Выводы

Использование алгоритмов трекинга в сочетании с моделями детекции объектов позволяет значительно улучшить их эффективность, решая такие проблемы, как пропуски в детекции, ложные срабатывания, нестабильность детекции и изменения масштаба и ракурса. Трекинг обеспечивает непрерывное и устойчивое отслеживание объектов, используя временную информацию и предсказательные модели, что делает системы компьютерного зрения более надежными и точными при выполнении реальных задач трекинга в различных сложных условиях.

3.2 Интеграция алгоритмов трекинга

3.2.1 Существующие решения

Пусть данное ответвление в отрасли еще довольно слабо развито, однако уже существуют примеры применения алгоритмов трекинга для повышения качества

детекции среди моделей в открытом доступе. Одной из таких моделей является YOLO v8, в которой применяется алгоритм DeepSORT (Simple Online and Realtime Tracking with a Deep Association Metric) [19].

Алгоритм DeepSORT расширяет возможности базового алгоритма SORT [58], добавляя метрику appearance для улучшения ассоциации объектов между кадрами. Этот алгоритм сочетает фильтр Калмана и расстояние Махалонобиса [59] для переноса информации о положении и движении объектов от одного кадра к другому. Основной процесс включает детекцию объектов с помощью YOLO v8, предсказание будущего состояния объектов с использованием фильтра Калмана и ассоциацию новых детекций с предсказанными состояниями объектов с помощью Венгерского алгоритма [60].

Главное улучшение в DeepSORT заключается в использовании метрики appearance, которая учитывает внешний вид объектов. Эта метрика была натренирована на отдельной нейронной сети, использующей более 1,100,000 изображений с более 1000 различных людей. Нейронная сеть создает вектор признаков (feature vector) размером 128x1 для каждого объекта, что позволяет алгоритму учитывать как динамику движения (предсказываемую фильтром Калмана), так и внешний вид объектов.

Для ассоциации объектов используется новая метрика, комбинирующая расстояние Махалонобиса (Dk) и косинусное расстояние (Da). Эта гибридная метрика применяется в Венгерском алгоритме для корректной идентификации объектов, минимизируя ошибки, такие как переключение ID при временной окклюзии.

DeepSORT является мощным инструментом для повышения качества детекции в модели YOLO v8. Он эффективно уменьшает количество ложных срабатываний и улучшает точность трекинга в условиях сложных движений и окклюзий. Высокие вычислительные требования и сложность настройки могут ограничить его использование в системах с ограниченными ресурсами, однако в целом, DeepSORT является одним из лучших решений для повышения качества детекции модели YOLO v8 в задачах, требующих высокой точности и надежности.

3.2.2 Разработанный программный модуль

Для решения указанной задачи по повышению качества работы модели детекции был разработан программный модуль, объединяющий модель детекции и выбранный алгоритм трекинга. В качестве детектора была выбрана передовая в сфере компьютерного зрения модель YOLO, которая активно использовалась при разработке системы контроля ручных операций в промышленном производстве на протяжении нескольких лет и которая показала себя как лучший и самый подходящий для разработки требуемого функционала системы детектор.

Перед применением описанных ранее подходов по использованию алгоритмов трекинга для повышения качества детекции, требуется понять, как в целом должен выглядеть рабочий процесс, включающий в себя одновременную работу и модели детекции, и алгоритма отслеживания, в итоге генерируя итоговый результат распознавания и локализации требуемого объекта на одном из последовательных кадров видеопотока. Используемый алгоритм рабочего процесса для разрабатываемой системы выглядит следующим образом:

- На первом кадре или группе кадров происходит начальная детекция, при которой идентифицируются и локализуются все объекты;
- Алгоритм трекинга инициализируется с использованием координат детектируемых объектов;
- Трекер отслеживает объекты на последующих кадрах, предсказывая их положение и обновляя свои данные модели;
- Детектор периодически применяется на следующих кадрах для обновления и проверки позиций объектов с помощью трекера (комбинированная детекция);
- Трекинг и детекция комбинируются для коррекции ошибок, сглаживания траекторий и фильтрации ложных срабатываний.

При написании части модуля, отвечающей за выбор и подключение алгоритмов трекинга, использовался весь доступный функционал языка Python и его библиотеки OpenCV. Библиотека частично предоставляла готовые модели

алгоритмов трекинга объектов на видео, что значительно упростило процесс разработки. Из алгоритмов трекинга были выбраны MOSSE [6], KCF [13], DeepSRDCF [35], AutoTrack [36], SiamRPN [15], SiamRPN++ [49] и SiamBAN [48], подробно рассмотренные и проанализированные ранее в тексте данной работы.

3.3 Анализ результатов испытаний

При проведении испытаний программного модуля, предназначенного для встраивания в систему контроля ручных операций, был создан обособленный от основной части приложения скрипт проверки. Его задачей являлось подключения лишь моделей детекции и трекинга и подсчет всех необходимых метрик в процессе обработки заранее подготовленных (записанных и размеченных) видеофайлов. Также скрипт проверял метрики качества трекинга для встроенного в YOLO v8 алгоритма DeepSORT, чтобы в дальнейшем можно было провести полноценный анализ всех подходов к решению проблем детекции (включая встроенный в YOLO v8 метод) и выбрать наилучший из них для использования в системе контроля ручных операций.

Во время проведения испытаний использовались особые метрики для оценки качества работы разработанного подхода в зависимости от выбранного алгоритма трекинга. Все упомянутые ранее проблемы моделей детекции довольно абстрактны и не имеют под собой четкого математического обоснования, что, соответственно, приводит к отсутствию способов оценки "величины" конкретной проблемы. В следствие этого, в процессе исследования способов решения проблем детекции были разработаны подходящие метрики, основанные на подсчете количества ложных срабатываний или допущенных пропусков модели детекции. Учитывая тот факт, что набор данных для валидации готового программного решения представляет из себя размеченные с помощью ограничивающих прямоугольников (bounding box) видео, становится возможным подсчет количества подобного рода ошибок модели детекции.

Разберем подробнее метрики для оценки качества решения каждой из исходных проблем моделей детекции. Начнем с проблемы пропусков детекций, число которых, как было замечено ранее, возможно посчитать для каждого заранее

размеченного видеофайла. Во время работы разработанного метода для каждого кадра создаются предсказания о положении объекта в кадре. Если вычислить для полученного в виде ограничивающего прямоугольника предсказания и корректного ограничивающего прямоугольника коэффициент пересечения (IoU) и проверить преодоление заранее определенного порога, то можно отнести текущее предсказание к корректному (true positive) или ошибочному (false positive). В случае проблемы пропусков детекций следует подсчитать количество false negative предсказаний, которые представляют из себя корректные (ground truth) ограничивающие прямоугольники для объектов, которые не были обнаружены, и разделить полученное число на общее количество кадров в видео потоке - полученное значение является долей пропусков детекций на видео. Чем значение данного коэффициента меньше, тем меньше было допущено пропусков детекций при применении модели. Данная величина отдаленно напоминает обратную к метрике детекции recall. Аналогичным образом можно посчитать долю ложных срабатываний модели. Чем больше будет значение доли ложных срабатываний, тем больше ошибок допускает модель. Данная величина отдаленно напоминает обратную к метрике детекции accuracy.

Для оценки качества решения проблемы нестабильности модели детекции требуется лишь посчитать долю нестабильных детекций, а именно детекция будет считаться нестабильной, если коэффициент пересечения IoU текущего предсказания с предсказанием на предыдущем кадре упал ниже требуемого порога при условии, что действительное положение объекта между кадрами не поменялось (для этого вычисляется степень пересечения между корректными ограничивающими прямоугольниками (ground truth) объекта на соседних кадрах, значение коэффициента должно быть выше определенного порога).

В случае с проблемой изменения масштаба и ракурса объекта в кадре также можно вычислить долю пропусков детекций для слишком большого изменения размера и формы ограничивающего прямоугольника, однако существует проблема с таким подходом, ведь при изменении ракурса ограничивающий прямоугольник может не поменять свою форму в случае, если объект интереса "объемный" с

каждого ракурса. Подобную проблему возможно решить, используя нейросети для сегментации изображений, однако в нашем случае проверочные видеофайлы были довольно малы, следовательно была возможность проверить и посчитать данные по данной метрике вручную для каждого кадра.

Дополнительно ко всем вышеперечисленным метрикам разработанный скрипт подсчитывал FPS работы для каждого подхода, что также позволит определить, какой из предложенных алгоритмов трекинга сможет при достаточно высоком качестве работы поддерживать работу всей системы в режиме реального времени.

Все полученные данные для стандартной модели YOLO v8 с отключенным алгоритмом DeepSORT, вместе с ним и с каждым выбранным отдельно алгоритмом трекинга вместо подхода DeepSORT представлены в Таблице 4. Также на Рис. 10-11 приведены примеры работы некоторых подходов и решение ими проблем моделей детекции для нескольких заранее размеченных видео файлов.

Таблица 4. Метрики качества для разработанных подходов, использующих YOLO v8 и алгоритмы трекинга с целью повышения эффективности работы модели детекции YOLO v8

Используемый подход	Доля пропусков детекции	Доля ложных срабатываний модели	Доля нестабильных детекций	Доля пропусков детекции при изменении масштаба и/или ракурса	Число кадров в секунду (FPS)	Подходит для режима реального времени
YOLOv8 + DeepSORT	0.05	0.07	0.45	0.11	72	Да
YOLOv8	0.18	0.2	0.76	0.3	81	Да
YOLOv8 + MOSSE	0.17	0.2	0.8	0.3	79	Да
YOLOv8 + KCF	0.15	0.18	0.74	0.26	69	Да
YOLOv8 + DeepSRDCF	0.05	0.08	0.4	0.1	3	Нет
YOLOv8 + AutoTrack	0.1	0.15	0.61	0.24	49	Да
YOLOv8 + SiamRPN	0.12	0.16	0.66	0.22	53	Да
YOLOv8 + SiamRPN++	0.08	0.11	0.54	0.19	32	Нет
YOLOv8 + SiamBAN	0.03	0.08	0.39	0.09	67	Да

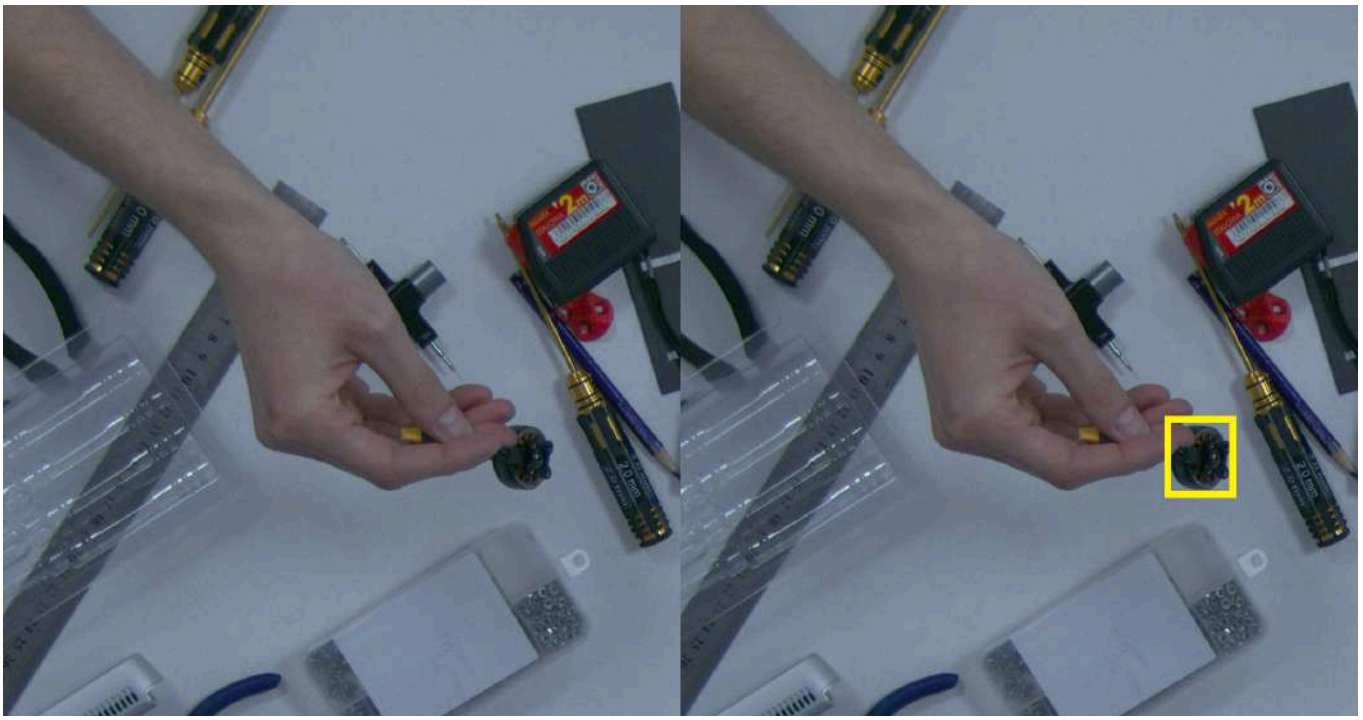

Рис. 10. Пример видеокадра в испытании по оценке решения проблемы пропусков детекций.
Слева - YOLOv8. Справа - YOLOv8 + SiamBAN.

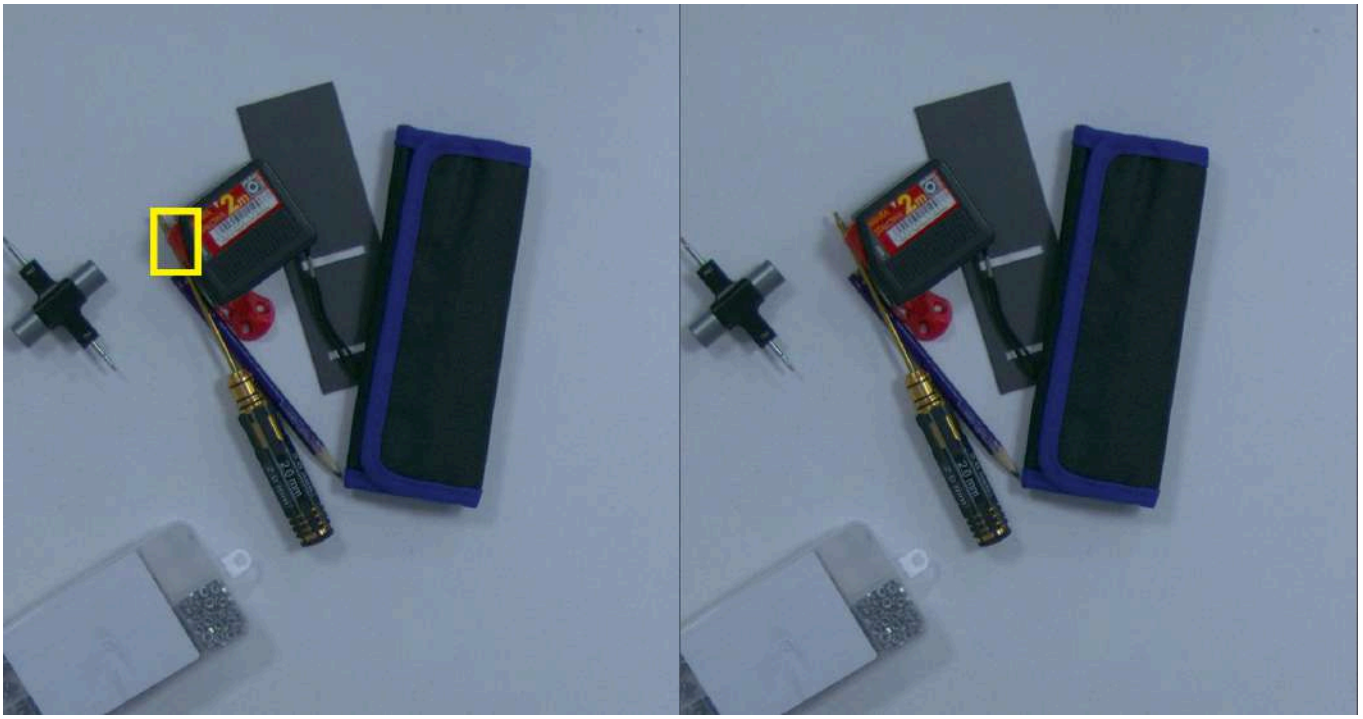

Рис. 11. Пример видеокадра в испытании по оценке решения проблемы ложных срабатываний.
Слева - YOLOv8. Справа - YOLOv8+SiamBAN.

Как видно по данным Таблицы 4, алгоритмы DeepSORT и SiamBAN демонстрируют наилучшие результаты по точности и стабильности, при этом обеспечивая высокую скорость обработки, подходящую для режима реального времени. Алгоритмы SiamRPN++ и DeepSRDCF имеют значительные ограничения по скорости, что делает их неподходящими для практических приложений в

реальном времени. Алгоритмы KCF и AutoTrack могут быть использованы вместе с моделью детекции YOLO v8, однако лишь в ограниченном количестве сценариев, где не требуется очень высокий FPS и высокий уровень качества детекции. Базовая модель YOLO v8 и ее комбинации с MOSSE и SiamRPN обеспечивают приемлемую скорость, но имеют более высокие показатели пропусков детекции и ложных срабатываний, что снижает их общую эффективность.

Из вышеперечисленных фактов можно сделать вывод, что встроенный в YOLO v8 алгоритм DeepSORT способен качественно работать, однако существует перспектива его улучшения, потому что разработанный программный модуль, который используется для внедрения в систему контроля ручных операций, способен заменить алгоритм DeepSORT, предустановленный в YOLO v8, моделью глубинного обучения SiamBAN, которая способна работать немного повысить качество работы модели детекции за счет небольшого падения FPS, что в нашем случае никак не отразится на эффективности работы системы стенда.

Таким образом, в данной работе был разработан подход по повышению качества работы модели детекции YOLO v8 с использованием модели SiamBAN за счет незначительного снижения скорости работы системы.

4 Реализация сценариев работы приложения

Разрабатываемое в данной работе приложение используется для контроля ручных операций в промышленном производстве. Оно предназначено для обеспечения правильности выполнения регламентированных операций при сборке изделий, выявления типовых ошибок и нарушений, таких как неправильная последовательность действий, ошибки в выборе компонентов и инструментов, а также нарушения техники безопасности. Основной целью стенда является повышение качества и надежности производственного процесса, особенно для предприятий оборонно-промышленного комплекса (ОПК), где высоки требования к точности и последовательности операций.

Для разработки новой версии приложения, ориентированной на сборку FPV-дронов, необходимо разработать алгоритмы работы нескольких новых

сценариев сборки изделия. В рамках данной работы создается программная основа этих сценариев. Введение новых сценариев сборки и адаптация алгоритмов детекции и трекинга позволят значительно повысить эффективность и точность системы, делая её пригодной для более сложных и разнообразных задач ручной сборки на производстве.

4.1 Технические требования и постановка задач

4.1.1 Технические требования

Для успешной реализации приложения, контролирующего ручные операции в сборочном производстве, необходимо соблюдение ряда технических требований к алгоритмам трекинга. Эти требования обеспечат высокую точность, надежность и скорость работы системы, соответствуя строгим стандартам производственного процесса:

- Алгоритмы трекинга должны обеспечивать высокие значения метрик точности (precision) и доли верных предсказаний (accuracy), что гарантирует минимальное количество пропусков объектов и ложных срабатываний. Показатель пересечения (Overlap Score или IoU) предсказанных и реальных ограничивающих рамок должен быть выше 0.75 для обеспечения точного определения положения объектов;
- Для обеспечения плавного и непрерывного трекинга объектов система должна работать с частотой не менее 30 кадров в секунду (FPS). Это позволит системе оперативно реагировать на изменения в зоне сборки и корректировать трек объектов в реальном времени;
- Алгоритмы трекинга должны быть устойчивы к изменяющимся условиям освещения, окклюзиям и изменению масштаба объектов. Это необходимо для обеспечения надежной работы системы в различных производственных условиях;
- Система должна эффективно распознавать и отслеживать мелкие детали размером от 10 мм и визуально схожие объекты. Это важно для предотвращения ошибок, связанных с неправильной идентификацией компонентов.

4.1.2 Постановка задач

Для выполнения вышеуказанных технических требований, в рамках данной работы ставятся следующие задачи:

- Изучение существующих алгоритмов трекинга, таких как MOSSE, KCF, DeepSRDCF, AutoTrack, SiamRPN, SiamRPN++ и SiamBAN. Тестирование и оценка производительности этих алгоритмов в условиях, имитирующих реальные производственные процессы;
- Адаптация алгоритмов для обеспечения работы с частотой не менее 30 FPS и минимальной задержкой обработки. Внедрение методов предсказания и сглаживания треков для повышения устойчивости к изменениям условий;
- Интеграция выбранных алгоритмов трекинга с моделями детекции (YOLO). Настройка совместной работы детекции и трекинга для обеспечения высокой точности и надежности системы;
- Проведение экспериментальных проверок на производственном стенде. Оценка работы системы в различных условиях и анализ полученных результатов для выявления и устранения возможных недостатков.

Эти задачи направлены на создание эффективной системы контроля ручных операций, которая сможет работать в условиях реального времени, обеспечивая высокое качество детекции и трекинга объектов в производственных процессах.

4.2 Адаптация алгоритмов

Для успешной реализации системы контроля ручных операций в сборочном производстве FPV-дронов, алгоритмы трекинга были адаптированы и модифицированы для конкретных сценариев сборки. Эти адаптации обеспечивают точное отслеживание деталей и корректную работу системы в условиях, характерных для сборочных процессов. В данном подразделе описаны конкретные изменения, внесенные в алгоритмы трекинга для решения задач, связанных с переносом деталей, исправлением специфических ошибок детекции и частичной сборкой компонент дрона.

4.2.1 Перенос винтов из коробки в зону сборки

Сценарий переноса винтов из конкретного отделения специального контейнера в зону сборки предполагает определение трека движения каждого винта по-отдельности. В дальнейшем эти же винты будут использованы для присоединения шасси к нижнему крепежу мотора через луч корпуса дрона, поэтому важно отслеживать их конкретные положения вплоть до наступления этого момента. К тому же, просчитав траекторию движения винтов в обратном направлении времени становится возможным определить тип и размер винтов, так как каждая зона контейнера замаркирована и используется для хранения винтов только одного конкретного размера и типа.

Для сценария переноса винтов из конкретного отделения коробки в зону сборки были выбраны несколько кандидатных алгоритмов: KCF, SiamRPN и SiamBAN. Эти алгоритмы были выбраны благодаря их достаточно высокой точности и устойчивости к изменениям масштаба и ракурса, что является критически важным при работе с мелкими и динамичными объектами, такими как винты. При этом каждый из этих алгоритмов можно использовать для решения задач в режиме реального времени, что тоже было одним из важных технических требований. Для обеспечения максимальной эффективности и повышения качества трекинга мелких деталей в данном сценарии были проведены оптимизации параметров алгоритмов (настройки гиперпараметров), включая настройку масштаба и чувствительности моделей к небольшим изменениям в положении и размере винтов. Интеграция предиктивных моделей позволила учитывать быстрые перемещения объектов, предсказывая траекторию движения винтов и корректируя треки в реальном времени. Дополнительно были добавлены функции предсказания траектории винтов, что позволило снизить вероятность пропусков и ложных срабатываний, обеспечивая более точное и стабильное отслеживание объектов на протяжении всего процесса их перемещения.

4.2.2 Исправление ошибок в классификации

Другим разрабатываемым в данный момент сценарием работы системы

является сборка корпуса дрона. Данный корпус является составным, следовательно для его полной сборки требуется последовательно соединять части дрона при помощи винтов. В зависимости от части корпуса используются винты разных форм и размеров. Среди этих винтов есть особенно сильно похожие винты (flat screw) размеров M2 и M3. При попытке классификации данных объектов модель YOLO v8 начинает часто путать имена классов, сохраняя при этом примерное расположение ограничивающего прямоугольника. Как видно на Рис.12, модель детекции может начать путать схожие по форме объекты. К тому же, зачастую это происходит не как появление лишь единственной детекции одного из двух классов, а появление ограничивающих прямоугольников сразу для двух классов. Получается так называемое «мигание» детекции, которое говорит о недостаточном качестве модели.

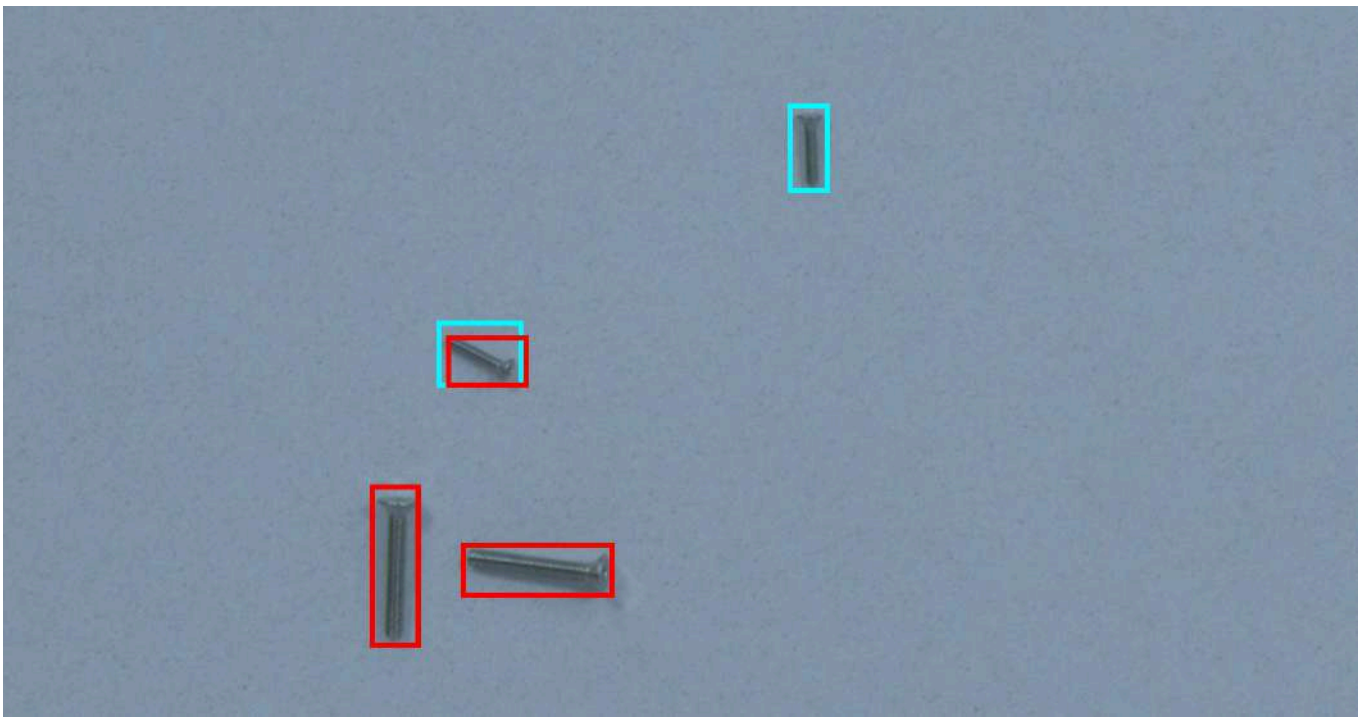

Рис. 12. Пример того, как YOLO путает винты M2 и M3 из-за их очень схожей формы.
Красный цвет соответствует детекции винта M3, голубой цвет - винт M2.

Для исправления ситуации, когда модель YOLOv8 путает детали между собой, были выбраны кандидатные алгоритмы SiamBAN и AutoTrack, которые обеспечивают высокую точность и устойчивость трекинга, что позволяет корректно идентифицировать детали и предотвращать их путаницу. Алгоритмы были выбраны из-за их способности эффективно справляться с различными ошибками детекции, особенно в условиях, когда необходимо точно различать визуально схожие объекты,

такие как винты M2 и M3 в одном из наборов данных для обучения модели детекции и написания сценария полноценной сборки корпуса дрона. Данные методы также успешно справляются с ситуациями, когда винт может быть ошибочно детектирован в пустой дырке луча или на фоне, благодаря мощным механизмам извлечения признаков и адаптивным моделям. Однако такой сценарий на данный момент является слишком сложным для разработки с точки зрения программирования, а также слишком трудозатратным в плане требуемых вычислительных ресурсов компьютера.

Для повышения точности детекции и устранения ложных срабатываний алгоритмы были объединен с моделью YOLOv8. Эта интеграция позволила использовать сильные стороны всех алгоритмов: YOLOv8 обеспечивает высокую точность детекции объектов, а SiamBAN или AutoTrack улучшают трекинг и устраняет ошибки детекции.

4.2.3 Отслеживание шасси и сборка моторов

Полный цикл сборки FPV-дрона предполагает сборку каждого из 4 моторов по-отдельности. Для отслеживания каждого мотора или шасси, присоединенного к каждому мотору, что является значительно более простой задачей, требуется разработать сценарий, основанный на алгоритмах трекинга множества объектов на видео. Возможность трекинга каждого мотора позволит автоматически определять, какие моторы на корпусе дрона уже собраны, а какие еще нет.

Для разработки данного сценария были выбраны алгоритмы MOSSE, KCF, AutoTrack, SiamRPN и SiamBAN в качестве кандидатов. Выбор пал на данные 5 алгоритмов отслеживания ввиду из достаточно высокой точности и скорости работы. Основной проблемой при разработке данного сценария сборки является необходимость производить отслеживание сразу нескольких объектов в кадре, что значительно повысит нагрузку на вычислительную систему и, соответственно, снизит общий FPS приложения. Исходя из данных рассуждений, было решено рассмотреть некоторые классические алгоритмы, которые способны выполнять обработку кадров заметно быстрее, чем модели глубинного обучения. К тому же, если учитывать тот факт, что в данном сценарии можно отслеживать не мотор, а

само посадочное шасси, то чаша весов еще больше начинает склоняться в сторону простых и быстрых алгоритмов трекинга, ведь шасси является довольно крупным и очень заметным объектов в кадре, отслеживать который должно быть крайне просто.

Все алгоритмы были модифицированы для работы с крупными компонентами и поддержания высокой частоты обновления (благодаря небольшой настройке гиперпараметров). Также была введена функция исключения собранных моторов из отображения на экране (часть пользовательского интерфейса).

4.2.4 Исправление ложных детекций

Последним из разработанных сценариев работы приложения является сценарий исправления ложных детекций модели YOLO. Во время работы первоначальной версии приложения по сборке FPV-дрона можно отчетливо заметить наличие большого количества ложных детекций, особенно в начале цикла сборки. Подобные ложные детекции специфичны конкретно для разработанного решения, поэтому их решение было вынесено в разработку отдельного сценария отслеживания подобных ложных предсказаний.

Для решения проблемы ложных срабатываний лучше всего подойдут алгоритмы SiamBAN и SiamRPN. SiamBAN выделяется своей высокой точностью и устойчивостью трекинга, эффективно различая визуально схожие объекты и минимизируя ложные детекции благодаря использованию мощных механизмов извлечения признаков и адаптивных моделей. SiamRPN сочетает в себе адаптивные методы и точное извлечение признаков, что улучшает способность алгоритма корректно идентифицировать объекты даже при изменениях их внешнего вида и условиях съёмки, тем самым снижая количество ложных срабатываний.

4.3 Процесс интеграции

После адаптации алгоритмов трекинга для конкретных сценариев сборки FPV-дронов необходимо интегрировать эти алгоритмы в существующую систему контроля ручных операций. В этом разделе представлено краткое описание процесса интеграции, включая технические аспекты и ключевые моменты. Алгоритм интеграции выглядит так:

1) Производится подготовка данных для обучения и тестирования моделей детекции и трекинга. Данные включают изображения и видео с аннотациями, отражающими положение и идентификаторы объектов;

2) Производится интеграция модели детекции (YOLOv8);

3) Производится интеграция алгоритмов трекинга, упомянутых в предыдущем разделе, их инициализация и настройка для работы с детекторами объектов;

4) Для обеспечения корректной работы системы выполняется синхронизация моделей детекции и трекинга объектов. Это включает периодическое обновление трекеров с использованием результатов детекции;

5) В конце каждого небольшого цикла работы системы производится тестирование качества работы системы. Это включает проверку точности детекции и трекинга, скорости обработки и устойчивости к изменениям условий.

Процесс интеграции алгоритмов трекинга в существующую систему контроля ручных операций включает несколько ключевых этапов, которые обеспечивают высокую точность и надежность системы. Данный подход позволяет эффективно контролировать сборочные операции FPV-дронов.

4.4 Результаты испытаний

Тестирование разработанных решений является важным этапом разработки системы контроля ручных операций в сборочном производстве FPV-дронов. В этом разделе представлены полученные результаты тестирования каждого из сценариев в виде таблиц. В соответствии с техническими требованиями к системе для каждого сценария сборки были измерены следующие метрики: Precision (точность), Accuracy (доля корректных предсказаний), Overlap Score (IoU) и FPS (кадры в секунду). Результаты представлены в таблицах 5-8.

Таблица 5. Результаты тестирования для сценария "Перенос винтов из коробки в зону сборки"

Алгоритм	Precision	Accuracy	Overlap Score	FPS
KCF	0.59	0.5	0.7	71
SiamRPN	0.75	0.63	0.78	45

SiamBAN	0.79	0.65	0.77	60

Таблица 6. Результаты тестирования для сценария "Исправление ошибок в классификации"

Алгоритм	Precision	Accuracy	Overlap Score	FPS
SiamBAN	0.83	0.67	0.81	63
AutoTrack	0.89	0.7	0.79	49

Таблица 7. Результаты тестирования для сценария "Отслеживание шасси и сборка моторов"

Алгоритм	Precision	Accuracy	Overlap Score	FPS
MOSSE	0.52	0.49	0.56	105
KCF	0.71	0.62	0.77	74
AutoTrack	0.92	0.71	0.9	46
SiamRPN	0.89	0.73	0.93	42
SiamBAN	0.95	0.8	0.96	58

Таблица 8. Результаты тестирования для сценария "Исправление ложных детекций"

Алгоритм	Precision	Accuracy	Overlap Score	FPS
SiamRPN	0.88	0.73	0.83	52
SiamBAN	0.92	0.68	0.8	59

Процесс тестирования алгоритмов трекинга включал тестирование на реальных данных, синтетические тесты и валидацию в реальном времени. Полученные результаты показали, что все алгоритмы имеют свои сильные и слабые стороны, и их производительность может варьироваться в зависимости от конкретного сценария сборки.

4.5 Анализ результатов испытаний

На основании результатов тестирования алгоритмов трекинга для различных сценариев сборки FPV-дронов была проведена оценка их эффективности. В этом разделе представлены сравнительный анализ алгоритмов и выбор наиболее

подходящих решений для каждого из сценариев.

4.5.1 Перенос винтов из коробки в зону сборки

Для сценария переноса винтов алгоритмы SiamRPN и SiamBAN показали наилучшие результаты и удовлетворили всем техническим требованиям. SiamRPN обеспечил высокую скорость обработки кадров при приемлемой точности, что делает его подходящим для задач в реальном времени. SiamBAN продемонстрировал более высокую точность и скорость работы, поэтому наилучшим выбором для сценария переноса винтов будет алгоритм SiamBAN.

На Рис. 13 представлен пример работы алгоритма SiamBAN в сценарии переноса винтов из коробки в зону сборки.

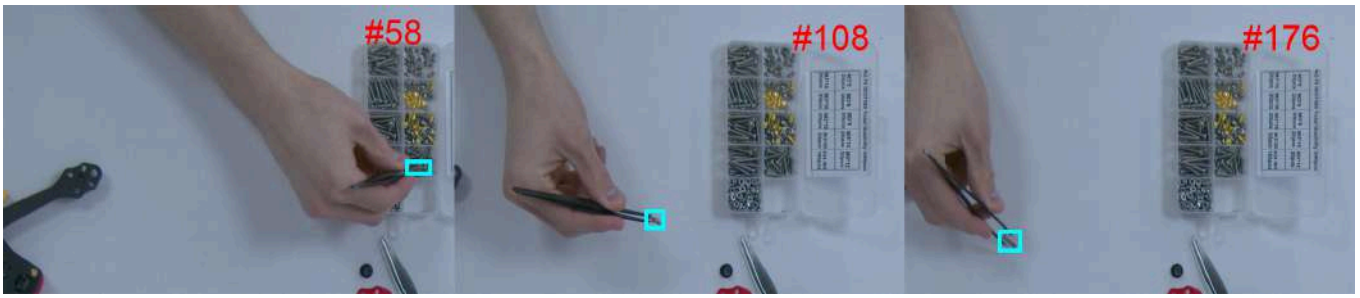

Рис. 13. Пример работы алгоритма SiamBAN в сценарии переноса винтов из коробки в зону сборки.

4.5.2 Исправление ошибок в классификации

В сценарии исправления ошибок классификации оба алгоритма показали высокий уровень точности, доли верных ответов и скорость обработки данных, однако, как показали эксперименты, на место алгоритма трекинга в этом сценарии сборки следует взять метод AutoTrack, который оказался способен наилучшим образом решать подобные задачи по исправлению ошибок в классификации и устранению возможных «миганий» детекций.

На Рис. 14 представлен пример работы алгоритма AutoTrack в сценарии исправления ошибок классификации.

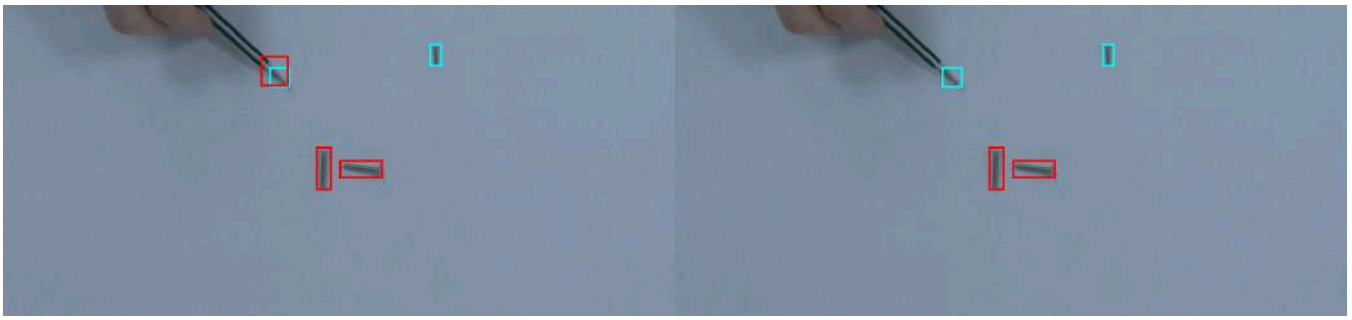

Рис. 14. Пример работы алгоритма AutoTrack в сценарии исправления ошибок классификации. Слева - без использования AutoTrack. Справа - с использованием AutoTrack.

4.5.3 Отслеживание шасси и сборка моторов

Для реализации работы сценария по отслеживанию моторов (или скорее шасси) и сборки моторов было выбрано довольно много кандидатов алгоритмов трекинга. Хоть алгоритмы AutoTrack, SiamRPN и SiamBAN показали отличные результаты скорости работы и значения метрик точности и доли верных ответов, не меньшего внимания в данной ситуации заслуживает алгоритм KCF. Данный алгоритм не достиг тех же результатов, что и более продвинутые версии трекеров, однако по результатам становится очевидно, что скорость работы данного подхода куда выше, чем у остальных, подходящих под технические требования. Точности работы данного алгоритма вполне достаточно для реализации работы сценария отслеживания шасси, которые являются довольно заметными объектами, настолько, что даже простые алгоритмы, основанные на корреляционных фильтрах способны достаточно качественно выполнять задачи отслеживания.

На Рис. 15 представлен пример работы алгоритма KCF в данном сценарии.

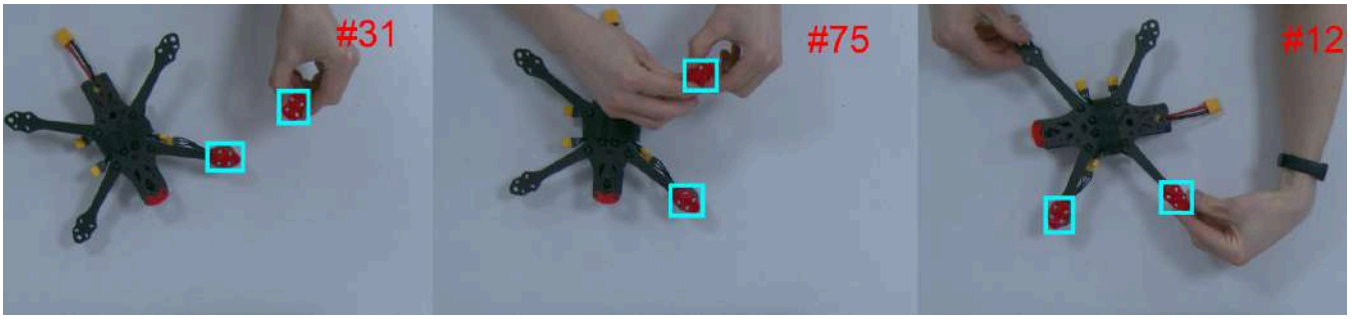

Рис. 15. Пример работы алгоритма KCF в сценарии отслеживания шасси и сборки моторов.

4.5.4 Исправление ложных детекций

Для последнего сценария по исправлению ложных детекций были проверены всего 2 алгоритма трекинга, которые показали внушительные результаты в скорости

обработки данных и полученных метриках. В данном случае выбор не столь очевиден, ведь в случаях с исправлением ложных детекций следует полагаться больше на метрику доли верных ответов и IoU, чем на точность. Этот вывод следует из размышлений о том, как в данном сценарии должны вести себя корректные и ошибочные ограничивающие прямоугольники. Основываясь на упомянутых фактах, выбор алгоритма для реализации данного сценария становится очевидным - SiamRPN.

На Рис. 16 представлен пример работы алгоритма SiamRPN в данном сценарии.

Рис. 16. Пример работы алгоритма SiamRPN в сценарии исправления ложных детекций. Слева - без использования SiamRPN. Справа - с использованием SiamRPN.

4.6 Выбор алгоритмов

На основании сравнительного анализа были выбраны наиболее эффективные алгоритмы трекинга для каждого сценария сборки FPV-дронов. Для сценария переноса винтов из коробки в зону сборки выбираем алгоритм SiamBAN. Для сценария исправления ошибок классификации выбираем алгоритм AutoTrack Для сценария отслеживания шасси и сборки моторов выбираем алгоритм KCF. Для сценария исправления ложных детекций выбираем алгоритм SiamRPN. Эти алгоритмы обеспечивают оптимальное сочетание точности и скорости для различных задач, что способствует повышению эффективности и надежности системы контроля ручных операций в сборочном производстве.

5 Система контроля сборки узлов FPV-дрона

5.1 Функциональные элементы системы

Система распознавания и интеллектуального контроля ручных операций в

промышленном производстве предназначена для автоматизации решения следующих задач:

- Контроль действий оператора-сборщика с помощью систем интеллектуального видеонаблюдения;
- Распознавание некорректных (не соответствующих белому списку) операций в процессе сборки деталей и конструкций;
- Распознавание несоблюдения техники безопасности во время работы на производстве;
- Предоставление отчетности (статистики) о процессе сборки изделия оператору-сборщику и руководителю производства.

На предприятии важную роль играет производительность труда и эффективность рабочего процесса. Данные, собираемые в процессе производства используются аналитиками для выявления слабых мест и оптимизации различных процессов. Приложение по контролю ручных операций также позволяет аккумулировать основную статистику по сборочному процессу.

Автоматизированная система “Испытательный стенда автоматизации и интеллектуального контроля ручных операций” включает следующие группы ключевых функциональных элементов:

- рабочее поле оператора и станина для размещения оборудования, объектов сборки, деталей;
- осветительное оборудование, камеры, проецирующее оборудование;
- аппаратная компьютерная платформа и соединительные провода;
- программные подсистемы.

Система адаптирована для работы в различных операционных системах (Linux, Windows). Компоненты, входящие в состав системы, основаны на монолитной архитектуре и доступны пользователям через графический интерфейс. Все компоненты, входящие в состав системы, построены в едином стиле и на одной технологии реализации, работают в формате десктопного приложения. Вся система состоит из нескольких подсистем. Для каждой

подсистемы приведем перечень выполняемых ею функций и задач:

Таблица 9 – Программная подсистема обработки входного потока данных

Функция	**Задачи**
Обработка данных с камер	1. Сбор входного видеопотока с камер высокого разрешения 2. Передача каждого кадра в модель компьютерного зрения для обработки и получения характеристик деталей и рабочей области
Обработка данных с внешних устройств	Обработка внешних сигналов пользователя, переданных системе от клавиатуры и мыши

Таблица 10 – Программная подсистема хранения данных

Функция	**Задачи**
Запись, хранения и модификация данных	1. Сохранение значений ранее загруженных данных в случае их изменения 2. Составление файлов JSON для дальнейшей работы с данными

Таблица 11 – Программная подсистема отрисовки вспомогательных элементов

Функция	**Задачи**
Хранение информации о всех вспомогательных визуальных элементах	Хранение информации о цвете, размере, местоположении на экране, толщине линии и других характеристиках граничных прямоугольников для рабочей области и областей хранения деталей
Анализ и расчет характеристик каждого сопроводительного элемента	1. Сбор данных о текущих размерах и положении рабочей области в кадре камеры 2. Вычисление корректных координат каждого вспомогательного элемента для отображения в рабочей области 3. Своевременное обновление характеристик
Визуализация сопроводительных элементов поддержки	1. Вывод элементов на мониторе вычислительной машины 2. Вывод элементов в рабочей области благодаря проектору

Таблица 12 – Программная подсистема обработки входного потока данных

Функция	**Задачи**

Обработка данных с камер	1. Сбор входного видеопотока с камер высокого разрешения 2. Передача каждого кадра в модель компьютерного зрения для обработки и получения характеристик деталей и рабочей области
Обработка данных с внешних устройств	Обработка внешних сигналов пользователя, переданных системе от клавиатуры и мыши

Таблица 13 – Аппаратная компьютерная платформа

Функция	**Задачи**
Запуск и работа всех программных подсистем	Запуск, функционирование всех программных подсистем
Аппаратное сопряжение с сетевой инфраструктурой (при необходимости)	Подключение к информационно-вычислительной сети (по умолчанию, по интерфейсу UTP/Ethernet)
Подключение другого оборудования стенда	Подключение видеокамер и др. устройств

Таблица 14 – Рабочее поле стола и станина

Функция	**Задачи**
Пространство для работы с деталями и объектом сборки	Создание пространства на рабочем столе для расположения деталей сборки и работы с ними
Несущая функция для расположенных на станине и рабочей зоне объектов	Обеспечение стабильности и непоколебимости и закрепленных частей стенда: камер, осветительных устройств, деталей и т.п.
Место размещения оператора	Стул, с опорой для спины

Таблица 15 – Рабочее поле стола и станина

Функция	**Задачи**
Получение последовательностей изображений (видео) рабочей зоны	Получение (съемка) последовательностей изображений (видео) рабочей зоны оператора в режиме реального времени
Передача данных на вычислительную аппаратную платформу	Передача видео данных на вычислительную аппаратную платформу в режиме реального времени

Таблица 16 – Рабочее поле стола и станина

Функция	**Задачи**
Освещение области сборки	Освещение области сборки до заданных значений
Отображение (проецирование) данных на рабочую область сборки	Отображение информации в режиме реального времени на область сборки: разметка зоны сборки, информирование о ходе сборки и ошибках и т.п.
Отображение (визуализация) информации	Отображение информации в режиме реального времени на экране монитора
Озвучивание информации (опционально)	Аудио-информирование оператора (по акустическому каналу) о ходе сборки и др. событиях при работе стенда

5.2 Сценарий контроля сборочного процесса

Слежение за рабочей областью оператора осуществляется с помощью видеокамеры высокого разрешения, расположенной над столом. Захватываемый камерой кадр поделен на произвольное число зон. За работу с зонами отвечает отдельный класс Zone. После обработки кадра нейросетью и получения координат прямоугольников деталей в рабочей области происходит вычисление их пересечения с зонами. Процент перекрытия детали и зоны для того, чтобы считать, что деталь находится в данной зоне – настраиваемый параметр. В демонстрационном варианте приложения на стенде в лаборатории считаем, что если пересечение детали и зоны больше 50%, то деталь находится внутри данной зоны.

Процесс сборки изделий на производстве строго регламентирован, и сборщик обязан его придерживаться. Алгоритм подразумевает единственно верную последовательность соединения деталей изделия. Этапы сборки выполняются последовательно, причем каждый из них может включать в себя две составляющие:

1) оператор раскладывает необходимое число деталей в определенных зонах;

2) оператор демонстрирует соединение деталей в камеру.

Пока все условия текущего шага не будут выполнены, переход на следующий шаг не произойдет. Система будет ожидать полного выполнения инструкций

текущего этапа. Информационные текстовые сообщения о действиях на текущем шаге выводятся на экран.

5.3 Работа основного приложения

Программа состоит из основного файла с кодом на языке python и нескольких пакетов, которые выполняют вспомогательные функции в коде, например, содержат описание необходимых абстракций и методов взаимодействия с ними.

Для запуска потребуется python версии от 3.7 до 3.9 (требования библиотеки pytorch) с установленными библиотеками - PyYaml, tqdm, scipy, pytorch, numpy, opencv, pandas, pytorch, pyyaml, requests, seaborn, setuptools, torchvision, tk, pyqt, qt, opencv, yolov5. Возможно использование виртуальных сред, таких как Anaconda.

Обратим внимание, что для быстродействия системы дополнительно требуется установить драйвера для графического чипа и пакет CUDA, а также проверить, что устанавливаемая версия pytorch поддерживает установленную версию компилятора.

После установки всех зависимостей переходим к корень каталога с кодом и выполняем команду python3 main_workflow.py. Во время выполнения откроется отдельное окно с демонстрацией работы программы. Для выхода из программы следует нажать клавишу “q” на клавиатуре. Для перехода вручную на следующий этап следует нажать клавишу “n” на клавиатуре.

В демонстрационном варианте приложения пользователю предложено собрать мотор квадрокоптера под контролем интеллектуальной системы. На экране появляются инструкции для сборки конкретного шага. Пошаговый пример работы приложений приведен ниже.

5.4 Пример работы приложения на кейсе сборки мотора дрона

Оператора сборщика требуют выполнить инструкции указанные на экране для соответствующего этапа сборки. Подсказки для совершения правильных действий на текущем этапе выводятся на втором мониторе в виде gif файлов. На

левом gif файле показаны необходимые для успешного завершения текущего этапа действия на текущем шаге, в то время как на правом изображении показано, каким будет следующий шаг. После выполнения инструкций текущего этапа на экране выводится соответствующее сообщение об успешном завершении текущего этапа сборки. Если какой-либо объект в кадре обводится красным цветом, то значит, что на текущем шаге он не должен находиться в данной области. Такие объекты необходимо перенести в нужные зоны и дождаться обводки его зеленым цветом, либо отсутствие выделения совсем. Приложение зациклено, что позволяет непрерывно собирать необходимое количество изделий не перезапуская его.

5.4.1 Этап 0

Оператора сборщика требуют расположить детали и инструменты в соответствующих зонах. Пример работы нулевого этапа приведен на Рис. 17.

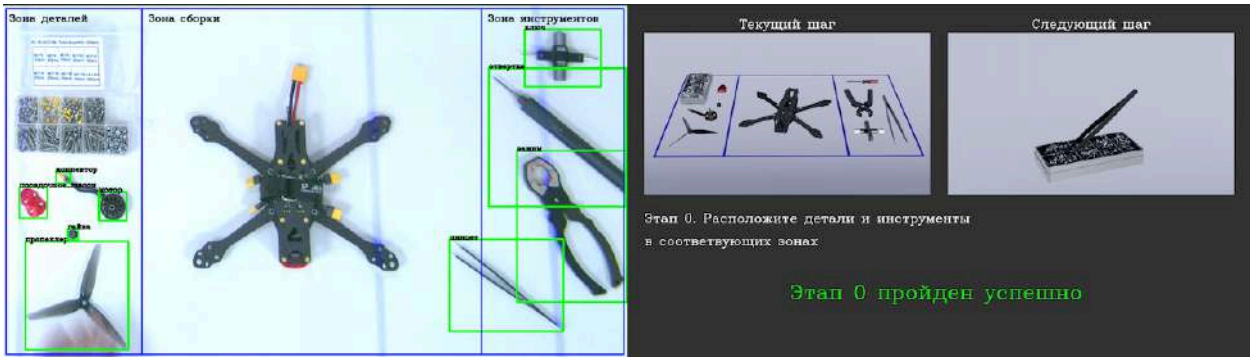

Рис. 17. Пример работы нулевого этапа

5.4.2 Этап 1

Оператора сборщика требуют взять пинцет и с его помощью перенести 4 винта М3 в зону сборки и вернуть пинцет в зону инструментов. Пример работы первого этапа приведен на Рис. 18.

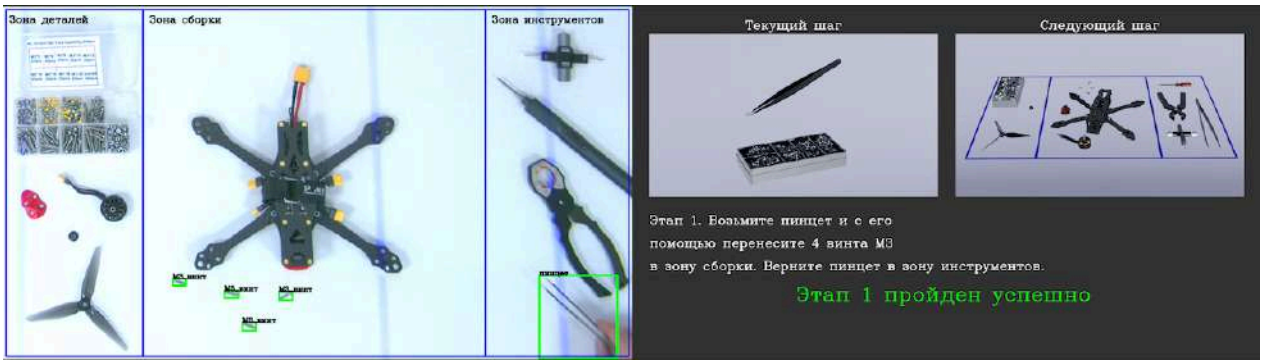

Рис. 18. Пример работы первого этапа

5.4.3 Этап 2

Оператора сборщика требуют переместить мотор и посадочное шасси в зону сборки. Пример работы второго этапа приведен на Рис. 19.

Рис. 19. Пример работы второго этапа

5.4.4 Этап 3

Оператора сборщика требуют взять отвертку и с ее помощью прикрепить посадочное шасси к мотору через луч с помощью винтов М3 и вернуть отвертку в зону инструментов. Пример работы третьего этапа приведен на Рис. 20.

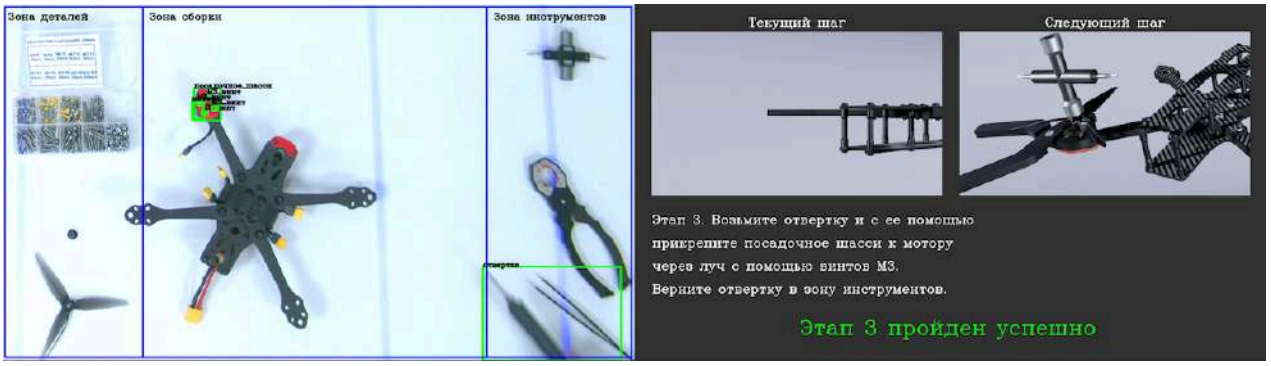

Рис. 20. Пример работы третьего этапа

5.4.5 Этап 4

Оператора сборщика требуют перенести гайку и пропеллер в зону сборки и с помощью ключа и зажима закрепить пропеллер и вернуть ключ и зажим в зону инструментов. Пример работы четвертого этапа приведен на Рис. 21.

Рис. 21. Пример работы четвертого этапа

5.5 Контроль за соединениями изделия

Помимо правильного расположения деталей в рабочей области и контроля за последовательностью их перемещения, необходимо следить за корректностью крепления деталей между собой. Для решения данной задачи система детектирует промежуточные соединения. Большая часть шагов совмещает в себе правильное расположение деталей и демонстрацию соединения. У метода определения корректности соединения с помощью одной камеры, расположенной строго над рабочей поверхностью есть очевидные недостатки. Один ракурс не позволяет точно сделать вывод о том, действительно ли детали прикреплены друг к другу правильным образом. Рассмотрим на смоделированном примере. На левом рисунке вид сверху соединения подшипника и куска балки. Этот кадр получает на вход нейросеть с верхней камеры. По данному виду невозможно понять, насколько кольцо “утоплено” в балку. Это можно понять лишь получив второй вид с другого ракурса как показано на правом рисунке, Рис. 22.

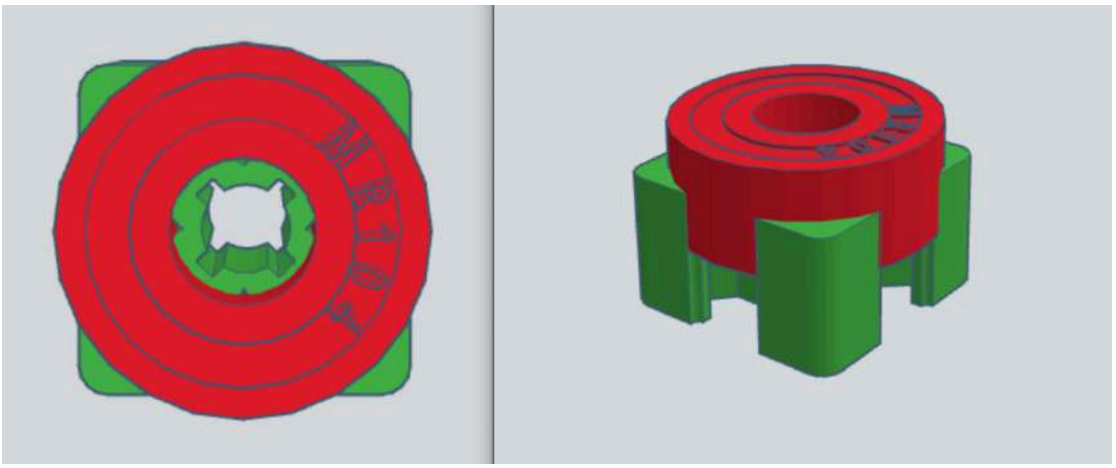

Рис. 22. Смоделированное соединение подшипника и балки

Для этого было решено использовать вторую камеру “Камера 2”, расположенную ниже, которая позволяет получить вид изделия сбоку, Рис. 23.

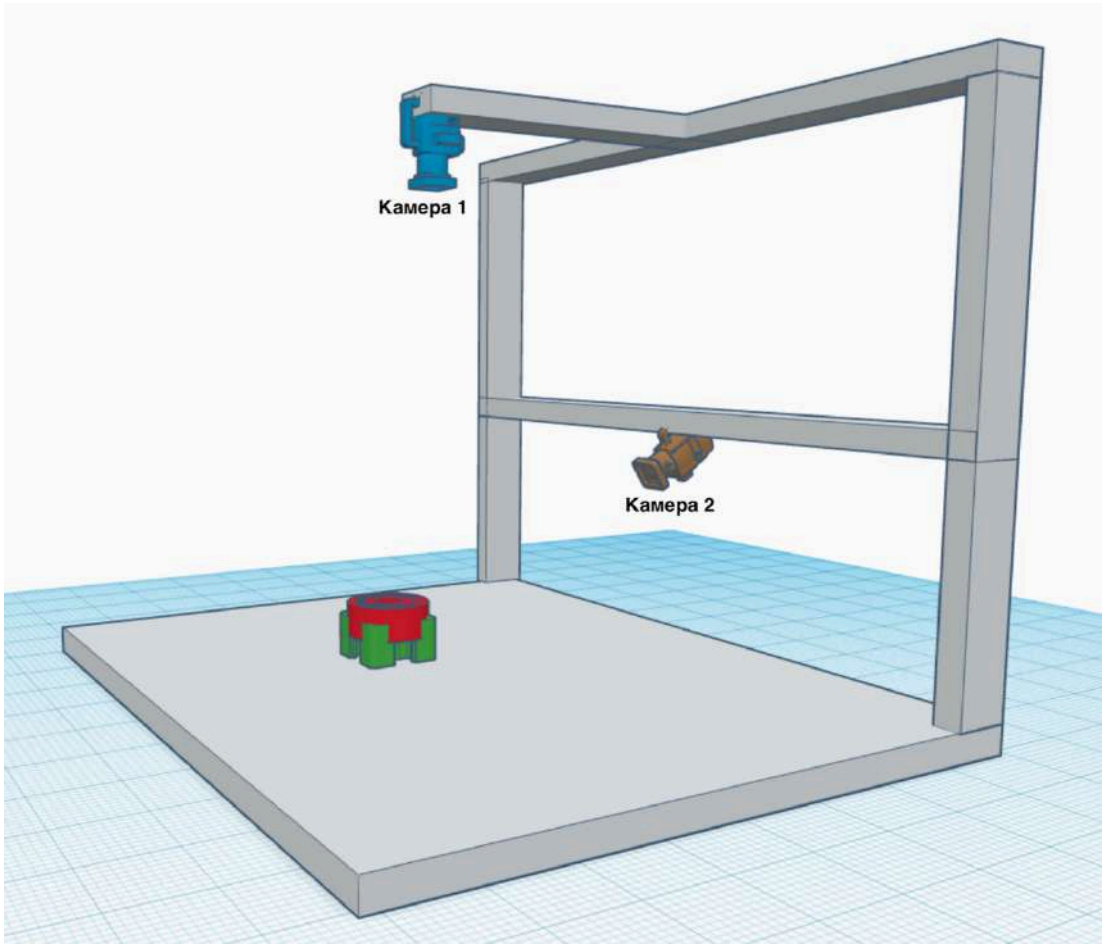

Рис. 23. Расположение двух камер для контроля промежуточных соединений

Таким образом, вторая камера позволяет увеличить угол обзора и уменьшить размеры мертвых зон, Рис. 24. Плоскостями схематично показаны области полей зрения камер.

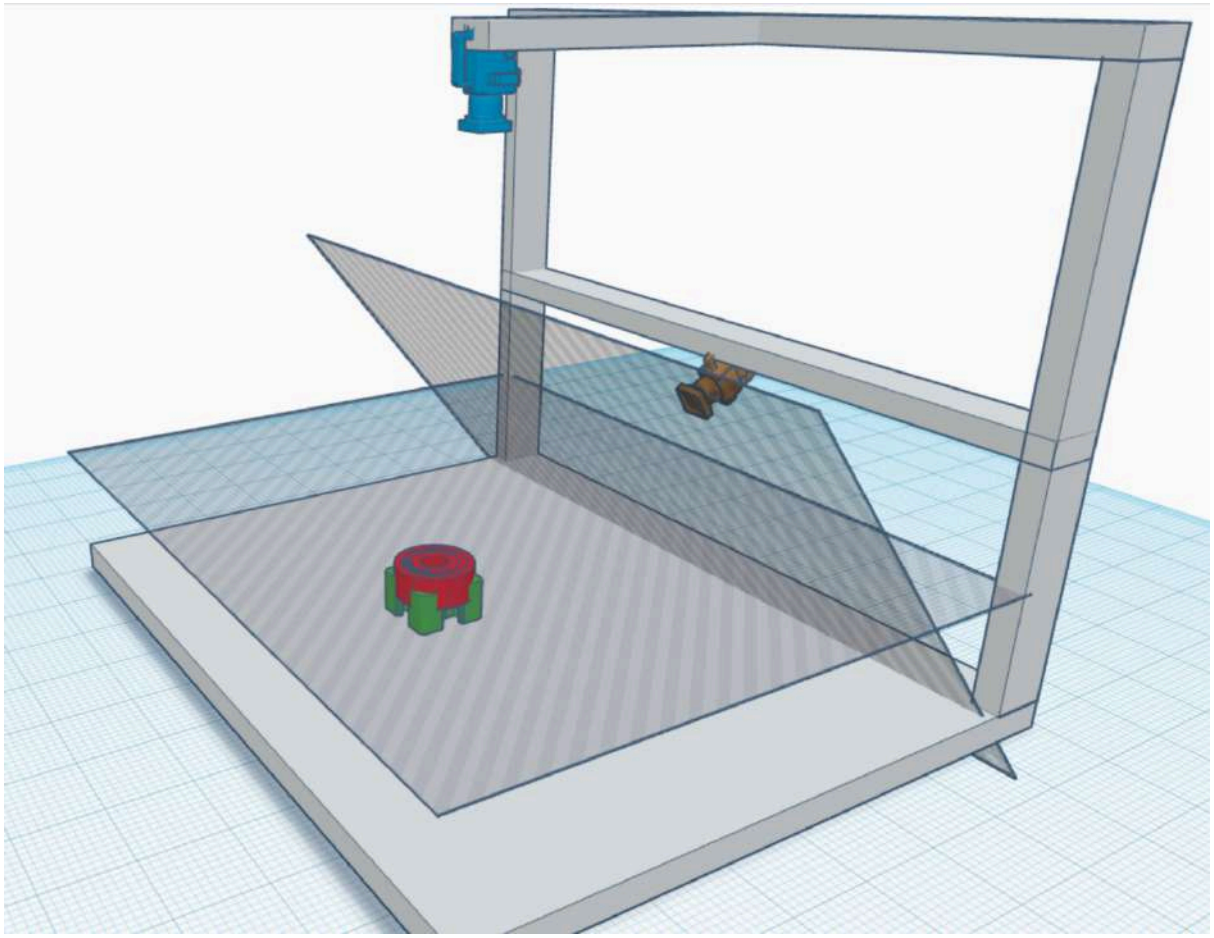

Рис. 24. Поля зрения камер для контроля промежуточных соединений

Рассмотрим процесс принятия решения системой о том, какое соединение сейчас демонстрируется в рабочей области. Изображение с верхней (ведущей) камеры поступает в модель детекции промежуточных соединений. Модель возвращает список из задетектированных соединений с вероятностями каждого из них. Далее идет сравнение полученных вероятностей с ручным порогом, который задается заранее в config файле приложения. Если соединение имеет наибольшую вероятность из всех вероятностей соединений, которые выше порога, то считаем, что камера 1 видит именно это соединение. После получения результата детекции с верхней камеры возможны два случая. Первый случай, когда модель либо не задетектировала ни одного соединения в кадре, либо все вероятности задетектированных соединений меньше порога. В этом случае программа начинает новый цикл обработки следующего кадра. Второй случай, когда модель уверена, что верхняя камера видит определенное соединение. Тогда весь предыдущий алгоритм повторяется для второй вспомогательной камеры. Кадр с камеры 2 обрабатывается

той же моделью. Далее идет сравнение полученных результатов. Если обе камеры задетектировали соединение, которое по инструкции должно быть на текущем этапе сборки, то приложение дает перейти к следующему этапу и выводит соответствующее сообщение об этом на экран. Пороги вероятностей детекции соединений для двух камер независимы и могут быть различными для каждой из камер. В приложении на стенде экспериментальным путем были подобраны значения порогов 0.8 для камеры 1 и 0.85 для камеры 2.

Таким образом, спроектированное система показывает возможность успешного применения интеллектуального контроля ручных операций во время сборки сложных технических устройств с большим числом мелких деталей. Использование промышленных камер и современных алгоритмов детекции позволяет в режиме реального времени осуществлять непрерывный контроль за действиями оператора-сборщика на рабочем месте.

6 Увеличение FPS приложения

6.1 Цель

Повышение количества кадров в секунду (FPS) в приложении для контроля ручных операций на производстве. Улучшение FPS предполагается осуществить путем оптимизации вычислений, в том числе через более эффективное распределение задач между графическим (GPU) и центральным (CPU) процессорами. Высокий FPS обеспечит более точное и быстрое отслеживание операций в реальном времени, что улучшит качество контроля и безопасность трудовых процессов.

6.2 Теоретическая основа

6.2.1 FPS

FPS (Frames Per Second) является мерой частоты смены кадров в приложении, определяя, насколько плавно отображается видео и графика. Чем выше FPS, тем более плавным для пользователя становится визуальный опыт. Низкий FPS может

привести к рывкам и задержкам, что негативно сказывается на взаимодействии пользователя с приложением.

6.2.2 Параллельные и конкурентные вычисления

Параллельное выполнение (Parallel execution) подразумевает физическое одновременное выполнение нескольких задач. Это достигается за счет использования многоядерных процессоров, где каждое ядро может обрабатывать отдельную задачу. Параллелизм идеально подходит для задач, которые можно разделить на независимые подзадачи, выполняемые одновременно без необходимости взаимодействия между ними.

Конкурентное выполнение (Concurrent execution) предполагает, что система может обрабатывать несколько задач в один и тот же момент времени. Это не обязательно означает, что задачи выполняются физически одновременно. В одноядерных системах это достигается за счет быстрого переключения контекста между задачами, создавая иллюзию одновременности. В многоядерных системах конкурентность может быть реализована с помощью параллелизма, но ключевой особенностью является возможность взаимодействия и совместной работы задач.

6.3 Методы исследования

6.3.1 Перевод вычислений на GPU

6.3.1.1.Теория

Использование GPU для выполнения вычислений позволяет значительно ускорить обработку данных благодаря его способности выполнять тысячи параллельных операций. GPU имеют значительно больше ядер по сравнению с CPU, что делает их идеальными для задач, требующих массового параллелизма.

Библиотеки, такие как CUDA (для графических процессоров NVIDIA) и OpenCL, предоставляют средства для разработки программ, использующих мощность GPU.

6.3.1.2.Практика

В данном исследовании был выполнен перевод вычислительной части приложения на GPU с использованием библиотеки CUDA .

В таблице 17 представлено сравнение производительностей различных моделей, выполняемых на CPU и двух GPU: NVIDIA GeForce RTX 3080 (обозначена как 0) и NVIDIA GeForce GTX 1660 (обозначена как 1). В последнем столбце таблицы указан процент увеличения FPS (кадров в секунду) по сравнению с выполнением на CPU.

Таблица 17. Производительность моделей при выполнении на различных комбинациях GPU и CPU

Назначение моделей на процессоры			Результаты	
palm model	drone tools model	drone details model	Количество кадров в секунду, обработанных системой при выполнении моделей, FPS	Процентное изменение FPS по сравнению с выполнением на CPU, %
cpu	cpu	cpu	10.585795500815214	0
0	0	0	10.760937562563763	1,65450071
0	0	1	10.795655313251011	1,982466149
0	1	0	10.97882685224355	3,712818289
0	1	1	9.892921512065698	-6,545318098
1	0	0	10.902879716688143	2,99537447
1	0	1	9.942854624773434	-6,073618898
1	1	0	10.01346692060467	-5,406571289
1	1	1	9.745736339833135	-7,935720664

Наилучшей комбинацией оказалось выполнение моделей на GPU 0 (NVIDIA GeForce RTX 3080), GPU 1 (NVIDIA GeForce GTX 1660) и снова GPU 0, что дало прирост FPS на 3,71% по сравнению с выполнением на CPU.

Однако этот прирост производительности является незначительным. В нашем

случае возможно, что вычислительные задачи не подходили для архитектуры GPU. Если задачи содержат множество последовательных операций и зависимостей, то использование GPU может не дать ожидаемого прироста производительности.

6.3.2 Многопоточность (Threading и ThreadPoolExecutor)

6.3.2.1.Теория

Многопоточность - это способность программы выполнять несколько потоков (нитей) одновременно. Каждый поток представляет собой отдельный путь выполнения внутри одного процесса.

Поток (Thread) - это наименьшая единица выполнения программы, которой операционная система выделяет время процессора. Каждый поток имеет собственный стек, регистры и счетчик команд, но разделяет общую область памяти с другими потоками того же процесса. Потоки позволяют эффективно использовать ресурсы процессора и повысить отзывчивость приложений.

Пул потоков (Thread Pool) - это набор предварительно созданных потоков, которые ожидают поступления задач для выполнения. Вместо создания нового потока для каждой задачи, задача назначается одному из свободных потоков в пуле. Это позволяет избежать накладных расходов на создание и уничтожение потоков, повышая производительность приложения. Пул потоков обычно управляется специальным планировщиком, который распределяет задачи между потоками.

GIL (Global Interpreter Lock) - это механизм в реализации Python, который позволяет только одному потоку выполнять байт-код в любой момент времени. Это ограничение было введено для упрощения реализации интерпретатора Python и предотвращения проблем с потоками при работе с объектами в памяти. Однако GIL также ограничивает возможности истинного параллелизма в Python, поскольку только один поток может выполняться одновременно.

6.3.2.2.Практика

В рамках исследования были созданы три отдельных потока, каждый из которых был назначен для обработки одной из следующих моделей:

- Palm Model
- Drone Tools Model
- Drone Details Model

Каждая модель была назначена на отдельный поток с целью распараллелить вычисления и повысить производительность приложения.

Однако, из-за ограничений, накладываемых GIL, только один поток мог выполнять байт-код в любой момент времени. Это означает, что потенциальные преимущества параллелизма были ограничены, и многопоточность не смогла обеспечить увеличения FPS.

Результаты исследования показали, что использование многопоточности (Threading и ThreadPoolExecutor) не привело к увеличению FPS в приложении. Это связано с тем, что GIL не позволяет реализовать параллелизм в Python.

6.3.3 Многопроцессная обработка (Multiprocessing)

6.3.3.1.Теория

Многопроцессная обработка, или multiprocessing, в Python — это метод параллельного выполнения задач, использующий отдельные процессы вместо потоков. Это позволяет эффективно использовать многопроцессорные и многоядерные системы, поскольку каждый процесс может выполняться на отдельном ядре процессора.

Процесс в контексте многопроцессной обработки представляет собой независимую единицу выполнения в компьютерной системе с собственным набором инструкций и состоянием. Каждый процесс обладает собственным виртуальным адресным пространством, что означает изоляцию процессов друг от друга и невозможность прямого взаимодействия с памятью или ресурсами других процессов без использования специальных механизмов межпроцессного взаимодействия.

Основное преимущество многопроцессной обработки перед многопоточностью заключается в том, что процессы не ограничены GIL (Global Interpreter Lock), характерным для потоков в Python. Это дает возможность

параллельной работы на разных ядрах процессора, обеспечивая истинный параллелизм и улучшение производительности задач. Каждый процесс имеет отдельное пространство памяти, что уменьшает риск конфликтов данных и упрощает синхронизацию, в отличие от потоков, которые работают в рамках одного процесса и делят общее пространство памяти. Однако управление процессами требует больше системных ресурсов, чем управление потоками, так как операционной системе нужно выделить дополнительные ресурсы для каждого запущенного процесса, что приводит к увеличению накладных расходов.

Несмотря на то, что GIL применяется в каждом процессе, созданном через многопроцессность, он не ограничивает параллелизм между процессами, так как они функционируют независимо друг от друга.

Однако при использовании механизмов межпроцессной коммуникации для обмена данными между процессами возникают следующие проблемы:

Гонки данных (Data races): Это ситуация, когда несколько процессов пытаются одновременно получить доступ к одним и тем же данным для чтения или записи. Может привести к тому, что один процесс перезапишет изменения, сделанные другим, что приведет к потере данных или непредсказуемому поведению программы. Например, если два процесса одновременно пытаются обновить одну и ту же запись в базе данных, результат может зависеть от того, какой процесс завершит свою операцию последним.

Deadlocks (Взаимные блокировки): Взаимная блокировка возникает, когда два или более процесса ожидают освобождения ресурсов, которые уже заняты другими процессами. Каждый из процессов ждет, пока другой освободит ресурс, но ни один из них не может продолжить выполнение, так как они заблокированы. Это может привести к полной остановке всех процессов, участвующих в блокировке.

6.3.3.2.Практика

Были проведены два эксперимента с использованием многопроцессности (multiprocessing).

В первом эксперименте один процесс отвечал за выполнение всех трех

моделей, а второй — за захват изображения и отрисовку. Однако это не привело к увеличению скорости работы, поскольку выполнение моделей, осуществляемое последовательно в одном процессе, оказалось самой ресурсоемкой задачей, и параллельная работа второго процесса не смогла компенсировать этот недостаток.

Во втором эксперименте задачи были распределены между тремя процессами, каждый из которых выполнял одну из моделей, в то время как основной процесс занимался захватом и отрисовкой. Это позволило каждой модели работать независимо и параллельно, что должно было способствовать повышению общей производительности.

Использование разделяемой памяти и очередей предотвратило гонки данных, обеспечивая упорядоченный и контролируемый доступ к данным. Однако, когда процессы должны совместно использовать результаты друг друга — например, когда один процесс должен отрисовывать фреймы, полученные после детекции другими процессами, — возникает взаимная блокировка. Это происходит, потому что каждый процесс зависит от результатов работы других процессов, и если один из процессов задерживается или ожидает ресурс, это блокирует всю систему.

6.3.4 Асинхронные вычисления (Asyncio)

6.3.4.1.Теория

Асинхронные вычисления в Python, реализуемые через модуль asyncio, представляют собой мощный инструмент для организации конкурентных вычислений. Позволяет программе продолжать выполнение, в то время как она ожидает завершения операций ввода-вывода или других длительных процессов. Это увеличивает общую эффективность, поскольку процессор может выполнять другие задачи во время ожидания.

Async I/O, Asynchronous Input and Output — асинхронный ввод/вывод подразумевает выполнение вычислений во время ожидания результата от ввода и вывода данных.

При использовании модуля asyncio программа сама принимает решение о том, когда ей нужно переключиться между задачами. Каждая задача взаимодействует с другими задачами, передавая им управление тогда, когда она к этому готова. Поэтому такая схема работы называется **«кооперативной многозадачностью»** (cooperative multitasking), так как каждая задача должна взаимодействовать с другими, передавая им управление в момент, когда она уже не может сделать ничего полезного.

Сопрограммы — это специальные функции, помеченные ключевым словом async, которые могут быть приостановлены и возобновлены. Они позволяют использовать оператор await для ожидания результата другой сопрограммы или операции ввода-вывода, не блокируя при этом основной поток выполнения программы. Это позволяет программе эффективно переключаться между задачами, улучшая производительность и отзывчивость.

Цикл событий (event loop) управляет выполнением сопрограмм и обработкой I/O операций, позволяя задачам переходить из состояния ожидания в состояние готовности и наоборот. Имеется единственный **цикл событий**, который занимается управлением всеми задачами. Задачи могут пребывать в некотором количестве различных состояний, самыми важными из которых можно назвать состояние готовности (ready) и состояние ожидания (waiting). Цикл событий на каждой итерации проверяет, имеются ли задачи, пребывающие в состоянии ожидания, которые завершены и оказались в состоянии готовности. Затем цикл берёт задачу, находящуюся в состоянии готовности, и выполняет её до тех пор, пока она не завершится, либо — до тех пор, пока не окажется, что ей нужно дождаться завершения другой задачи.

Футуры (futures) же представляют собой объекты, которые обещают предоставить результат асинхронной операции в будущем.

В целом, асинхронное программирование в Python позволяет избежать блокировок и взаимоблокировок задач, увеличивая общую эффективность

использования процессора. Программа сама решает, когда переключаться между задачами, что делает код конкурентным и кооперативным. Асинхронный ввод/вывод позволяет выполнять вычисления во время ожидания результата от ввода и вывода данных, что особенно полезно при решении задач, зависящих от подсистемы ввода/вывода, и обеспечивает потокобезопасность.

6.3.4.2.Практика

В рамках исследования была внедрена асинхронная обработка с использованием asyncio, что способствовало оптимизации управления операциями ввода-вывода. Асинхронные методы применялись для параллельного распознавания объектов на изображениях из видеопотока, что достигалось благодаря использованию ключевых слов async и await, а также функции asyncio.gather, обеспечивающей одновременное выполнение множества асинхронных задач.

Асинхронные операции инициировались в функции run_three_models, где с помощью await asyncio.gather(...) запускались множественные экземпляры функции detect_objects, каждый из которых независимо обрабатывал кадр и выявлял объекты.

Использование await перед asyncio.gather указывает на приостановку выполнения текущей функции — в данном случае run_three_models — до завершения всех переданных в asyncio.gather асинхронных задач. В бесконечном цикле while True происходит непрерывное получение кадров и их асинхронная обработка, что позволяет обрабатывать изображения по мере поступления, минуя ожидание завершения предыдущих операций.

Применение асинхронности в процессе реального времени для распознавания объектов способствует увеличению производительности и эффективности, так как позволяет выполнять несколько задач одновременно, сокращая время ожидания окончания длительных операций ввода-вывода. Это критически важно для систем, требующих быстрой обработки большого объема данных, как в случае с видеопотоком. Благодаря этому достигается значительное повышение частоты

кадров (FPS), поскольку приложение может обрабатывать другие задачи, не ожидая окончания операций ввода-вывода.

6.4 Результаты исследования

В таблице 18 представлены результаты исследования, где сравниваются различные эффективные методы вычислений по их производительности (FPS) и процентному приросту по сравнению с базовой моделью. Как видно из таблицы, перевод модели на ГПУ (графический процессор) и использование асинхронных вычислений приводит к увеличению FPS и процентного прироста производительности на 38%.

Таблица 18. Результаты исследования

	Скорость обработки, FPS	Улучшение производительности, %
Исходная модель кода	10.585795500815214	0
Модель, оптимизированная для ГПУ	10.97882685224355	3,712818289
Модель с асинхронными вычислениями	14.180416196292715	33,95702000
Модель, оптимизированная для ГПУ с асинхронными вычислениями	14.628262387698413	38,18765332

6.5 Заключение

Исследование было направлено на повышение количества кадров в секунду (FPS) в приложении для контроля ручных операций на производстве. Целью было улучшение FPS за счет оптимизации вычислений, включая более эффективное распределение задач между графическим (GPU) и центральным (CPU) процессорами. Высокий FPS необходим для точного и быстрого отслеживания операций в реальном времени, что способствует улучшению качества контроля и безопасности трудовых процессов.

В ходе исследования вычислительная часть приложения была переведена на

GPU с использованием библиотеки CUDA. Сравнение производительности различных моделей, выполняемых на CPU и двух GPU (NVIDIA GeForce RTX 3080 и NVIDIA GeForce GTX 1660), показало, что наилучший прирост FPS достигается при выполнении модели Palm Model на GPU NVIDIA GeForce RTX 3080, затем Drone Tools Model на GTX 1660 и Drone Details Model снова на RTX 3080, что дало прирост на 3,71%. Однако этот прирост оказался незначительным, что может быть связано с несоответствием вычислительных задач архитектуре GPU, особенно если задачи содержат множество последовательных операций и зависимостей.

Также были созданы три отдельных потока для обработки трех разных моделей: Palm Model, Drone Tools Model и Drone Details Model. Каждая модель обрабатывалась в отдельном потоке для параллелизации вычислений. Однако ограничение GIL в Python, которое позволяет выполнять байт-код только в одном потоке за раз, не позволяет реализовать параллелизм.

Были проведены эксперименты с многопроцессностью. В первом эксперименте один процесс выполнял все модели, а второй занимался захватом изображения и отрисовкой, но это не увеличило скорость работы. Во втором эксперименте задачи были распределены между тремя процессами, каждый из которых выполнял одну модель, в то время как основной процесс занимался захватом и отрисовкой. Однако, такая схема приводила к проблемам взаимной блокировки, поскольку каждый процесс зависел от результатов, обработанных другими процессами.

Применение асинхронной обработки через asyncio значительно повысило эффективность работы. Асинхронные методы включали запуск нескольких задач по обнаружению объектов, каждая из которых независимо анализировала кадры. Благодаря асинхронности, система могла обрабатывать новые изображения параллельно, не дожидаясь окончания предыдущих задач.

В результате, использование асинхронных вычислений и перевод вычислений на графический процессор привело к увеличению FPS и процентного прироста

производительности на 38%. Это подтверждает, что асинхронная обработка является наиболее эффективным методом для приложений, требующих быстрой обработки данных в реальном времени, и позволяет достичь высокого FPS, улучшая качество контроля и безопасность производственных процессов.

7 Съемка и разметка обучающих наборов данных

Задача заключалась в создании и разметке фотографий для формирования датасетов, которые будут использоваться для обучения алгоритмов машинного обучения. Выполнялись следующие этапы работы:

1) Съемка фотографий различных деталей.
2) Ручная разметка фотографий с использованием платформы makesense.ai.
3) Создание разнообразных наборов данных для обеспечения высокого качества обучения модели.

Разметка данных является фундаментальной частью процесса разработки моделей машинного обучения. Без качественных и разнообразных данных любая модель будет недостаточно точной и надежной. Работа по созданию и разметке фотографий деталей гарантирует, что система обучается на данных, которые охватывают все возможные сценарии и вариации, способствуя тому, что система будет эффективно работать в реальных условиях, где возможны различные непредвиденные ситуации. Это включает в себя:

1) Съемку деталей в различных условиях и с использованием разных камер (Basler и Logitech).
2) Разметку данных для разных типов деталей и их комбинаций.
3) Создание датасетов, которые включают перекрытия и сложные фоны.

В рамках проекта для съемки использовались две разные камеры: Basler и Logitech. Каждая из этих камер имеет свои уникальные преимущества, что позволило создать более разнообразные датасеты.

Камера Basler обладает высоким разрешением (1920x1080), что обеспечивает исключительное качество и детализацию изображений. Использование Logitech

было также необходимо для захвата изображения деталей с различных ракурсов и освещения. Использование двух камер позволяет компенсировать возможные ограничения одной камеры за счет преимуществ другой.

Платформа makesense.ai стала идеальным выбором для разметки данных из-за своих преимуществ, таких как удобство использования, поддержка различных форматов данных, возможности для совместной работы и высокая производительность.

7.1 Описание выполненных этапов работы

7.1.1 Разметка фотографий различных винтов и гаек

Этот процесс включает разметку нескольких типов винтов и гаек, каждый из которых имеет свои особенности и характеристики: M3_cup_screw, M3_button, M3_flat_screw_black, M3_flat_screw, M3_screw_gasket, M3_insert_nut, M3_ M2_flat_screw, M2_locknut. Всего различных деталей 9.

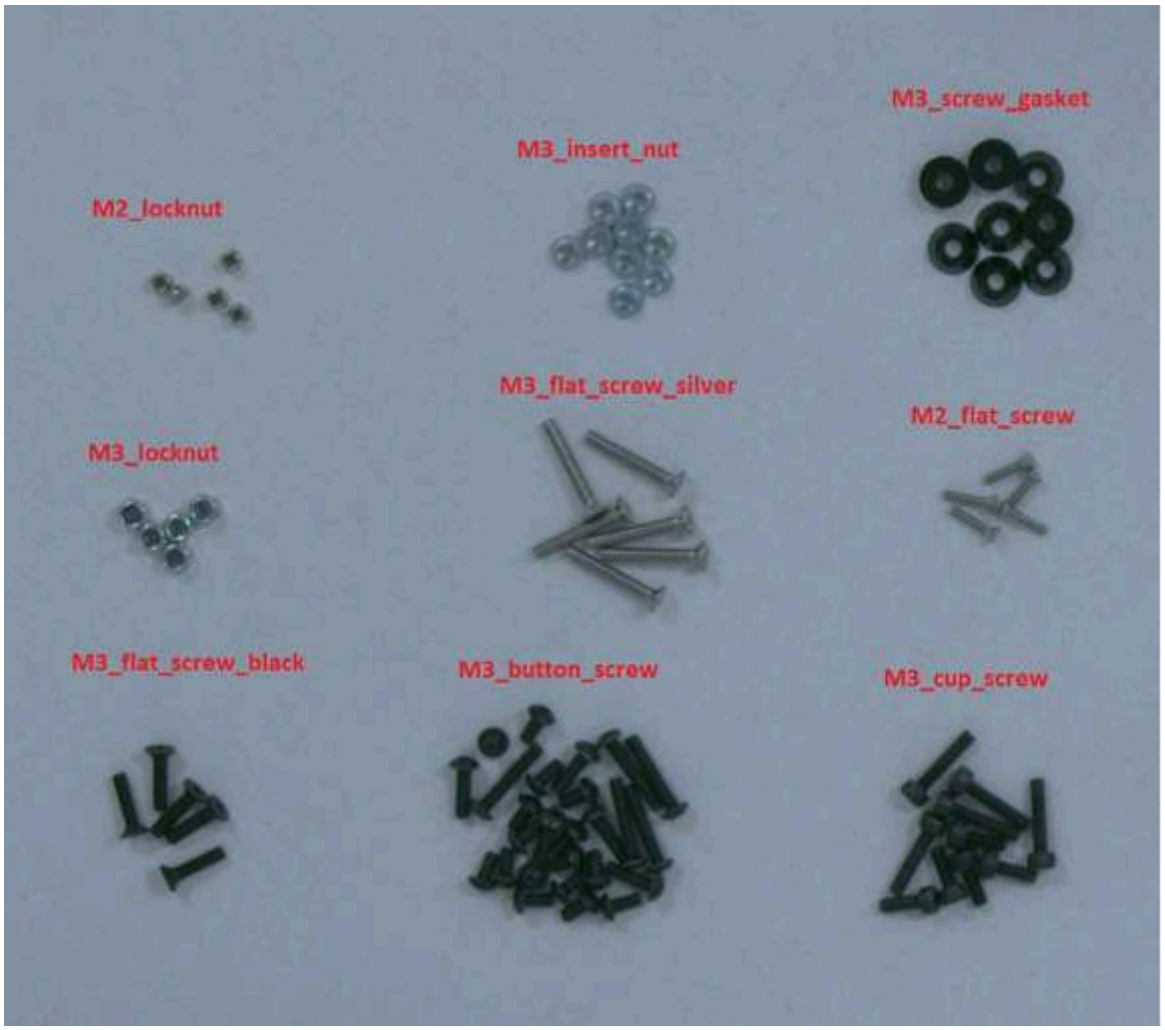

Рис. 25. Винты и гайки для разметки новых наборов данных

Было сделано по 10 фотографий каждого вида винтов и гаек (по 5 шт. деталей одного класса на фото), и 20 фотографий всех видов винтов и гаек вместе (по 5 шт. одно класса на фото, всего деталей на фото было 45 шт.). В сумме 110 фотографий.

Рис. 26. Пример разметки в программе makesence.ai

Разметка такого большого количества объектов на одной фотографии требует значительного внимания. Это сложный процесс, так как каждый объект должен быть точно идентифицирован и размечен. Процесс разметки включал следующие этапы:

1) Импорт изображений: все фотографии, снятые на камеры, были загружены на платформу makesense.ai.

2) Создание разметок: для каждой фотографии вручную создавались разметки, обозначающие положения и границы каждой детали.

3) Проверка и корректировка: после создания разметок фотографии проверялись на точность, и при необходимости вносились корректировки для обеспечения высокого качества данных.

Такие групповые фотографии с различными типами винтов и гаек помогают модели научиться различать и классифицировать объекты в более сложных сценариях, способствуя повышению точности.

7.1.2 Съёмка и разметка фотографий детали nut_bolt_M5_CW

Деталь nut_bolt_M5_CW является важной частью конструкции дрона. Она

представляет собой небольшую гайку с болтом, используемую для крепления различных компонентов. Несмотря на свои компактные размеры, эта деталь играет критическую роль в надежности и устойчивости дрона. Были использованы две камеры: Basler и Logitech. Во время съемки было сделано 300 фотографий, которые включали в себя:

1) 70 шт. фотографий на Basler и 30 шт. фотографий на Logitech детали в пинцете;

2) 70 шт. фотографий на Basler и 30 шт. фотографий на Logitech деталей в группах.

3) 70 шт. фотографий на Basler и 30 шт. фотографий на Logitech деталей на корпусе дрона.

В производственном процессе детали часто перемещаются с помощью инструментов, таких как пинцеты. Фотографирование в пинцете моделирует этот сценарий, что помогает системе обучаться распознавать детали в процессе сборки.

Снимки деталей в группах помогают модели обучаться различать отдельные объекты даже в случае, когда они находятся близко друг к другу или частично перекрываются.

Снимки деталей на корпусе дрона включают более сложные фоны и окружение, что помогает алгоритму различать детали даже в условиях, когда они находятся рядом с другими компонентами. Процесс разметки включал следующие этапы:

1) Импорт изображений: все фотографии, снятые на камеры, были загружены на платформу makesense.ai.

2) Создание разметок: для каждой фотографии вручную создавались разметки, обозначающие положения и границы детали nut_bolt_M5_CW, а также разметки других попадающих в кадр деталей дрона.

3) Проверка и корректировка: после создания разметок фотографии проверялись на точность, и при необходимости вносились корректировки для обеспечения высокого качества данных.

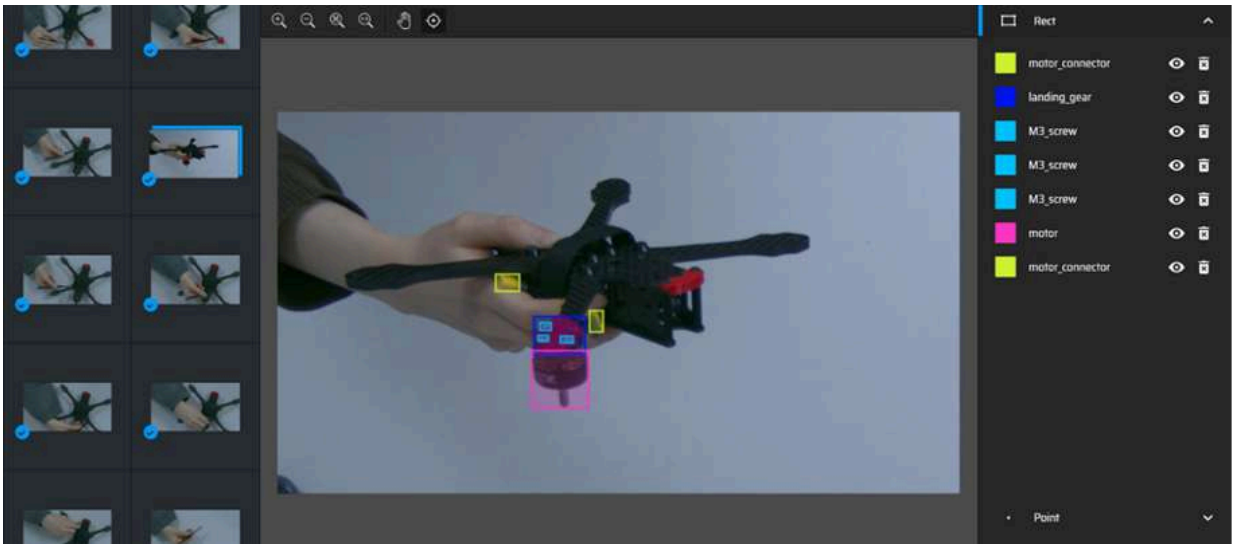

Рис. 27. Пример размеченной фотографии дрона в программе makesence.ai

7.1.3 Съёмка и разметка фотографий деталей рамы

Для съёмки и разметки были выбраны следующие детали рамы: side_plate, arms_stopper, top_plate, rack, bottom_plate, arms_plate.

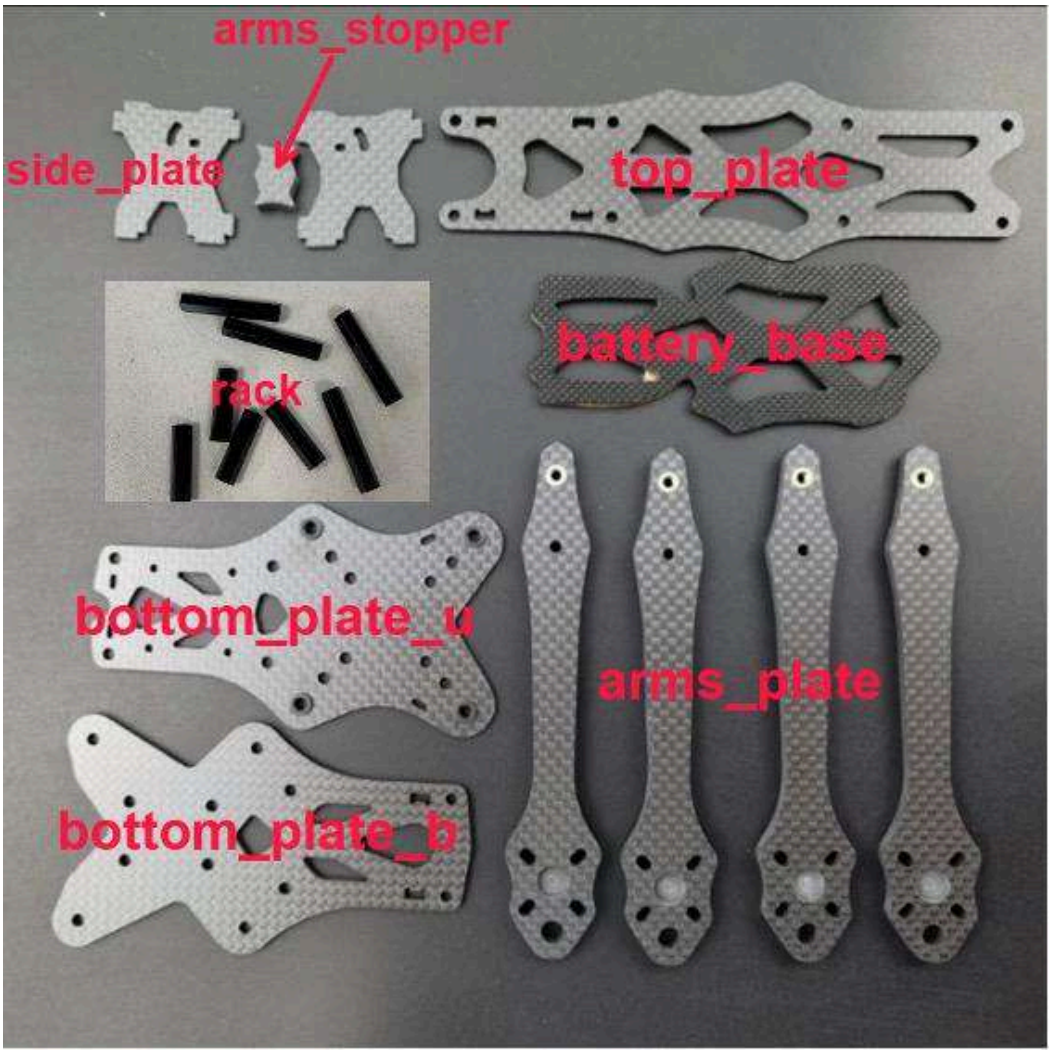

Рис. 28. Детали рамы дрона для разметки новых наборов данных

В общей сложности было сделано 240 фотографий, которые включали в себя:

1) Фотографии отдельных классов деталей: для каждого класса было сделано по 20 фотографий (120 фотографий).
2) Фотографии со всеми классами деталей на фото: было сделано 50 фотографий, включающих все классы деталей на разных фонах.
3) Фотографии с различными комбинациями классов деталей: было сделано 50 фотографий с различными комбинациями классов деталей.
4) Фотографии с перекрытиями деталей: было сделано 20 фотографий, на которых одна деталь наложена на другую.

Данная съемка аналогично размечалась вручную на той же платформе.

7.1.4 Съемка и разметка дополнительных наборов данных инструментов для сборки дрона

Разметка инструментов для сборки дрона позволяет системе не только распознавать детали, но и идентифицировать используемые инструменты, что необходимо для полного контроля процесса сборки. В съемке участвовали следующие объекты:

- electric_screwdriver (электрическая отвертка)
- hex_screwdriver (шестигранная отвертка)
- motors_clamp (зажим для моторов)
- pilers (плоскогубцы)
- tweezers (пинцет)
- universal_key (универсальный ключ)

Всего с инструментами было проведено три съёмки. Итоговое количество снимков инструментов составляет 520 фотографий. Аналогично была проведена разметка.

7.1.5 Съемка и разметка дополнительных наборов данных для промежуточных этапов сборки дрона

Съемка и разметка дополнительных наборов данных для промежуточных этапов сборки дрона включала создание и разметку фотографий различных деталей мотора. Для данного датасета были размечены следующие объекты:

- M3_screw
- landing_gear
- motor
- nut_bolt_M5_CW
- propeller
- motor_connector
- connection

Всего, за пять съемок, было размечено 646 фотографий. Датасет постепенно дорабатывался, добавлялись фотографии проблемных и сложных случаев для улучшения качества данных.

8 Обучение моделей детекции и классификации объектов

В нашей работе мы использовали различные наборы данных:

- детали мясорубки
- этапы сборки мясорубки
- изображения рук людей
- детали дрона
- этапы сборки дрона
- инструменты для сборки дрона
- винты для сборки дрона

На части из этих наборов данных ранее была обучена модель YOLOv5. Так как новая модель YOLOv8, согласно отчету разработчиков, превосходит результаты работы модели YOLOv5, и менее требовательна к вычислительным ресурсам, было принято решение обучить на всех перечисленных наборах данных более современную модель.

Обучение моделей проводилось с использованием ресурсов суперкомпьютерного кластера НИУ ВШЭ. Для того, чтобы получить устойчивые к различным условиям окружающей среды (освещение, расположение камер, и другие) модели, мы применяли различные аугментации изображений, среди

которых: афинные преобразования, наложение случайного шума, преобразования цветовой палитры и гистограммы освещенности, преобразование мозаики (создание одного изображения из нескольких, накладывающихся друг на друга).

8.1 Детали мясорубки

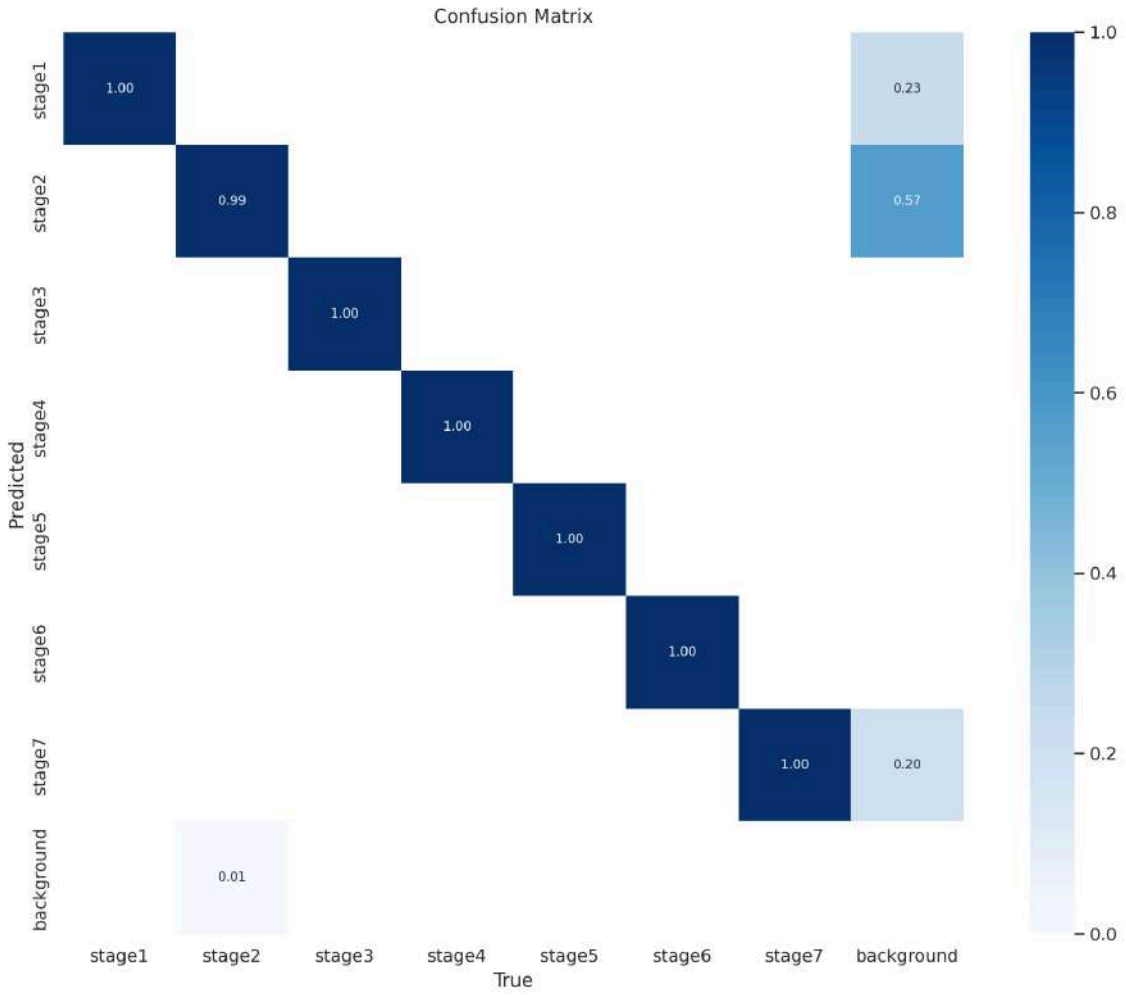

Рис.29. Матрица ошибок обученной модели YOLOv8 на наборе данных с деталями мясорубки

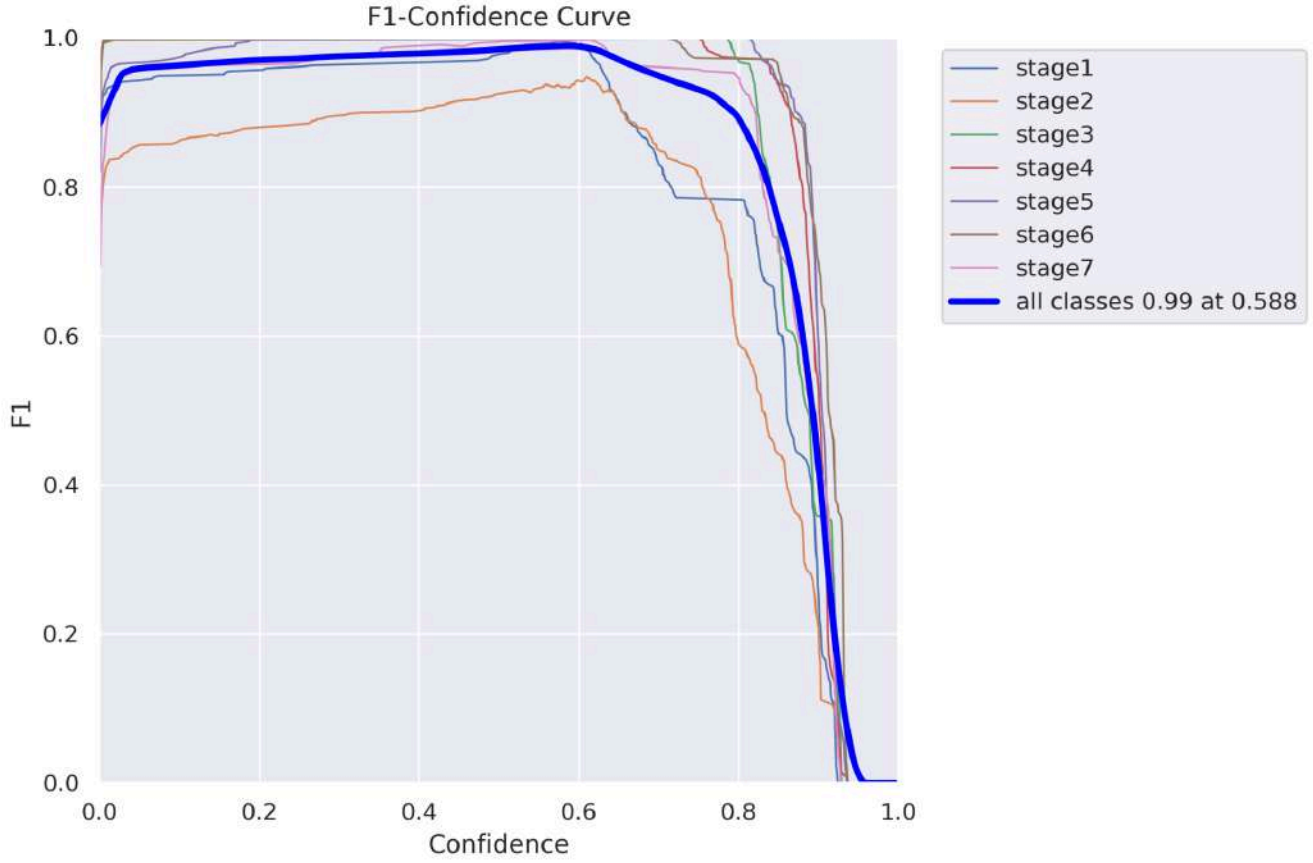

Рис. 30. Кривые F1-Confidence обученной модели YOLOv8 на наборе данных с деталями мясорубки

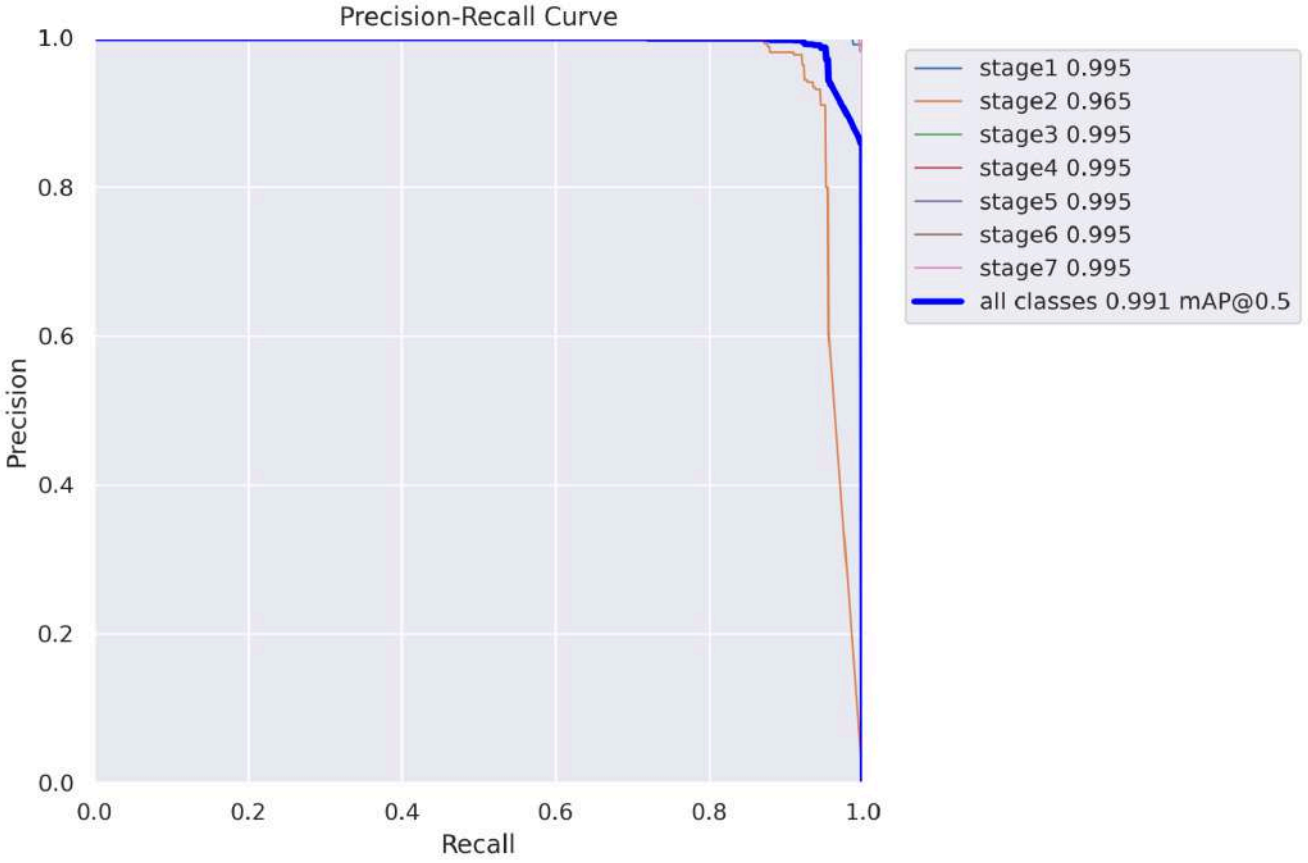

Рис. 31. Кривые Precision-Recall обученной модели YOLOv8 на наборе данных с деталями мясорубки

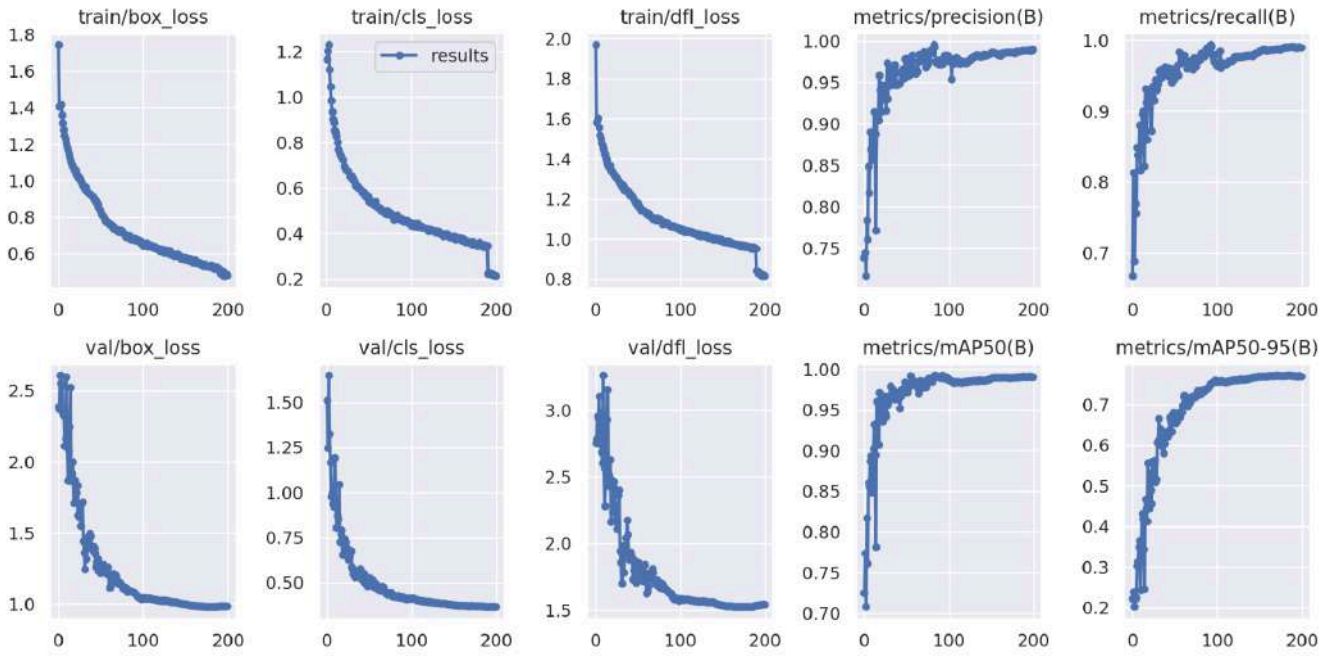

Рис. 32. Графики процесса обучения модели YOLOv8 на наборе данных с деталями мясорубки

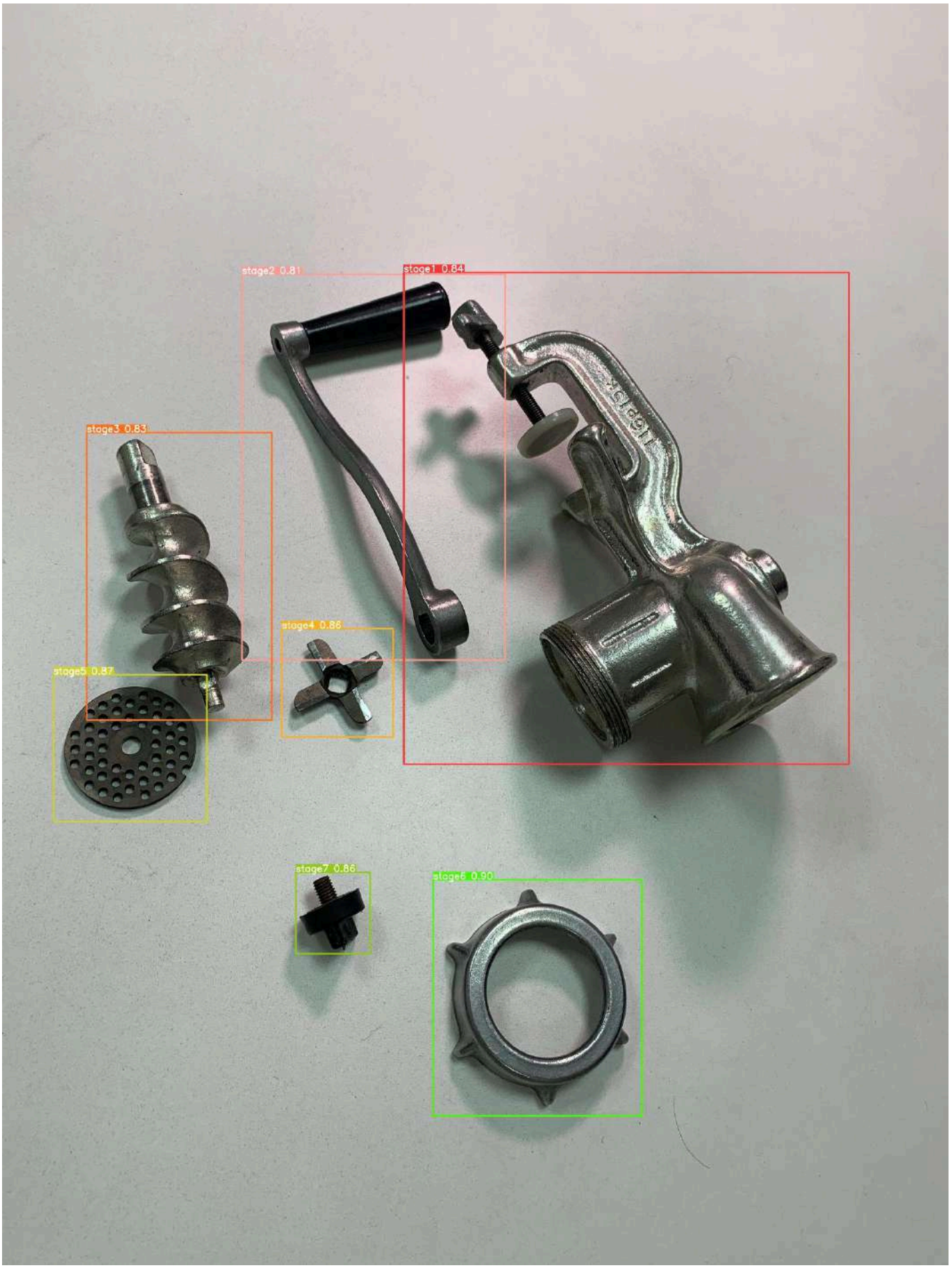

Рис. 33. Пример работы модели YOLOv8, обученной на наборе данных с деталями мясорубки

8.2 Этапы сборки мясорубки

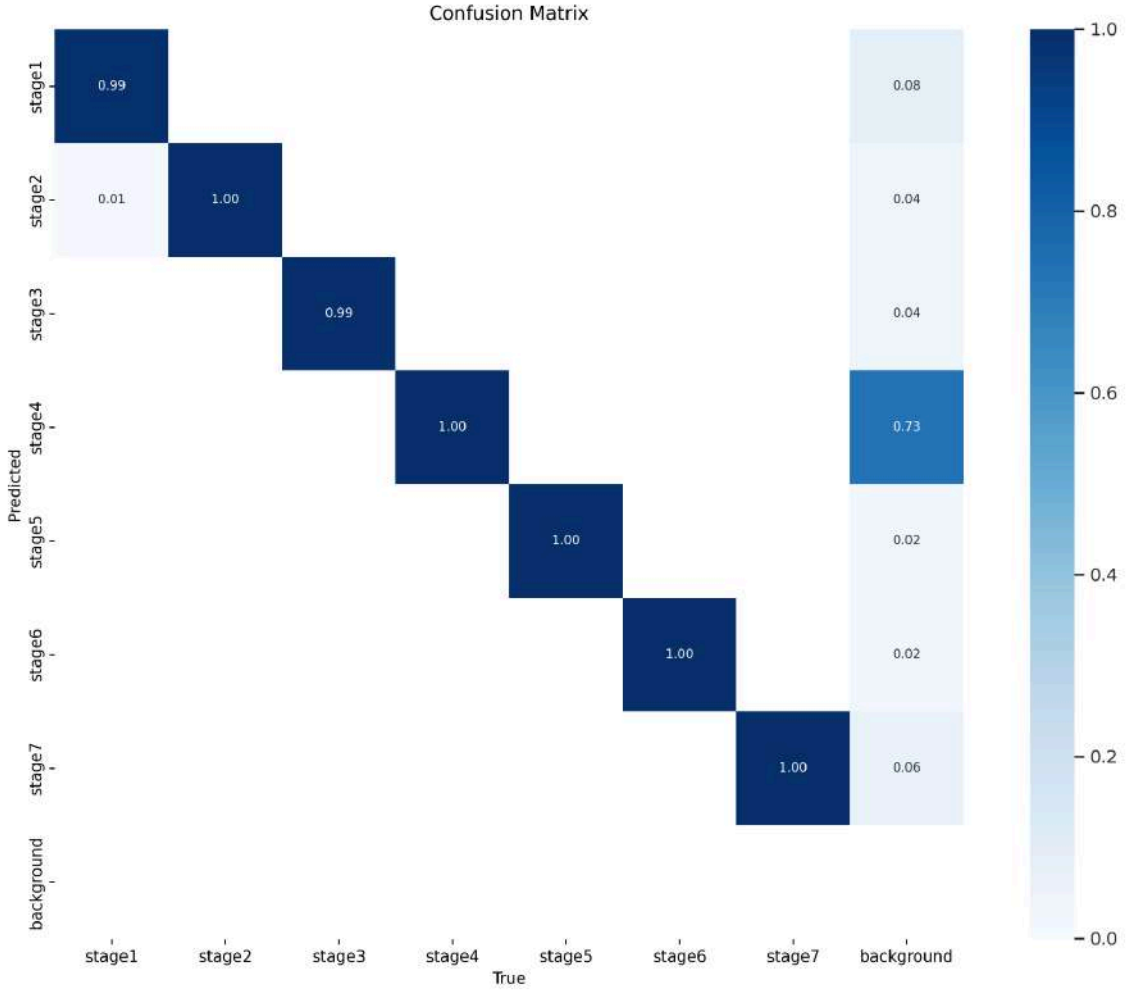

Рис. 34. Матрица ошибок обученной модели YOLOv8 на наборе данных с этапами сборки мясорубки

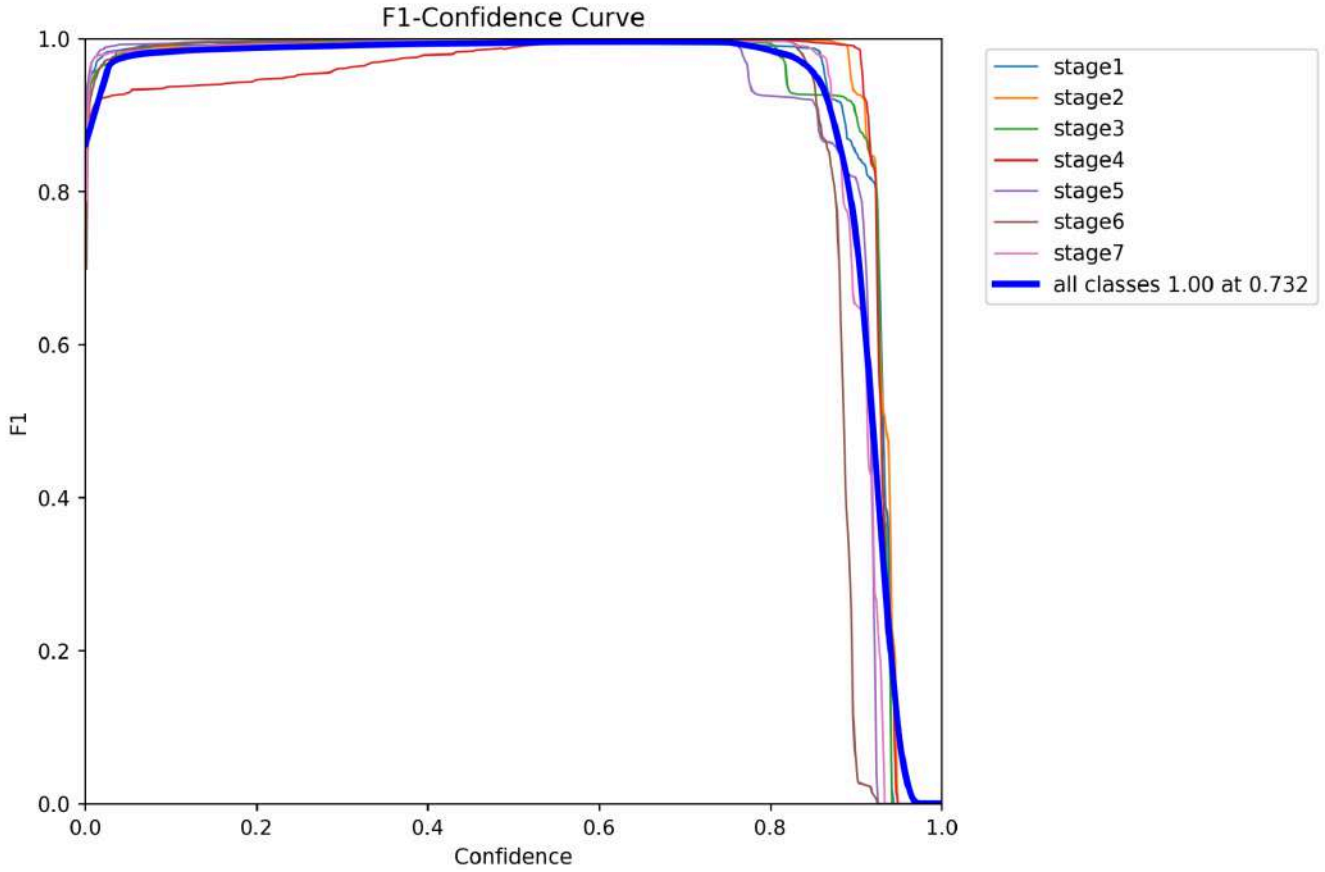

Рис. 35. Кривые F1-Confidence обученной модели YOLOv8 на наборе данных с этапами сборки мясорубки

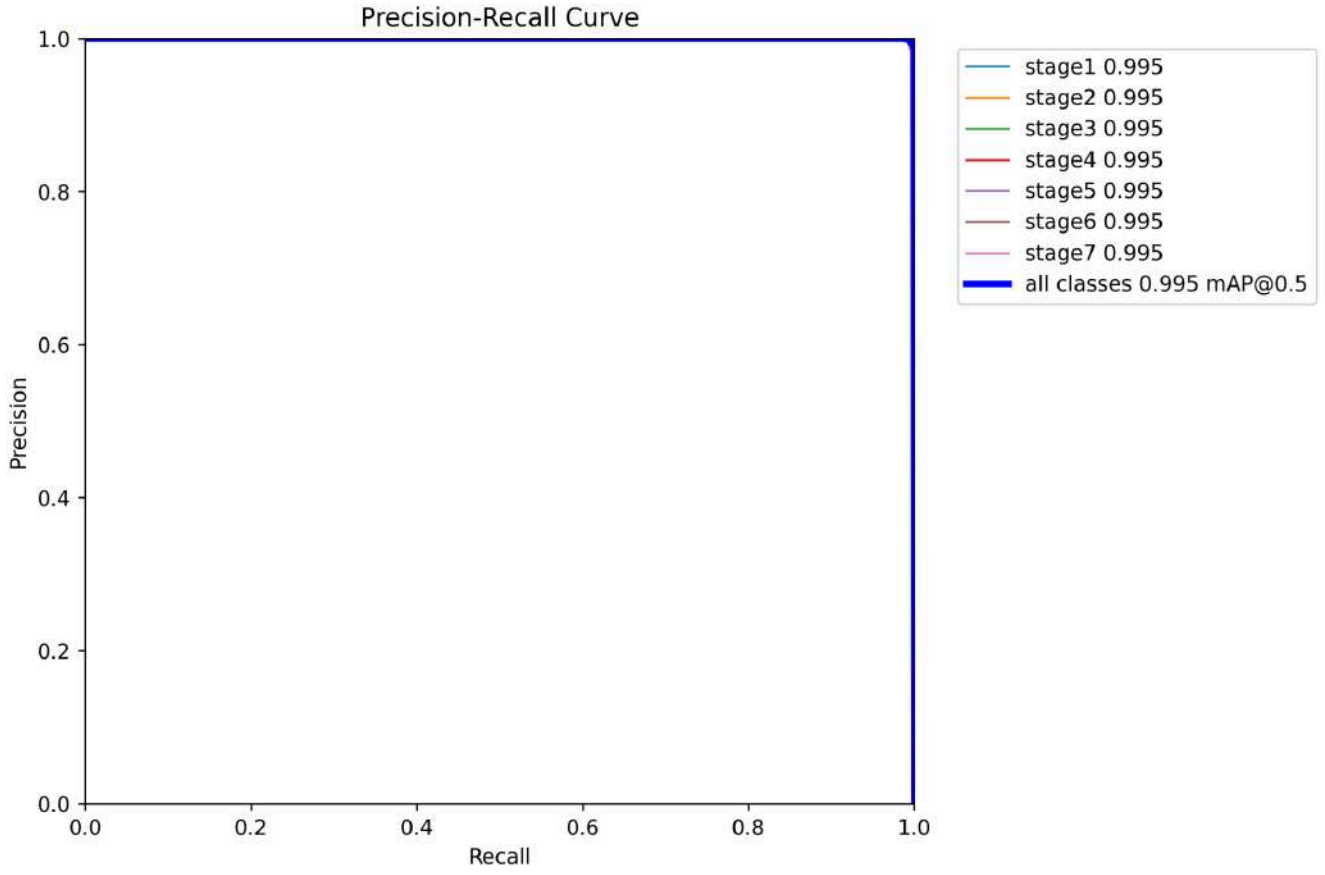

Рис. 36. Кривые Precision-Recall обученной модели YOLOv8 на наборе данных с этапами сборки мясорубки

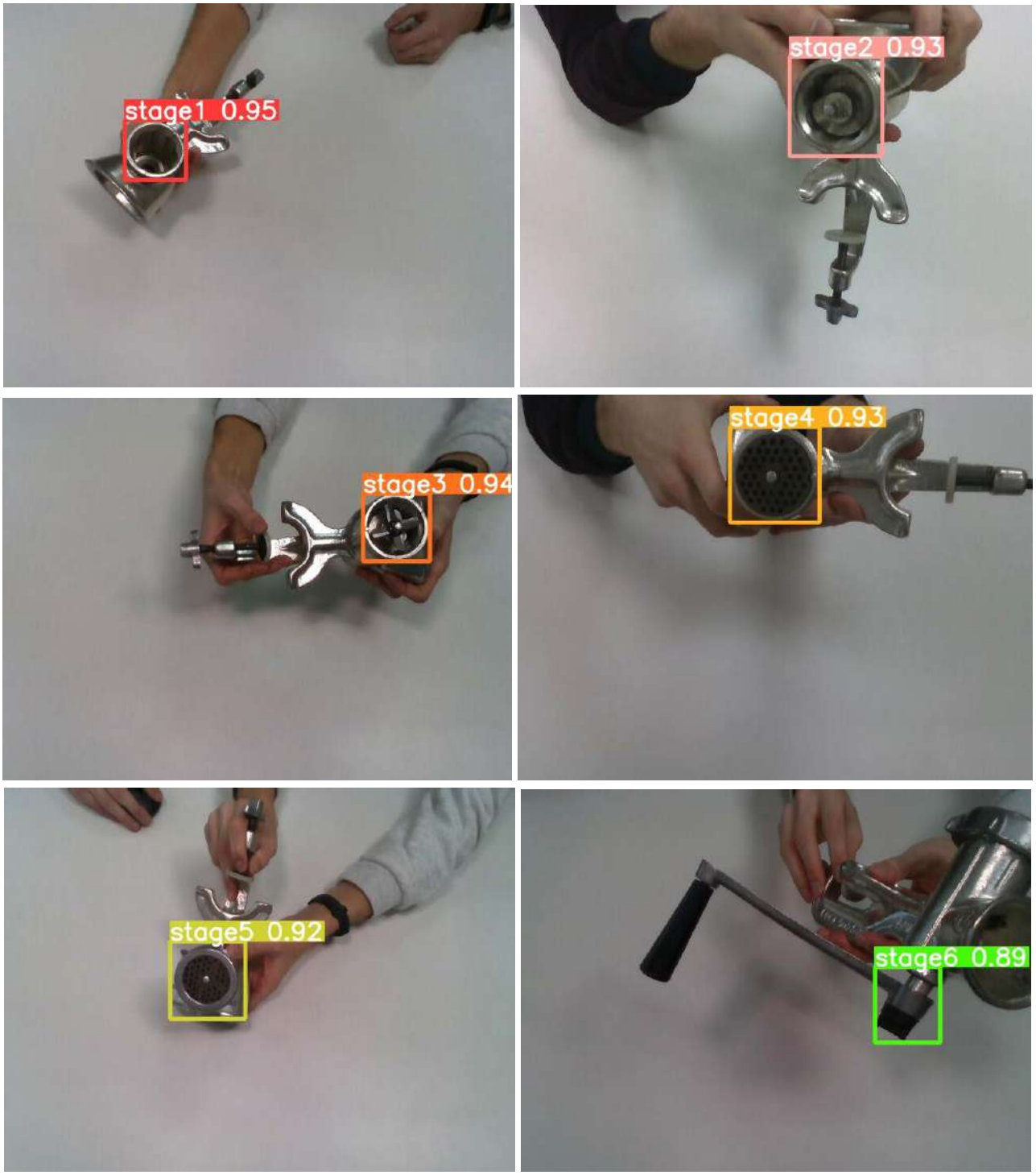

Рис. 37. Пример работы модели YOLOv8, обученной на наборе данных с этапами сборки мясорубки

8.3 Изображения рук людей

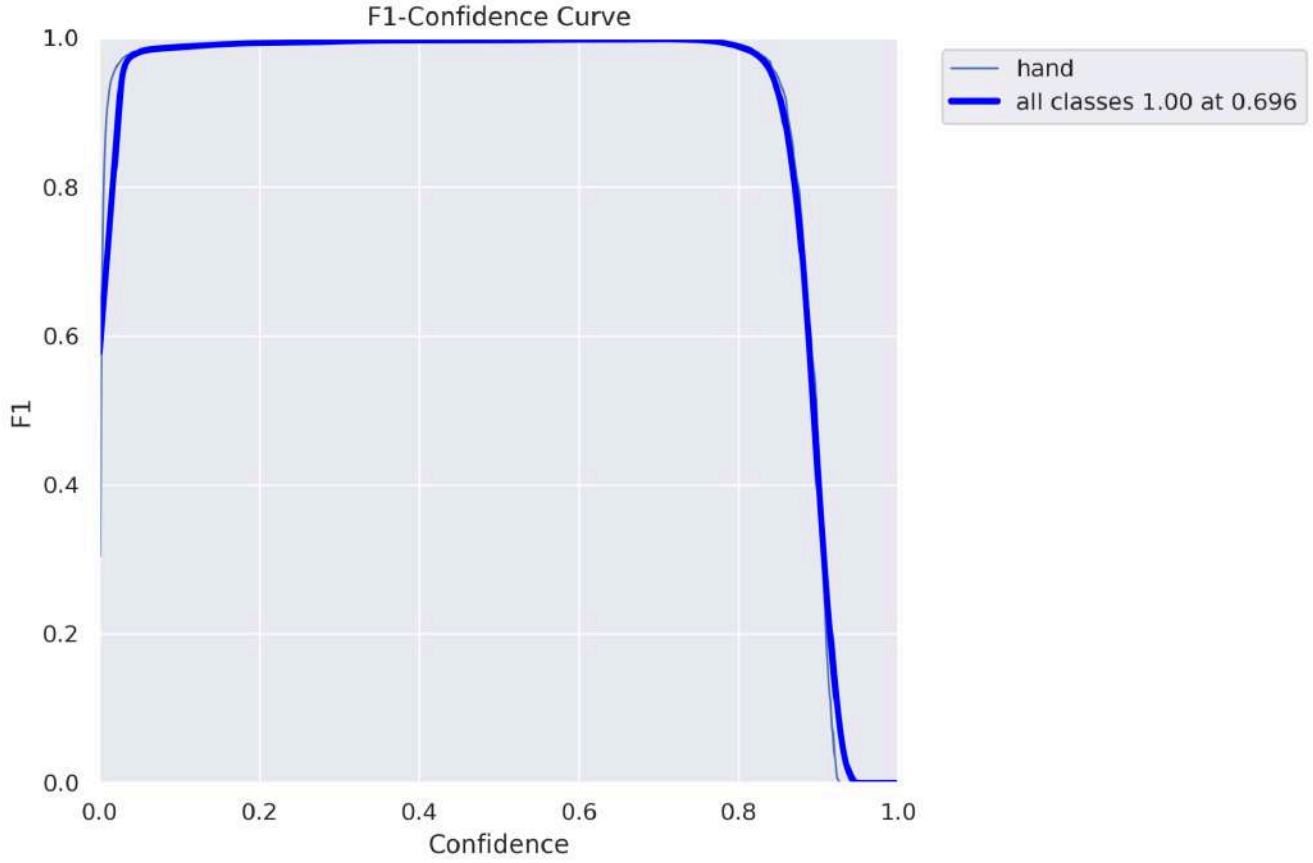

Рис. 38. Кривая F1-Confidence обученной модели YOLOv8 на наборе данных с руками людей

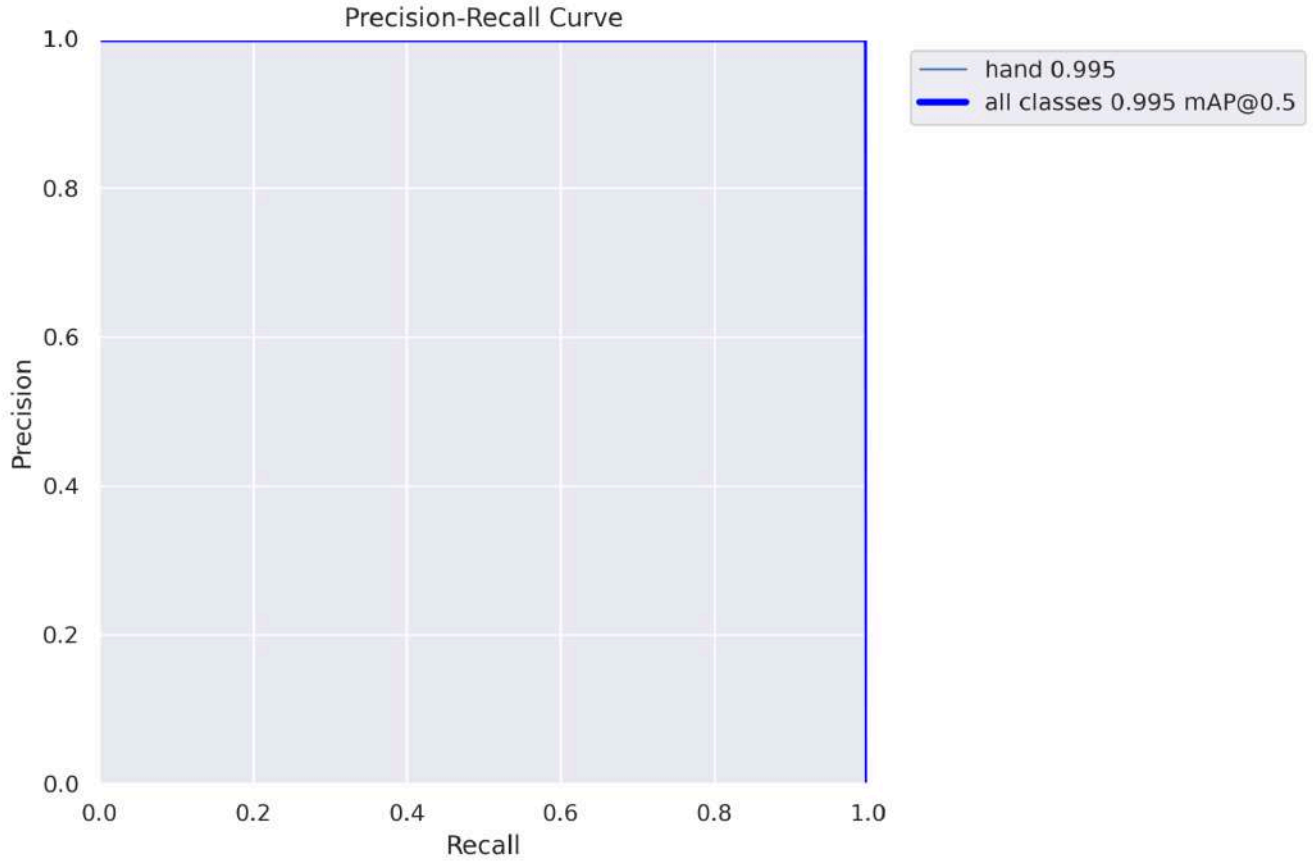

Рис. 39. Кривая Precision-Recall обученной модели YOLOv8 на наборе данных с руками людей

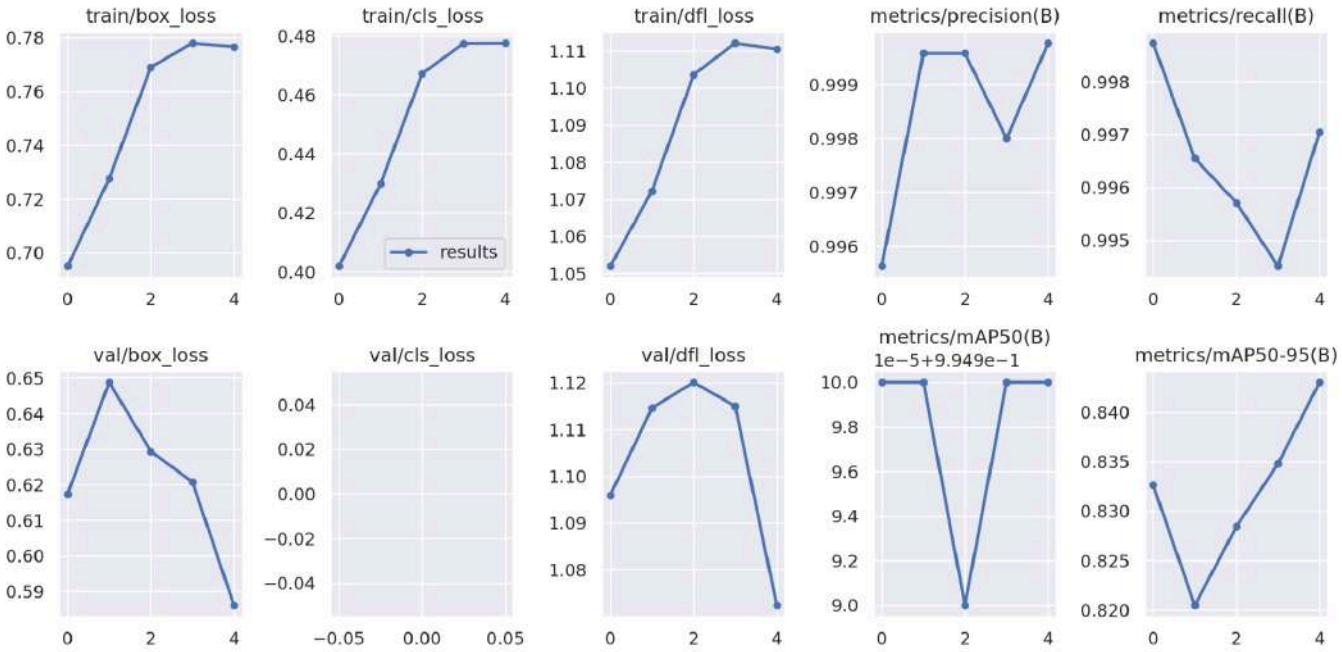

Рис. 40. Графики процесса обучения модели YOLOv8 на наборе данных с руками людей

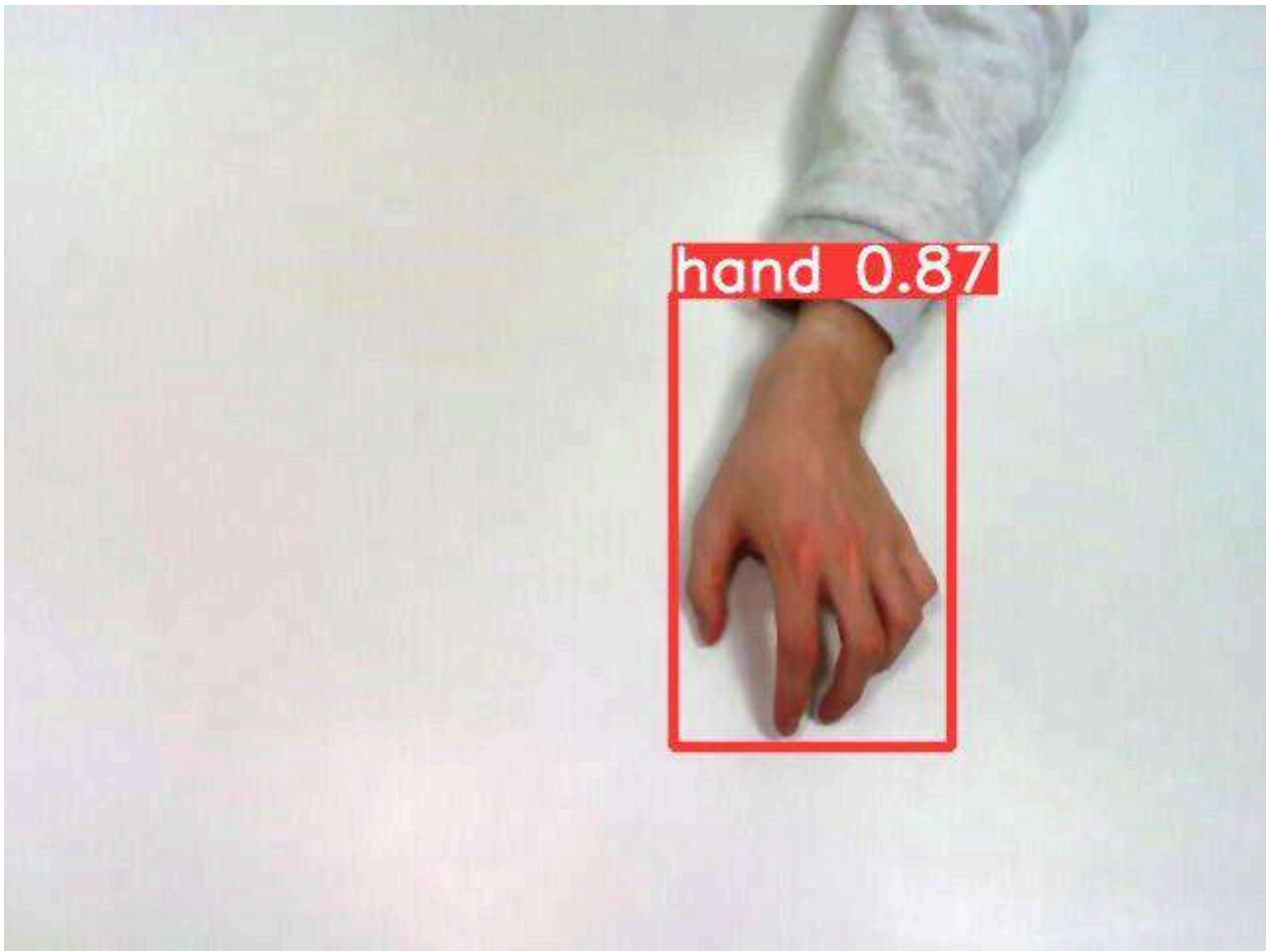

Рис. 41. Пример работы модели YOLOv8, обученной на наборе данных с руками людей

8.4 Детали дрона

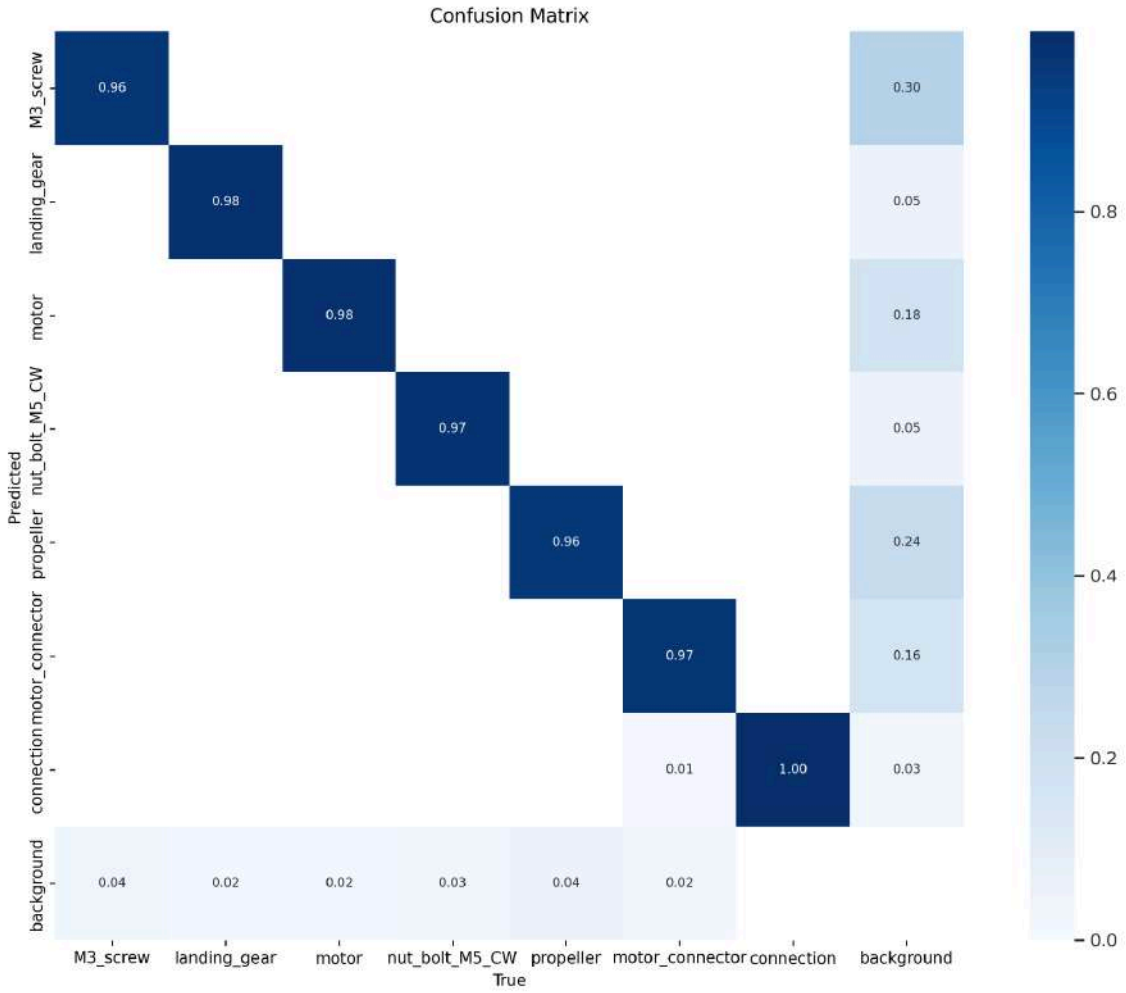

Рис. 42. Матрица ошибок обученной модели YOLOv8 на наборе данных с деталями дрона

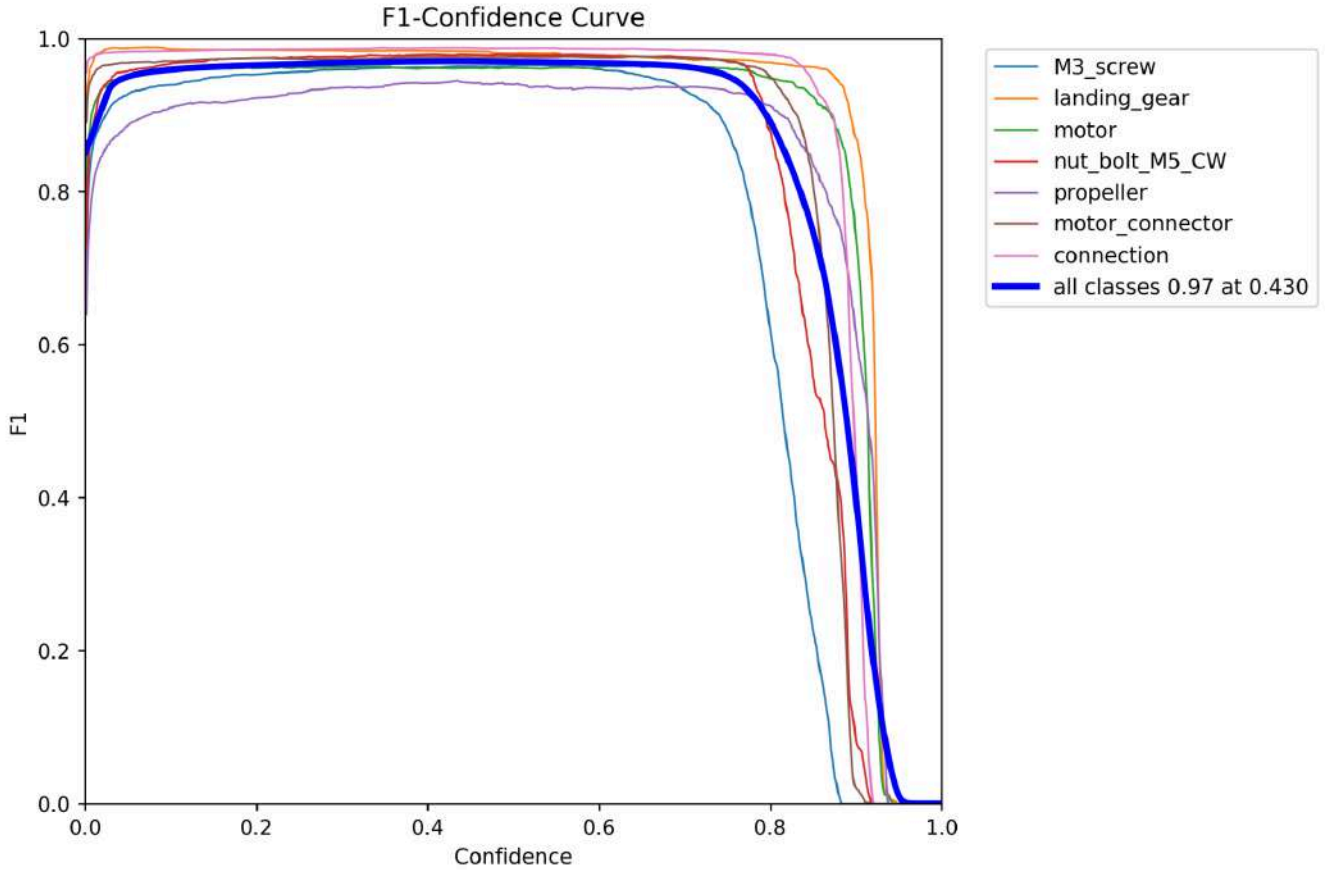

Рис. 43. Кривые F1-Confidence обученной модели YOLOv8 на наборе данных с деталями дрона

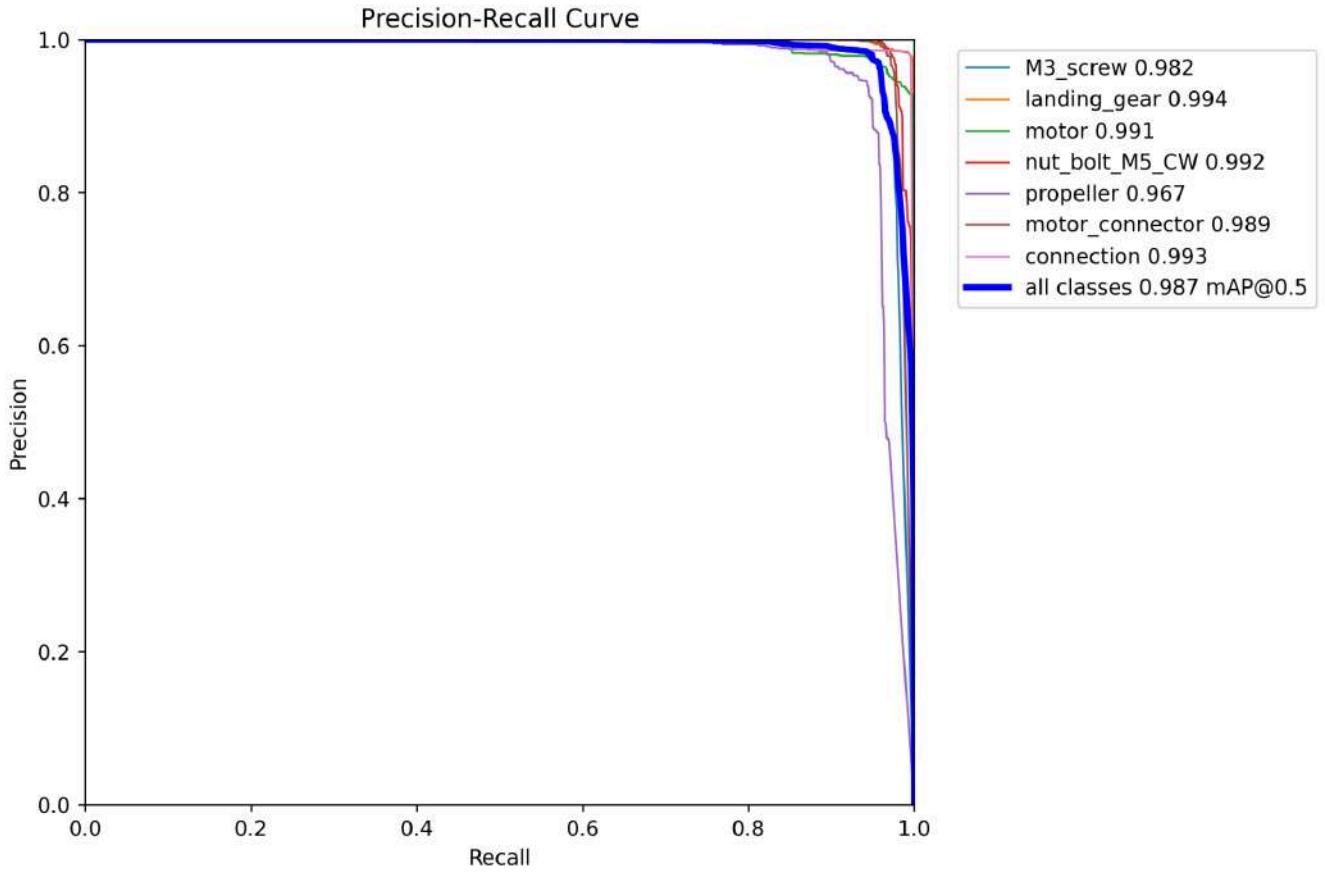

Рис. 44. Кривые Precision-Recall обученной модели YOLOv8 на наборе данных с деталями дрона

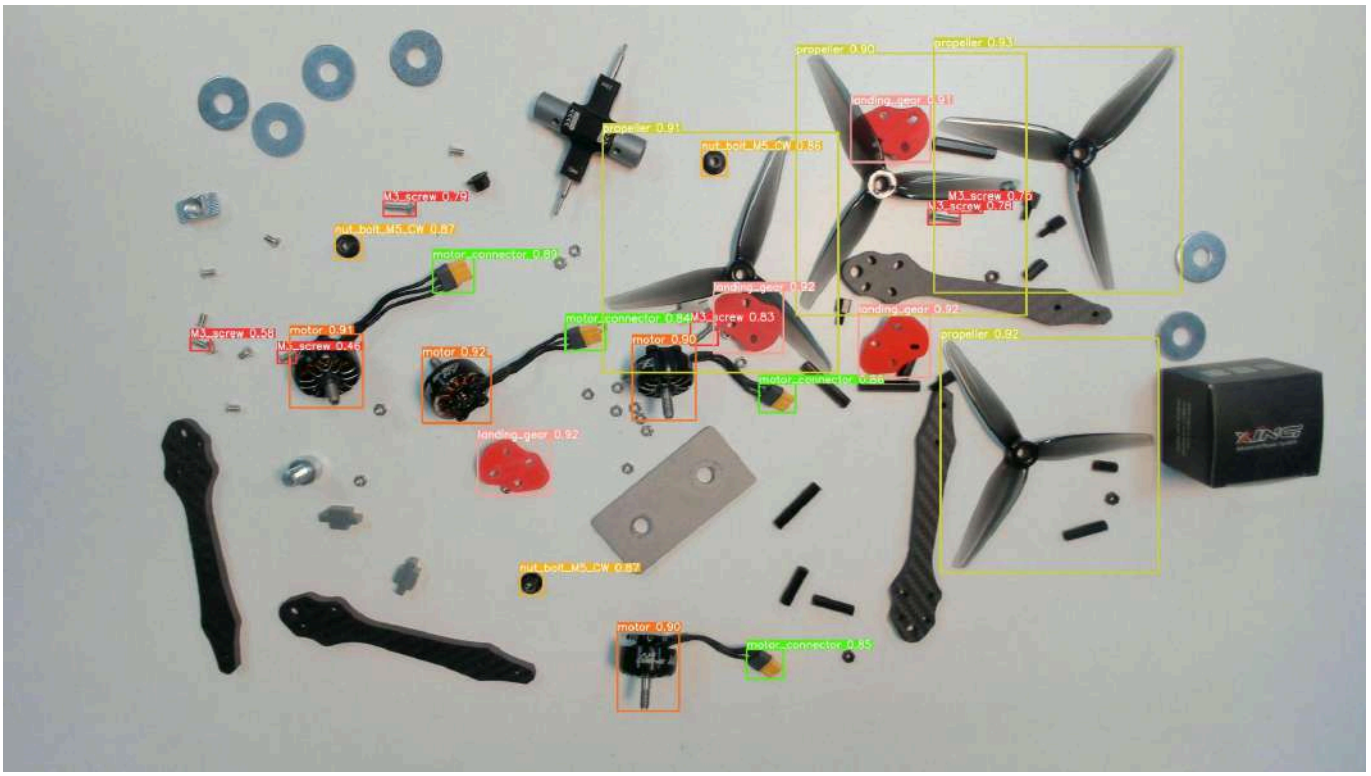

Рис. 45. Пример работы модели YOLOv8, обученной на наборе данных с деталями дрона

8.5 Этапы сборки дрона

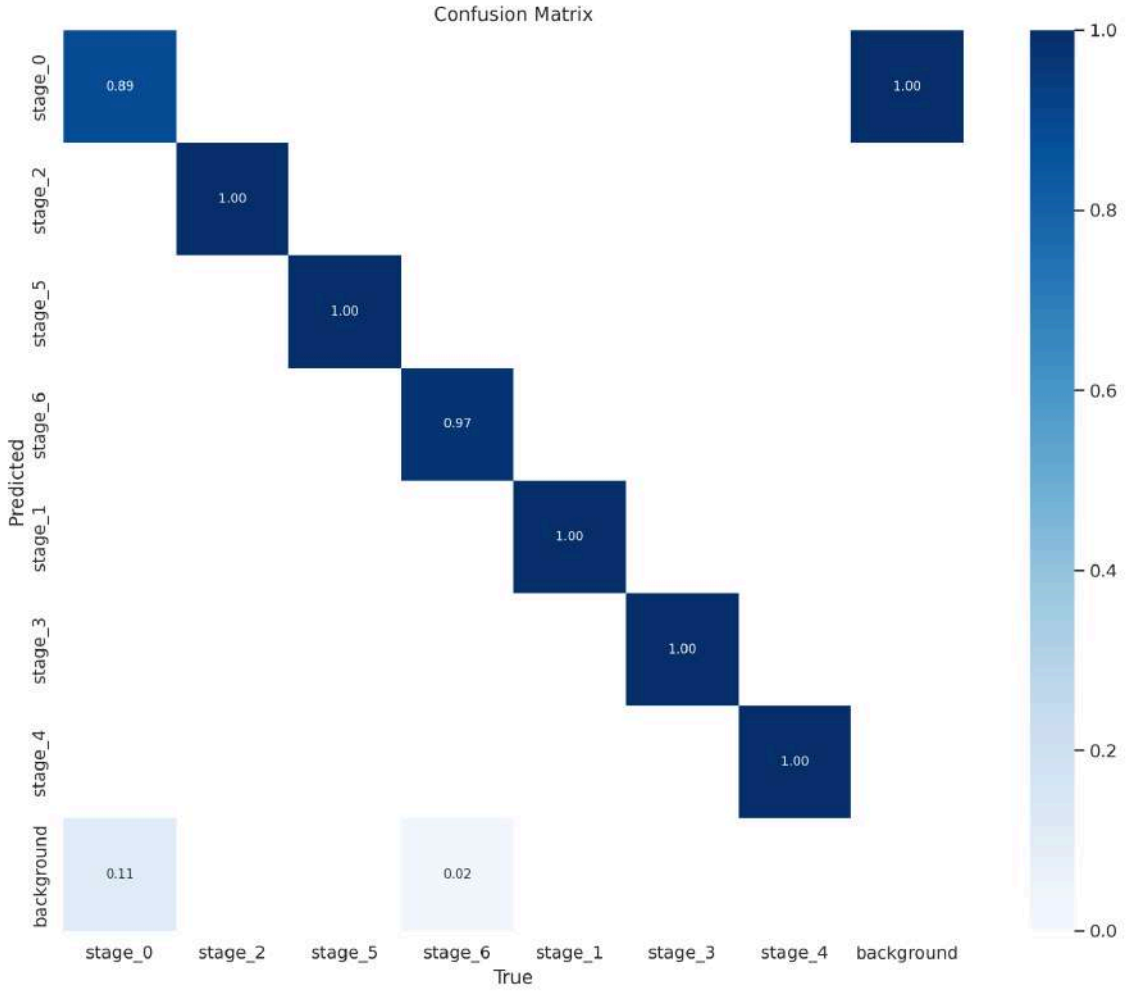

Рис. 46. Матрица ошибок обученной модели YOLOv8 на наборе данных с этапами сборки дрона

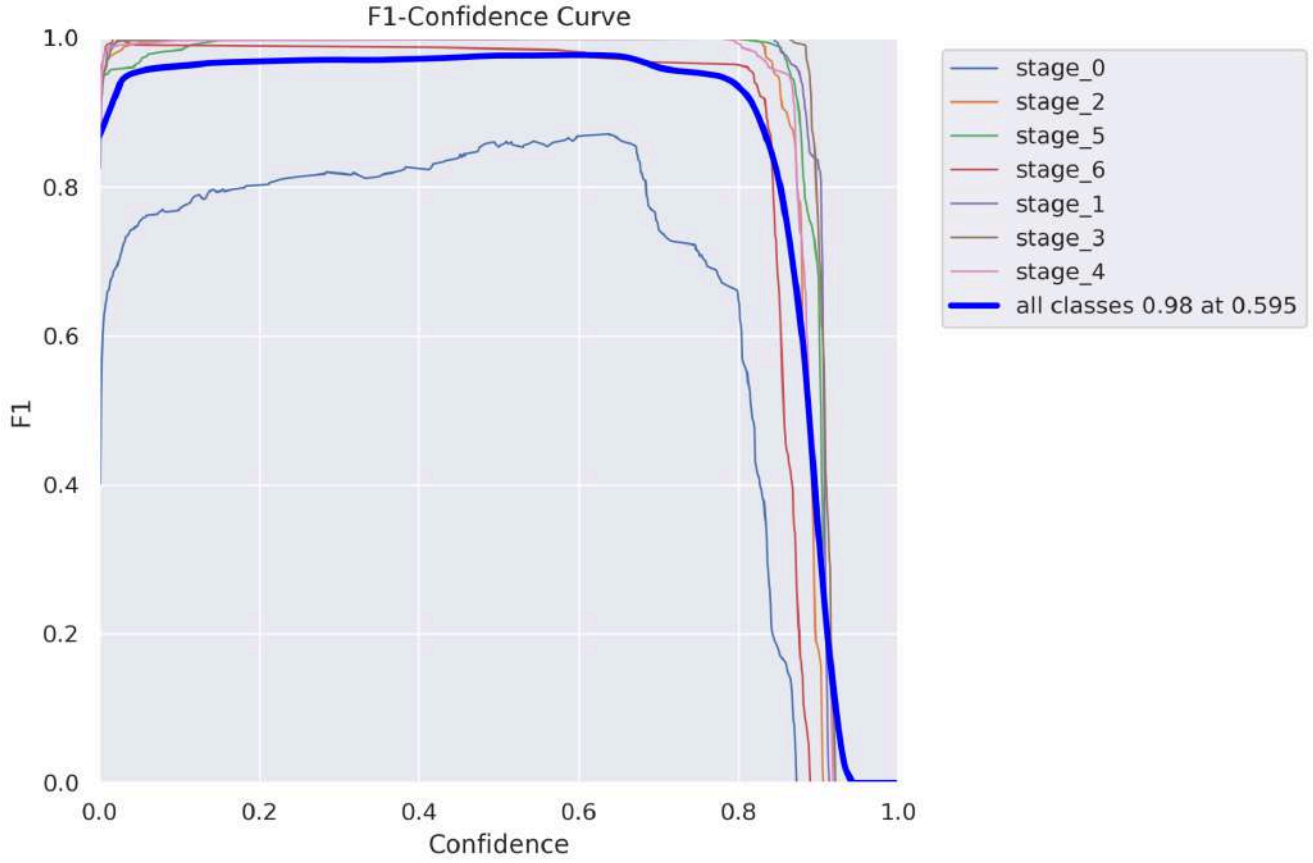

Рис. 47. Кривые F1-Confidence обученной модели YOLOv8 на наборе данных с этапами сборки дрона

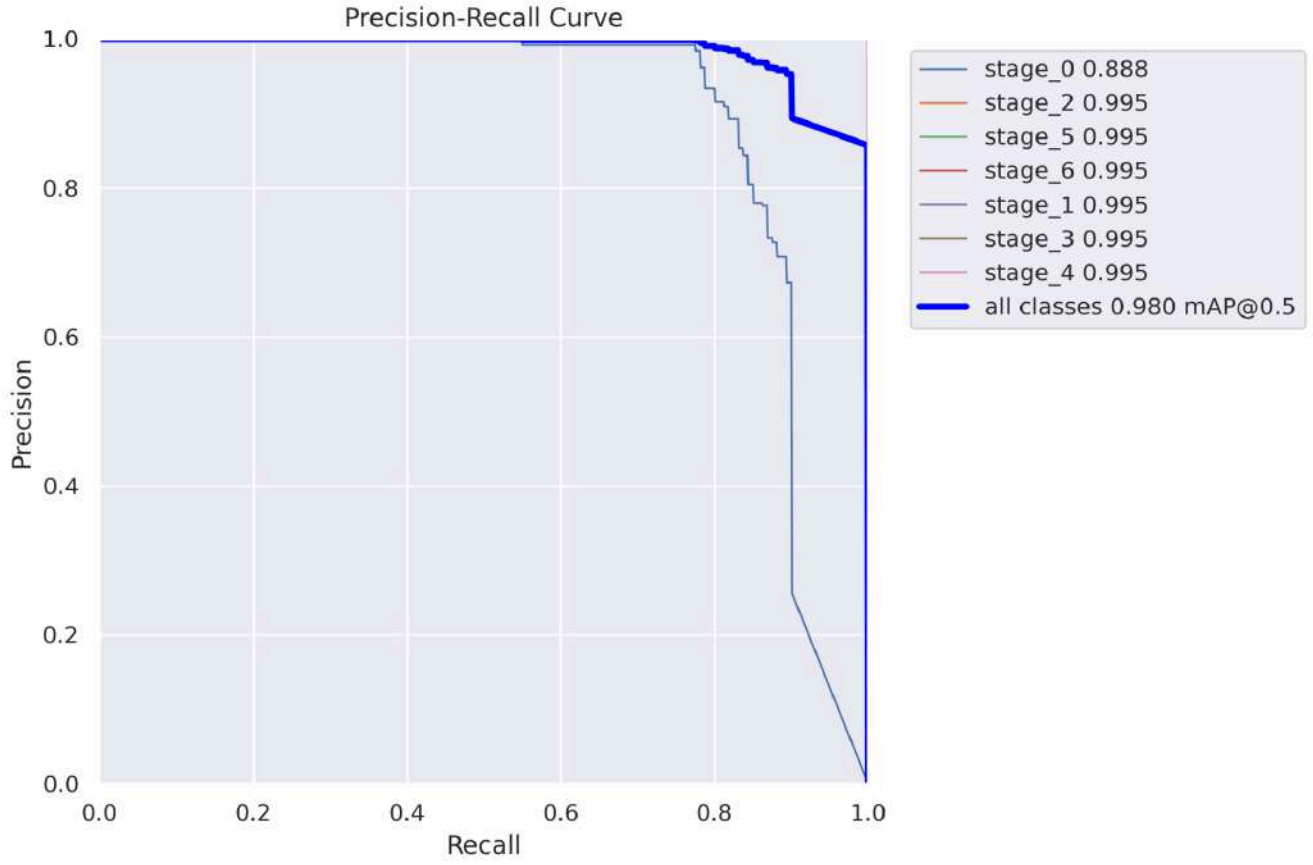

Рис. 48. Кривые Precision-Recall обученной модели YOLOv8 на наборе данных с этапами сборки дрона

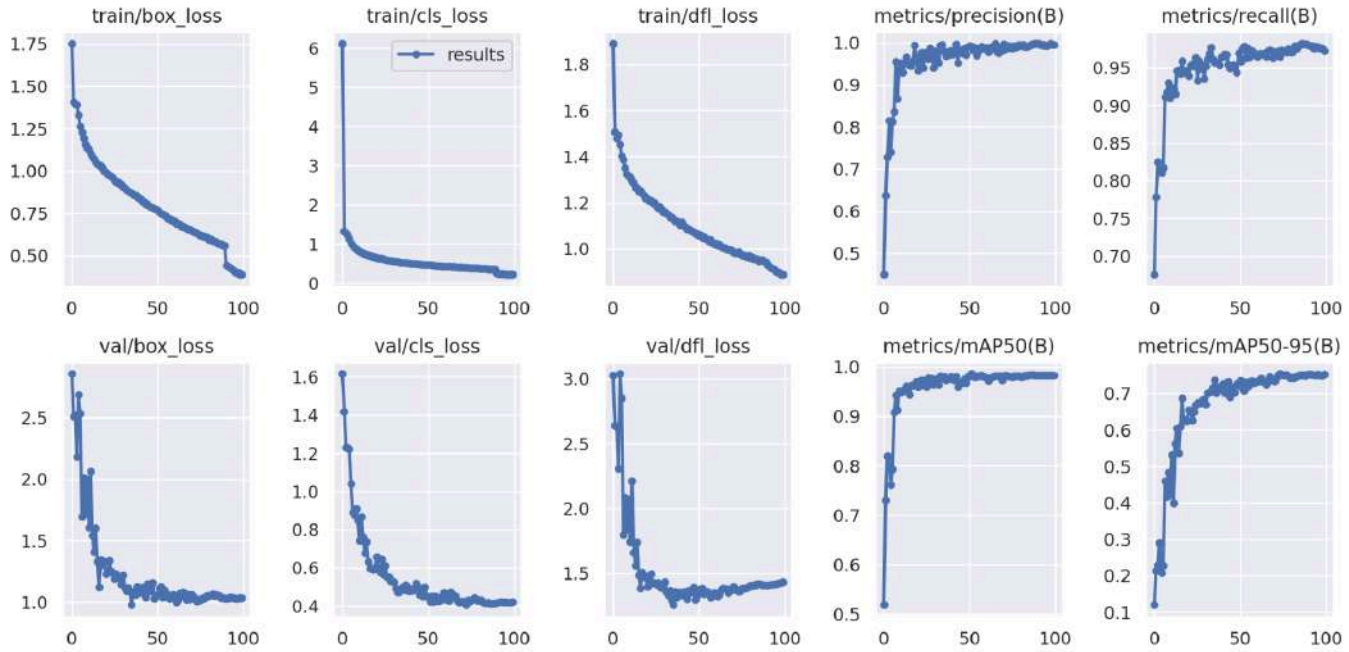

Рис. 49. Графики процесса обучения модели YOLOv8 на наборе данных с этапами сборки дрона

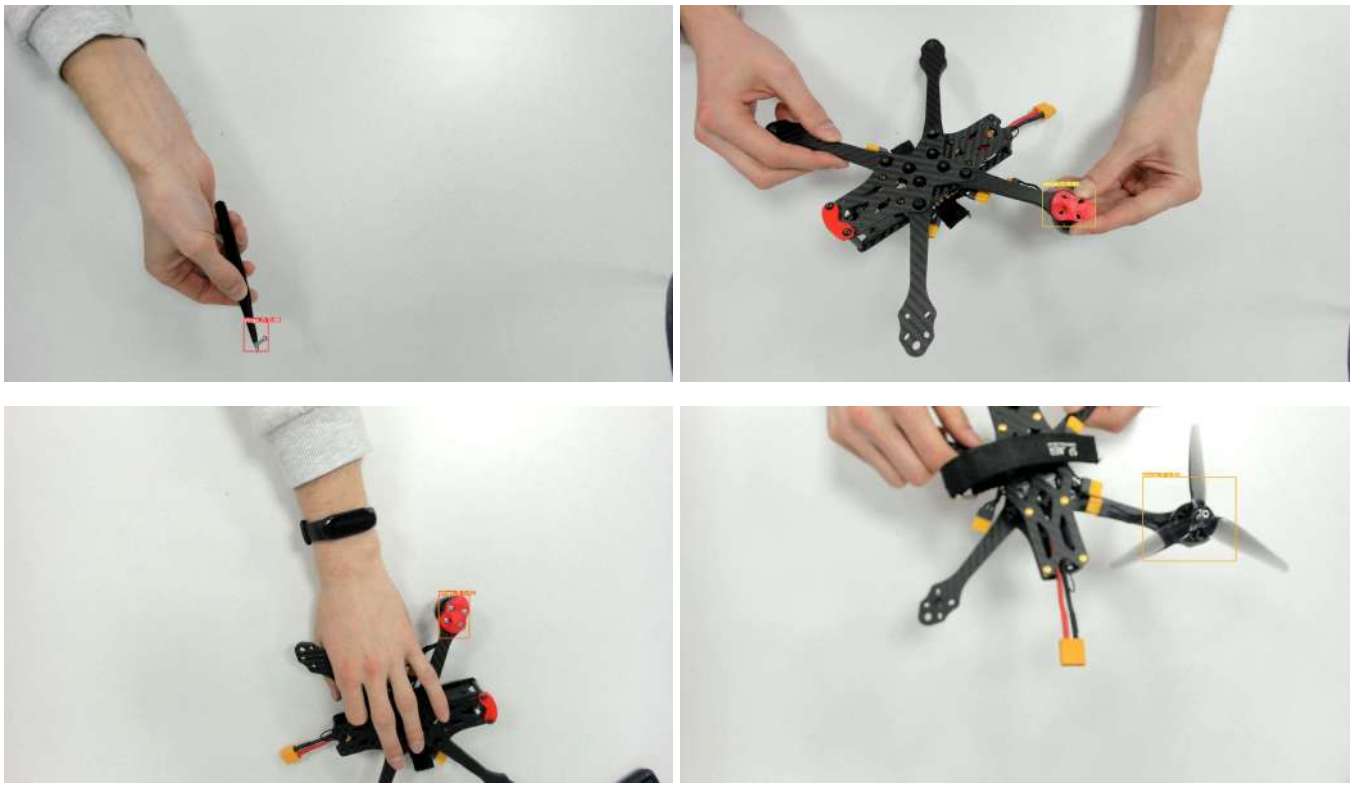

Рис. 50. Пример работы модели YOLOv8, обученной на наборе данных с этапами сборки дрона

8.6 Инструменты сборки дрона

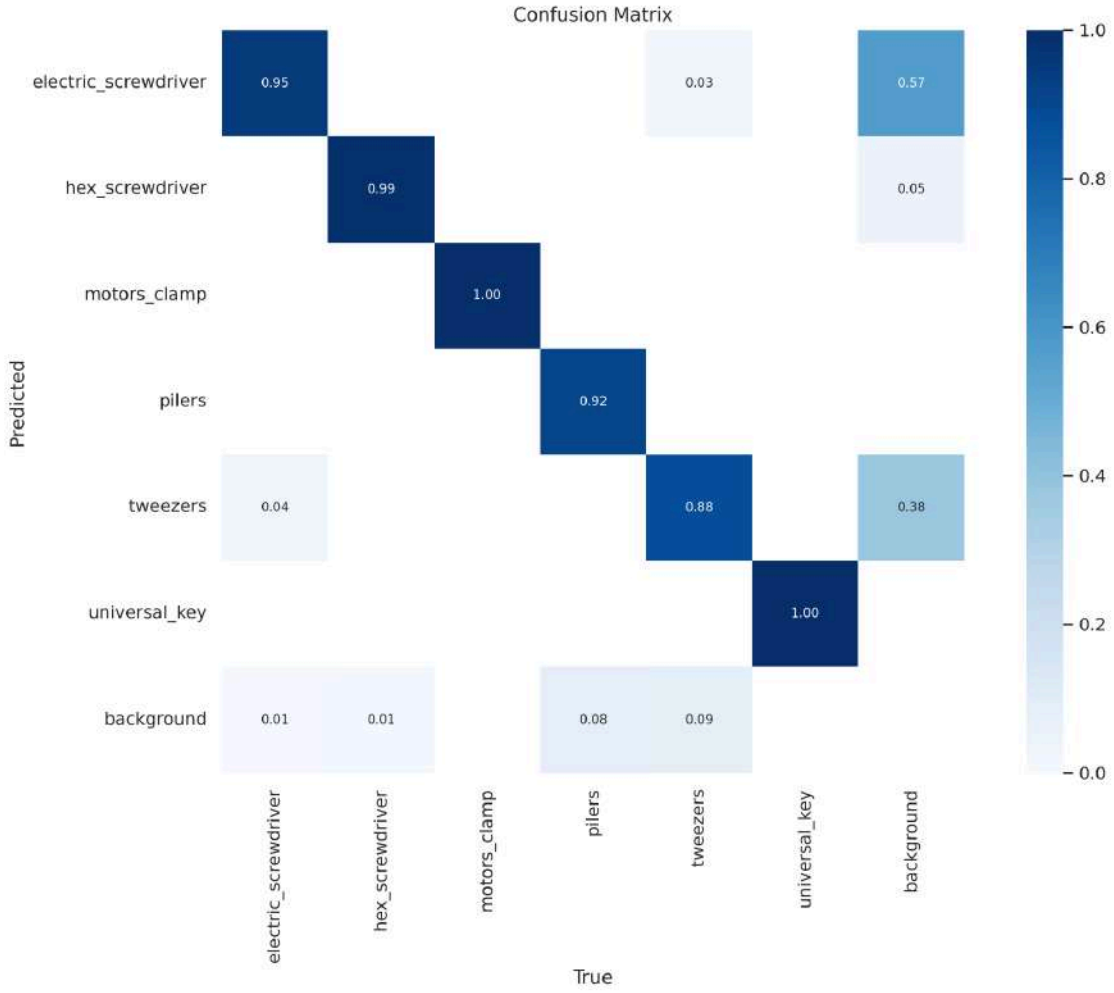

Рис. 51. Матрица ошибок обученной модели YOLOv8 на наборе данных с инструментами для сборки дрона

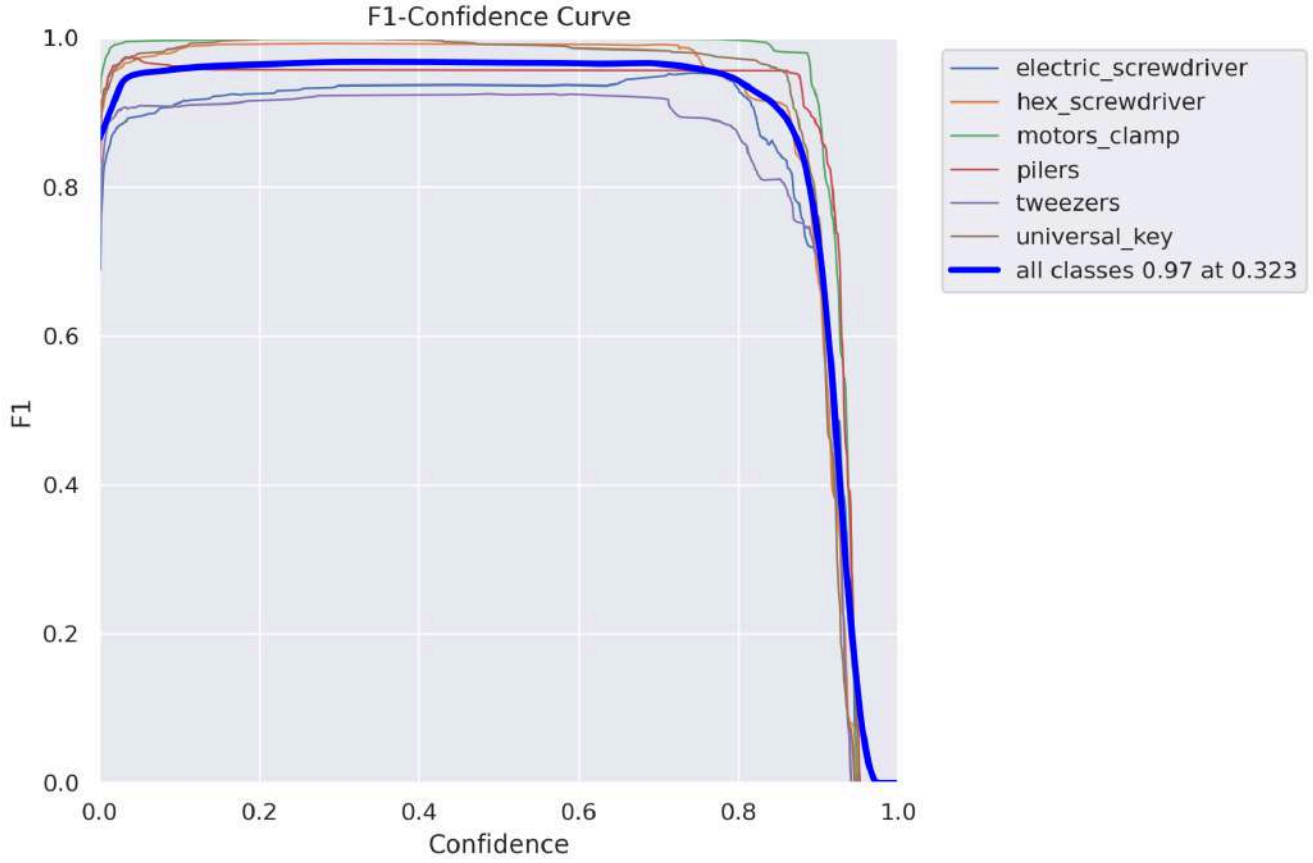

Рис. 52. Кривые F1-Confidence обученной модели YOLOv8 на наборе данных с инструментами для сборки дрона

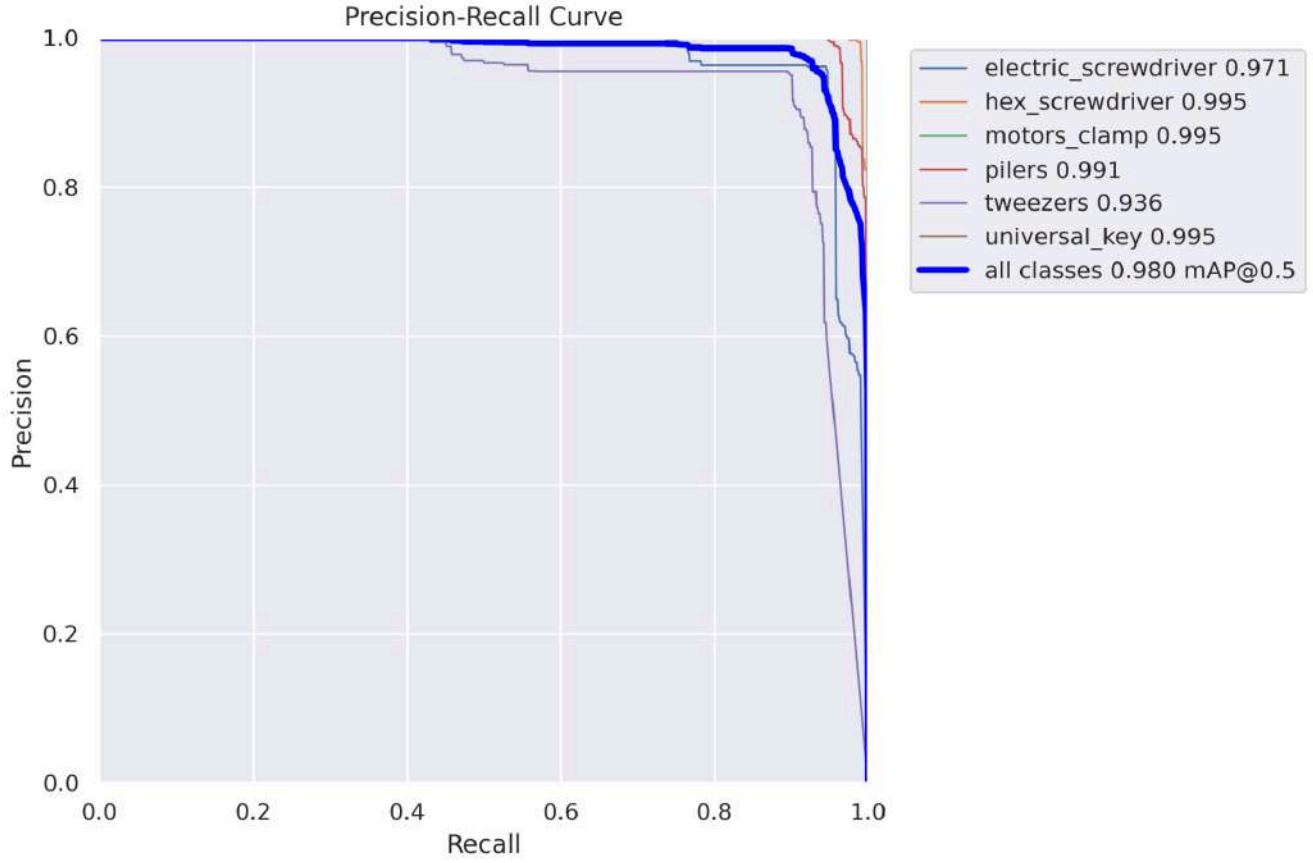

Рис. 53. Кривые Precision-Recall обученной модели YOLOv8 на наборе данных с инструментами для сборки дрона

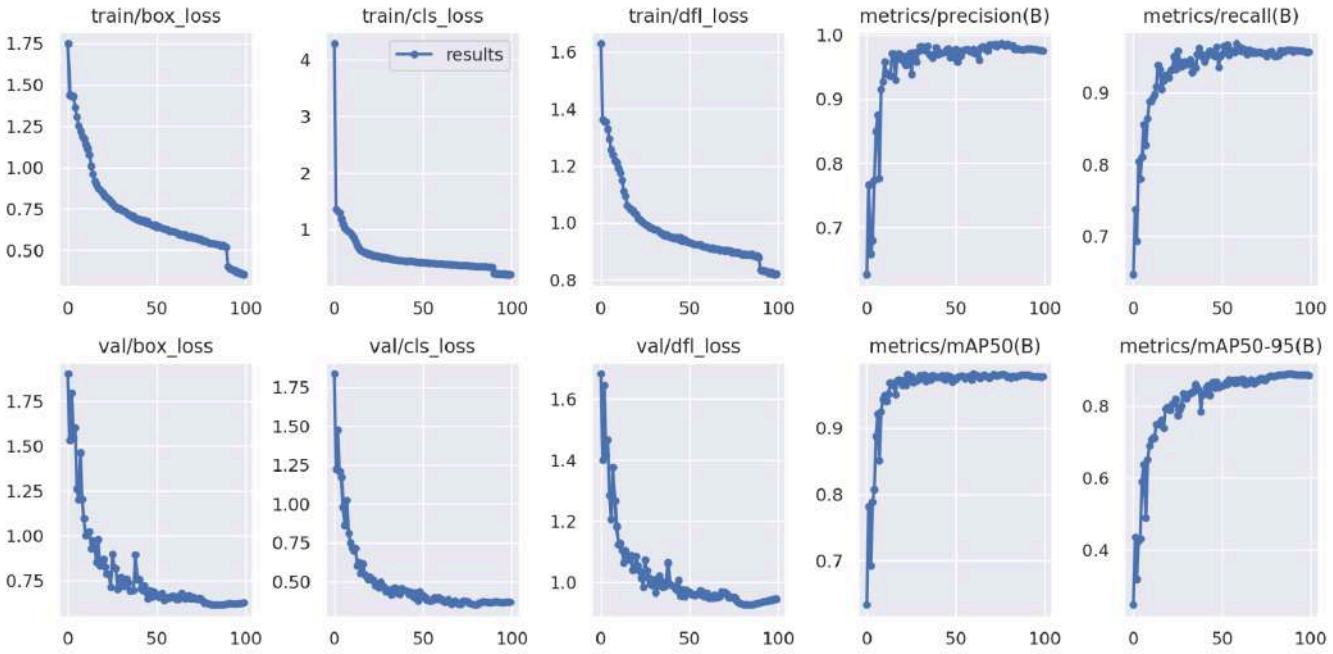

Рис. 54. Графики процесса обучения модели YOLOv8 на наборе данных с инструментами для сборки дрона

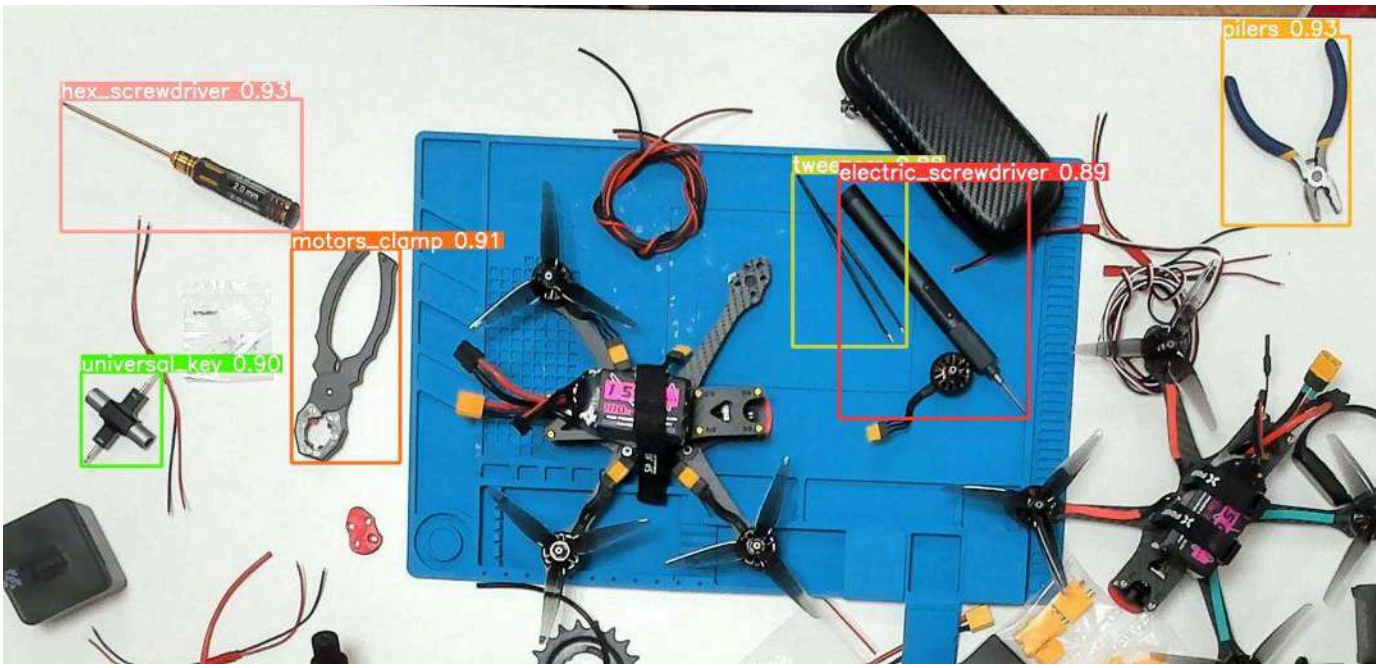

Рис. 55. Пример работы модели YOLOv8, обученной на наборе данных с инструментами для сборки дрона

8.7 Винты для сборки дрона

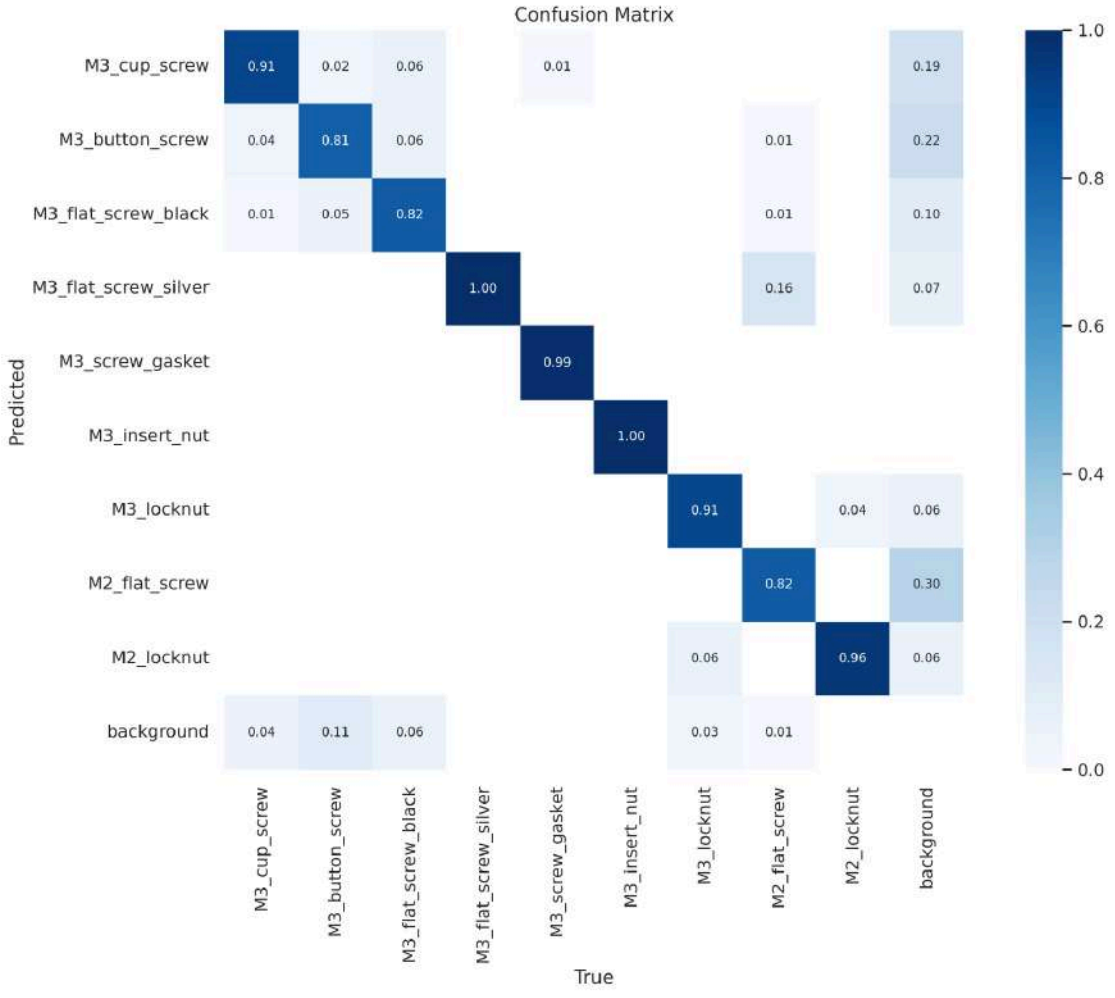

Рис. 56. Матрица ошибок обученной модели YOLOv8 на наборе данных с винтами для сборки дрона

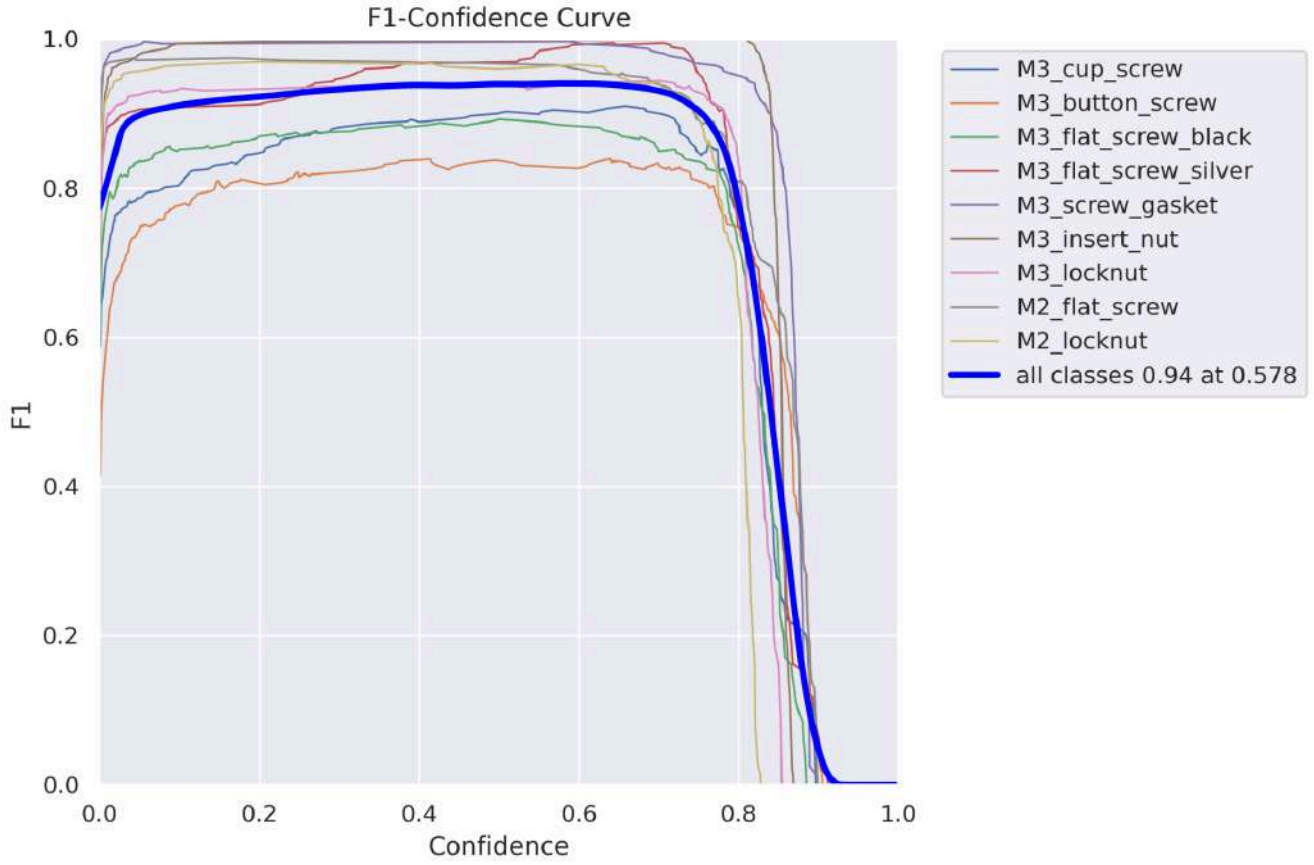

Рис. 57. Кривые F1-Confidence обученной модели YOLOv8 на наборе данных с винтами для сборки дрона

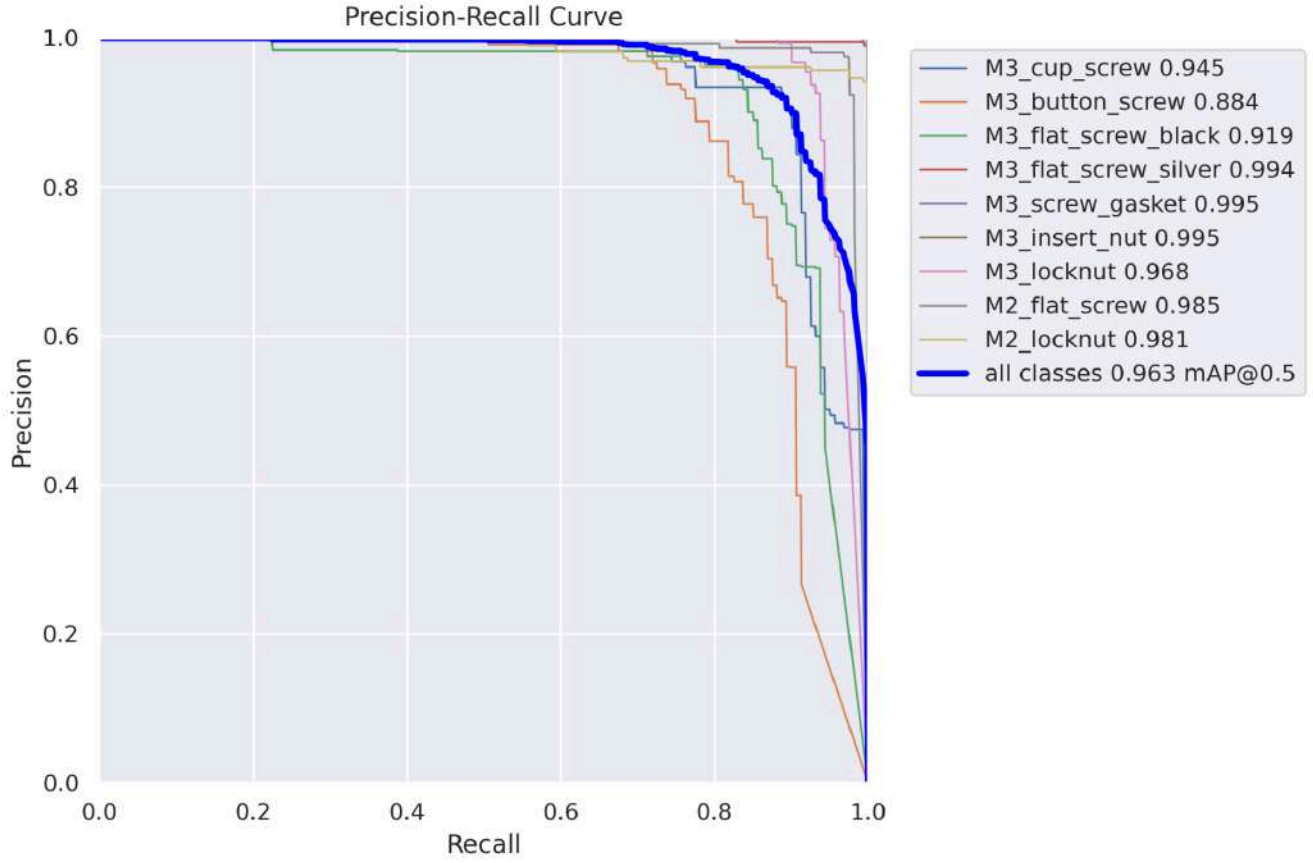

Рис. 58. Кривые Precision-Recall обученной модели YOLOv8 на наборе данных с винтами для сборки дрона

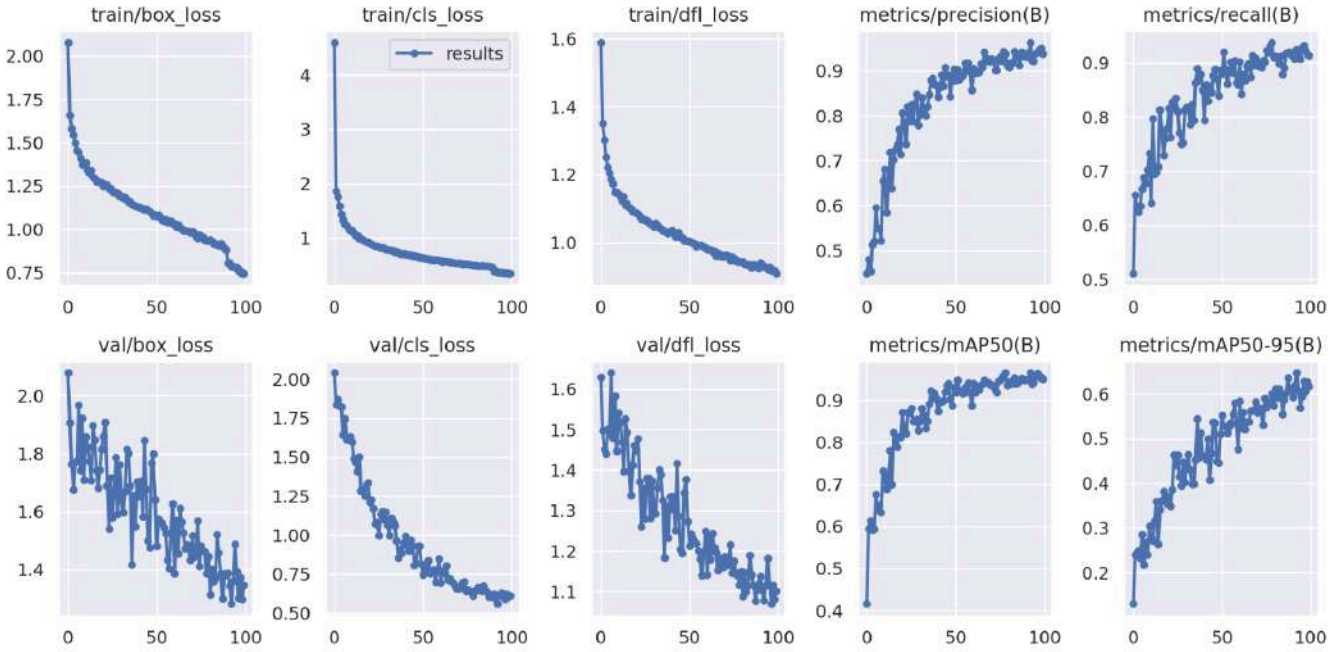

Рис. 59. Графики процесса обучения модели YOLOv8 на наборе данных с винтами для сборки дрона

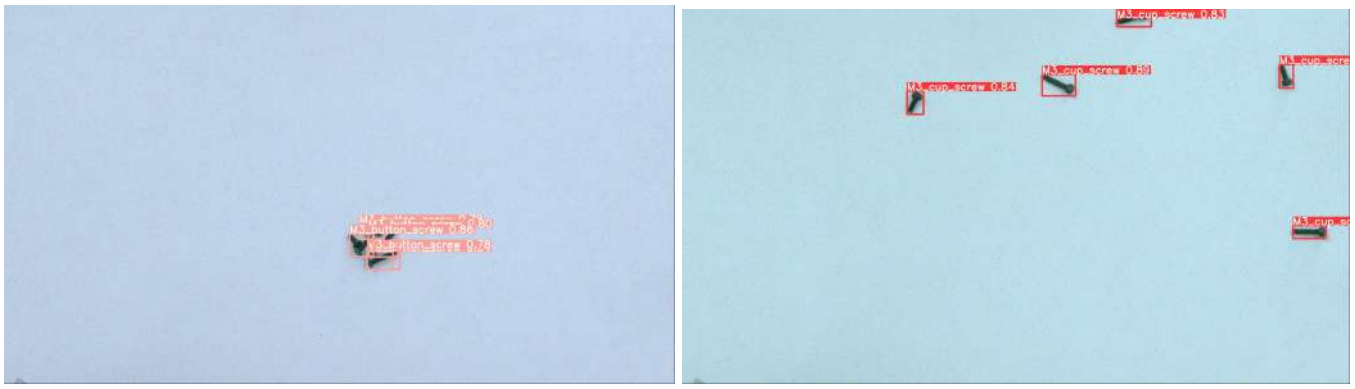

Рис. 60. Пример работы модели YOLOv8, обученной на наборе данных с винтами для сборки дрона

9 Заключение

Результатом данной работы стало улучшение программно-аппаратного комплекса, решающего задачу контроля ручных операций в промышленном производстве. Приложение для контроля сборочного процесса включает в себя слежение не только за основным процессом сборки, но и за соблюдением правил техники безопасности на рабочем месте. Решение представляет из себя многопоточное приложение с множеством сценариев контроля, которое может быть быстро адаптировано под разные отрасли промышленности.

В ходе работы были изучены такие алгоритмы трекинга как MOSSE, KCF, DeepSRDCF, AutoTrack, SiamRPN, SiamRPN++ и SiamBAN. Эти алгоритмы были адаптированы и протестированы для различных сценариев сборки FPV-дронов, таких как перенос винтов в зону сборки, исправление ошибок классификации, контроль за сборкой шасси и моторов. Также алгоритмы трекинга были использованы для повышения качества детекции модели YOLOv8. Проведенные эксперименты показали, что использование алгоритмов трекинга значительно повышает точность и надежность системы. Наиболее эффективными оказались алгоритмы AutoTrack, SiamRPN и SiamBAN, которые обеспечивают баланс между высокой точностью и скоростью работы, что делает их оптимальными для применения в реальном времени.

С целью дополнительного повышения качества работы приложения по контролю ручных операций в области сборки были сняты и размечены дополнительные наборы данных для обучения моделей детекции, а также было

разработано решение по повышению FPS приложения при помощи конкурентных вычислений и использования ресурсов GPU.

Разработанная система способна работать в реальном времени и обеспечивать высокую точность контроля ручных операций. Приложение было успешно протестировано на проектном стенде и готово к внедрению в реальных производственных условиях.

10 Список литературы

1. Мотив Нейроморфные технологии // URL: https://motivnt.ru/интеллектуальная-технология-контрол/ (дата обращения: 27.04.2023).
2. Lee K. H., Hwang J. N. On-road pedestrian tracking across multiple driving recorders //IEEE Transactions on Multimedia. – 2015. – Т. 17. – №. 9. – С. 1429-1438.
3. Bonin-Font F., Ortiz A., Oliver G. Visual navigation for mobile robots: A survey //Journal of intelligent and robotic systems. – 2008. – Т. 53. – С. 263-296.
4. Haritaoglu I., Harwood D., Davis L. S. W/sup 4: real-time surveillance of people and their activities //IEEE Transactions on pattern analysis and machine intelligence. – 2000. – Т. 22. – №. 8. – С. 809-830.
5. Tang S. et al. Multiple people tracking by lifted multicut and person re-identification //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 3539-3548.
6. Bolme D. S. et al. Visual object tracking using adaptive correlation filters //2010 IEEE computer society conference on computer vision and pattern recognition. – IEEE, 2010. – С. 2544-2550.
7. Bertinetto L. et al. Fully-convolutional siamese networks for object tracking //Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14. – Springer International Publishing, 2016. – С. 850-865.
8. Yin J. et al. A unified object motion and affinity model for online multi-object

tracking //Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. – 2020. – С. 6768-6777.

9. Shen J. et al. Multiobject tracking by submodular optimization //IEEE transactions on cybernetics. – 2018. – Т. 49. – №. 6. – С. 1990-2001.
10. Kart U. et al. Object tracking by reconstruction with view-specific discriminative correlation filters //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2019. – С. 1339-1348.
11. Lu X. et al. Video object segmentation with episodic graph memory networks //Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. – Springer International Publishing, 2020. – С. 661-679.
12. Lu X. et al. Learning video object segmentation from unlabeled videos //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2020. – С. 8960-8970.
13. Henriques J. F. et al. High-speed tracking with kernelized correlation filters //IEEE transactions on pattern analysis and machine intelligence. – 2014. – Т. 37. – №. 3. – С. 583-596.
14. Danelljan M. et al. Accurate scale estimation for robust visual tracking //British machine vision conference, Nottingham, September 1-5, 2014. – Bmva Press, 2014.
15. Li B. et al. High performance visual tracking with siamese region proposal network //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2018. – С. 8971-8980.
16. Arulampalam M. S. et al. A tutorial on particle filters for online nonlinear/non-Gaussian Bayesian tracking //IEEE Transactions on signal processing. – 2002. – Т. 50. – №. 2. – С. 174-188.
17. Redmon J. et al. You only look once: Unified, real-time object detection //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – С. 779-788.
18. Redmon J., Farhadi A. YOLO9000: better, faster, stronger //Proceedings of the

IEEE conference on computer vision and pattern recognition. – 2017. – С. 7263-7271.

19. Vats A., Anastasiu D. C. Enhancing retail checkout through video inpainting, yolov8 detection, and deepsort tracking //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2023. – С. 5529-5536.
20. Пименова М. Б. Применение фильтра Калмана в задачах трекинга воздушных объектов //Политехнический молодежный журнал. – 2019. – №. 12. – С. 8-8.
21. Ядерный метод // Википедия. [2024]. Дата обновления: 08.01.2024. URL: https://ru.wikipedia.org/?curid=7631825&oldid=135438551 (дата обращения: 08.01.2024).
22. Horn B. K. P., Schunck B. G. Determining optical flow //Artificial intelligence. – 1981. – Т. 17. – №. 1-3. – С. 185-203.
23. Lucas B. D., Kanade T. An iterative image registration technique with an application to stereo vision //IJCAI'81: 7th international joint conference on Artificial intelligence. – 1981. – Т. 2. – С. 674-679.
24. Bouguet J. Y. et al. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm //Intel corporation. – 2001. – Т. 5. – №. 1-10. – С. 4.
25. Maybeck P. S. The Kalman filter: An introduction to concepts //Autonomous robot vehicles. – New York, NY : Springer New York, 1990. – С. 194-204.
26. La Scala B. F., Bitmead R. R. Design of an extended Kalman filter frequency tracker //IEEE Transactions on Signal Processing. – 1996. – Т. 44. – №. 3. – С. 739-742.
27. Julier S. J., Uhlmann J. K. Unscented filtering and nonlinear estimation //Proceedings of the IEEE. – 2004. – Т. 92. – №. 3. – С. 401-422.
28. Nummiaro K., Koller-Meier E., Van Gool L. An adaptive color-based particle filter //Image and vision computing. – 2003. – Т. 21. – №. 1. – С. 99-110.
29. Comaniciu D., Meer P. Mean shift: A robust approach toward feature space analysis //IEEE Transactions on pattern analysis and machine intelligence. –

2002. – Т. 24. – №. 5. – С. 603-619.

30. Bradski G. R. Computer vision face tracking for use in a perceptual user interface //Intel technology journal. – 1998. – Т. 2.
31. Casasent D. Unified synthetic discriminant function computational formulation //Applied Optics. – 1984. – Т. 23. – №. 10. – С. 1620-1627.
32. Mahalanobis A., Kumar B. V. K. V., Casasent D. Minimum average correlation energy filters //Applied Optics. – 1987. – Т. 26. – №. 17. – С. 3633-3640.
33. Mahalanobis A. et al. Unconstrained correlation filters //Applied Optics. – 1994. – Т. 33. – №. 17. – С. 3751-3759.
34. Bolme D. S., Draper B. A., Beveridge J. R. Average of synthetic exact filters //2009 IEEE conference on computer vision and pattern recognition. – IEEE, 2009. – С. 2105-2112.
35. Danelljan M. et al. Convolutional features for correlation filter based visual tracking //Proceedings of the IEEE international conference on computer vision workshops. – 2015. – С. 58-66.
36. Li Y. et al. AutoTrack: Towards high-performance visual tracking for UAV with automatic spatio-temporal regularization //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2020. – С. 11923-11932.
37. Wang Q. et al. Dcfnet: Discriminant correlation filters network for visual tracking //arXiv preprint arXiv:1704.04057. – 2017.
38. Danelljan M. et al. Learning spatially regularized correlation filters for visual tracking //Proceedings of the IEEE international conference on computer vision. – 2015. – С. 4310-4318.
39. LeCun Y., Bengio Y., Hinton G. Deep learning //nature. – 2015. – Т. 521. – №. 7553. – С. 436-444.
40. Wang T. et al. Generative neural networks for anomaly detection in crowded scenes //IEEE Transactions on Information Forensics and Security. – 2018. – Т. 14. – №. 5. – С. 1390-1399.
41. Simonyan K., Zisserman A. Very deep convolutional networks for large-scale

image recognition //arXiv preprint arXiv:1409.1556. – 2014.
42. Szegedy C. et al. Going deeper with convolutions //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2015. – С. 1-9.
43. He K. et al. Deep residual learning for image recognition //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2016. – С. 770-778.
44. Huang G. et al. Densely connected convolutional networks //Proceedings of the IEEE conference on computer vision and pattern recognition. – 2017. – С. 4700-4708.
45. Hochreiter S., Schmidhuber J. Long short-term memory //Neural computation. – 1997. – Т. 9. – №. 8. – С. 1735-1780.
46. Cho K. et al. On the properties of neural machine translation: Encoder-decoder approaches //arXiv preprint arXiv:1409.1259. – 2014.
47. Shi Y. et al. Deep LSTM based feature mapping for query classification //Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: Human language technologies. – 2016. – С. 1501-1511.
48. Chen Z. et al. Siamese box adaptive network for visual tracking //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2020. – С. 6668-6677.
49. Li B. et al. Siamrpn++: Evolution of siamese visual tracking with very deep networks //Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. – 2019. – С. 4282-4291.
50. Karasulu B., Korukoglu S. A software for performance evaluation and comparison of people detection and tracking methods in video processing //Multimedia Tools and Applications. – 2011. – Т. 55. – С. 677-723.
51. Li H., Shen C., Shi Q. Real-time visual tracking using compressive sensing //CVPR 2011. – IEEE, 2011. – С. 1305-1312.
52. Čehovin L., Leonardis A., Kristan M. Visual object tracking performance measures revisited //IEEE Transactions on Image Processing. – 2016. – Т. 25. –

№. 3. – С. 1261-1274.

53. Kristan M. et al. A two-stage dynamic model for visual tracking //IEEE transactions on systems, man, and cybernetics, part B (cybernetics). – 2010. – Т. 40. – №. 6. – С. 1505-1520.
54. Luiten J. et al. Hota: A higher order metric for evaluating multi-object tracking //International journal of computer vision. – 2021. – Т. 129. – С. 548-578.
55. Bernardin K., Stiefelhagen R. Evaluating multiple object tracking performance: the clear mot metrics //EURASIP Journal on Image and Video Processing. – 2008. – Т. 2008. – С. 1-10.
56. Wu Y., Lim J., Yang M. H. Object Tracking Benchmark (2015) //IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2014. – С. 1834-1848.
57. Kristan M. et al. The visual object tracking vot2015 challenge results //Proceedings of the IEEE international conference on computer vision workshops. – 2015. – С. 1-23.
58. Bewley A. et al. Simple online and realtime tracking //2016 IEEE international conference on image processing (ICIP). – IEEE, 2016. – С. 3464-3468.
59. Расстояние Махаланобиса // Википедия. [2023]. Дата обновления: 11.08.2023. URL: https://ru.wikipedia.org/?curid=2458133&oldid=132293816 (дата обращения: 11.08.2023).
60. Венгерский алгоритм // Википедия. [2023]. Дата обновления: 17.04.2023. URL: https://ru.wikipedia.org/?curid=2246561&oldid=129883801 (дата обращения: 17.04.2023).