СОДЕРЖАНИЕ

Введение

С каждым днем увеличивается потребность автоматизации контроля производственного процесса и качества производимых товаров, актуальной становится задача автоматического управления производственным процессом. В настоящее время существует достаточно большое количество открытых решений для конкретных, индивидуальных подзадач, но они не объединены в единую универсальную систему. Такие системы являются коммерческими, засекреченными и дорогими, что не позволяет осуществить их массовое внедрение. В данном проекте описывается создание универсальной системы контроля производственного процесса на основе алгоритмов компьютерного зрения и глубокого обучения для анализа видеопотока. Рассмотрены задачи распознавания дефектов, класса объекта и его положения в пространстве, производится анализ перемещения объекта, качественное решение которых позволит управлять производственным процессом в реальном времени. Спроектированную систему в течение следующего этапа планируется сопроводить графическим интерфейсом, который откроет ее более широкому кругу пользователей. Качество созданного решения будет оценено посредством его применения на реальных производствах.

Цель данного проекта – автоматическое управление производственным процессом по видеопотоку, в том числе: автоматическая оценка качества выпускаемой и принимаемой продукции и результатов технологических операций, автоматическое принятие решений в ходе процесса и автоматическое отслеживание объектов.

Задачи, решаемые в течение первого этапа рассмотрены в указанных ниже главах:

- Разработка архитектуры библиотеки (рассматривается в главе 1).
- Поиск, сбор, разметка и обогащение данных (описана в главе 2).
- Определение положения объекта (детали производственного оборудования, человека, машины) в пространстве и расстояния между ними (глава 3).
- Распознавание дефектов и прочих образований на материале (глава 4) .
- Распознавание номера объекта (автомобиля, вагона поезда, других объектов на производстве или в городской среде) (рассмотрено в главе 5).
- Анализ перемещения объектов (людей, машин и других) в пределах установленной области (описан в главе 6).
- Определение размеров объекта (для автомобилей и прочих т/с и для руды на конвейере) (глава 7).
- Возможность системы обучиться решать одну из указанных выше задач для новой категории объектов при условии предоставления обучающих данных – данная задача рассматривается во всех главах, так как алгоритмы проектировались с учетов возможности дообучения.

ОСНОВНАЯ ЧАСТЬ

1. Глобальная архитектура системы

Основная причина разработки данного фреймворка – предоставить полный инструментарий для построения конвейеров самых часто используемых задач компьютерного зрения, а также следует предусмотреть возможность расширения для реализации дополнительных возможностей. Универсальность всей планируемой системы не позволяет использовать один алгоритм, поэтому для каждой конкретной задачи потребуется выбор оптимальной архитектуры нейронной сети. Поставленные задачи планируется решать с помощью детекции объектов, их локализации, сегментации, трекинга и предсказания траектории движения, распознавания символов и классификации объектов. Результатом работы должен быть фреймворк, позволяющий
строить готовые конвейеры для различных задач компьютерного зрения, а также изменять функционал уже готовых конвейеров путём передачи в него разных моделей нейронных сетей. В разделе описана разработанная блочная система, которая позволит строить гибкие конвейеры, в которых можно изменять архитектуры нейронных сетей.

1.1 Требования к системе

1.1.1 Функциональные требования

1. Библиотека должна поддерживать входные видеопотоки с веб камер, из файлов, с IP камер.
2. Библиотека должна быть способна работать с моделями классификации, детекции и сегментации.
3. Библиотека должна быть способна рисовать ограничивающие рамки для задач детекции, маски для задач сегментации и писать метки для задач классификации, в том числе текстовые.
4. Библиотека должна поддерживать несколько источников ввода.

5. Библиотека должна быть способна работать с различными архитектурами нейронных сетей.
6. Библиотека должна иметь модуль трекинга для подключения к моделям детекции.
7. Библиотека должна уметь работать как с CPU, так и с GPU.
8. Библиотека должна уметь подсчитывать количество объектов для задач детекции.
9. Библиотека должна позволять сохранять веса моделей машинного обучения и их архитектуру.

1.1.2 Нефункциональные требования

1. Библиотека должна быть реализована на Python.
2. Библиотека должна быть реализована с помощью PyTorch.
3. Возможность расширения новым функционалом для задач компьютерного зрения.
4. Скорость работы конвейера должна не уступать аналогам.

1.2 Архитектура

В идею заложено создание структур данных, которые будут называться метаданные, содержащие изображения и информацию о них. Эта информация, как и сами изображения внутри могут меняться в зависимости от компонента, в который они переданы. Компонентами будут служить классы, которые наследуются от абстрактного класса, имеющего метод do, который необходимо реализовать всем компонентам. Каждый из этих компонентов будет обрабатывать данные по-своему и возвращать тот же объект, что был ему передан, но с измененными полями. Таким образом, достигается масштабируемость и гибкость функционала фреймворка.

1.2.1. Диаграмма пакетов

Каждый пакет в пакете components на рисунке 2.1 является следствием решения одного из функционального требования.

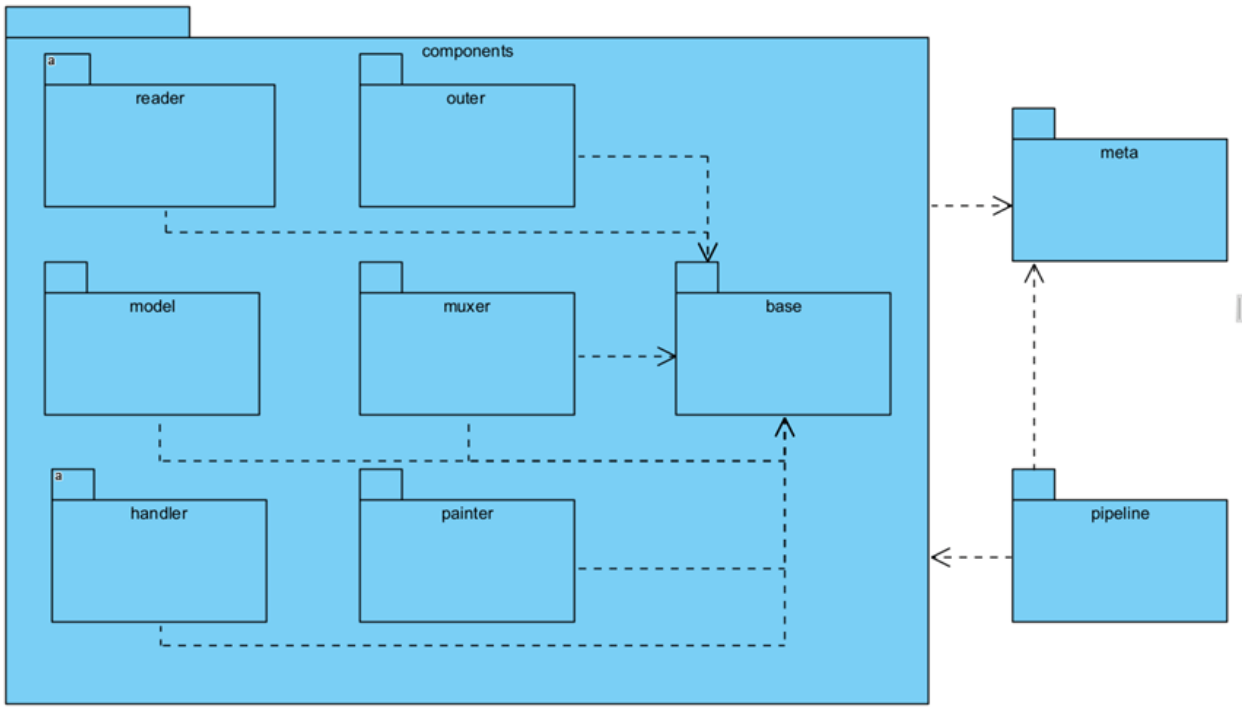


Рисунок 1.1 - Диаграмма пакетов системы

- reader – пакет с компонентами, выполняющими функцию чтения изображений и видеопотоков;
- muxer – пакет, содержащий компонент для объединения потоков данных из различного количества источников, считанными компонентами пакета reader;
- model – пакет, содержащий компоненты для работы с моделями фреймворка PyTorch.
- painter – пакет, содержащий компоненты для отрисовки ограничивающих рамок, масок и классов. А также содержит компонент для объединения нескольких потоков данных в один в виде сетки.
- handler – пакет содержащий компоненты, для фильтрации предсказаний, а также подсчёта объектов.

- base – пакет, содержащий класс, от которого наследуются все компоненты, перечисленные выше. Он реализует необходимые методы для работы со всем компонентами фреймворка.
- meta – пакет, содержащий структуры данных, которыми оперируют компоненты фреймворка.
- pipeline – пакет, содержащий конвейер, класс, который передает структуры данных meta между компонентами. Также собирает и закрывает компоненты.

1.2.2 Диаграмма классов пакета Meta

На рисунке 1.2 изображены классы, которые являются хранилищем для данных, полученных из компонентов. Это структуры данных, которые

являются переносчиками информации между компонентами, как для работы с ними, так и для работы с конвейером.

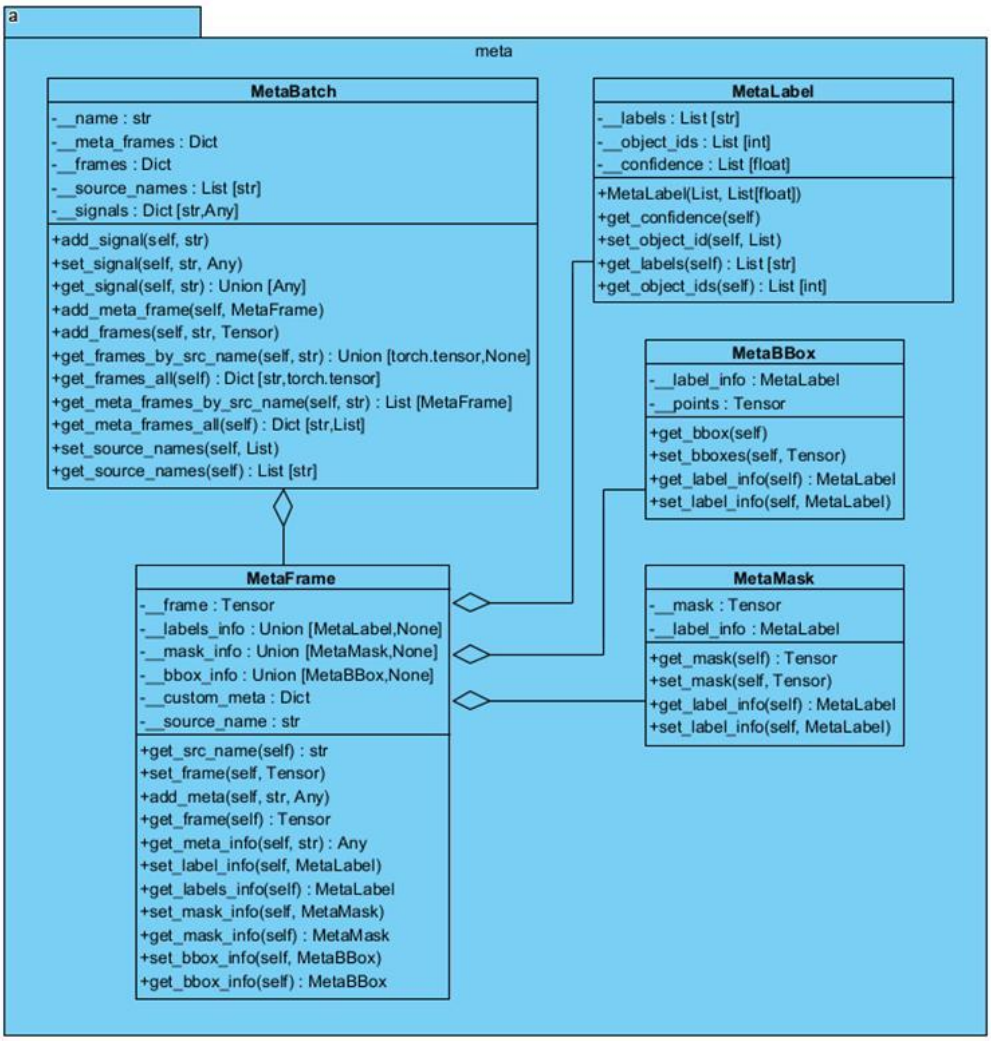


Рисунок 1.2 – диаграмма классов пакета meta

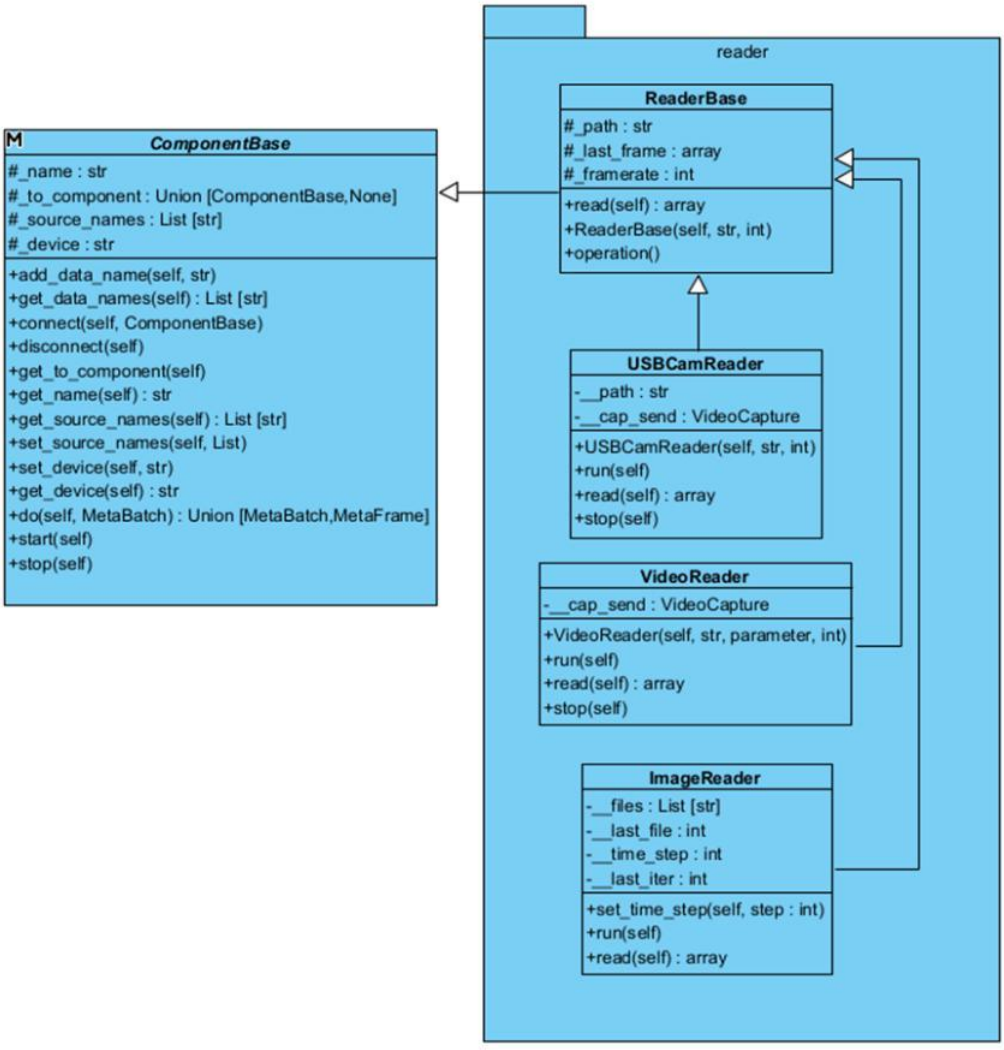
- MetaLabel – класс, хранящий информацию о классах, изображений, масок и ограничивающих рамок;
- MetaBBox – класс, хранящий информацию об ограничивающих рамках и их классах;
- MetaMask – класс, хранящий информацию о масках и классах.
- MetaFrame – класс, хранящий изображение, а также всю информацию, полученную от компонентов об этом изображении.

- MetaBatch – класс, хранящий все изображения из различных источников. Также ModelBatch хранит сигналы, получаемые из компонент, и служит в роли шины.

Для задач, не относящихся к классификации, сегментации или детекции в классе MetaFrame есть поле, которое предназначено для хранения данных, которые могут пригодиться при расширении функционала.

1.2.3 Диаграмма классов пакета reader

На рисунке 1.3 представлена подробная диаграмма классов пакета

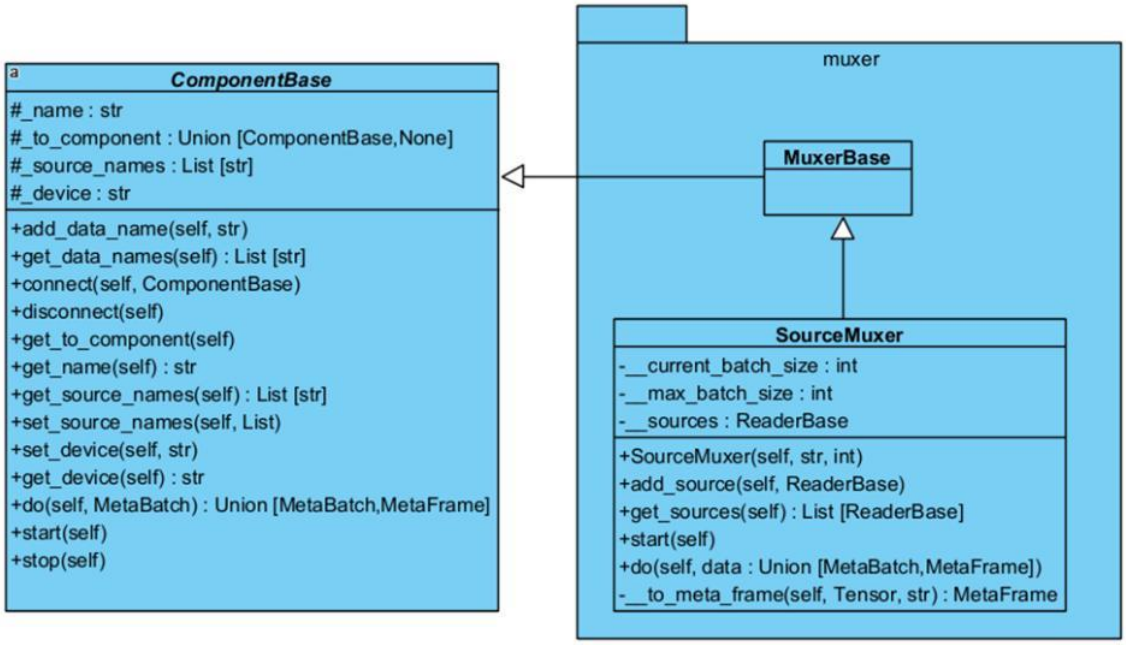


reader. Задача этих компонентов чтение данных из различных источников.

Рисунок 1.3 – Диаграмма классов пакета reader

1.2.4. Диаграмма классов пакета muxer

На рисунке 1.4 изображен пакет muxer. Задача этих компонентов объединение видеопотоков путём добавлений MetaFrame в MetaBatch, с



которым работают все компоненты.

Рисунок 1.4 – Диаграмма классов пакета muxer.

- MuxerBase – абстрактный класс без полей и методов, необходим для того, чтобы пользователь имел возможность от наследоваться и реализовать свой класс Muxer. Класс MuxerBase служит логическим обособлением от класса ComponentBase.
- SourceMuxer – класс, объединяющий потоки данных из разных источников и формирующий из MetaFrame пакеты, передавая в MetaBatch.

Данный компонент позволяет получать видеопотоки из различных источников, получаемых компонентами-наследниками ReaderBase. Также оборачивает изображения в объекты типа MetaFrame, которые содержат само изображение, а также название источника, из которого оно было получено.

Благодаря этому, каждый из компонентов сможет использовать видеопотоки, полученные из нужных источников, что позволит строить длинные конвейеры, которые на выводе будут давать разный результат в зависимости от источника ввода.

- ReaderBase – абстрактный класс содержащий метод read(self), который необходимо реализовать всем классам наследникам. Этот метод считывает кадр из источника и возвращает его.
- CamReader – класс наследник класса ReaderBase, реализующий чтение из веб-камеры и ip-камеры. Хранит последнее прочитанное изображение в качестве защиты от потери кадров.
- VideoReader – класс наследник класса ReaderBase, реализующий чтение видеофайлов.
- ImageReader – класс наследник класса ReaderBase, реализующий чтение изображений.

Для работы в реальном времени можно либо считывать по одному кадру, либо больше, но быть готовым, что видеопоток будет отставать на какое-то время. Возвращает изображение в формате numpy.ndarray.

1.2.5 Диаграмма классов пакета model

Для реализации компонентов model необходимо, чтобы передаваемые модели имели соответствующий формат входных и выходных данных. Для соответствия этому формату необходимо переопределить метод forward для

класса Module из фреймворка PyTorch пакета nn. Структура классов изображена на рисунке 1.5.

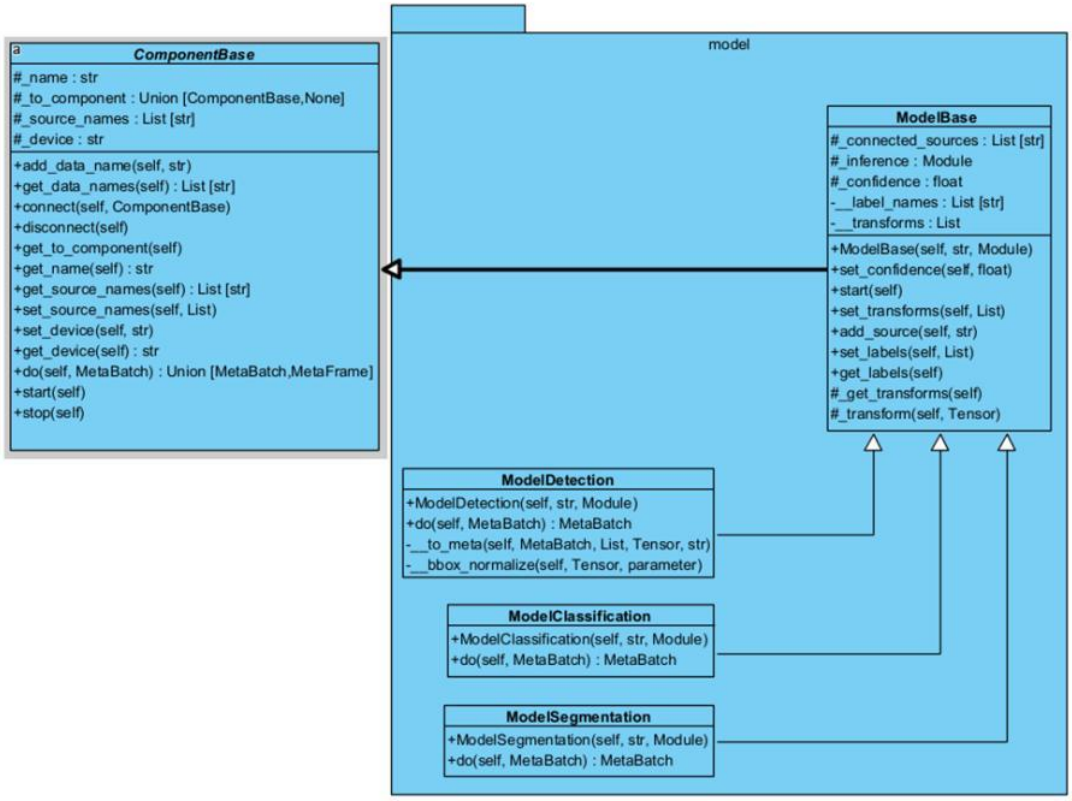


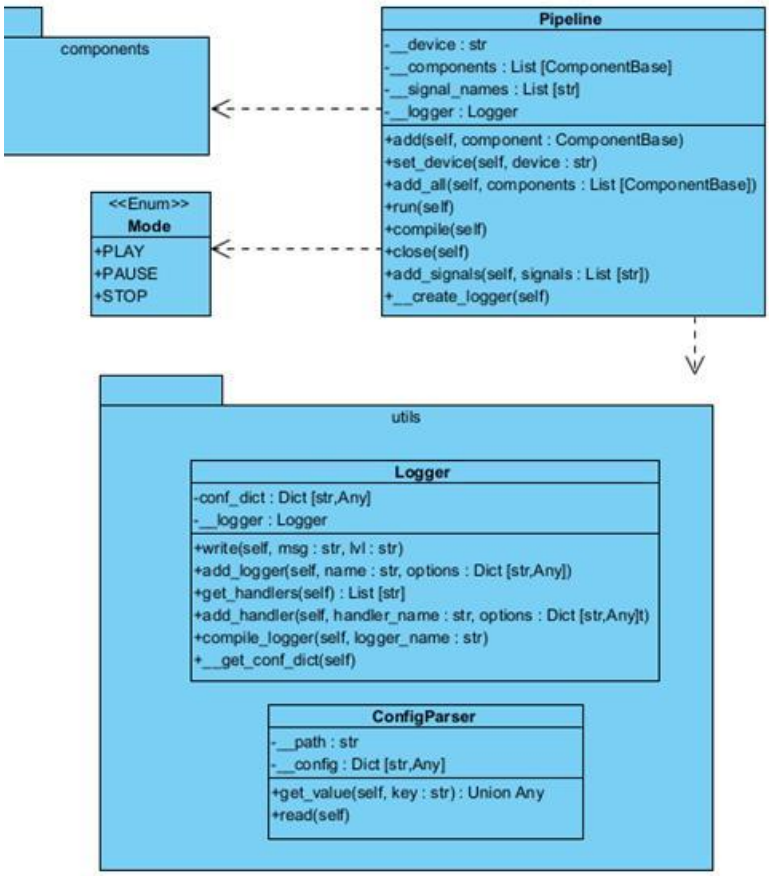
Рисунок 1.5 - Диаграмма классов пакета model

- ModelBase – базовый класс компонентов моделей, содержащий поля и методы, необходимые всем типам моделей компьютерного зрения.
- ModelDetection – компонент, который на вход принимает модель детекции типа torch.nn.Module, метод forward, которой возвращает словарь с ключами boxes, labels, scores, которые хранят тензор ограничивающих рамок, численное значение метки класса и уверенности в ответе соответственно.
- ModelClassification – компонент, который на вход принимает модель типа torch.nn.Module, метод forward, которой возвращает тензоры вероятностей принадлежности изображения к классу.
- ModelSegmentation – компонент, который на вход принимает модель типа torch.nn.Module, метод forward которой возвращает тензор масок

равный числу классов, где каждая маска отвечает за обнаружение класса соответствующего порядку расположения классов.

1.2.6 Диаграмма классов пакета handler

Пакет handler на рисунке 1.6 реализует компоненты постобработки и



анализа данных.

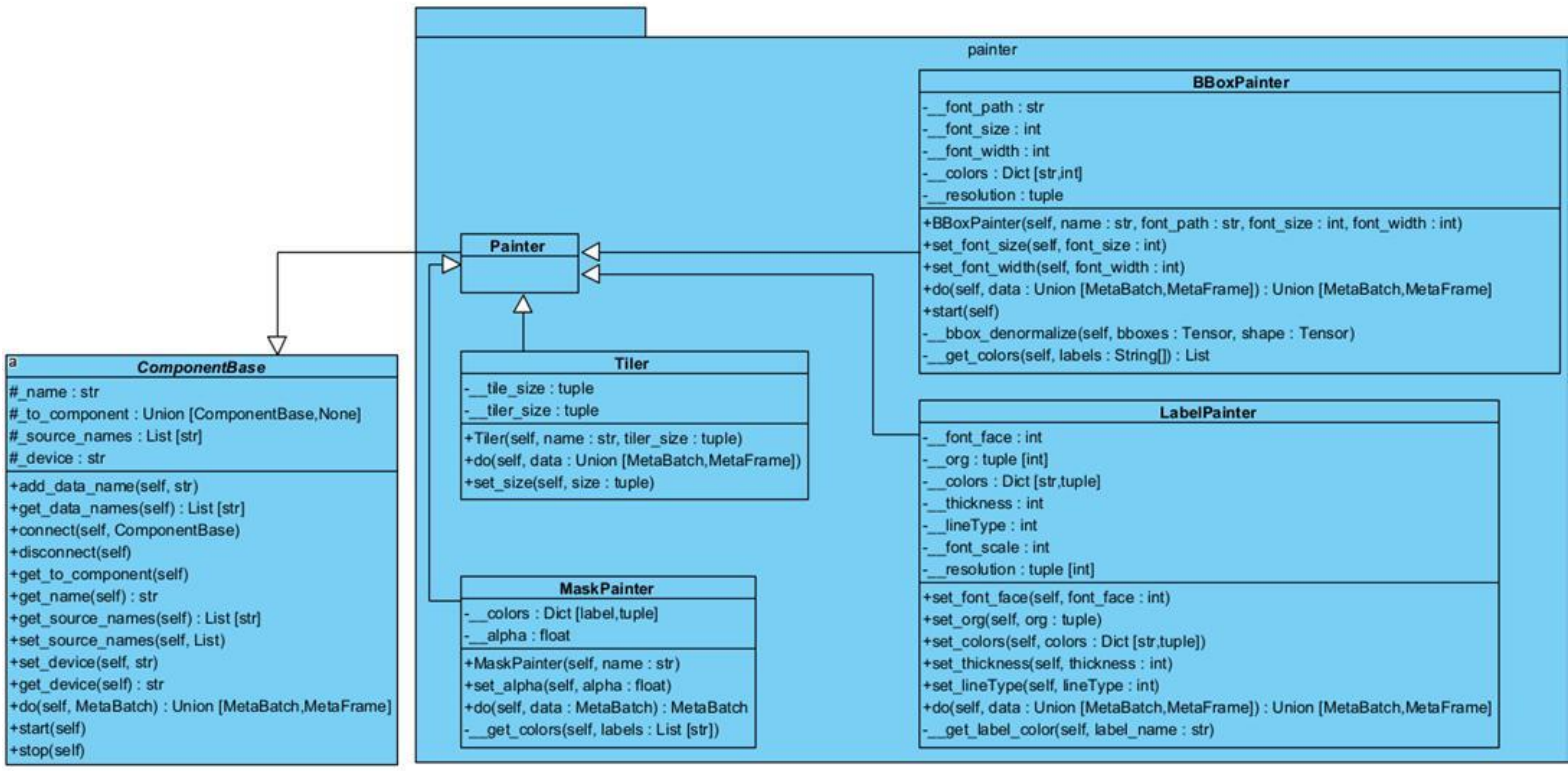
Рисунок 1.6 - Диаграмма классов пакета handler

- Counter – компонент, считающий количество объектов, пересекших выделенную зону. Объекты могут считать, как с учётом наличия уникального идентификатора, так и без.
- Filter – компонент, удаляющий переданные ему классы из предсказания. Что позволяет переиспользовать модели, убирая ненужные предсказания. Что позволит брать готовые модели на огромное

количество классов и просто указывать, как метки мы хотим идентифицировать.

1.2.7 Диаграмма классов пакета painter

На рисунке 1.7 изображен пакет диаграммы классов painter, их задача



отрисовка приведение изображение к определенному формату.

Рисунок 1.7 – Диаграмма классов пакета painter

- Painter – абстрактный класс, имеющий значение логического деление на подгруппу.
- LabelPainter – класс, отвечающий за написание класса, полученного из моделей классификации на изображение. По умолчанию предсказанный класс пишется вверху слева от кадра, но возможно пользовательская настройка.
- BBoxPainter – класс, отвечающий за нанесение ограничивающих рамок и класса объектов, полученных из моделей детекции на изображение.

- MaskPainter – класс, отвечающий за нанесение масок, полученных из моделей сегментации, на изображение.
- Tiler – класс, отвечающий за объединение изображений, полученных из разных источников, в один кадр. Таким образом, появляется возможность записи или вывода результатов всех видеопотоков в один экземпляр.

Первые три предыдущие компонента поддерживают возможность настройки уникального цвета для каждого класса, если параметр не передан, то компоненты сами генерируют уникальный цвет.

1.2.8 Диаграмма классов пакета outer

За вывод результата отвечают компоненты пакета outer, представленные на рисунке 1.8.

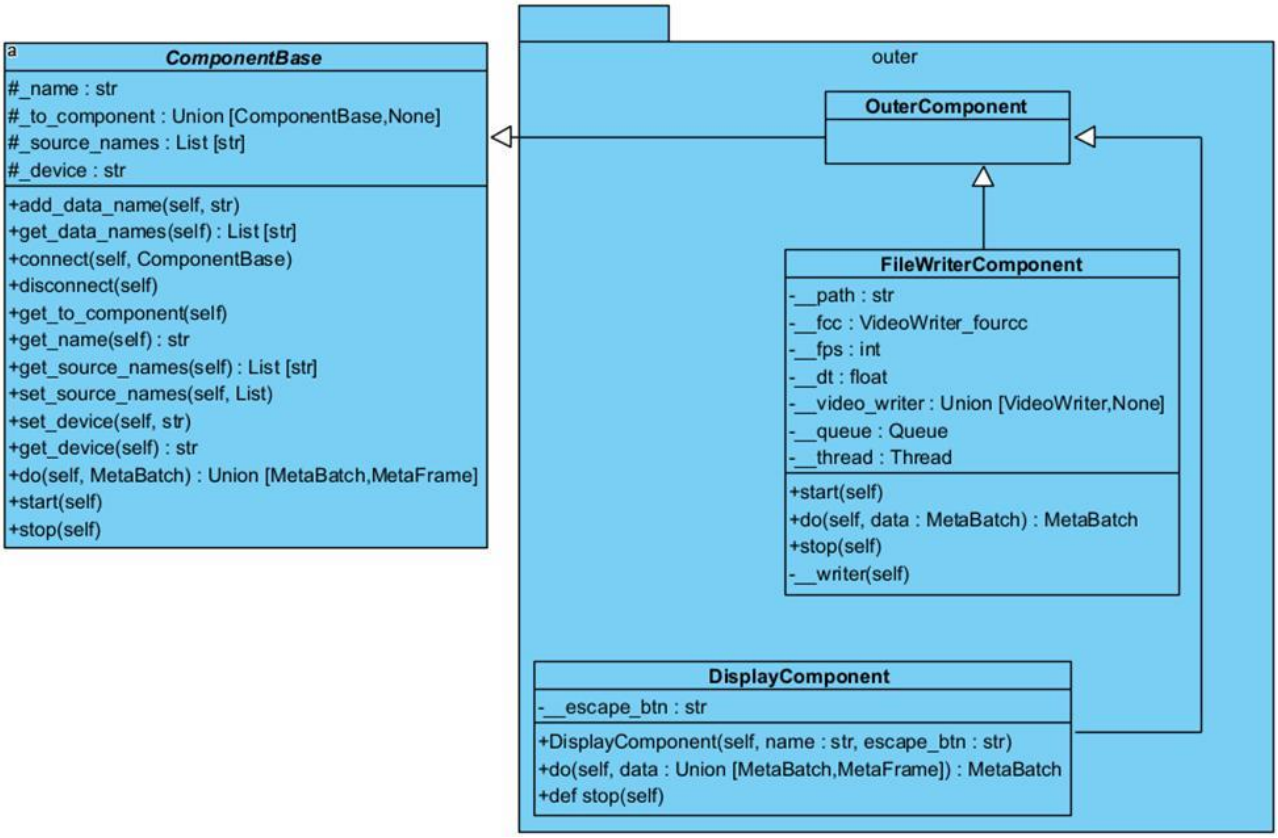
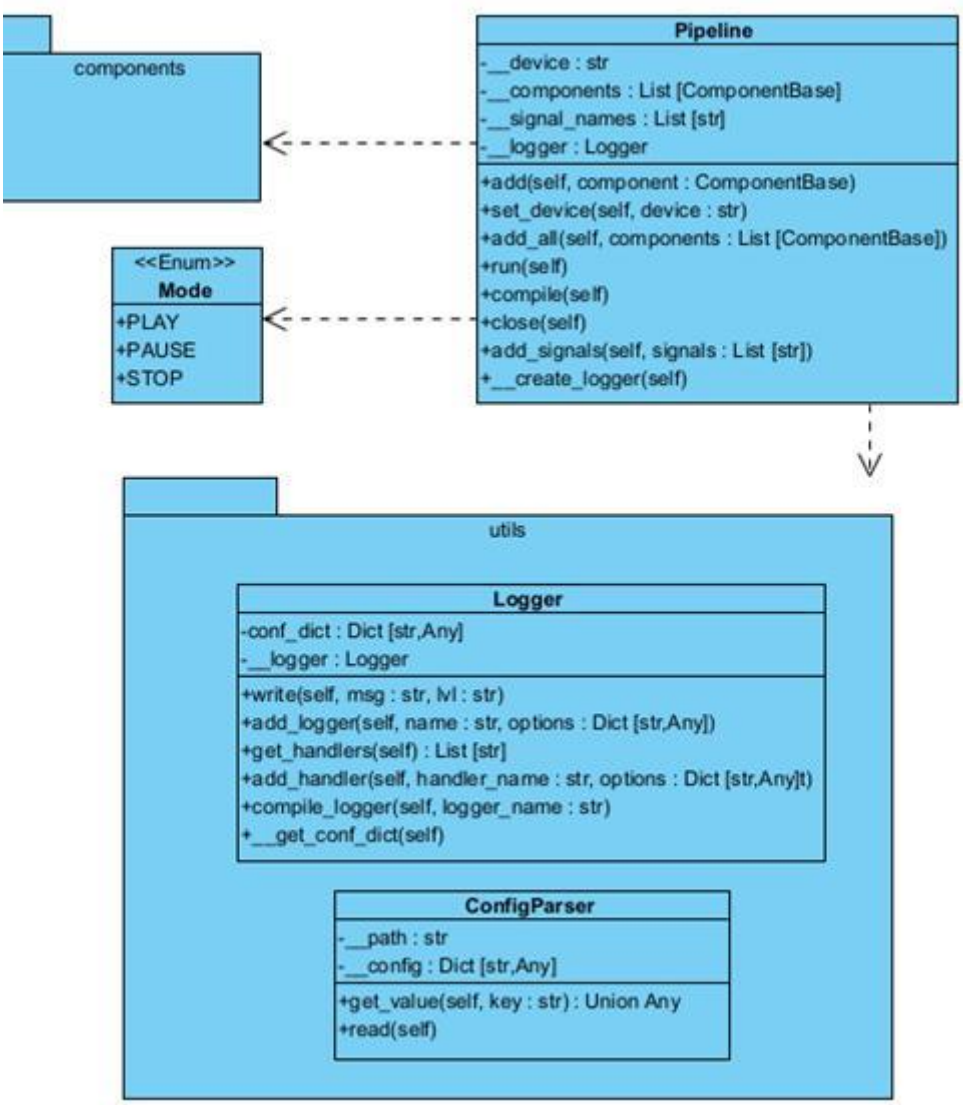


Рисунок 1.8 - Диаграмма классов пакета outer

- OuterComponent – абстрактный класс, отвечающий за логическое деление компонентов пакета outer на подгруппу. Переопределяет параметр _source_names, который служит компонентам для указания с какими потоками видеоданных они работают. В компонентах, наследующихся от OuterComponent, это значение будет равно списку из одного элемента tiler, он указывает, что результат вывода будет осуществлен лишь для изображений, полученных путем обработки компонентом Tiler.
- FileWriterComponent – класс, отвечающий за запись результата в видеофайл. Компонент в отдельном потоке считывает данные очереди для записи, сделано это для того, чтобы была возможность равномерной записи видеопотока в файл.
- DisplayComponent – класс, отвечающий за вывод результата на экран.

1.2.9 Диаграмма классов pipeline

Для того, чтобы связать все компоненты в одну цепочку, а также задавать им цикличный вызов метода do во всех компонентах необходим



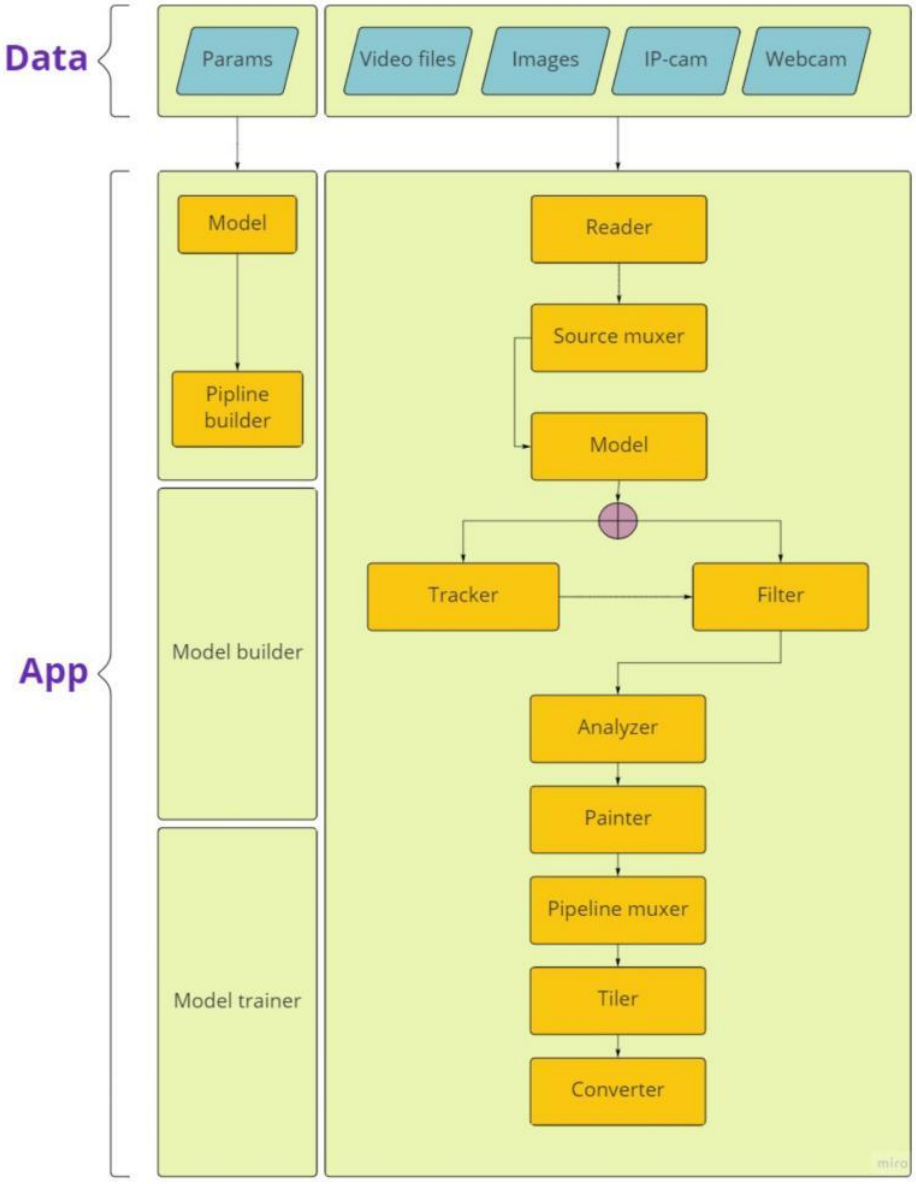
контейнер, структура которого представлена на рисунке 1.9.

Рисунок 1.9 – Диаграмма классов pipeline

- Pipeline – класс, отвечающий за проверку, построение, работу и закрытие компонентов. Он передает данные между компонентами, а также считывает сигналы, такие как Mode.
- Mode – класс перечисления, нужен для задания сигнала состояния, эти сигналы передают компоненты, а pipeline считывает и обрабатывает.

- Logger - класс логирования, позволяет выводить информации о состоянии в консоль и файл, а также указывать уровни и настраивать встроенный Logger из модуля config языка Python.

Пример конвейера в рамках архитектуры



На рисунке 1.10 показан пример конвейера в рамках готового приложения

Рисунок 1.10 - Архитектура конвейера в приложении

Итоговый список используемых технологий

На основе архитектуры и требований был составлен список технологий.

1. язык программирования Python 3.7;
2. фреймворк opencv-python 4.5.5.64;
3. фреймворк pytorch 1.11.0;
4. часть пакета pytorch проект torchvision 0.12.0;
5. библиотека numpy 1.21.5;
6. GStreamer 1.20.2
7. unittest 26.4
8. Git
9. github.com

1.3 Оптимизация скорости работы алгоритмов

Благодаря тому, что фреймворк реализован блочно, скорость работы алгоритмов не зависит от фреймворка. Скорость обработки самим фреймворком этих алгоритмом занимает тысячные секунды.

1.4 Тестирование качества, скорости, отказоустойчивости системы

1.4.1 Скорость работы

Для проверки скорости работы проведено сравнение с фреймворком, частично выполняющего те же самые задачи. Фреймворк DeepStream[1] от компании Nvidia позволяет строить конвейеры с различными нейронными сетями. Для тестирования были выбраны различные архитектуры для задач классификации, детекции и сегментации. Результаты тестирования можно посмотреть на рисунке 1.10.

[1] https://developer.nvidia.com/deepstream-sdk

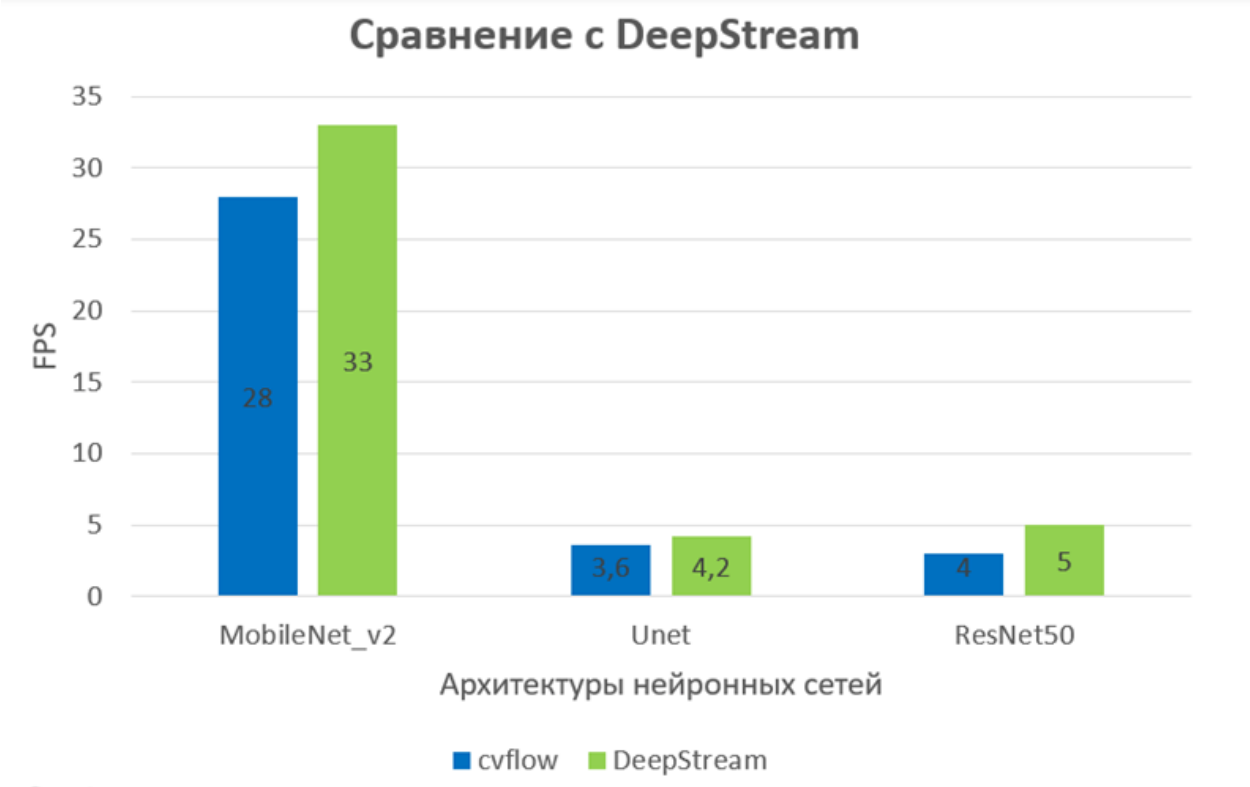


Рисунок 1.10 - Диаграмма сравнения скорости работы функционально идентичных конвейеров фреймворков DeepStream и cvflow

Учитывая, что DeepStreeam написан на компилируемом языке C, но при этом не соответствует некоторым требованиям для наших целей, результат можно считать хорошим, поэтому производительность полученного фреймворка считаем приемлемыми.

Также на примере архитектуры ResNet50 были протестированы скорости выполнения конвейеров для задач классификации, детекции и сегментации для 1 или нескольких потоков. Результат можно увидеть на таблице 1.1. По этим результатам можно увидеть влияние количества источников на скорость работы конвейеров.

Таблица 1.1 - Таблица влияния количества источников для основных задач cv.

Компоненты	Кадров в секунду

Resnet50 + Компонент сегментации для 1 видеопотока. Разрешение 240 на 320	3 fps
Resnet50 + Компонент сегментации для 2 видеопотока. Разрешение 240 на 320	1.6 fps
Resnet50 + Компонент детекции для 1 видеопотока. Разрешение 240 на 320	4 fps
Resnet50 + Компонент детекции для 2 видеопотока. Разрешение 240 на 320	2.3 fps
Resnet50 + Компонент классификации для 1 видеопотока. Разрешение 240 на 320. Результат на рисунке 3.10	9 fps
Resnet50 + Компонент классификации для 2 видеопотока. Разрешение 240 на 320	6 fps

Можно заметить, что скорость уменьшается в диапазоне от 1.5 до 2 раз. Таким образом появляется задача оптимизации влияния количества источников на скорость работы.

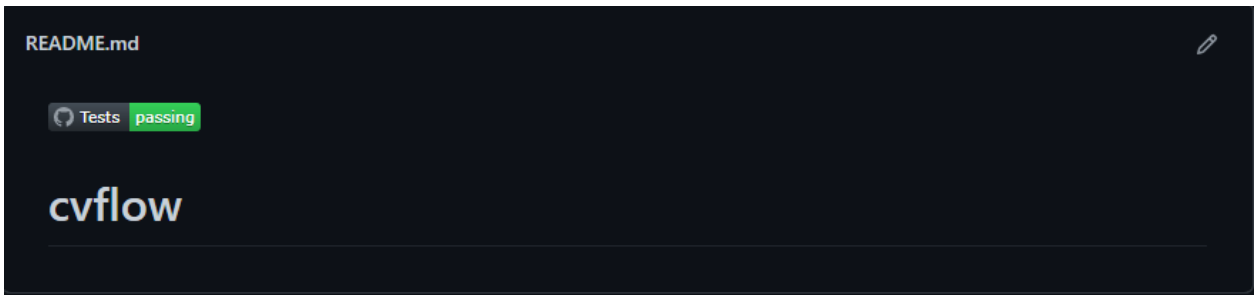
1.4.2 Отказоустойчивость

На данный момент задача отказоустойчивости заключается в непредвиденных сбоях в работе компонентов конвейера. Для того, чтобы при постоянной работе системы не было поломок, каждый компонент сохраняет информацию, поступившую до обработки в текущем компоненте. Далее, если внутри компоненты произойдет сбой, то в следующую компоненту пойдут данные, сохраненные до поломки, таким образом конвейер не перестанет работать, а продолжит функционировать. Если откажет источник ввода видеопотока, то обработку проходят последние успешно сохраненные кадры.

Конвейер не станет запускаться только в том случае, когда ни один из источников ввода не работает успешно.

1.4.3 Качество

Для постоянной проверки качества, код библиотеки был покрыт unit тестами. Проверены входные и выходные данные для каждого компонента, а также граничные условия. Для поддержания корректности работы была



написана конфигурация, которая автоматически запускает тестирование библиотеки на github, тем самым указывая на работоспособность библиотеки. На рисунке 1.11 показано, как выглядят полностью пройденные тесты, указывая на качество последней версии.

Рисунок 1.11 - Файл ReadMe, указывающий на полностью пройденные тесты на github

1.5 Поиск ошибок и некорректной работы

В качестве методологии разработки фреймворка была использована спиральная модель. Для поиска ошибок было использовано unit тестирование, с помощью которого были найдены и исправлены некоторые ошибки. Часть ошибок была найдена при имплементации новых алгоритмов в фреймворк, но была также успешно исправлена. Далее, по мере работы и развития фреймворка все ошибки будут исправляться таким же образом.

2. Поиск, сбор, анализ, разметка и обогащение данных

2.1 Создание требований по данным для обучения алгоритмов

При реализации проекта мы стремимся создать универсальную библиотеку, которая будет применима не только к узкому классу задач, на решение

которых была обучена, но и к новым задачам, что учитывалось в требованиях к данным.

- Для тренировки алгоритмов распознавания положения объекта требуется набор, содержащий минимум 5000 фотографий предметов, сделанных с двух камер. К нему должна быть доступна разметка в виде карт глубины или карт разницы.
- Для оценки качества распознавания положения объекта в пространстве необходимо минимум 5 коротких видео с одной камеры, а лучше с нескольких, где в камеру показывают “шахматную доску” фиксированного размера для калибровки камеры. Необходимо определять, где находится объект, поэтому двигаться он должен рядом с линейкой.
- Для тренировки и тестирования алгоритмов нахождения дефектов требуются наборы фотографий разных поверхностей, в том числе дерева, металла, бетона и мрамора.
- Для решения задачи распознавания номера объекта необходимо найти/создать соответствующий набор данных. Он должен представлять собой аннотированные изображения (либо кадры видео). Аннотация должна включать в себя информацию о локализации номеров (координаты прямоугольника минимальной площади, содержащего номер), а также наборы символов, характеризующие сами номера. Кроме этого, нужно учесть географическую привязку, то есть получить набор данных с изображениями автомобилей на русских номерах.
- Для тренировки распознавания номеров автомобилей требуется размеченный набор минимум из 10000 изображений. Для распознавания номеров остальных объектов планируется применить дообучение, поэтому требуется 100-500 изображений объектов каждой категории (вагоны, бруски металла, посылки). На объектах номер может

располагаться как на специальных табличках, так и на бумаге, а может быть написан краской на самом объекте.

- Для определения размеров объектов требуется разметка изображений. В данном разделе предполагается определение размера кузова машины, что для дообучения существующих моделей требует минимум по 25 изображений каждой категории.

2.2 Используемые наборы данных

В проекте выделено несколько основных задач: определение положения объектов в пространстве, распознавание дефектов, распознавание номеров объектов, анализ перемещения объектов, определение размеров объектов. По всем направлениям был произведен поиск открытых наборов данных, его результаты представлены в таблице.

Ссылки на используемые наборы данных будут представлены в разделах отчета, соответствующих задачам.

Таблица 2.1. Наличие в открытом доступе данных по задачам проекта и необходимость сбора и разметки. Цветом отмечена степень необходимости сбора данных.

Задача	Существующие данные	Количество и оценка	Сбор
Определение положения объектов в пространстве	Есть только 2 датасета и те не по задаче.	2 плохих	Обучающий и тестовый
Распознавание дефектов	Есть наборы фотографий металла, бетона и ткани и других	38, много	Не нужен

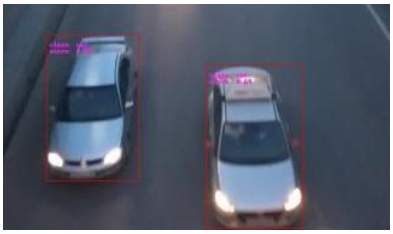
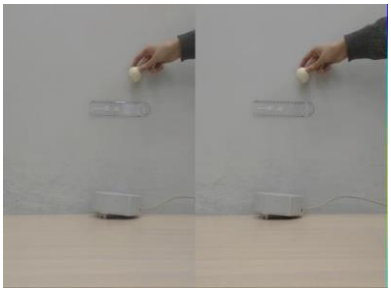
	поверхностей с дефектами и без.		
Распознавание номера объекта	Для распознавания номеров автомобилей есть данные. Других объектов – нет.	2 + 2, частично есть	Обучающий и тестовый для не автомобилей
Анализ перемещения объектов	Видео в открытом доступе. Трансляции на Youtube с камер видеонаблюдения. Набор данных KITTY.	2	Не нужен
Определение размеров объектов	Достаточных данных для определения размеров т/с и для детекции предметов на конвейере нет.	2 частично	Обучающий и тестовый по разным задачам

2.3 Сбор и разметка данных

По результатам поиска данных в открытых источниках было собрано и размечено 5 наборов изображений, их описание и размер представлены в таблице 2.2. Более подробно собранные и размеченные данные будут описаны в разделах, соответствующих задачам. На этап 2 намечена задача по определению перемещений в ложных условиях, уже начата разработка алгоритмов по этой теме и собран набор виде №5.

Таблица 2.2. Описание собранных и размеченных наборов данных.

№	Описание	Число объектов	Пример изображения

1	Фотографии вагонов для распознавания их номеров с разметкой номеров (баундинг бокс номера + цифры/буквы).	40 + будет собрано еще 60	
2	Фото машин для классификации типа кузова (и определения его высоты). Классы: легковая, кроссовер, фургон, автобус, грузовик, велосипед, другое т/с, человек.	802	
3	Синтетический набор изображений для определения положения объектов в пространстве и расстояний можду ними.	10000	
4	Набор видеозаписей движения небольших объектов.	10 видео по 15-20 секунд	
5	Набор видеозаписей автомобильных дорог в разное время суток. Будет использован позже, при адаптации алгоритмов к сложным условиям среды.	10 видео по 2 минуты	

3. Алгоритмы определения положения объектов в пространстве и расстояния между ними

В рамках данного этапа требуется создание алгоритмов определения взаимного положения объектов нескольких в пространстве с высокой точностью по трём осям (X-Y-Z) и расчёта расстояния между ними. Для определения перемещения оси глубины выбрано использование стереозрения, т.е. одновременной съёмки видео с двух камер и построение карты глубины. На основе знания расстояний до объектов и характеристик камеры (фокусное расстояние, размер матрицы, разрешение итогового изображения) возможно определение точного расстояния между несколькими выбранными объектами с помощью триангуляции.

Исследование источников показало, что существующие модели определения расстояний в основном используются для автоматического управления автомобилями и рассчитаны на определение расстояния до больших объектов (автомобиля, дерева, человека), соответственно имеют допустимую для этих случаев погрешность 0.05-0.3 метра на расстоянии 4 и более метров, имеют значительную погрешность при определении близких объектов и не всегда учитывают мелкие объекты.

В рамках первого этапа был реализован программный модуль библиотеки, предназначенный для определения расстояния до заданных пользователем объектов для последующего расчёта расстояния между ними (в метрических величинах) относительно расстояния до объектов. Основным преимуществом разработанного программного модуля является спецификация на определении расстояния до небольших произвольных объектов, на которых не производилось предварительное обучение, возможность быстрого масштабирования.

Данный функционал реализован благодаря использованию стерео-зрения, т.е. съемки с двух камер одновременно. Системы, работающие с использованием одной камеры, определяют расстояние до известных объектов, на которых модели были обучены. Система, работающая с использованием двух камер, реагирует на смещение объектов относительно друг друга, что позволяет отслеживать расстояние до произвольных объектов. Особенностями использования технологии является:

- необходимость использования двух одинаковых по характеристикам камер;
- необходимость калибровки камер для построения карты глубины;
- сложность разметки данных.

Для калибровки камер, т.е. получения внутренних параметров (фокусного расстояния, оптического центра, коэффициента радиального искажения объектива) и внешних параметров (поворотом и смещением) камеры относительно некоторой мировой системы координат) разработан программный модуль средствами открытой библиотеки OpenCV. Для калибровки необходимо предоставить снимки с используемых камер с изображением шахматной доски фиксируемого размера. Данное действие позволяет получить параметры для уточнения дальнейшего определения расстояния. Также из изображения получаются данные о матрице камеры, необходимые для преобразования расстояния, измеряемого в пикселях, в метрические величины.

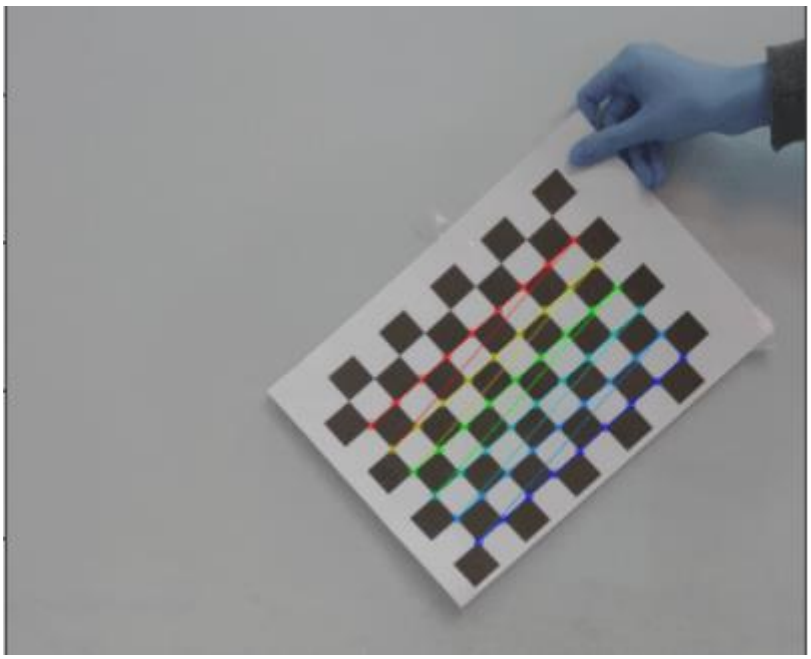


Рис. 3.1 – Визуализация калибровки камеры с помощью шахматной доски.

Исследование литературы позволило выбрать несколько открытых моделей стерео зрения в качестве базы. В качестве критерия выбора использовалась метрика 3-px error, т.е. процент пикселей, более чем на 3 пикселя удалённых от своего целевого положения [1]. По такому критерию были отобраны модели MobileStereoNet [2] и CDN-GA-Net [3]. MobileStereoNet реализует легковесные блоки на основе MobileNet, адаптируя их к задаче стерео зрения. CDN-GA-Net применяет к задаче стерео-зрения функцию потерь на основе расстояния Вассерштейна [4]. Модели протестированы на снятом датасете с двух камер, результат которого показал необходимость дообучения на собственном для получения приемлемых результатов для небольших произвольных объектов.

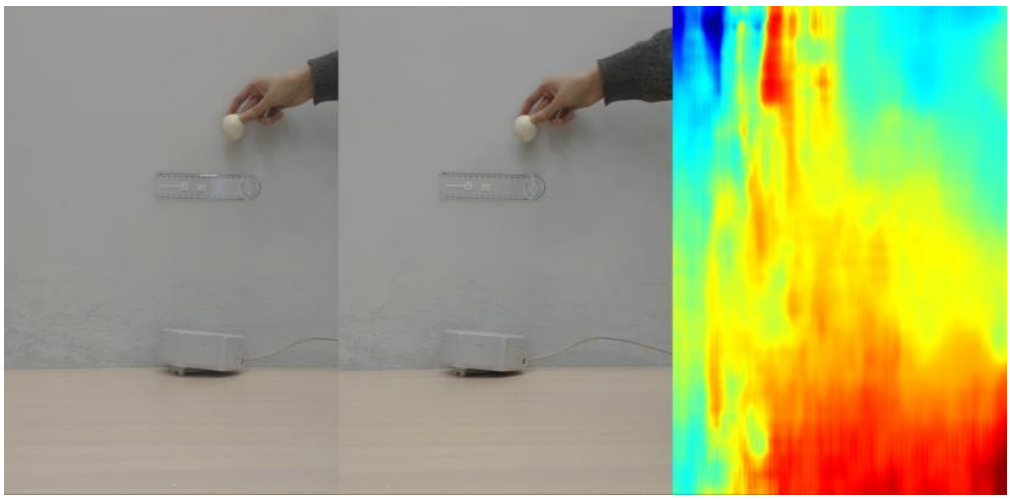


Рис. 3.2 – Результат предсказания без дообучения

Существуют несколько датасетов для стерео-зрения. Самыми известными из них являются: датасет KITTI [1], представляющий собой набор с уличных съемок окружающей среды, предназначающийся для использования в первую очередь в задачах, связанных с беспилотными автомобилями; синтетический датасет Sceneflow [5], представляющий собой набор созданных в среде 3D-моделирования Blender объектов. Исследование последнего показали, что его составляют несколько частей: модели мультфильмов, аналогичные KITTI уличные сцены и случайные объекты, помещенные в виртуальную среду. Более близким к требуемой задаче является датасет Sceneflow, поэтому было принято решение о создании аналогичного датасета, соответствующего целевой задаче.

Для сбора синтетического датасета была выбрана среда разработки blender, в которую были помещены более 20 моделей инструментов, устройств и бытовых вещей. Размер объектов был нормирован до 1 метра в физическом аналоге. На один кадр помещается от 3 до 8 случайных объектов, расположение, угол поворота и цвет объектов так же случайно. В качестве фона выбраны фотографии промышленных помещений. Объекты, помещенные на изображения, имеют размер от 10 до 50 сантиметров и расположение на расстояние 0.5-2 метра от камеры. Далее карта глубины,

полученная с помощью blender, была переведенаа в карту разницы по формуле:

$$disparity = (baseline * focal\ length) / depth,$$

где disparity – целевая переменная значения разницы между изображениями, baseline – константа, коэффиециент трансляции координат изображения в координаты реального мира, focal length – фокусное расстояние, depth – глубина.

Рис. 3.3 – Пример генерации изображений и карты разницы.

Было сгенерировано 1000 изображений со случайными объектами. На данном датасете было произведено дополнительное обучение моделей с последующим тестированием. Задачей тестирования является сопоставление полученных расстояний до объектов с помощью моделей с реальными расстояниями.

Рис. 3.4 - Пример результатов работы дообученной модели MobileStereoNet

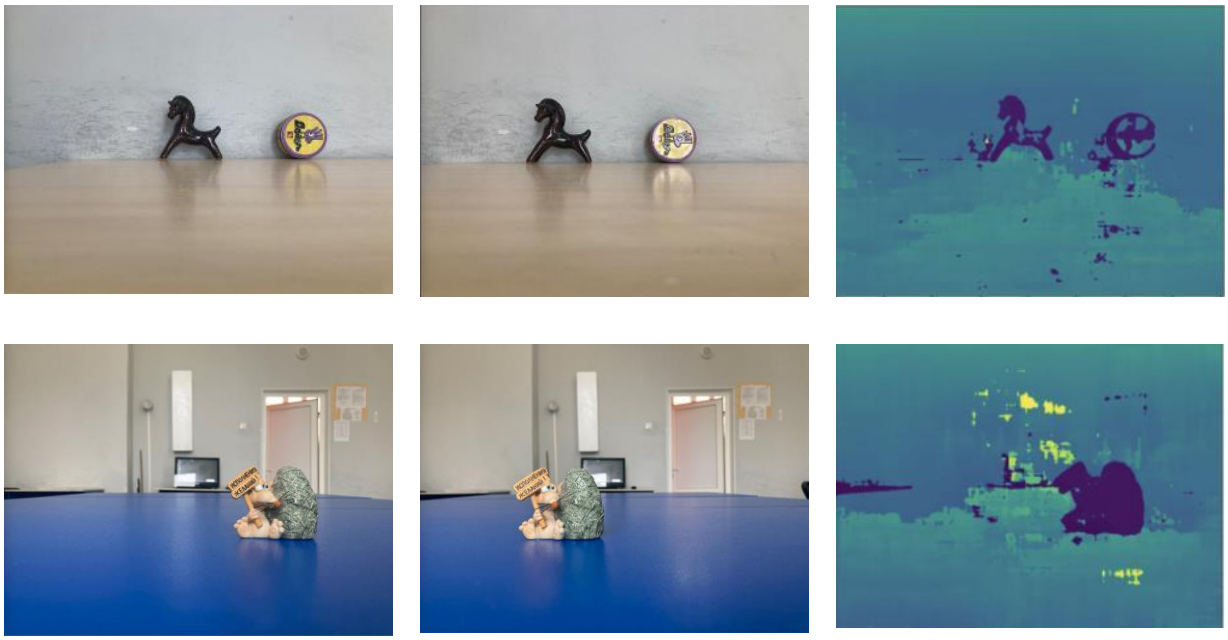


Рис. 3.5 - Пример результатов работы дообученной модели CDN-GaNET

В результате дообучения видно, что целевые объекты имеют чёткие контуры и точно выделяются на фоне окрущающей среды. Снимаемые объекты находятся на расстоянии 50-70 сантиметров от камеры и имеют размеры 5-20 сантиметров. Последним этапом вычисления является перевод значение полученной разницы между кадрами в метрические величины по формуле

$$depth = (baseline * focal\ length) / disparity,$$

где disparity – переменная значения разницы между изображениями, baseline – константа, коэффициент трансляции координат изображения в координаты реального мира, focal length – фокусное расстояние, depth – целевая переменная расстояния до объекта. На основе этого были полученные следующие значения расстояния до объектов:

№	Модель	Высота объекта	Расстояние до объекта	Реальное расстояние до объекта	Ошибка
1	MobileStereoNet	0.1 м	0.63 м	0.6 м	30 мм
2	MobileStereoNet	0.2 м	0.64 м	0.6 м	40 мм
3	MobileStereoNet	0.1 м	0.68 м	0.7 м	20 мм
4	MobileStereoNet	0.1 м	0.35 м	0.25 м	100 мм
5	CDN-GaNET	0.1 м	0.62 м	0.6 м	20 мм
6	CDN-GaNET	0.2 м	0.56 м	0.6 м	40 мм
7	CDN-GaNET	0.1 м	0.75 м	0.7 м	50 мм
8	CDN-GaNET	0.1 м	0.4 м	0.25 м	150 мм

В результате тестирования моделей видно, что ошибка определения расстояния для объектов, расположенных в 0.6-0.7 метрах от плоскости камеры составляет менее 7% относительно расстояния (0.02-0.03 метра). Значительная ошибка формируется для объектов, расположенных в пределах 0.5 метров до камеры, что может свидетельствовать о недостаточном количестве аналогичных данных в обучающем датасете.

Расстояние между объектами таким образом определяется по формуле:

$l = d * L/f - L$, $L = L_{px} * baseline$

где l – искомое расстояние между объектами в сантиметрах, d – расстояние до объектов, L – расстояние между объектами в м, f – фокусное расстояние, baseline – количество px в одном метре. Данные о расстоянии до объектов получены с помощью съемки с двух камер, отслеживаются на изображении с левой камеры, так как предсказывается глубина изображения с левой камеры, согласно датасету Sceneflow [5]. Два объекта выделяются с помощью выделения пользователем и рассчитывается расстояния между крайней правой точкой левого объекта и крайней левой точкой правого объекта.

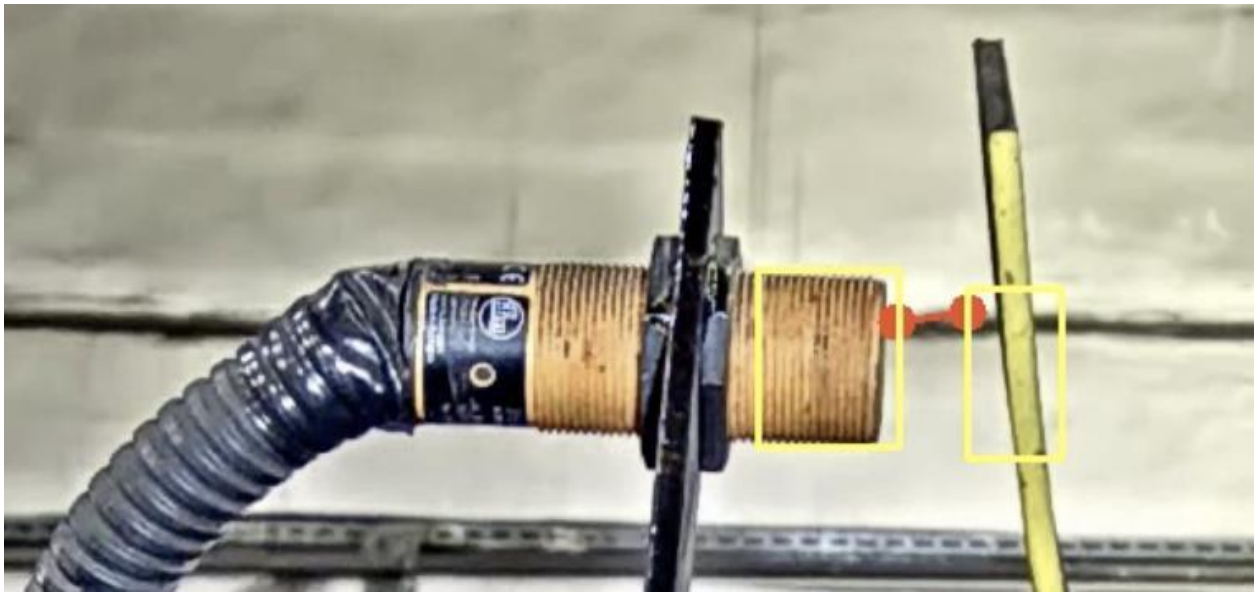


Рис. 3.6 - Пример расчёта определения расстояния в пикселях между двумя объектами

№	Расстояние до объектов	Предсказанное расстояние между объектами	Реальное расстояние между объектами	Ошибка
1	70 см	113 мм	101 мм	12 мм
2	70 см	93 мм	101 мм	7 мм
3	60 см	105 мм	100 мм	5 мм
4	60 см	94 мм	100 мм	6 мм
5	60 см	95 мм	100 мм	5 мм

Таким образом расстояние между объектами можно определить с точность до 10 мм на расстоянии до 0.7 м.

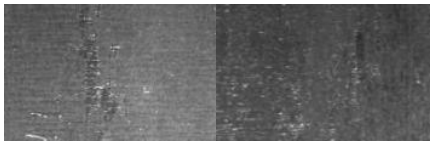
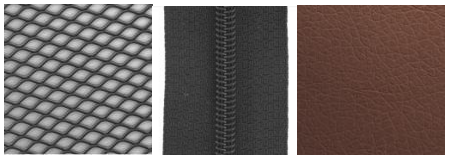
4. Алгоритмы распознавание дефектов и прочих образований на материале

В рамках данного этапа требуется разработать алгоритм, который бы распознавал дефекты на любом материале любых указанных пользователем видов. Планируется разработать и реализовать универсальный подход к поиску дефектов, то есть не по отдельной модели на каждый материал (дерево, сталь, бетон, мрамор и так далее, материалом множество), а одну модель для всех материалов. Качество, достигаемое такой моделью, будет меньше, но она не будет требовать дообучения под каждую конкретную задачу, хотя мы не исключаем эту возможность и добавляем соответствующий функционал.

Архитектура универсальной модели основана на классической сети VGG19, признаки которой хорошо отражают паттерны на изображении. Используются 4 сверточных слоя предобученной сети VGG19, затем идут два обучаемых сверточных слоя, а затем два полносвязных слоя.

Были проанализированы методы нахождения дефектов и отобраны данные для этой задачи. Эксперименты проводились на фотографиях дерева[2], металла[3], мрамора[4] и разных поверхностей и предметов[5], статистики по этим наборам изображений представлены в таблице 4.1.

Таблица 4.1. Статистики наборов данных с дефектами на материалах.

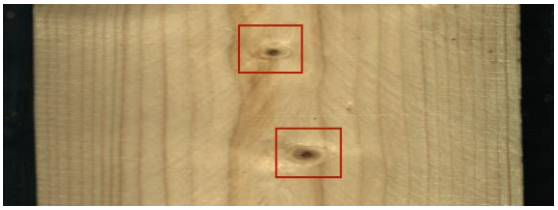
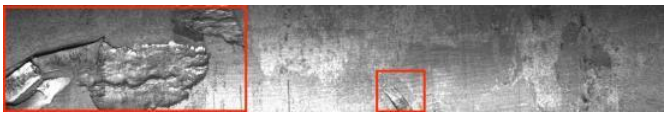
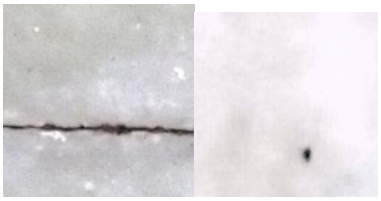
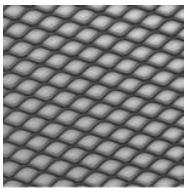
Набор данных	Классы	Число фото	Пример
Дерево	6+1 (Без дефектов + 6 типов дефектов)	6652	
Сталь	4+1	21500	
Мрамор	4+1	2937	
Разное	15 видов поверхностей	6612	

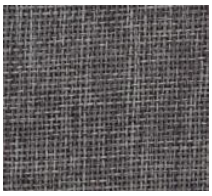
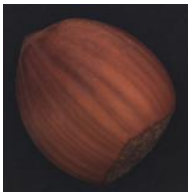
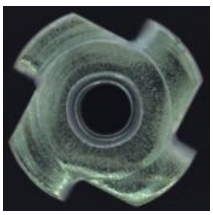
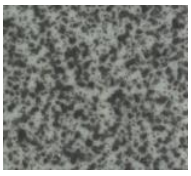
Сеть была обучена на тренировочных частях данных о дефектах дерева и стали и протестирована на их тестовых частях по достигаемой точности (метрике

[2] https://zenodo.org/record/4694695#.Yp3HVWBBzgR
[3] https://www.kaggle.com/datasets/pronomuos/steel-patches
[4] https://www.kaggle.com/wardaddy24/marble-surface-anomaly-detection-2
[5] https://www.kaggle.com/thtuan/mvtecad-mvtec-anomaly-detection

F_1). Также сеть была протестирована на разных типах поверхностей. Результаты представлены в таблице 4.2.

Таблица 4.2.

Данные	F_1-мера	Пример изображения
Дерево	0.82	
Сталь	0.72	
Мрамор	0.71	
bottle	0.60	
grid	0.58	
zipper	0.51	

cable	0.52	
carpet	0.55	
hazelnut	0.56	
leather	0.56	
metal_nut	0.54	
pill	0.57	
screw	0.53	
tile	0.66	

toothbrush	0.50	
transistor	0.57	
capsule	0.50	

Как можно видеть из таблицы, точность нахождения дефектов на дереве и металле больше 0,7, как и было заявлено в гранте. Дефекты на мраморе визуально похожи на дефекты на дереве, модель не обучалась на этом наборе изображений и никогда не видела поверхность мрамора, тем не менее смогла предсказать наличие и класс дефектов с точность более 0,7. На наборе изображений разнообразных предметов и поверхностей достигается точность 0,5-0,6, что вызвано их непохожестью на обучающую выборку.

Ожидаемая точность по F_1-мере достигнута, но не для всех видов поверхностей, что требует дальнейшего улучшения модели. В рамках следующего этапа планируется доработать архитектуру модели и определять, сколько нужно данных для получения точность более 0,7.

5. Распознавание номеров объектов

В рамках данного этапа требуется создание алгоритма распознавания номеров автомобилей по видеопотоку. Такой алгоритм состоит из двух частей: часть детекции (позволяющая локализовать номер на кадре) и часть генерации

(генерирующая набор символов по части кадра). Кроме этого, важно, чтобы алгоритм позволял делать распознавание номеров автомобилей на дорогах общественного пользования в реальном времени.

Анализ различных источников в интернете позволил найти несколько соответствующих требованиям из раздела 2.1 наборов данных, из которых был выбран лучший[6] (с точки зрения качества разметки и количества примеров). В нем содержится 25000 изображений для обучения модели.

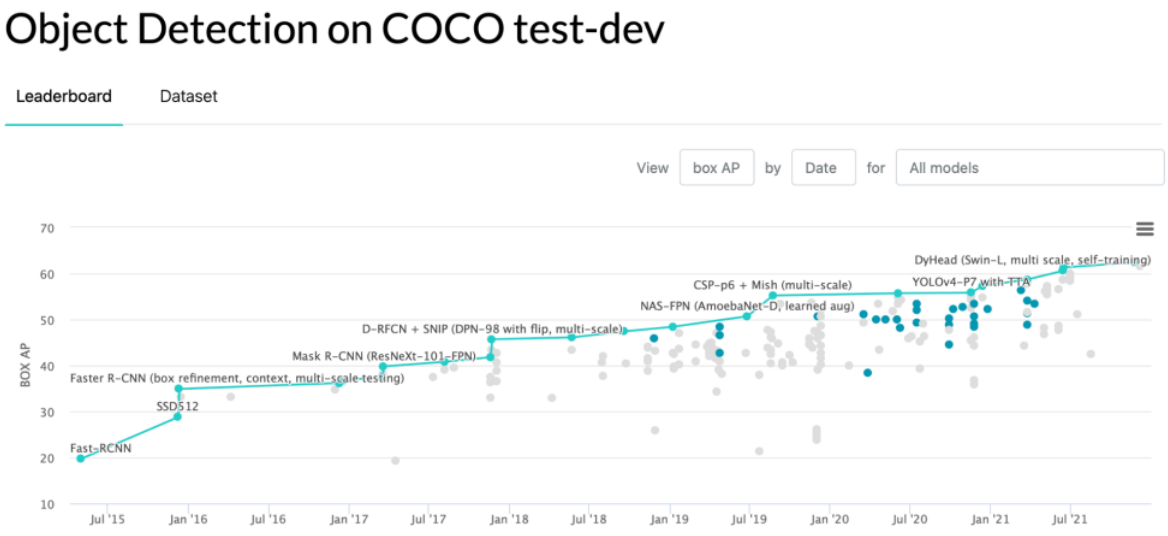
Для выбора модели детекции был проведен анализ литературы, который позволил выявить лучшие модели в настоящее время с учетом скорости работы:

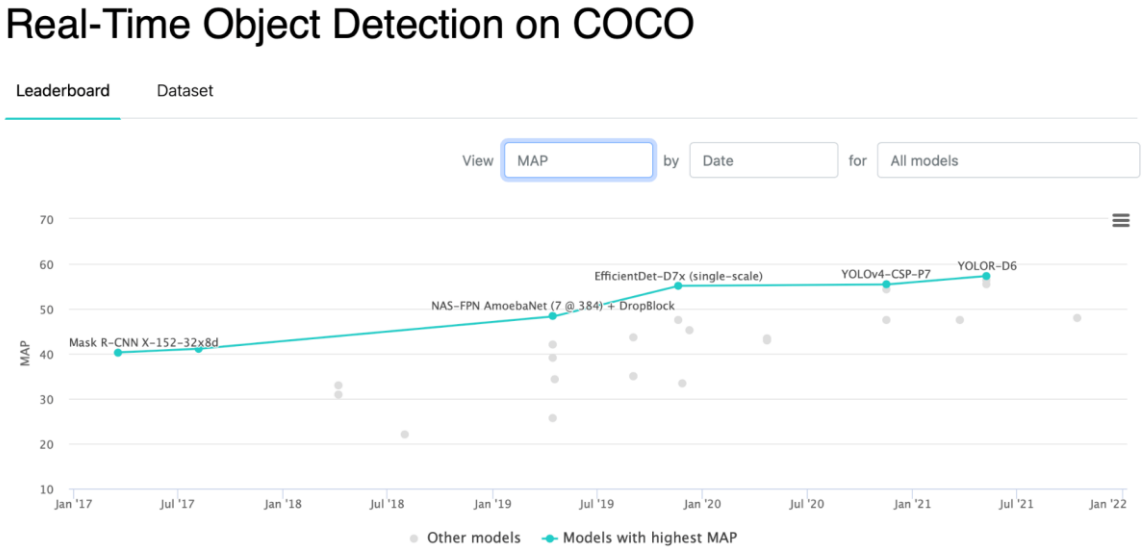
Модель	Протестировано на COCO	Комментарий
Mask RCNN [11]	box AP 39.8	Довольно старая, но проверенная временем модель, весьма медленная, что не очень хорошо для процессинга каждого кадра видео (~5fps)
YOLOv4-v5 [10, 15]	box AP ~55.8	Работают в реальном времени, много дискуссий насчет того, какая лучше в плане точности (v4 или v5), v5 не от darknet, но работает еще быстрее.
DyHead [12]	box AP 60.6	Достаточно новая модель от Microsoft, работает на self-attentions (swin transformer). Two-stage – то

[6] https://www.kaggle.com/c/car-plates-ocr-made

		есть, вероятно, работает не слишком быстро. Из-за новизны есть потенциальные проблемы с дообучением.
Soft Teacher [13]	Box AP 61.3	Новая модель от microsoft, основанная на Swin transformer.

В качестве лучшей модели была выбрана модель YOLOv5, она обладает сравнимой точностью детекции с SOTA моделями и при этом впечатляющей скоростью работы.





Аналогично был проведен анализ литературы для анализа SOTA моделей для распознавания текста по изображению (Optical character recognition или OCR). Существует множество решений, использующих архитектуру Transformer [14], но все они работают достаточно медленно, поэтому лучшей моделью была определена модель CRNN.

Полученный набор данных был преобразован для дообучения модели YOLOv5, само дообучение происходило в течение 10 эпох (после появлялся эффект переобучения). Ниже можно увидеть результат работы дообученной модели детекции.

Далее, дообученная модель YOLOv5 была использована, чтобы составить набор данных, содержащий только изображения номеров (crops). Аннотации для номеров были взяты из исходного набора. На полученном наборе данных была дообучена модель CRNN. Точность распознавания (доля случаев, когда

распознанный номер совпал полностью с действительным) на тестовой выборке составила **96%**.

Для изображения выше модель генерирует следующий результат:
[['C861KT190', 'C931KM76', 'C441AK76']]

Модели YOLOv5 (детекции) и CRNN (распознавания) были объединены в компоненту реализуемой библиотеки, выполняющую распознавание номеров автомобилей по видеопотоку и встроены в архитектуру библиотеки.

6. Алгоритмы для анализа перемещения объектов, захвата и подсчета объектов

В рамках реализации первого этапа проекта был реализован программный модуль библиотеки, предназначенный для подсчёта легковых автомобилей, грузовиков, велосипедов, мотоциклов, пешеходов или любых других объектов, заданных пользователем.

Основным преимуществом разработанного модуля является возможность построения масштабируемой эффективной промышленной системы в кратчайшие сроки, способной решать широкий спектр задач как на дорогах общественного пользования и в торговых центрах, так и на КПП, цехах и конвейерных лентах заводов. Такое удобство использования обуславливается предусмотренным функционалом для разметки пространства, дообучения моделей детекции семейства, интегрированной предобученной системы детекции и трекинга объектов, а также простотой совместного использования с остальными компонентами библиотеки.

Возможность использования в любой среде, помещении или в уличных условиях, обуславливается возможностью задания областей и потоков с помощью графического интерфейса, работающего следующим образом:

пользователю выводится на экран первый кадр видео или любой кадр с камеры наблюдения, после чего он может начертить любое количество линий, предназначенных для подсчёта объектов в потоке, проходящем через нарисованную линию, или же область произвольной формы и заштриховать её для определения находится ли тот или иной обьъект в заданной области или нет. После задания линий, необходимо нажать кнопку сохранить, в результате чего заданные линии будут преобразованы в конфигурационный файл, указываемый при запуске детектора. Интерфейс и результат работы программы задания конфигурации области анализа продемонстрированы ниже, на рисунках 6.1 и 6.2.

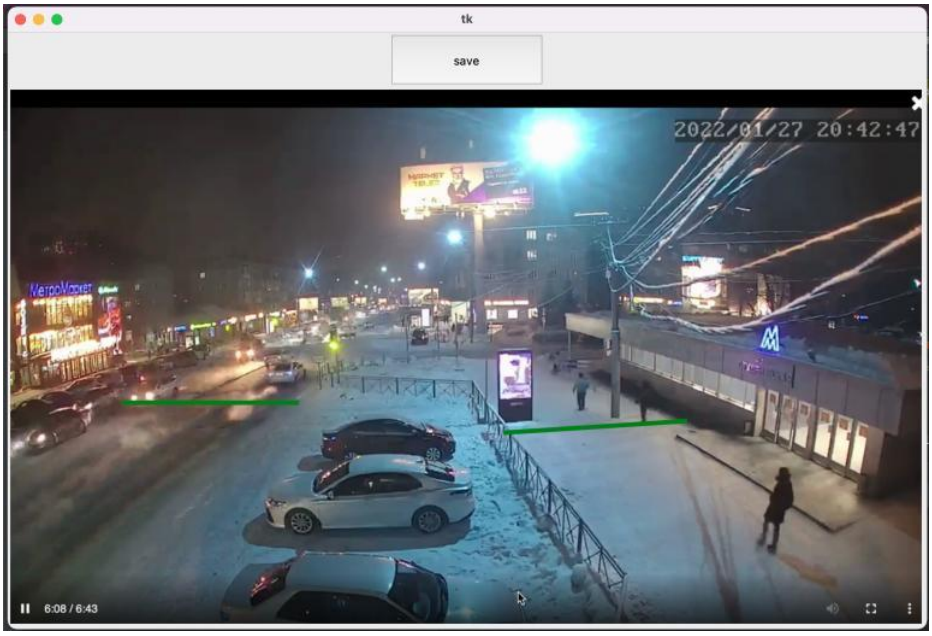


Рис. 6.1, Интерфейс задания конфигурации области

Рис. 6.2, Пример заданной области

Разработанный функционал может быть использован вместе с любым алгоритмом детекции и трекинга объектов, однако по умолчанию для детекции объектов используется предобученная модель YOLOv5 [15], а для отслеживания распознанных объектов алгоритм корреляционного мультитрекинга [16]. Использованные решения демонстрируют высокую точность. Необходимость использования трекинга обуславливается возможностью ускорения работы системы путём уменьшения нагрузки на аппаратную часть пользователя, и, тем самым, удешевления процесса разработки более специализированных решений при несущественной потере точности детекции.

В процессе работы системы анализа потоков на экране отображается информация о различных объектах, посчитанных системой, демонстрируются рамки объектов, позволяющие оценить точность детекции и трекинга объектов. Для разных типов объектов рамки отображаются в разных цветах. При пересечении линии объектом происходит его учет в системе, а также может быть подан звуковой или текстовый сигнал в консоли. При

необходимости задания более сложного поведения при пересечении объектом линии, разработанная система может быть использована как API.

В ходе первичного тестирования реализованного модуля была продемонстрирована высокая точность и надежность в ночных, дневных и заводских условиях. Процесс работы программы продемонстрирован на рисунках 6.3 и 6.4.

Рис. 6.3, пример определения заезда в заданную область

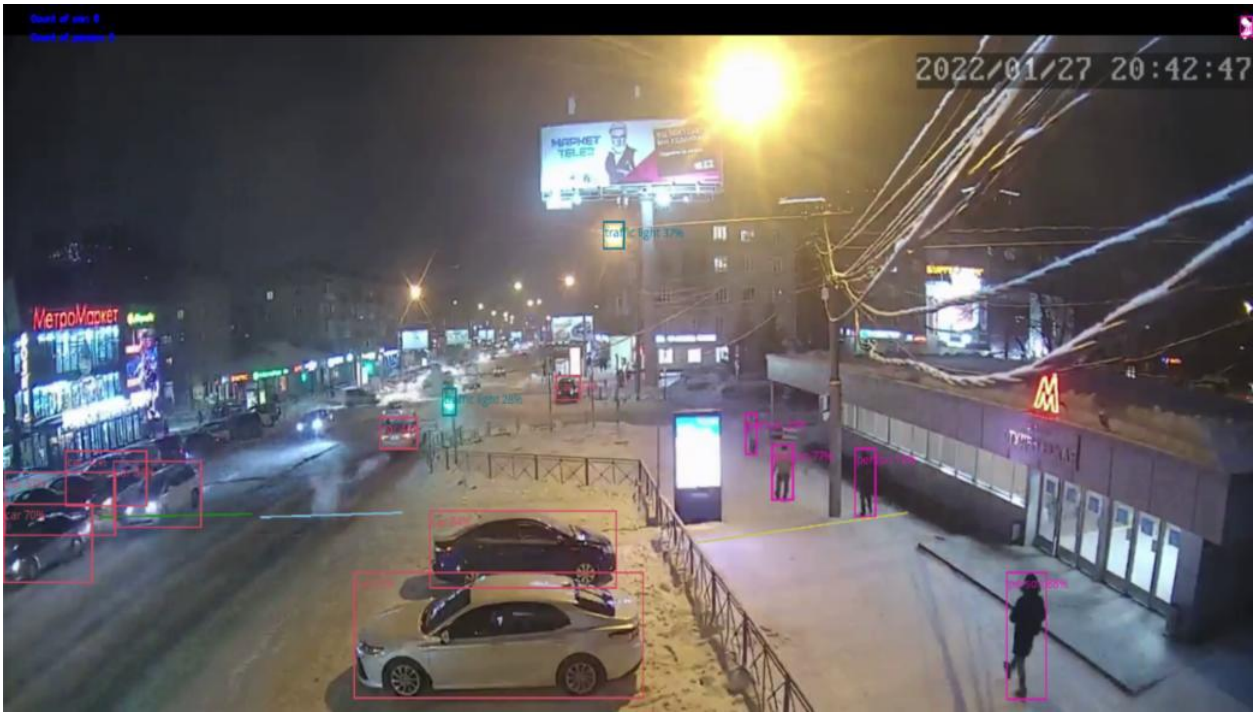


Рис. 6.4, пример подсчёта ТС и пешеходов в потоках

7. Алгоритмы определения размеров объектов

По итогам разработки модуля, описанного в главе 6, была получена система для анализа перемещения объектов, однако этот модуль не позволяет получать информацию о самих объектах. Вместе с тем, наиболее важным применением этой системы в промышленных условиях является именно анализ передвижения различных транспортных средств, поэтому в рамках проекта был также реализован алгоритм для определения размеров кузовов транспортных средств.

Разработанный алгоритм позволяет классифицировать транспортные средства по типу кузова и, таким образом, определять их размеры. Это может быть необходимо при анализе поставок на заводах, фильтрации т/с, которые могут угрожать безопасности персонала или при проектировке контрольно-пропускных пунктов, однако использование этого модуля не ограничивается приведёнными ситуациями. Процесс работы модуля продемонстрирован на рисунке 7.1. Точность классификации при IoU 0.5 составила 0,962.

Рисунок 7.1, пример определения размеров кузова

Гранулометрия

Гранулометрический анализ – распределение камней руды по крупности, характеризующееся процентным выходом от массы или количества кусков руды. Гранулометрический состав – важный физический показатель качества руды. Состав руды влияет на дальнейшее дробление и измельчение, так как при преобладании крупных камней мельницу необходимо убыстрять, при преобладании мелких – замедлять для повышения эффективности работы агрегата.

Стандартом для оценки распределения частиц является ситовой анализ, применяемый к забору пробы (1 $м^3$) руды из контейнера и дальнейшего

просеивания сквозь сита разной размерности. Точность данного метода низкая, так как он работает на допущении, что распределение камней в пробе и контейнере совпадают. Дорогостоящими методами, требующими дополнительного оборудования, являются инфракрасная трёхмерная съемка, трёхмерное лазерное сканирование, флуоресцентный рентнегорадиометрический анализ. Существующие промышленные решения являются пропиетарными. Продукты для гранулометрического анализа конфигурируются для конкретного производства в связи с отличающимися условиями. Чаще всего предприятия предлагается готовый аппаратно-программный комплекс, сконфигурированный под конкретное предприятие, однако, это не всегда допустимо.

Автоматизация построения распределения камней по размеру может быть реализована с помощью применения компьютерного зрения к существующим данным, которым является видео с конвейера [9].

Осуществление гранулометрического анализа происходит следующим образом:

1. получение изображения с камеры;
2. предобработка изображения;
3. определение камней на конвейере;
4. дополнительный шаг: отслеживание камней;
5. построение распределения.

Цель **предобработки изображений** – подготовить исходные данные изображения для извлечения необходимой нам информации, повысить вероятность определения именно тех объектов, которые нам необходимы. Исходные кадры с конвейера требуют предобработки, так как имеют низкую яркость, и объекты сложно различимы. Дополнительную сложность вносит однородность объектов. Задача предобработки - выделить объекты для дальнейшего анализа.

Предварительная обработка кадра (рис. 7.1) включает в себя:

1. Кадрирование изображения для выделения ленты конвейера. Кадрирование является наиболее простым шагом предобработки и позволяет не выявить лишних деталей.
2. Уменьшение шума путём применения размытия по Гауссу. Фильтр Гаусса часто применяется для сглаживания изображения, удаления шумов и улучшения структуры изображения.
3. Эквализация гистограммы. Данная операция позволяет повысить яркость и контраст изображения. Цель эквализации - преобразование гистограммы таким образом, чтобы она соответствовала нормальному закону распределения.

Рисунок 7.1 - Процесс предобработки изображения

Основная проблема исследования – **определение камней на конвейере**. Для решения этой задачи необходимо произвести сегментацию изображения, выделив пиксели, принадлежащие камням или фону (к фону так же относятся элементы конвейера). Существуют два ключевых подхода к сегментации: на основе выделения границ и на основе метода выращивания регионов, а также смешанный подход. Простые методы не могут полностью решить задачу детектирования камней на плоскости, однако, их использование эффективно в качестве этапов сегментации. Метод водораздела относится к методам выращивания регионов и является одним из самых популярных и эффективных методов сегментации.

Суть метода водораздела заключается в обнаружении маркеров [6], которые будут являться точками локального минимума, точек, находящихся на возвышенности и точек, находящихся на склоне, с которых вода разных цветов будет сливаться к центру водоёма.

Алгоритм работы метода водораздела следующие:

a. В местах локального минимума образуются отверстия, через которые вода начинает заполнять поверхность.
b. Если вода с двух сторон возвышенности готова слиться в один бассейн, то необходимо установит перегородку между бассейнами.
c. Когда вода заполнит всё изображение, то алгоритм можно остановить.

Использование водораздела часто приводит к сильной сегментации, поэтому чаще всего используют водораздел на основе маркеров. Маркер являет собой точку локального минимума, на основе которого будет выбран бассейн. Маркеры возможно расставлять как вручную, так и автоматически.

Рисунок 7.2 – Процесс расстановки маркеров и вычисление маски

Оценка точности определения камней на конвейере была произведена с помощью матрицы ошибок (таблица 1), в которой сравнивались маски сегментации, построенные вручную и с помощью нейронной сетью. Если пересечение между реальной и сегментированной маской превышало 70%, то сегментация признается истинно положительной (TP, True positive), иначе результат признавался ложноположительным (FP, Falsepositive). Истинно отрицательный результат (TN, true negative) не применим к данной задаче, и на всех изображениях равен нулю.

Выделенный объект классифицировался как ложноотрицательный результат (FN, False negative), если камень был определён как фон, т.е. не

определён вообще. Вычисление происходило на основе 10 случайно выбранных изображений конвейера, прошедших предобработку (размер 256x256 пикселей, 8-бит, градации серого). На основе этих данных были вычислены следующие метрики: точность (1) и полнота (2). Точность позволяет определить долю верно определённых объектов относительно всех объектов.

Таблица 7.1 – Матрица ошибок и метрики метода водораздела

	Реальное положение		Точность	Полнота
Вычисленное	Камни	Фон	60%	84%
Камни	TP = 736	FP = 490		
Фон	FN = 134	TN = 0		

Из таблицы 7.1, построенной на базе выборки из 10 изображений, видно, что метод водораздела не обеспечивает высокую точность, но значение полноты, превышающее значение точности, позволяет сделать вывод, что это происходит вследствие выделения большого количества лишних (false positive) объектов, таких как детали конвейера. Таким образом, метод водораздела не подходит для определения камней на конвейере, однако его можно использовать как базовый метод для обучения нейросети.

В качестве нейронной сети выбрана сеть U-Net [7], которая считается одной из стандартных архитектур свёрточных нейронных сетей, используемых для задач сегментации изображений . Особенность данной нейронной сети в том, что она позволяет сегментировать область по классу, т.е., создать маску, позволяющую разделить изображение на несколько классов [8].

Изображения обучающей выборки, полученные с помощью метода водораздела, были модифицированы с использованием графического

редактора. Убраны ошибочно выделенные объекты, такие как детали конвейера, убраны распознанные как крупные камни области песка (ошибка разделения мелких частиц), выделены границы для нескольких камней, распознанных как один, или убраны для камней, распознанных как несколько.

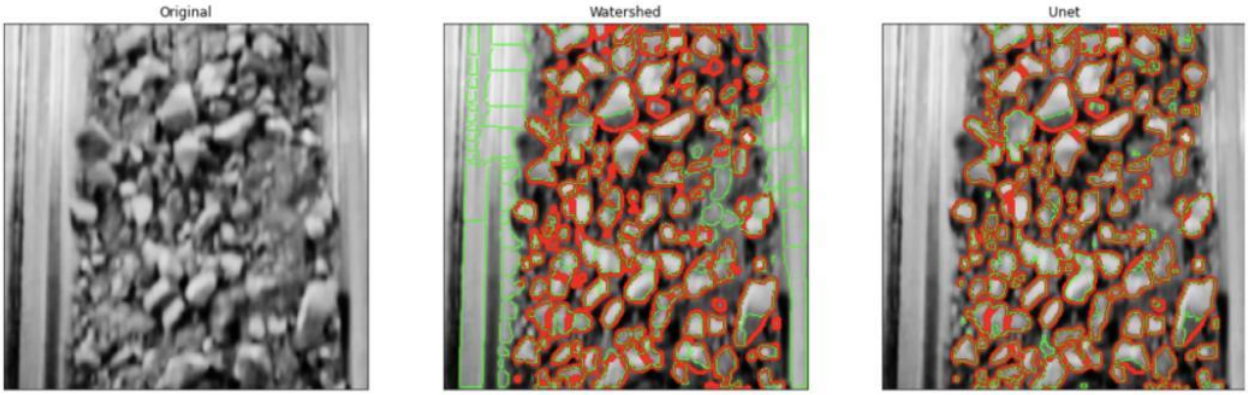


Рисунок 3 – Сравнение результатов сегментации методами водораздела и с помощью нейронной сети

Экспериментальным образом было подобрано время обучение в течение 5 эпох и 2000 шагов в каждой эпохе. Полученное значение функции потерь 0.15, полученная точность - 98.5 обучения. Проверим результат работы алгоритма на тестовой выборке, не пересекающейся с обучающей. Согласно таблице 2, достигнута точность около 95%, что значительно превышает метод водораздела. Относительно классических методов компьютерного зрения, нейронная сеть обучена таким образом, что не выделяет области мелких камней как один большой камень (ошибка разграничения частиц), что является плюсом (рис.3). Однако, при использовании этого метода, как и все методов, использующие двумерные изображения, сложно избежать ошибку профиля, то есть неправильного определения реального размера камни из-за тени.

Таблица 7.2 – Матрица ошибок и метрики нейронной сети

	Реальное положение	Точность	Полнота

Вычисленное	Камни	Фон	94,4%	94,9%
Камни	TP = 947	FP = 56		
Фон	FN = 23	TN = 0		

Временные затраты на вычисление маски одного изображения (аппаратное обеспечение CPU XeonProcessors @2.3Ghz, GPU Tesla K80 12GB VRAM, RAM 13 GB) можно разделить на следующие: метод водораздела – 0,04 с, нейронная сеть – 0,09 с. Согласно построенной матрице ошибок и вычисленным метрикам по 10 изображениям (таблицы 7.1, 7.2) можно сделать вывод, что нейронная сеть Unet обеспечивает точность детектирования, превосходящую метод водораздела в 1,57 раза, а время обработки одного изображения продолжительностью 0,09 с позволяет работать в реальном времени.

Финальным этапом является **построение распределения** по результатам детектирования камней на ленте конвейера мы можем построить диаметр Ферета (опоясывающая рамка, перпендикулярная осям X и Y) для каждого из камней.

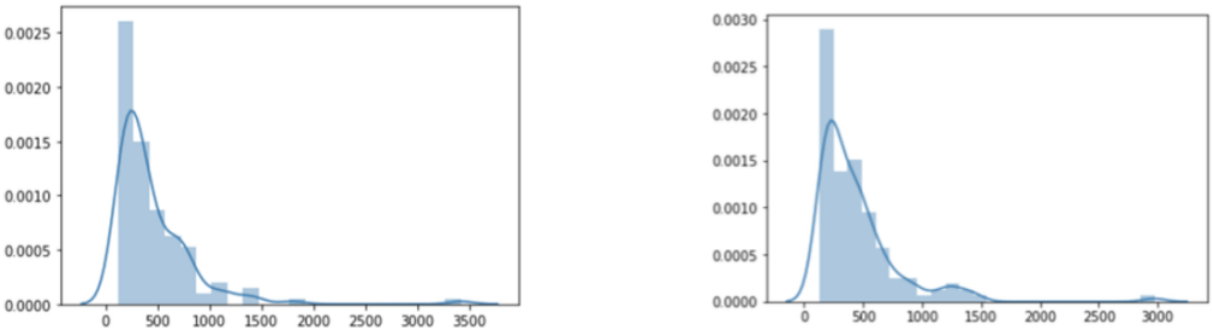


Рисунок 7.5 – Построенное распределение

На основе этих данных мы можем построить гистограмму (рис.5), которая будет показывать распределение камней по размеру на данном участке конвейера. В качестве размера берётся площадь полученного прямоугольника.

На построенном примере распределения мы можем видеть преобладание камней небольшой площади. На основе накопленных данных распределения определяются пороговые значения размеров камней, по которым можно определять крупность для управления мельницей.

В результате данного этапа получен набор методов для автоматизации гранулометрического анализа, в том числе для детектирования камней на конвейере и определения частоты детектирования.

Наилучший результат детектирования показало использование нейронной сети Unet, предназначенной для семантической сегментации изображений. Нейронная сеть была обучена на выборке, созданной с помощью метода водораздела, скорректированного вручную.

Заключение

В первой части проекта была реализована общая архитектура библиотеки, предоставляющей полный инструментарий для построения конвейеров самых часто используемых задач компьютерного зрения. При разработке архитектуры и ее реализации были учтены требования к считыванию видеопотока, работе в реальном времени, решении задач детекции, классификации и сегментации и многие другие. Достигнута масштабируемость и гибкость функционала фреймворка.

Изучены открытые источники данных с учетом требований к данным. По двум из пяти задач найдено достаточное количество данных и соответствующая разметка. Для решения остальных задач собрано и размечено 5 наборов необходимых изображений и видеозаписей.

Разрабатывались алгоритмы распознавания положения производственных объектов в пространстве и расстояний между ними, алгоритмы детекции дефектов дефектов поверхностей, алгоритмы распознавания номеров объектов, алгоритмы анализа перемещений объектов и алгоритмы определения размеров объектов. По всем алгоритмам достигнуты требуемые показатели, кроме анализа местоположения, где указанной в заявке точности не позволили достичь характеристики объективов видеокамер. Итого, заявленная в рамках гранта F1-мера в задачах многоклассовой классификации более 0,7 достигнута. Была заявлена точность локализации 0,5 мм, но не были указаны характеристики камеры и расстояние до объектов. При расстоянии до объектов в 70 см получена точность в 1 см, что является успешным результатом и применено на практике. Точность распознавания номеров объектов более 95%.

Разрабатываемые на данном этапе решения успешно внедрены на предприятиях Евраз, Норникель, КБ Раскат.

Список использованных источников

1. Geiger A. и др. Vision meets robotics: The KITTI dataset // The International Journal of Robotics Research. 2013. Т. 32. № 11. С. 1231–1237.

2. Shamsafar F. и др. MobileStereoNet: Towards Lightweight Deep Networks for Stereo Matching [Электронный ресурс]. URL: https://arxiv.org/abs/2108.09770.

3. Garg D. и др. Wasserstein Distances for Stereo Disparity Estimation [Электронный ресурс]. URL: https://arxiv.org/abs/2007.03085.

4. Panaretos V. M., Zemel Y. Statistical Aspects of Wasserstein Distances // Annual Review of Statistics and Its Application. 2019. Т. 6. № 1. С. 405–431.

5. Mayer N. и др. A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation // 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). : IEEE, 2016.

6. Gonzalez, M., Ballarin, V. Automatic marker determination algorithm for watershed segmentation using clustering // Latin American Applied Research. – 2009, July. – Vol. 39. – P. 225–229.

7. Ronneberger, O., Fischer, P., Brox, T. U-Net Convolutional Networks for Biomedical Image Segmentation // Medical Image Computing and Computer-Assisted Intervention (MICCAI). – Vol. 9351. – Springer, 2015. – P. 234–241. – (LNCS). – URL: http://lmb.informatik.uni-freiburg.de/ Publications/2015/RFB15a (дата обращения 05.05.2022).

8. Liu, X., Ore image segmentation method using U-Net and Res_Unet convolutional networks // RSC Adv. – 2020. – Vol. 10, Is. 16. – P. 9396–9406. — URL: http://dx.doi.org/10.1039/C9RA05877J (дата обращения 14.05.2022).

9. Hamzeloo, E., Massinaei, M., Mehrshad, N. Estimation of particle size distribution on an industrial conveyor belt using image analysis and neural networks // Powder Technology. – 2014. – Vol. 261. – P. 185–190.

10. Bochkovskiy A., Wang C., Liao H. M. Yolov4: Optimal speed and accuracy of object detection // CoRR. — 2020. — Vol. abs/2004.10934. — arXiv : 2004.10934

11. Mask R-CNN / Kaiming He, Georgia Gkioxari, Piotr Doll´ar, Ross B. Girshick // CoRR. — 2017. — Vol. abs/1703.06870. — arXiv : 1703.06870.

12. Dynamic head: Unifying object detection heads with attentions / Xiyang Dai, Yin- peng Chen, Bin Xiao et al. // CoRR. — 2021. — Vol. abs/2106.08322. — arXiv : 2106.08322.

13. End-to-end semi-supervised object detection with soft teacher / Mengde Xu, Zheng Zhang, Han Hu et al. // CoRR. — 2021. — Vol. abs/2106.09018. — arXiv : 2106.09018.

14. Attention is all you need / Ashish Vaswani, Noam Shazeer, Niki Par- mar et al. // Advances in Neural Information Processing Systems / Ed. by I. Guyon, U. Von Luxburg, S. Bengio et al. — Vol. 30. — Curran Associates, Inc., 2017. — Access mode: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

15. Glenn, Jocher ultralytics/yolov5: v6.1 - TensorRT, TensorFlow Edge TPU and OpenVINO Export and Inference / Jocher Glenn. — Текст : электронный // zenodo.org : [сайт]. — URL: https://github.com/ultralytics/yolov5 (дата обращения: 10.06.2022).

16. Visual object tracking using adaptive correlation filters / S. B. David. — Текст : электронный // researchgate.net : [сайт]. — URL: https://www.researchgate.net/publication/221362729_Visual_object_tracking_using_adaptive_correlation_filters (дата обращения: 10.06.2022).