



**МОСКОВСКИЙ АВТОМОБИЛЬНО-ДОРОЖНЫЙ
ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ (МАДИ)**

Кафедра «Высшая математика»

Курсовой проект

По дисциплине **ФУНКЦИОНАЛЬНЫЙ АНАЛИЗ**

на тему:

«Анализ расстояния до движущегося авто по соседней (левой-правой)
полосе»

Выполнил:

Серов П.Г.

Группа 36ПМ

Подпись _____

Руководитель курсовой работы:

д.т.н., к.ф.-м.н., доцент, заведующий кафедрой Яшина М.В.

Курсовой проект защищен с оценкой «_____»

Подпись _____ «_____» _____ 20 ____ г.

Содержание

1. Введение.....	3
2. Исходные данные.....	4
3. Постановка задачи.....	5
3.1. Фокусное расстояние камеры.....	6
3.2. Использование детекторов.....	7
3.3. Ручная проверка.....	7
4. Алгоритм.....	8
4.1. Фокусное расстояние.....	8
4.2. Детекторы.....	10
5. Действия пользователя.....	13
6. Сравнение результатов.....	14
7. Вывод.....	17
8. Список литературы.....	18

1. Введение

С наступлением века информационных технологий транспортная отрасль также нуждается в использовании современных подходов к решению проблем, снижению количества дорожно-транспортных происшествий (ДТП), оптимизацией процесса управления.

В данной работе поставлена задача анализа расстояния до автомобиля, движущегося по соседней (левой-правой) полосе.

Для ее решения, будут рассмотрены различные средства мониторинга дорожно-транспортной инфраструктуры путём обработки видеоряда.

2. Исходные данные

На вход работы алгоритма подаётся видеофайл формата «*.mp4» любой продолжительности.

Например, видеофрагмент «test.mp4» длительностью 00:01:00(ЧЧ:ММ:СС).

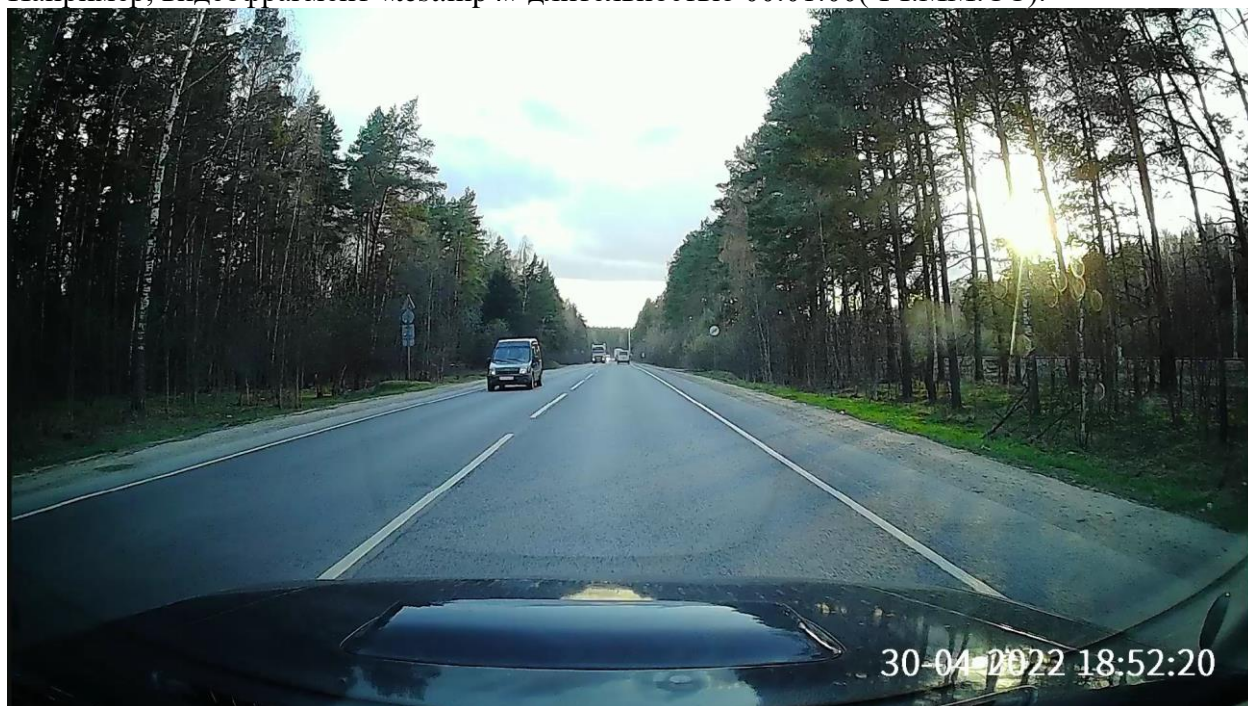
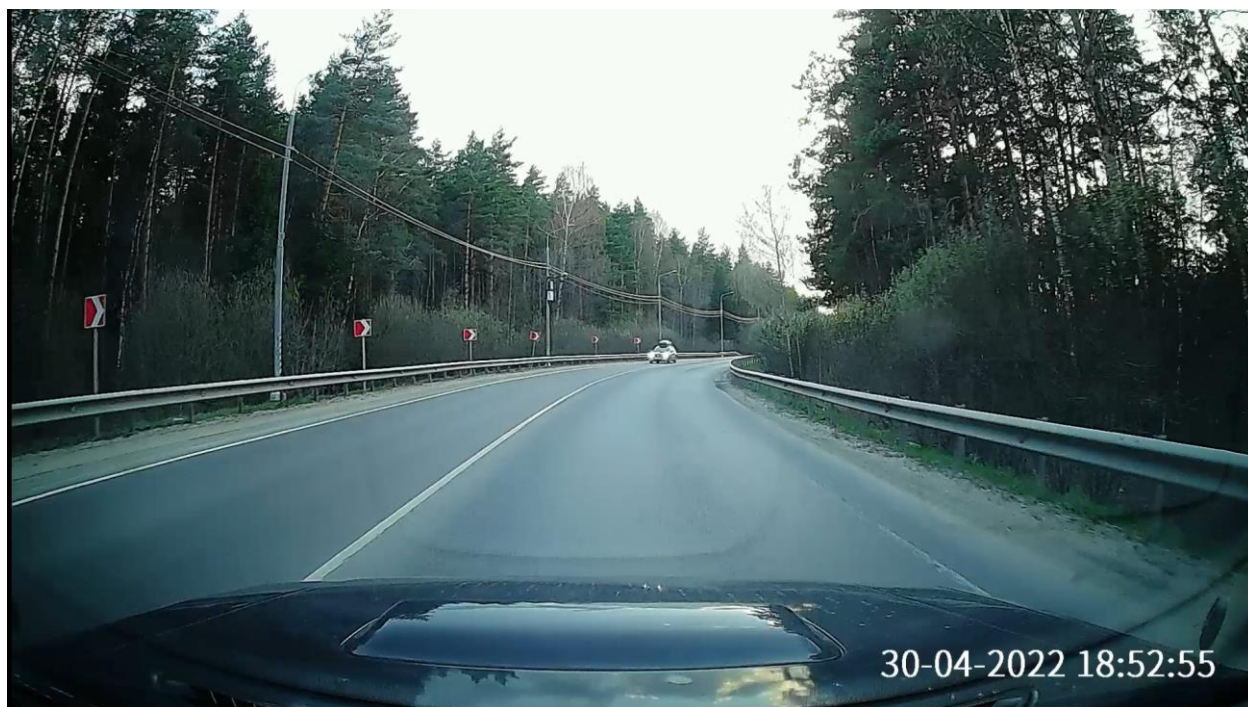


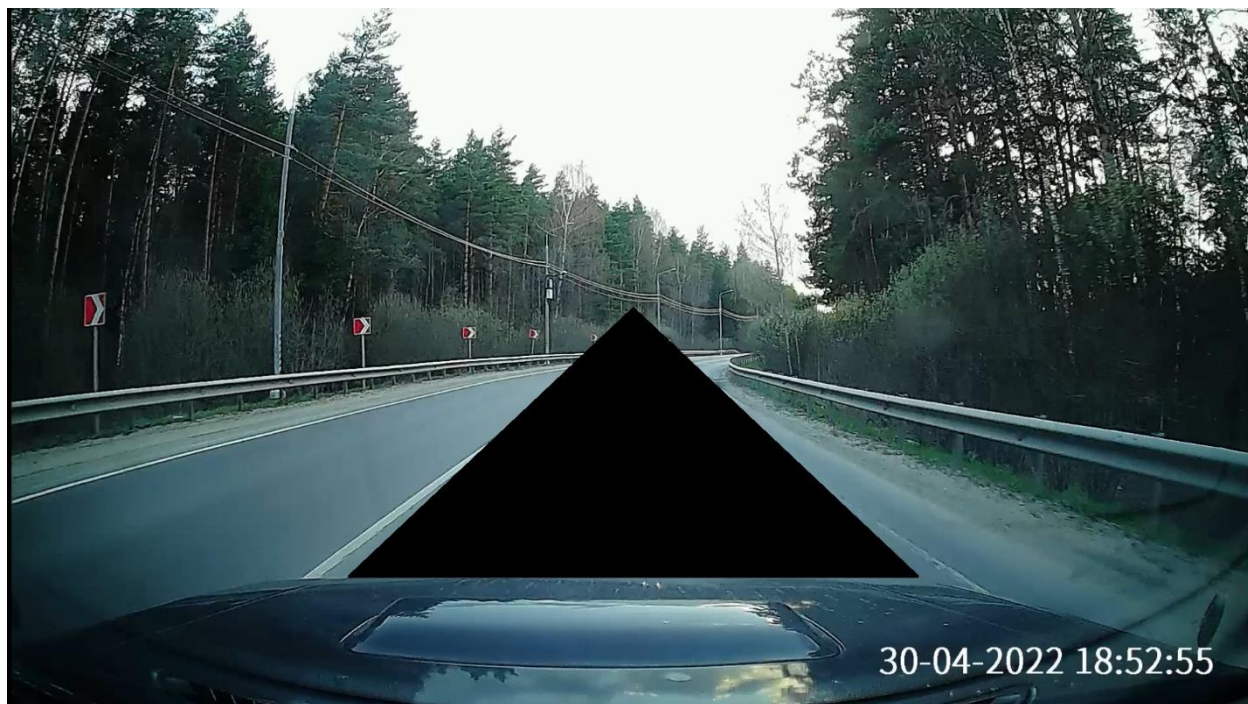
Рис. 1 Пример кадра исходных данных

3. Постановка задачи

Прежде всего, хотелось бы отметить, что, в связи с ограниченностью исходных данных (использование 1 камеры), погрешность, связанная с искажением объектов, удаляющихся от центра кадра, ничтожна мала и, используя маски, перекрывающие полосу, по которой движется автомобиль с камерой или обрезаая видео, мы теряем данные об автомобилях на поворотах, например:



Поворот дорожного полотна



Использование маски

Исходя из вышесказанного, считаю целесообразным использовать информацию о расстоянии до автомобиля на соседних полосах из готовых(обработанных) кадров видео.

Традиционно, для анализа расстояния до автомобилей движущихся по соседним (левой-правой) полосе, используют следующие методы:

- 1) 3D-датчики (Лидары) – прибор, представляющий собой дальномер, т.е. измеряет расстояние путем излучения света (лазера) и замера времени возвращения этого отражённого света на приёмник.
- 2) Использование 2 камер и определения параллакса, т.е. изменение видимого кажущегося положения объекта, наблюдаемого с разных точек и измеренное как угол (или половинный угол) между направлениями от наблюдателя на объект.

Но в данной работе, есть лишь видеоизображение с 1 камеры, поэтому для решения поставленной задачи воспользуемся двумя следующими способами:

3.1. Фокусное расстояние камеры

Первым способом был выбран поиск расстояния до объекта с помощью геометрической оптики.

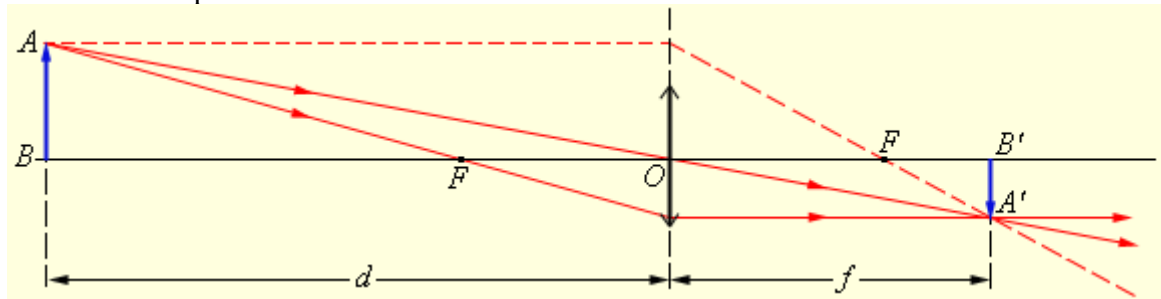


Рис. 2 Построение изображения в тонкой линзе

На этой схеме d — расстояние от линзы до объекта, а f — фокусное расстояние линзы и пусть D — расстояние от линзы до изображения объекта (на матрице или плёнке)

Формула тонкой линзы связывает эти три расстояния:

$$\frac{1}{d} + \frac{1}{f} = \frac{1}{D}$$

Теперь ещё раз посмотрим на оптическую схему: $h = AB$ — это линейный размер объекта съёмки, а $H = A'B'$ — размер его уменьшенного изображения. Нетрудно заметить, что $h = d \tan \alpha$, а $H = D \tan \alpha$ (это следует из свойств прямоугольного треугольника). Подставив эти величины в формулу тонкой линзы, увидим, что $\tan \alpha$ сокращается, и в результате получим следующее уравнение:

$$d = \frac{(f(H + h))}{H}$$

«Неудобная» величина D ушла, а остальные мы знаем или можем легко вычислить. На основе этого уравнения получаем вот такую формулу расстояния до объекта:

Следовательно, в нашем случае, h — это реальная высота автомобиля, H — это размер изображения автомобиля на матрице фотоаппарата. Чтобы получить значения H , воспользуемся следующей формулой:

$$H = \frac{l * c}{100 * S}$$

где l – высота объекта в пикселях, c – длинна матрицы по вертикали, а S – размер изображения в пикселях по вертикали.

Проанализировав метаданные изображения с той же камеры, на которую был снят исходные видеофайл, получим f — фокусное расстояние нашей камеры.

В формулах я использую среднюю высоты ТС:

1,5 м – легковые автомобили

3 м – грузовики и автобусы

А теперь перейдем к более сложному этапу данного способа. А, именно, прежде чем определять расстояние до автомобиля, нам сначала надо найти этот автомобиль. Для этого воспользуемся готовой архитектурой, а именно моделью машинного обучения. Модель является сверточной нейронной сетью и полностью готова к использованию, с достаточной точностью, без необходимости ее обучения.

3.2. Использование детекторов

В качестве второго способа, было решено использовать детекторы, построенные с помощью библиотеки OpenCV, которые регистрируют изменение цвета внутри своих границ. Таким образом, расставив детекторы на определенном расстоянии друг от друга, привязанном к реальным показателям, то можно в момент активации детектора считывать расстояние до соседних транспортных средств.

В данном методе нам не понадобится распознавать автомобили, но нужно будет с помощью градации серого регистрировать отличия пикселей в детекторе, от выбранных заранее (стандартизированных), пикселей асфальта.

3.3. Ручная проверка

Также проведем ручной анализ расстояния до автомобилей, чтобы оценить точность того или иного способа.

4. Алгоритм

Прежде чем использовать любой из двух способов, нужно подготовить исходные данные. Для этого, с помощью цикла, видео разбивается на кадры и покадрово передается в следующие алгоритмы.



Рис.3 Пример кадра №25

4.1. Фокусное расстояние

Покадровая обработка

После разбиения каждый кадр в виде матрицы размерностью (1920, 1080, 3) - изображению разрешением 1920x1080 пикселей в 3 цветовых каналах, подается на вход алгоритму распознавания.

После определения наличия транспортного средства, архитектура отрисовывает специальную коробку, обозначающую границу найденного транспортного средства. Полученные координаты высоты автомобиля в пикселях передаются в формулу подсчета расстояния до объекта. Затем, на кадре отмечается тип транспортного средства и вычисленное расстояние до него.

После окончания работы алгоритма на всем видеофайле, т.е. после прохождения через цикл всех кадров, на экран выводится время работы алгоритма.

Выходные данные

Обработанный кадр сохраняется в заданную папку под названием «Frame + номер кадра».

Также, в отдельный текстовый файл «distance.txt», выводится:

Frame - значения текущего кадра,

Car/Truck - счетный номер и тип ТС на конкретном кадре

Distance - расстояние до ТС

transportCount - количество машин в кадре, чтобы можно было, выбрав любой кадр, открыть соответствующее изображение и наглядно посмотреть результат работы.

Заметим, что для грузовиков и автобусов используется иная формула подсчёта расстояния, чем для легковых автомобилей в связи с их отличными размерами.

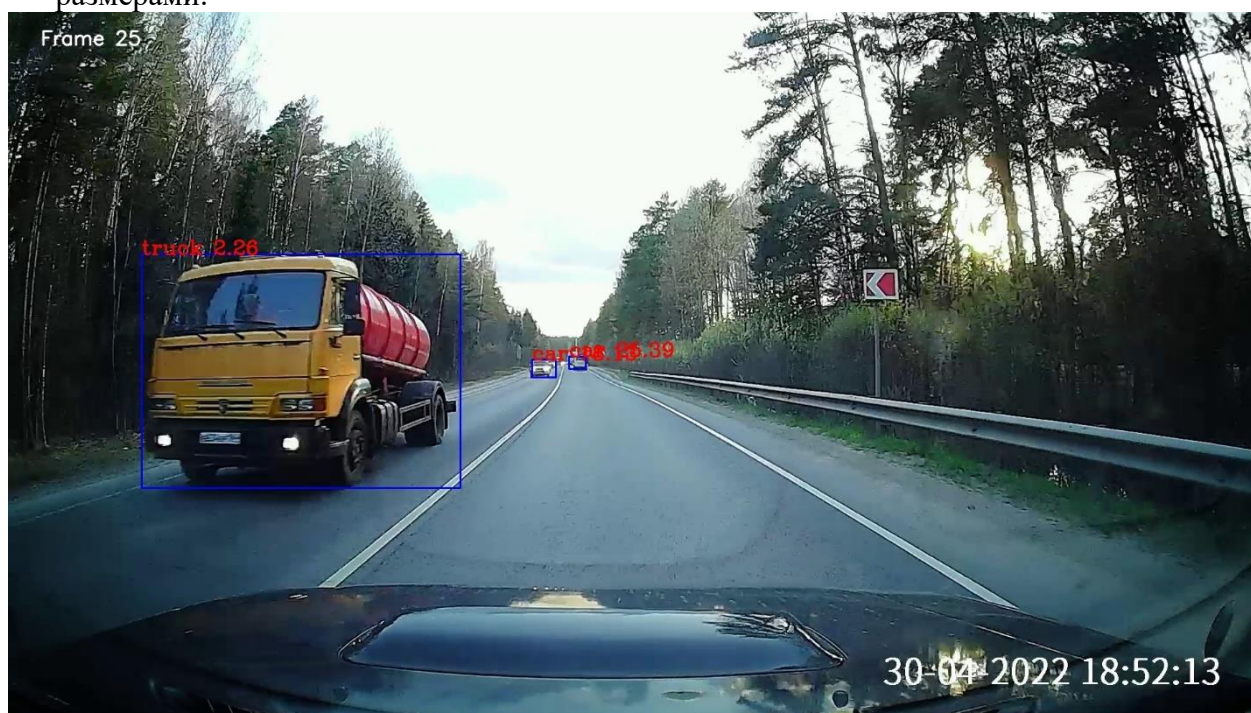




Рис. 4 Пример обработанного кадра № 25

 distance (3) – Блокнот

Файл Правка Формат Вид Справка

```
frame = 25; car = 0; distance = 18.13; transportCount = 1
frame = 25; truck = 1; distance = 2.25; transportCount = 2
frame = 25; car = 2; distance = 25.39; transportCount = 3
```

 distance (2) – Блокнот

Файл Правка Формат Вид Справка

```
frame = 25; car = 0; distance = 18.13
frame = 25; truck = 1; distance = 2.25
frame = 25; car = 2; distance = 25.39
```

Рис. 5 Пример кадра №25, файл distance.txt

Время работы

Тип устройства	Время работы (ЧЧ:ММ:СС)
CPU 2 ядра, 4 потока, 2.6-3.5 GHz	00:57:52
Сервис colab.google.com Использование GPU	00:05:34

Time cost = 0:57:52.397719

Рис. 6 CPU Время работы

Time cost = 0:05:34.804978

Рис. 7 GPU colab.research.google.com Время работы

4.2. Детекторы

Покадровая обработка

Перед запуском пользователь может настроить количество детекторов и их размер.

После запуска, пользователь выбирает на свое усмотрение, куда выставить детектор.



Рис. 8, Рис. 9 Разные размеры детекторов

После выставления детекторов, нажатием кнопки активируется дальнейшая работа программы. Внутри детекторов происходит градация серого и высчитывается средний цвет по пикселям. Если он отличается от начального(стандартного) цвета асфальта хотя бы на 4%, то программа считает, что проехала машина.

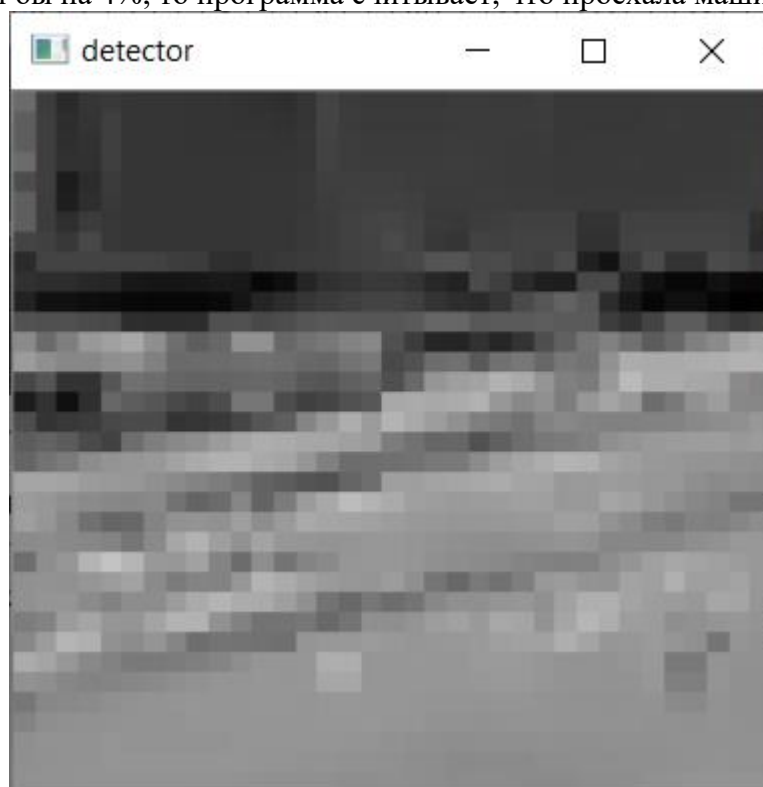



Рис. 10 Градация серого внутри детектора

Выходные данные

На каждом обработанном кадре в текстовый файл «distance_detector.txt» выводится Матрица вида:

F – Номер кадра	№ детектора	№ детектора	№ детектора	№ детектора	№ детектора	№ детектора	Кол-во автомобилей в кадре
--------------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------------------------

 detection_detectors –

Файл Правка Формат

F 1 2 3 4 5 6

0	0	0	0	0	0	0	0
1	1	0	1	0	0	1	3
2	1	0	1	0	0	1	3
3	1	0	1	0	0	1	3
4	1	1	1	0	0	1	4
5	1	1	1	0	0	1	4
6	1	1	1	0	0	1	4
7	1	1	1	0	0	1	4
8	1	1	1	0	0	1	4
9	1	1	1	0	0	1	4

Рис. 11 Пример с 6 детекторами

Расстояния до объекта зависят от того, как пользователь поставил детекторы и как для себя определил их удаленность.

Время работы:

Time cost = 0:03:04.978766

5. Действия пользователя

Функция draw_outputs

- 1) Указать путь к текстовому файлу «FileName.txt», в который будет записываться обработанная информация вида:

```
frame = Номер кадра; car = номер транспортного средства на кадре; distance =  
расчитанная дистанция; transportCount = количество ТС на кадре.  
f = open('distance.txt', 'a')
```

- 2) В зависимости от исходных данных задаем следующие параметры:
AvgCarSize - средняя высота легкового автомобиля в метрах
AvgTruckSize - средняя высота грузового автомобиля/автобуса в метрах.
FLength - фокусное расстояние используемой видеокамеры.

Функция video_to_frames

- 1) Запустить функцию, подав на вход путь к нужному видеофайлу

```
video_to_frames(['test.MP4'])
```


6. Сравнение результатов

Для сравнения результатов я отдельно вывел в файлы «.txt» количество распознанных машин на каждом кадре для двух методов. С помощью библиотеки pandas я считал эти данные из файла и построил 2 графика, а так же сравнил общее количество распознанных машин.

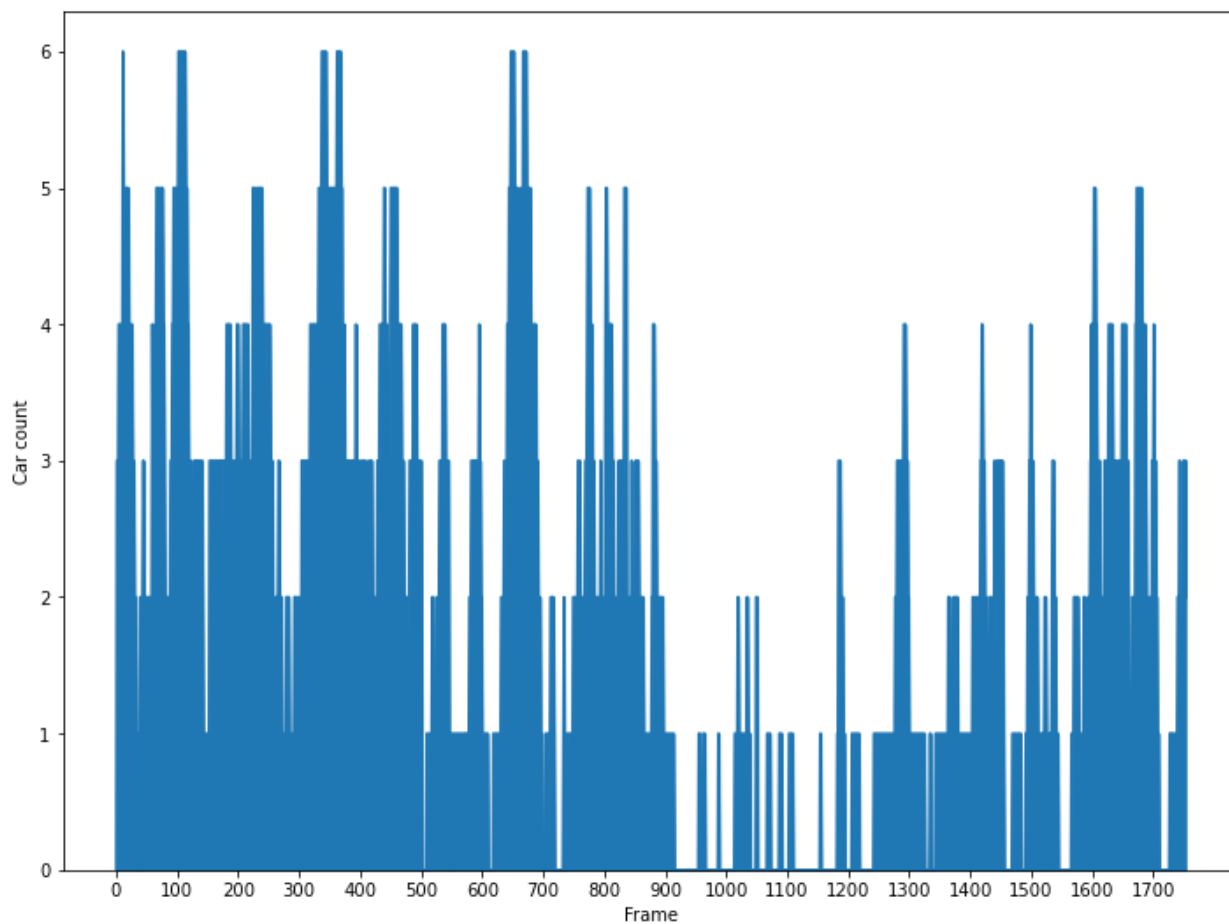


Рис. 12 График по детекторам

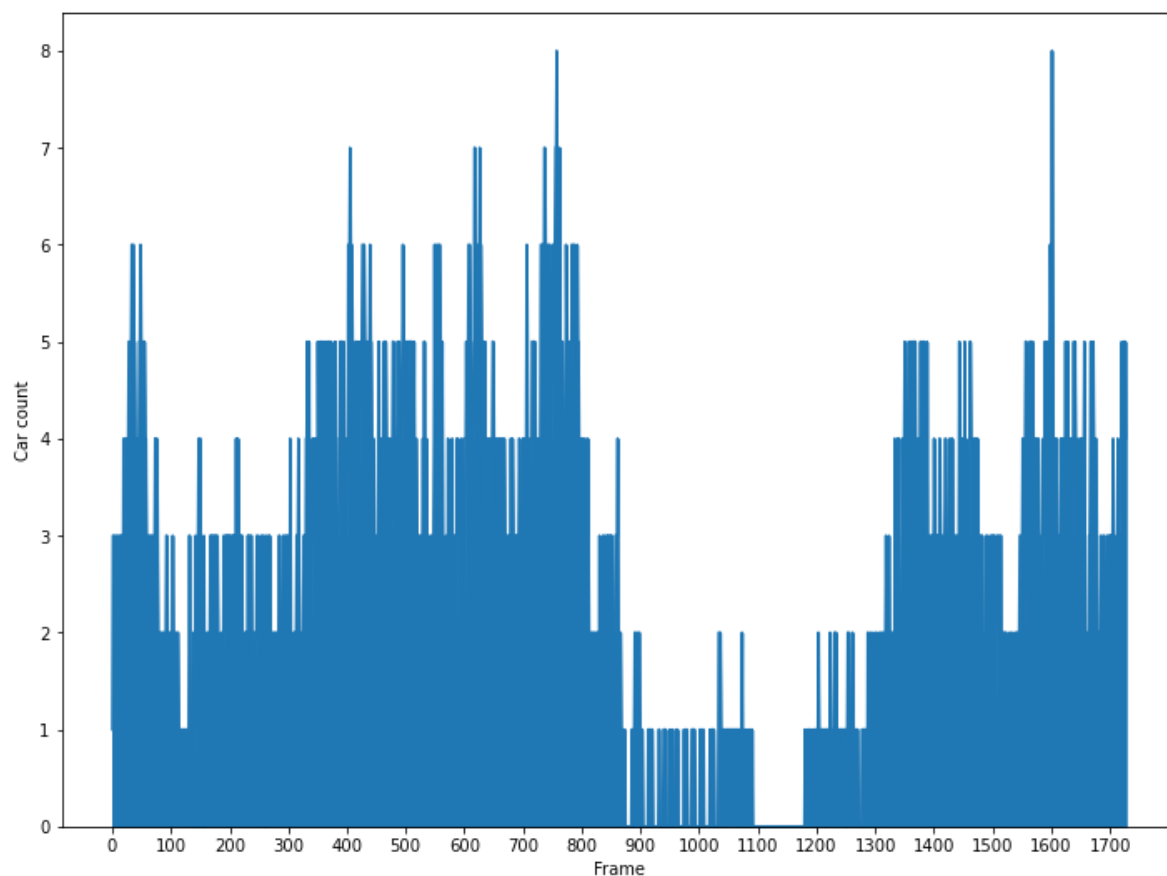


Рис. 13 График YoloV3

Сравнивая графики можем увидеть, что на промежутке от 900-1300 было распознано крайне мало машин и одним и другим способом, что подтверждают кадры до обработки:

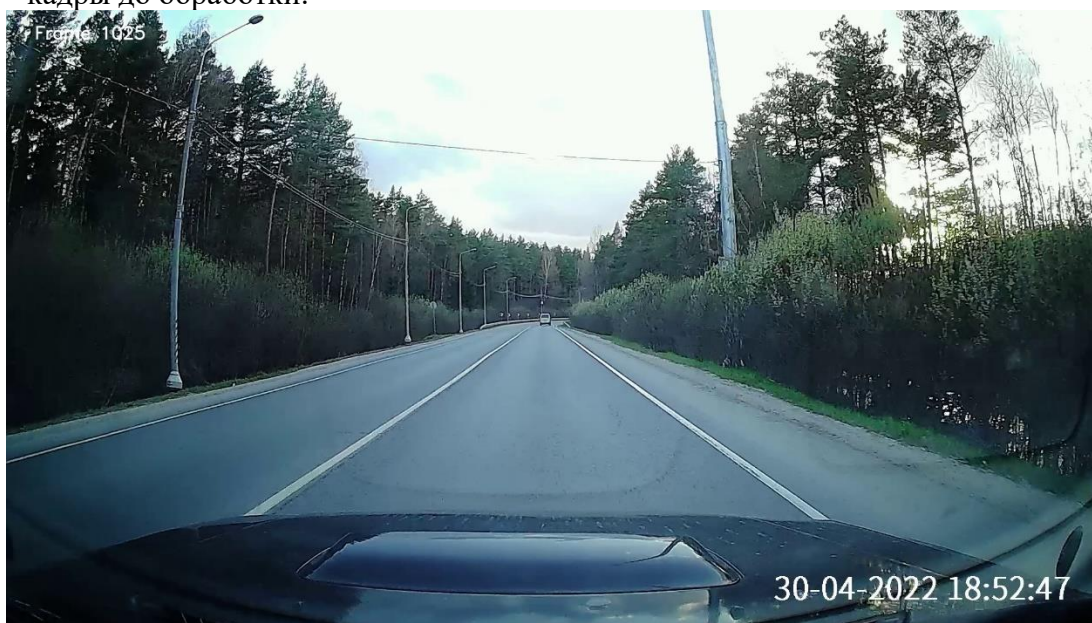


Рис. 14 Кадр №1025

Значит, с распознаванием серьёзных проблем нет.

Общее число распознанных машин.

Детекторы	Машинное обучение
3499	3088

Сравнивая эти результаты, можно заметить, что метод машинного обучения выигрывает у метода, с использованием детекторов, так как положение детекторов определяет сам человек и не всегда точно может определить расстояние и одна и та же машина может одновременно активировать сразу 2 детектора.

Подведём итоги

Метод машинного обучения

Метод машинного обучения хорошо показал себя при распознавании, но, к сожалению, в питоне нет нужных библиотек для создания маски, чтобы ограничить распознавание ТС, движущихся на одной с нами полосе или, стоящим на парковке, поэтому надо обрезать видеоряд до подачи в алгоритм. Но несмотря на это, мы всё равно можем открыть любой кадр нашего видеоряда и получить нужную информацию о расстоянии до соседних автомобилей.

Второй проблемой являются случаи оценивания высоких легковых машин, программой, как «грузовиков». Но мы можем настроить алгоритм на дообучение прямо на рабочих данных, что позволит в будущем лучше распознавать и классифицировать ТС.

Метод детекторов

Метод детекторов показал неплохие результаты при распознавании, но при анализе расстояния из-за того, что пользователь может неточно оценить расстояние, расставляя детекторы, что приведет к неточному анализу расстояния. Также, несколько детекторов могут быть активированы одним и тем же автомобилем на одном и том же кадре.

Исходя из вышесказанного можно сделать вывод, что метод детекторов больше подходит для оценочного суждения расстояния (очень близко / близко / средняя дистанция / далеко / очень далеко).

Можно сделать вывод, что метод машинного обучения показал себя лучше метода детекторов, несмотря на длительность его вычислений, но их как было показано можно нивелировать, всего лишь, имея доступ в интернет.

7. Вывод

Подводя итог можно сказать, что несмотря на то, что метод машинного обучения показал себя неплохо, все ещё сложно точно определить расстояние до объектов, движущихся по соседней полосе и при этом сделать программу максимально универсальной. Поэтому, методы с двумя камерами или с использованием оптических дальномеров превосходит на несколько уровней, но в бюджетном варианте решение нашей задачи имеет место быть.

Также мы рассмотрели различные средства мониторинга дорожного полотна и научились определять расстояние до автомобилей движущихся по соседней(левой-правой) полосе.

8. Список литературы

1. Буйначев, С. К. Основы программирования на языке Python : учебное пособие / С. К. Буйначев, Н. Ю. Боклаг. – Екатеринбург : Изд-во Урал. ун-та, 2014. – 91
2. <https://habr.com/ru/post/556404/> (дата обращения 28.04.2022)
3. Learning OpenCV: Computer Vision with the OpenCV Library / Bradski Gary, Kaehler Adrian, 2008: Изд-во O'Reilly Media
4. Физика. Оптика: учебник / Ю.И. Тюрин, И.П. Чернов, Ю.Ю. Крючков. – Томск: Изд-во Томского политехнического университета, 2009 – 240с