Санкт-Петербургский государственный университет

Кафедра информатики

Зайцев Дмитрий Игоревич

Обнаружение столкновений на парковке со стоящим транспортным средством

Бакалаврская работа

Научный руководитель:
к. ф.-м. н. Салищев С. И.

Рецензент:
к. т. н. Ронжин А. Л.

Санкт-Петербург
2021

SAINT-PETERSBURG STATE UNIVERSITY

Department of Informatics

Dmitry Zaitsev

Detecting collision with a stationary vehicle at parking

Graduation Thesis

Scientific supervisor:
Candidate of Physical and Mathematical Sciences Sergey Salishchev

Reviewer:
Candidate of Sciences Alexander Ronzhin

Saint-Petersburg
2021

Оглавление

Введение

В настоящее время, искусственный интеллект начал пользоваться огромной популярностью, связанной с появлением вычислительных мощностей, которых хватает на конструирование мощных нейросетей.

Нейронные сети применяют для распознавания образов и классификации, кластеризации, предсказания или аппроксимации. Машинное обучение применяется для решения сложных задач, которые требуют аналитические вычисления, похожие на то, что делает человеческий мозг. Примеров использования нейросетей очень много. Так, например, они используются в экономике для предсказания курса валют, в голосовых ассистентах в смартфонах или редакторах фотографий.

Проблема обнаружения и классификации часто используется в компьютерном зрении для распознавания рукописного текста, для распознавания лиц или для детектирования номеров транспортных средств с камер видеонаблюдения. В последнее время стала популярна идея беспилотного автомобиля. Для реализации этой идеи также нужны компьютерное зрение и машинное обучение.

Для беспилотного автомобиля существует множество подзадач такие как обнаружение пешеходов, детектирование и классификация знаков дорожного движения, обнаружение светофоров и линий разметки, детектирование других транспортных средств, предсказание поведения пешеходов и прочих машин и множество других задач.

Одной из таких подзадач является обнаружение столкновений автомобилей на парковочной стоянке. Это позволит беспилотному транспортному средству информировать владельца о случившимся происшествии.

1. Цели и задачи работы

1.1. Цель работы

Целью данной дипломной работы является обнаружение столкновений на парковочной стоянке со стоящим транспортным средством с использованием нейронных сетей.

1.2. Поставленные задачи

Для достижения обозначенной цели были поставлены следующие задачи:

- Поиск и сбор данных
- Применение, обученной CNN, для обнаружения автомобилей
- Построение кинематической модели
  - Построение математической модели
  - Калибровка камеры
  - Определение расстояния до автомобиля
- Сигнализирование при столкновении транспортных средств

2. Обзор

2.1. Поиск данных

Для поиска данных столкновения транспортных средств на парковочной стоянке был выбран бесплатный видеохостинг YouTube. Видео представляют собой запись с регистратора припаркованного автомобиля, с которым происходит столкновение.

Так же для генерации видео со столкновениями автомобилей был рассмотрен вариант симуляции автономного вождения. Для выбора симулятора были проанализированы несколько уже существующих вариантов таких как AirSim [10], Webots [12], Gazebo [5] и CARLA [3]. Из перечисленных автосимуляторов для данной задачи больше подошел CARLA. Одним из главных достоинств этого программного обеспечения, не считая открытости кода, является то, что это симулятор городской среды, позволяющий пользователю самому ее настраивать, в том числе изменять погодные условия и освещение, чего нет в упомянутых ранее симуляторах.

2.2. Выбор CNN для обнаружения автомобилей

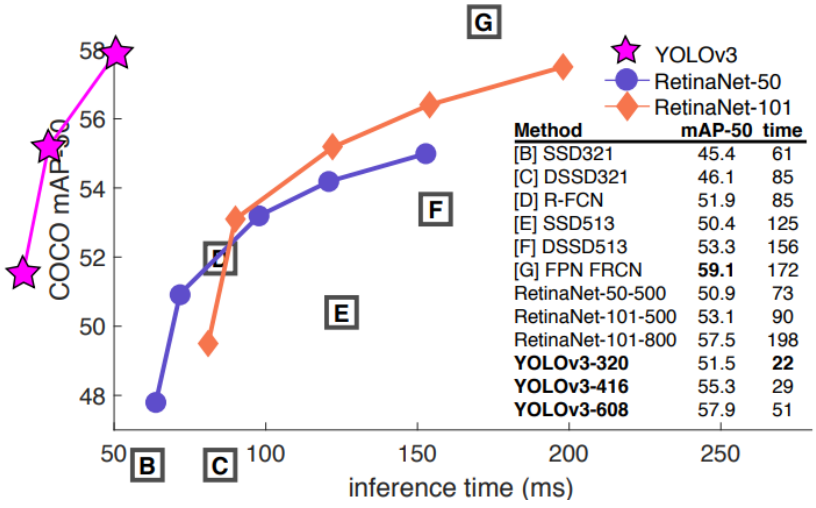
Для обнаружения на видео автомобилей было рассмотрено несколько вариантов CNN: серия архитектур на основе R-CNN, YOLO и RetinaNet.

Нейросети архитектуры R-CNN[20] такие, как R-CNN, Fast R-CNN и Faster R-CNN относятся к two-stage архитектурам и основаны на подходе предположения регионов. На первом этапе происходит поиск регионов, возможно содержащих в себе какие-то объекты. На втором этапе классификатор определяет, является ли регион объектом или фоном. В архитектуре Fast R-CNN улучшением было применение интеграции RPN(Region Proposal Network) с CNN на втором этапе для классификации объектов.

Нейронные сети архитектуры YOLO[9] относятся к one-stage архитектурам и предсказывают обрамляющие регионы с вероятностями классов, применяясь ко всему изображению. В отличие от других ней-

росетей, YOLO не использует плавающее окно, а делит изображение на ячейки с помощью сетки. После этого каждая ячейка отвечает за предсказание нескольких содержащих регионов и для каждого показатель уверенности и за предсказание вероятностей классов. Из-за того, что в архитектуре нет явных циклов for, сеть работает быстро. Это позволяет распознавать объекты в реальном времени.

Архитектура нейросети RetinaNet[11][15] состоит из FPN (Feature Pyramid Network), основной сети, предназначенной для вычисления карты признаков входного изображения, и двух дополнительных, предназначенных для классификации объектов на основе сгенерированных признаков и вычисления границы bounding box объекта с помощью регрессии. Обученный с помощью функции ошибки (Focal Loss) детектор RetinaNet позволяет получить высокое качество распознавания на уровне two-stage подходов со скоростью на уровне one-stage.



Method	mAP-50	time
[B] SSD321	45.4	61
[C] DSSD321	46.1	85
[D] R-FCN	51.9	85
[E] SSD513	50.4	125
[F] DSSD513	53.3	156
[G] FPN FRCN	**59.1**	172
RetinaNet-50-500	50.9	73
RetinaNet-101-500	53.1	90
RetinaNet-101-800	57.5	198
YOLOv3-320	51.5	**22**
YOLOv3-416	55.3	29
YOLOv3-608	57.9	51

Рис. 1: YOLOv3 и RetinaNet в тесте COCO 50 Benchmark

Поскольку в нейросетях архитектуры R-CNN явно используются циклы перебора по ограниченным регионам, замедляя работу обнаружения объекта, было решено использовать обученные на датасете COCO [4] нейронные сети: RetinaNet и YOLO, позволяющие достаточно точно и быстро находить объекты. На Рис. 1 изображено сравнение моделей YOLOv3 и RetinaNet. В качестве основной сети для RetinaNet была использована ResNet50. В качестве нейронной сети архитектуры YOLO была выбрана YOLOv3[8]. Главной особенностью YOLOv3 от

предыдущих версий является то, что на выходе есть три слоя, каждый из которых рассчитан на обнаружения объектов разного размера.

Для нахождения транспортных средств на видео была использована библиотека ImageAI [1] языка программирования Python.

3. Построение кинематической модели

Для построения математической модели необходимо определить дистанцию до авто, его ориентацию относительно камеры и корпуса машины и, возможно, наличие удара, поскольку встряска на видео не является надежным признаком.

3.1. Методы вычисления расстояния до объекта

В рамках дипломной работы было рассмотрено несколько вариантов определения расстояния до автомобиля по фотографии.

3.1.1. Определение расстояния методом параллакса

Способ, основанный на явлении параллактического смещения и предусматривающий вычисление дистанции на основе измерений длины одной из сторон (базиса – AB) и двух углов A и B в треугольнике ACB. Определить дистанцию до объекта можно по формуле (1)

$$D = \frac{L}{2 \sin \alpha/2} \tag{1}$$

Где D - расстояние до объекта, L - базис, α - угол смещения
Пример использования этого метода приведен на Рис. 2.

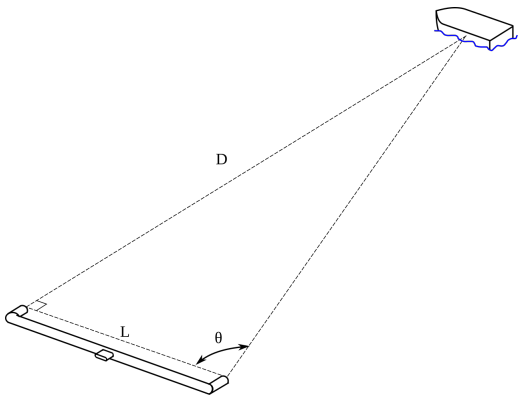


Рис. 2: Измерение дистанции при помощи параллакса

Однако, данный метод для решения поставленной задачи не подхо-

дит, поскольку видеорегистратор, на который ведется съемка, не передвигается.

3.1.2. Вычисление расстояния до наблюдаемого объекта по изображениям со стереопар

В приведенной работе [17] описан алгоритм для вычисления дистанции до предмета при помощи двух камер. На Рис. 3 приведена схема системы, из которой выводится формула для нахождения расстояния до объекта.

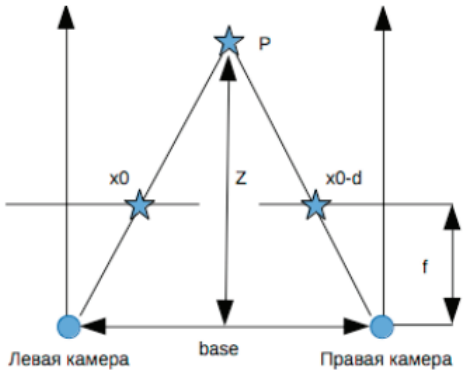


Рис. 3: Схема системы с двумя камерами

Из подобия треугольников следует формула (2)

$$\frac{(base–d)}{(Z–f)} = \frac{base}{Z} \tag{2}$$

Откуда получается формула (3).

$$Z = f \cdot \frac{base}{d} \tag{3}$$

Где Z — искомая дистанция до предмета, f — фокусное расстояние камеры стереопары, $base$ — расстояние между стереопарой, d — диспаратность точки x_0 объекта P.

Однако, этот метод тоже не подойдет для решения поставленной задачи, поскольку видеорегистратор чаще всего имеет одну камеру.

3.1.3. Вычисление расстояние через отношение окружающих объектов

Идея этого метода заключается в том, чтобы, зная размеры и другие величины до объекта на видео(например человека или фонарный столб) найти из соотношений реальных габаритов автомобиля и опорного предмета дистанцию до машины.

Однако, так как мы не можем дать гарантию, что на видео обязательно будут дорожные полосы, люди или другие объекты, этот способ нам также не подойдет.

3.1.4. Вычисление расстояния через угловые размеры объекта

Опираясь на статью [18] был получен метод вычисления дистанции до автомобиля через его угловые размеры.

Пусть у объекта заданы две точки с координатами образа (x_{1p}, y_{1p}) и (x_{2p}, y_{2p}) на изображении. Для каждой точки применим процедуру нормализации (4).

$$(x_n, y_n) = Normalize(x_n, y_n, c_x, c_y, f, s, dist) \tag{4}$$

Где c_x, c_y — координаты центра оптической оси в пикселях, f — фокусное расстояние измеряемое в пикселях, s — соотношение сторон пикселя, $dist$ — вектор искажения дисторсии.

Процедура нормализации позволяет перевести координаты изображения в систему координат фокальной плоскости с учетом смещения оптической оси, искажения дисторсии и соотношения сторон пикселя.

$$x' = x_p - c_x \tag{5}$$

$$y' = (y_p - c_y) \cdot s \tag{6}$$

$$(x, y) = Undistort(\frac{x'}{f}, \frac{y'}{f}, k) \tag{7}$$

$$(x_n, y_n) = (x \cdot f, y \cdot f) \tag{8}$$

Где $Undistort$ — функция, позволяющая убрать эффект дисторсии

для точки. Получив точки (x_{1n}, y_{1n}) и (x_{2n}, y_{2n}) найдем угловые размеры объекта по формуле (9):

$$\delta = arccos(\frac{x_{1n}x_{2n} + y_{1n}y_{2n} + f^2}{\sqrt{x_{1n}^2 + y_{1n}^2 + f^2} \cdot \sqrt{x_{2n}^2 + y_{2n}^2 + f^2}}) \tag{9}$$

Получив угловые размеры и реальные габариты транспортного средства, можно найти дистанцию до него по формуле (10):

$$D = \frac{H}{2tg(\frac{\delta}{2})} \tag{10}$$

Где D — расстояние до машины, H — реальная высота автомобиля, δ — угловые размеры транспортного средства.

3.1.5. Вычисление расстояния по углу наклона и высоте камеры

Опираясь на статьи [19] и [2] было решено использовать алгоритм, который использует высоту установленного видеорегистратора, угол наклона камеры и ее внутренние характеристики.

По формуле тонкой линзы (11), схема которой изображена на Рис. 4, зная фокусное расстояние и размеры объекта на изображении, можем получить дистанцию до автомобиля.

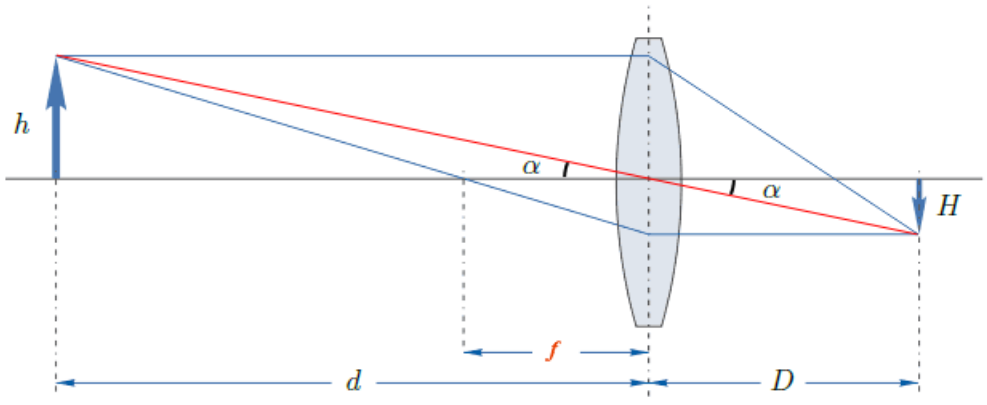


Рис. 4: Схема тонкой линзы

$$\frac{1}{d} + \frac{1}{D} = \frac{1}{f} \tag{11}$$

Где d — дистанция от линзы до объекта, D — расстояние от линзы до изображения объекта, а f — фокусное расстояние линзы.

Сделаем замену (12) и получим формулу (13), переписав (11):

$$h = d \cdot \tan(\alpha); H = D \cdot \tan(\alpha) \tag{12}$$

$$1 + \frac{h}{H} = \frac{d}{f} \tag{13}$$

Откуда получаем итоговую формулу (14):

$$d = \frac{f(H + h)}{H} \tag{14}$$

3.1.6. Итоговые методы для нахождения дистанции от камеры до автомобиля

Проведя анализ вышеперечисленных методов, было решено использовать два способа вычисления расстояния до объекта с помощью угла наклона и высоты видеорегистратора и через угловые размеры транспортного средства.

3.2. Предобработка видео

Поскольку необходимое для нахождения дистанции до автомобиля фокусное расстояние видеорегистратора изначально не дано, было решено построить матрицу внутренних характеристик камеры.

3.2.1. Матрица внутренних характеристик камеры

Матрица внутренних параметров камеры представляет собой верхнюю треугольную матрицу (15) размера 3×3, содержащую только параметры оптической системы и фотоприемника камеры.

$$K = \begin{pmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{15}$$

Где f_x, f_y — соответствуют фокусному расстоянию, измеренному в ширине и высоте пикселя, u_0, v_0 — соответствуют координатам принципиальной точки, а γ вычисляется по формуле (16)

$$\gamma = f_y \cdot \tan(\phi) \tag{16}$$

Где ϕ — угол наклона пикселя.

Зная значения f_x, f_y из матрицы и размеры пикселя, можно получить фокусное расстояние видеорегистратора.

Для вычисления матрицы внутренних параметров, было решено воспользоваться функцией калибровки камеры из библиотеки OpenCV языка программирования Python [6][7]

3.2.2. Калибровка камеры

Для калибровки камеры были использованы 8 опорных точек на автомобиле, запечатленном на видео. Для получения реальных точек пространства была использована инструкция по эксплуатации соответствующего автомобиля [16]. Для получения координат отображенных опорных точек с разных положений, было решено выделить из видео 11 кадров, и на них найти двумерные точки, являющимися проекциями реальных трехмерных опорных точек. Пример такого кадра изображен на Рис. 5.

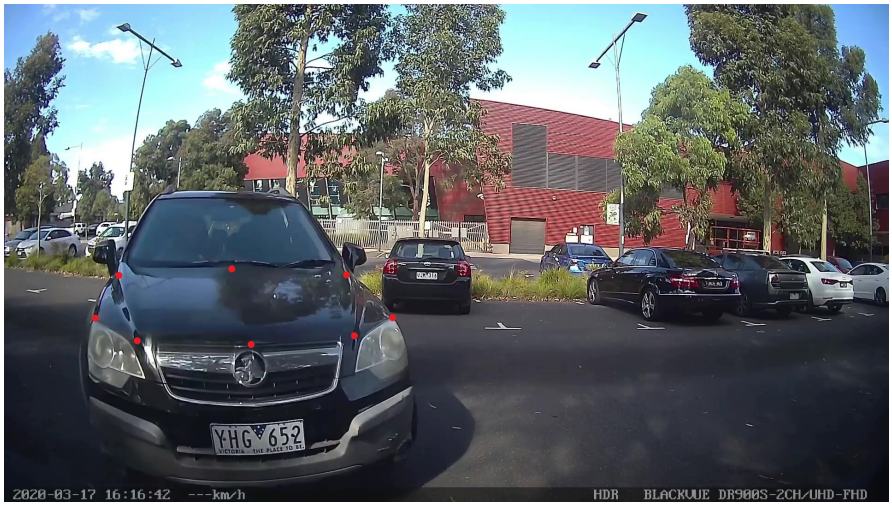


Рис. 5: Кадр из видео с отмеченными опорными точками

4. Вычисление расстояния до автомобиля с учетом угла наклона видеорегистратора

Для вычисления дистанции необходимо учитывать угол наклона γ видеорегистратора, схема камеры, оптическая ось которой расположена под углом, изображена на Рис. 6.

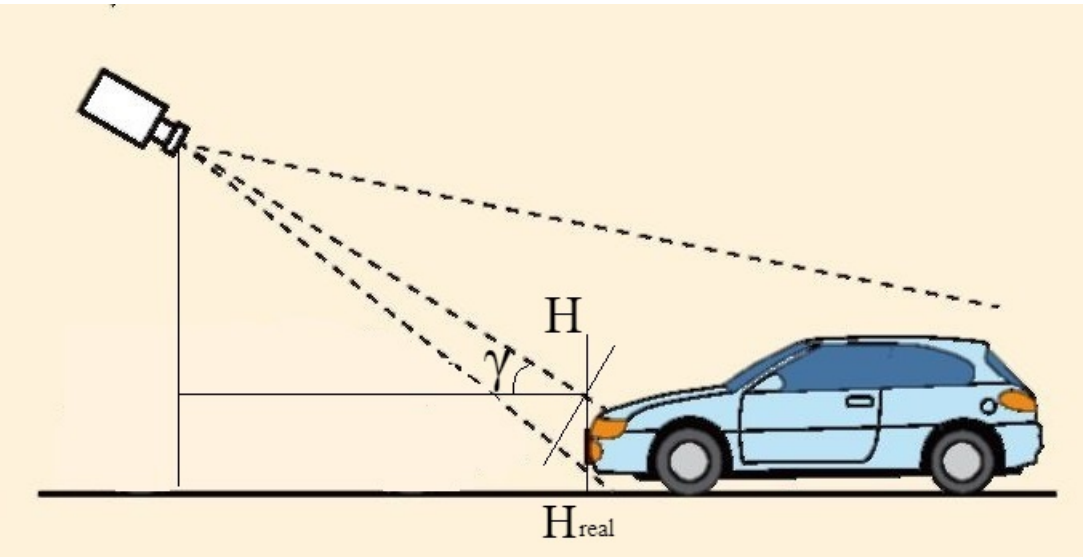


Рис. 6: Схема камеры, расположенной под углом

Поскольку проекция камеры тоже зависит от угла наклона, то высота транспортного средства вычисляется по формуле (17)

$$H = H_{real} \cdot \cos(\gamma) \tag{17}$$

Где H_{real} — Высота реального автомобиля, γ — угол наклона камеры.

В случае метода вычисления дистанции до объекта с помощью высоты и угла наклона, используя формулу тонкой линзы и фокусное расстояние, преобразованное из пикселей в метры, получаем расстояние L_0 от линзы видеорегистратора до проекции авто на плоскость ортогональную оптической оси. Для получения расстояния от проекции камеры на землю до этой проекции автомобиля необходимо домножить L_0 на $\cos(\gamma)$.

В случае метода с использованием угловых размеров объекта, получив расстояние из угловых размеров и реальной высоты автомобиля,

получим так же дистанцию от камеры до плоскости, ортогональной оптической оси видеорегистратора. Поэтому полученную высоту тоже нужно домножить на $\cos(\gamma)$

5. Алгоритм

Для решения задачи проверки столкновения автомобиля со стоящим транспортным средством было решено использовать следующий алгоритм.

Для получения внутренних параметров видеорегистратора вызвать функцию калибровки камеры. Разбить видео по кадрам и для каждого применить поиск автомобилей с помощью нейронных сетей и применить функцию измерения расстояния до транспортного средства, которое ближе всего находится к объекту, с которого ведется видеосъемка. Если дистанция стала меньше определенного значения $distnce_{min}$ и произошла встряска на видео, то сигнализируем пользователю, что произошло столкновение.

Псевдокод данного алгоритма Alg. 1 представлен ниже.

Сам код проекта можно посмотреть в репозитории [13]

```
def calibrate() //функция для калибровки видеорегистратора
def detectCars() //функция, определяющая автомобили на
изображении
def getDistance() //функция для расчета расстояния до
автомобиля по фотографии
def isShake() //функция проверяющая наличие встряски на видео
def signal() //функция, сигнализирующая пользователю, что
произошло столкновение
calibrate()
frame = video.firstFrame()
while frame! = video.lastFrame do
    detectCars()
    distance = getDistance()
    if distance < minDistanceisShake() then
        signal()
    else
end
```

Algorithm 1: Псевдокод для определения столкновения автомобиля на парковочной стоянке

6. Эксперименты

В рамках дипломной работы для проверки работы алгоритма были поставлены следующие эксперименты.

6.1. Эксперименты, основанные на видео с YouTube

В этих экспериментах использовалось одно и то же видео, но с разными нейронными сетями для распознавания объектов и разными методами определения расстояния до автомобиля. Ниже приведены фотографии с применением алгоритма, который в качестве входных данных принимал видео с YouTube.

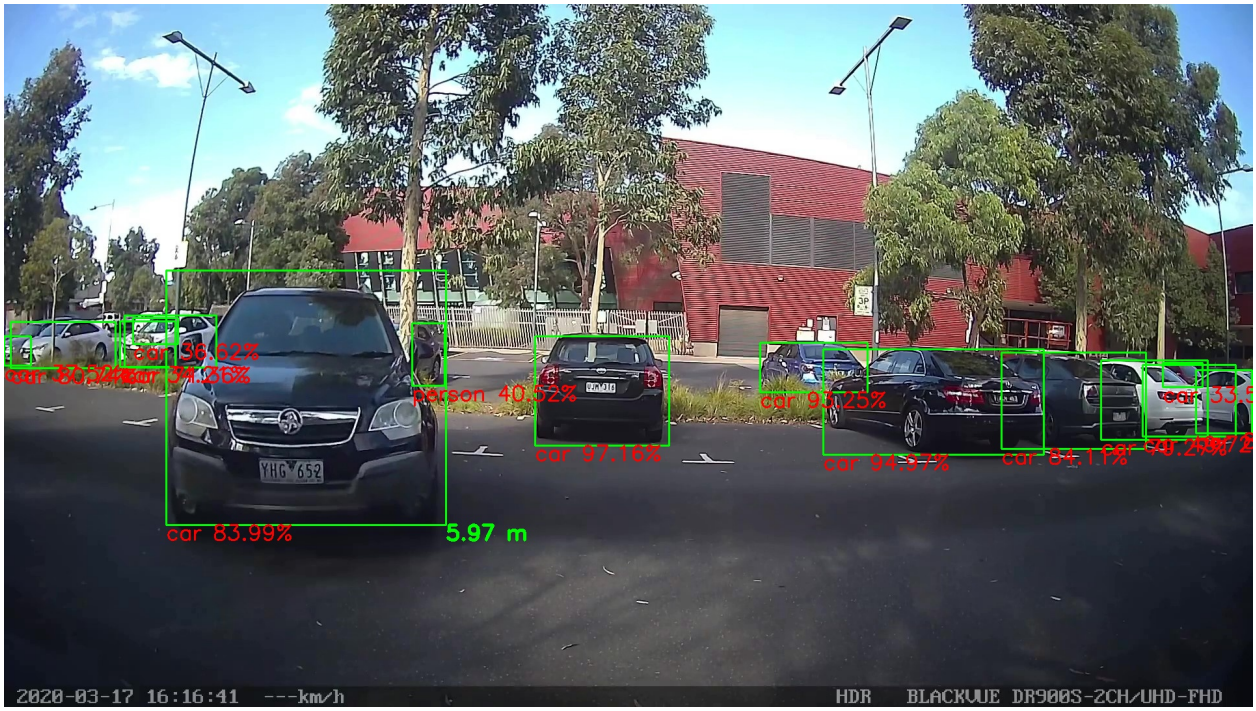


Рис. 7: Алгоритм с использованием RetinaNet и метода углового размера объекта

Рис. 8: Алгоритм с использованием RetinaNet и метода тонкой линзы

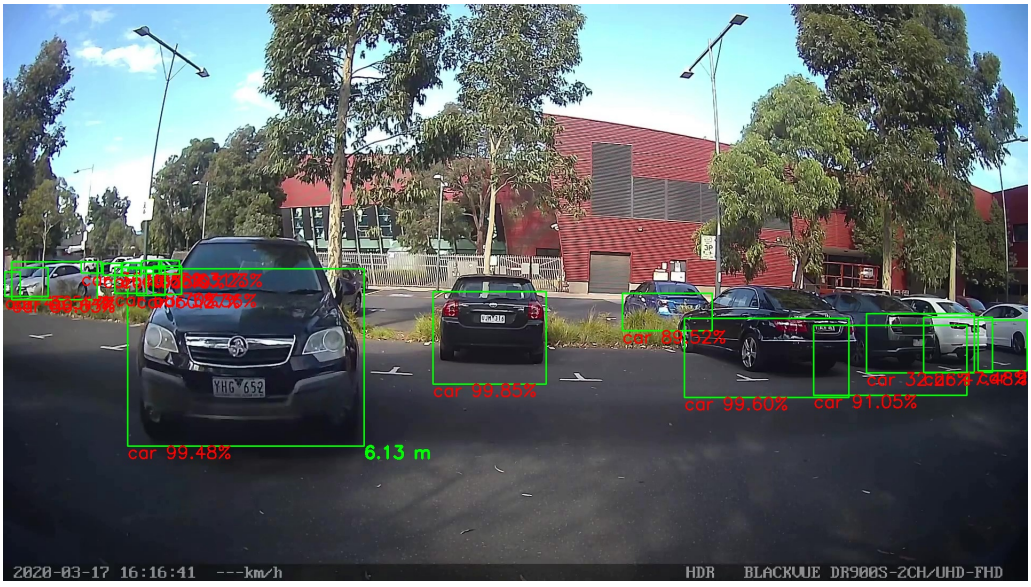


Рис. 9: Алгоритм с использованием YOLOv3 и метода углового размера объекта

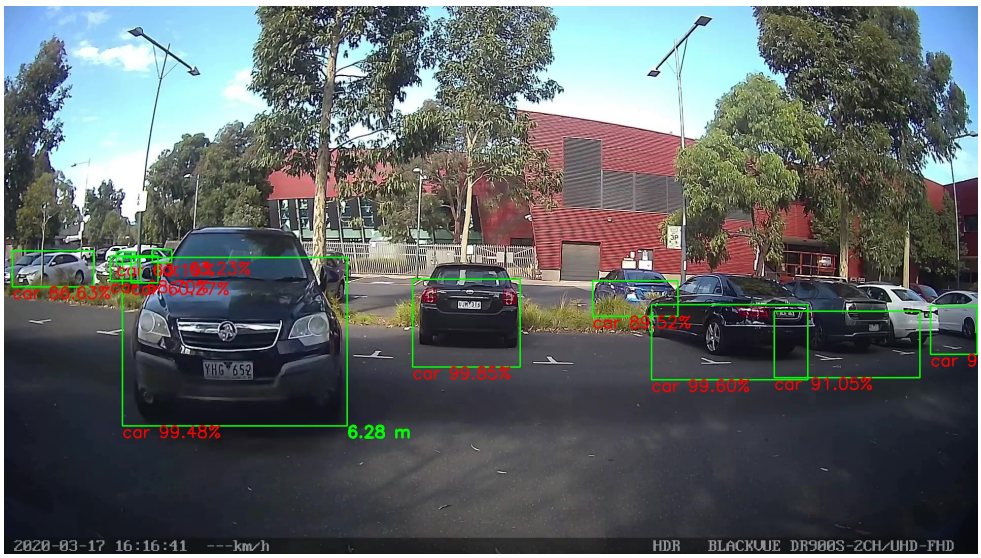


Рис. 10: Алгоритм с использованием YOLOv3 и метода тонкой линзы

6.2. Эксперименты, основанные на сгенерированных видео

Поскольку точное реальное расстояние по видео с YouTube определить нельзя, было решено провести ряд таких же экспериментов, использовав видео со сгенерированной на симуляторе CARLA аварии. Ниже приведены фотографии с применением алгоритма, для видео, полученного с помощью симулятора.

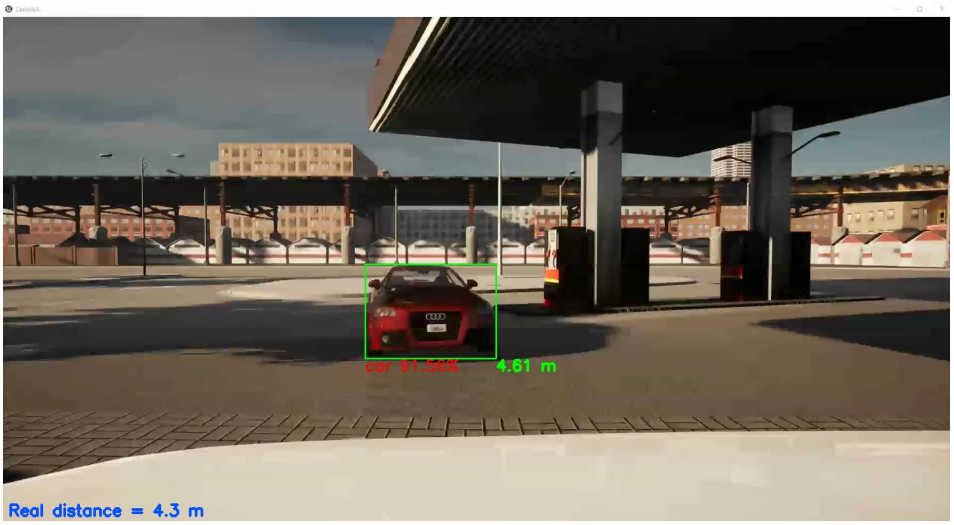


Рис. 11: Алгоритм с использованием RetinaNet и метода углового размера объекта

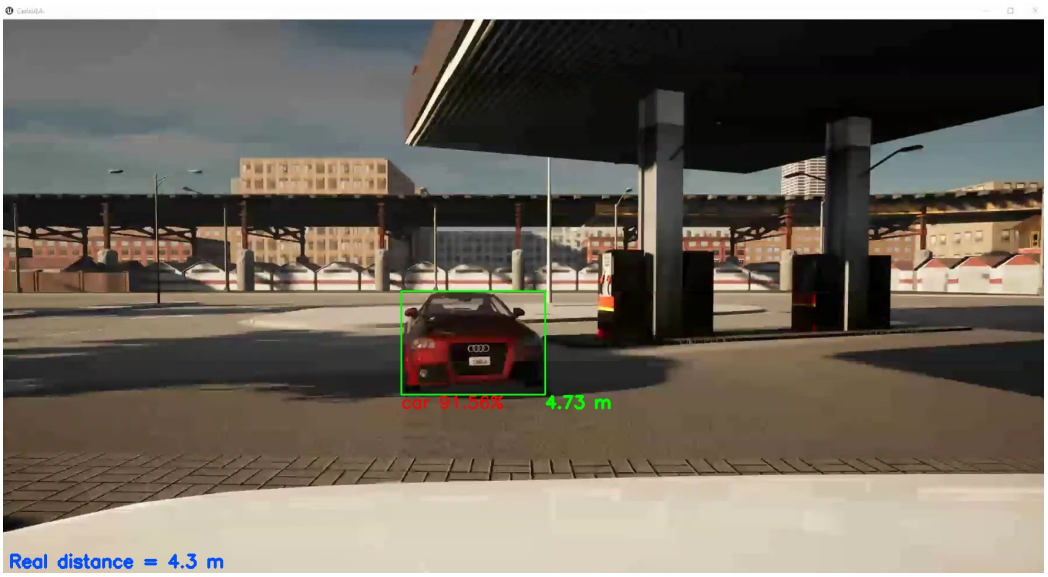


Рис. 12: Алгоритм с использованием RetinaNet и метода тонкой линзы

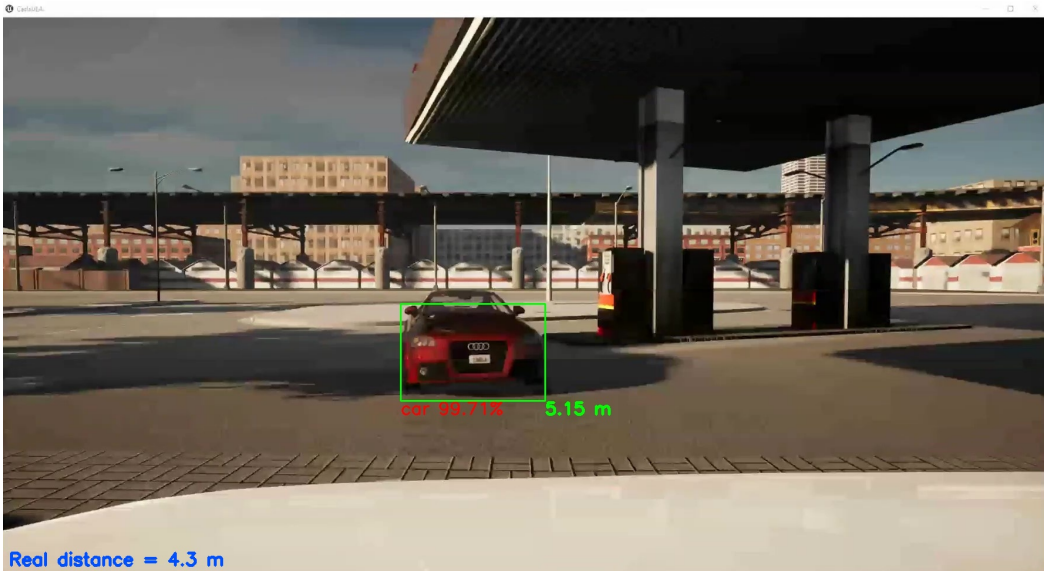


Рис. 13: Алгоритм с использованием YOLOv3 и метода углового размера объекта

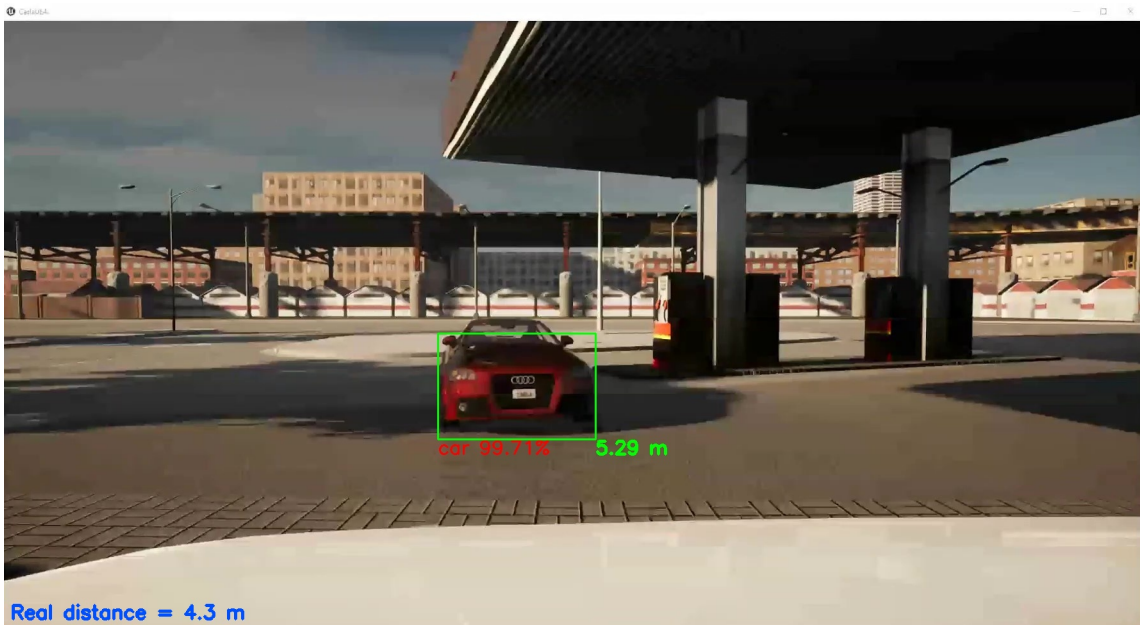


Рис. 14: Алгоритм с использованием YOLOv3 и метода тонкой линзы

6.3. Результаты экспериментов

Результаты экспериментов, приведенных в таблицах (1)(2), показали, что расстояние до автомобиля более точно считается в случае использования нейронной сети RetinaNet с применением метода угловых размеров объекта.

	YOLO с использованием метода тонкой линзы	YOLO с использованием углового расстояния	RetinaNet с использованием метода тонкой линзы	RetinaNet с использованием углового расстояния
Расстояние до объекта	6.28m	6.13m	6.09m	5.97m

Таблица 1: Расстояние до автомобиля разными способами по видео, найденного на YouTube

	YOLO с использованием метода тонкой линзы	YOLO с использованием углового расстояния	RetinaNet с использованием метода тонкой линзы	RetinaNet с использованием углового расстояния	Реальная дистанция в симуляторе
Расстояние до объекта	5.29m	5.15m	4.73m	4.61m	4.3m

Таблица 2: Расстояние до автомобиля разными способами по видео сгенерированного на симуляторе CARLA

Более подробную информацию об экспериментах можно посмотреть в репозитории [14]

Так как для измерения дистанции до объектов из автомобиля используются в основном стереокамера или камера, но ее внутренние параметры не вычисляются, а передаются в качестве входных параметров, найти аналогов, позволяющих определять расстояние до объекта по фотографии не удалось.

7. Итоговые результаты

В рамках дипломной работы были достигнуты следующие результаты:

- Были найдены и сгенерированы видео со столкновениями автомобилей.
- Был проведен анализ существующих автосимуляторов.
- Был проведен анализ трех нейронных сетей для определения объектов на изображении.
- Был проведен анализ способов измерения расстояния до объекта по фотографии.
- Был изучен метод калибровки камеры.
- Был представлен алгоритм проверки столкновения автомобиля со стоящим транспортным средством, используя запись видео с одной камеры и оценку расстояния по фото, аналогов которому не было найдено.
- Были поставлены эксперименты с использованием разных методов для измерения дистанции до автомобиля, разных нейронных сетей для обнаружения объектов на двух типах видео.

Список литературы

[1] AI Image. Официальная английская документация для ImageAI. — URL: https://clck.ru/SV667.

[2] Apoorva Joglekar Devika Joshi Richa Khemani Smita Nair Shashikant Sahare. Depth Estimation Using Monocular Camera. International Journal of Computer Science and Information Technologies, Vol. 2. — 2011. — URL: https://clck.ru/UYBhL.

[3] CARLA. Официальный сайт симулятора городского вождения CARLA. — URL: http://carla.org/.

[4] COCO. Набор данных для обнаружения и сегментации. — URL: https://cocodataset.org/#home.

[5] Gazebo. Официальный сайт Gazebo. — URL: http://gazebosim.org/.

[6] OpenCV. OpenCV Camera Calibration and 3D Reconstruction. — URL: https://clck.ru/Ud3gk.

[7] OpenCV. OpenCV Fisheye camera model. — URL: https://clck.ru/Ud3hM.

[8] Pjreddie.com. Официальный сайт нейронной сети YOLO3. — URL: https://clck.ru/SV5xi.

[9] Redmon Joseph, Farhadi Ali. YOLOv3: An Incremental Improvement. — URL: https://pjreddie.com/media/files/papers/YOLOv3.pdf.

[10] Shital Shah Debadeepta Dey Chris Lovett Ashish Kapoor. AirSim: High-Fidelity Visual and PhysicalSimulation for Autonomous Vehicles. — 2017.

[11] Tsang Sik-Ho. Review: RetinaNet — Focal Loss (Object Detection). — URL: https://clck.ru/SadZ6.

[12] Webots. Официальный сайт Webots. — URL: http://www.cyberbotics.com/.

[13] ZaicevDima. Репозиторий с дипломным проектом. — URL: https://github.com/ZaicevDima/DiplomProj.

[14] ZaicevDima. Эксперименты дипломного проекта. — URL: https://github.com/ZaicevDima/DiplomProj/tree/master/Experiments.

[15] balezz. RetinaNet на LADDv2. — URL: https://github.com/balezz/LaExperiments.

[16] Гаврилов А. Н. Петров А. М. Горфин И. С. Chevrolet Cruze. Руководство по эксплуатации, техническому обслуживанию и ремонту. Ремонт без проблем. — 2011. — URL: https://clck.ru/Ud3yf.

[17] С. Ильясов Э. Молодой учёный № 14 (118). Технические науки. — ООО «Издательство Молодой ученый», 2016. — URL: https://moluch.ru/archive/118/pdf/618/.

[18] Шишалов И. С. Погорский Н. В. Филимонов А. В. Громазин О. А. СПОСОБ ОПРЕДЕЛЕНИЯ РАССТОЯНИЯ ДО ОБЪЕКТА ПРИ ПОМОЩИ КАМЕРЫ (ВАРИАНТЫ). Технические науки. — Общество с ограниченной ответственностью "ДиСиКон", 2016. — URL: https://clck.ru/UhLsA.

[19] Экспонента ЦИТМ. Семантическая сегментация изображения с камеры автомобиля для построения ADAS систем. — 2020. — URL: https://clck.ru/UY6VP.

[20] Я. Бендриковский А. Выделение объекта из видеопотока с помощью глубинного обучения. — 2017. — URL: https://dspace.spbu.ru/bitstream/11701/10721/1/diploma.pdf.