Санкт-Петербургский государственный университет

Кафедра системного программирования

Группа 21.Б10-мм

Создание ground truth data для дорожных знаков в ADAS

Громова Арабелла Михайловна

Отчёт по учебной практике
в форме «Решение»

Научный руководитель:
доцент кафедры системного программирования, к.т.н., Ю. В. Литвинов

Консультант:
Инженер-программист АО «Кама» М. С. Осечкина

Санкт-Петербург
2023

Оглавление

Введение

ADAS (Advanced Driver Assistance Systems) — это система дополненной реальности для помощи водителю на дороге. ADAS используется для повышения безопасности дорожного движения и сокращения дорожно-транспортных происшествий. Такие системы не могут полностью предотвратить аварии, но они могут уменьшить влияние некоторых человеческих факторов, которые приводят к ДТП [1]. Одной из функций ADAS является детектирование и распознавание дорожных знаков.

Как и любую другую систему, ADAS нужно тестировать и иметь возможность отслеживать, улучшили ли внесённые в новой версии изменения работу системы по сравнению с предыдущей. Для этого нужны эталонные данные (англ. ground truth data) дорожных знаков, с которыми нужно сравнивать результат работы системы.

Существует большое количество публичных датасетов дорожных знаков, но их недостаточно, так как датасеты в основном содержат европейские дорожные знаки. Также датасеты должны быть достаточно разнообразными: изображения, видео должны быть сняты в любую погоду, в любое время года. Существует датасет российских дорожных знаков RTSD[1], который содержит 179138 кадров. Кадры сняты в разное время года, в разные погодные условия. Основными недостатками датасета являются неточность разметки и разметка данных квадратом, а не прямоугольником с меняющимися пропорциями.

Существует множество инструментов разметки, которые позволяют размечать данные. Некоторые инструменты поддерживают только ручную разметку данных, например, LabelImg[2]. Однако ручная разметка ground truth data — очень долгий и не всегда точный процесс, поэтому требуется автоматизировать процесс разметки. Некоторые инструменты, например Hasty.ai[3], позволяют автоматически размечать данные, но, как правило, данные инструменты являются платными либо для

[1]RTSD: https://www.kaggle.com/datasets/watchman/rtsd-dataset (дата обращения 28.11.2022)
[2]LabelImg: https://github.com/heartexlabs/labelImg (дата обращения 28.11.2022)
[3]Официальный сайт: https://hasty.ai/annotation (дата обращения 28.11.2022)

разметки дорожных знаков пользователь должен самостоятельно обучить нейросеть. Обучение нейросети требует большую обучающую выборку, также пользователь может столкнуться с проблемами при обучении нейросети, например, с её переобучением или недообучением.

В связи с этим, предлагается разработать инструмент, который в автоматизированном режиме будет создавать эталонные данные для дорожных знаков. Поскольку процесс автоматизированный, то допустима помощь пользователя: он может отсекать ложно-положительные срабатывания или выделять знак на кадре, или регион для поиска знаков.

1. Постановка задачи

Целью работы является разработка инструмента для создания эталонных данных для дорожных знаков. Для её выполнения были поставлены следующие задачи.

Задачи на осенний семестр:

1. Ознакомиться с уже существующими инструментами разметки данных.
2. Изучить существующие подходы к задаче обнаружения дорожных знаков.
3. Реализовать алгоритм детектирования дорожных знаков.

Задачи на весенний семестр:

1. Реализовать полуавтоматический инструмент для создания эталонных данных для дорожных знаков на основе созданного алгоритма обнаружения дорожных знаков.
2. Провести пользовательское тестирование.

2. Обзор

В этой главе будет проведен обзор наиболее популярных существующих инструментов разметки данных, а также рассмотрены алгоритмы детектирования дорожных знаков.

Все инструменты разметки были найдены в поисковой системе Google. Для рассмотрения были взяты первые три ссылки по запросу «автоматические инструменты разметки данных», а также первые три ссылки по запросу «инструменты разметки данных».

2.1. Инструменты разметки данных

В данном разделе рассмотрены инструменты, автоматизирующие процесс разметки данных.

2.1.1. CVAT

CVAT[4] — бесплатный open-source веб-инструмент, который используется для разметки данных. Инструмент предоставляет возможность размечать данные для задач машинного обучения, одной из которых является задача распознавания объектов. Для этой задачи в CVAT можно использовать обученные детекторы, с помощью которых можно автоматически размечать данные. Но предоставленные детекторы не обучены обнаружению дорожных знаков, кроме знака STOP. В CVAT пользователь имеет возможность с помощью своей обученной модели размечать данные.

2.1.2. Auto-Annotate Tool

Auto-Annotate Tool [5] — бесплатный open-source инструмент для автоматического аннотирования данных. Данный инструмент работает только с изображениями и использует полигональную сегментацию.

[4]Официальный сайт: https://www.cvat.ai/ (дата обращения 30.11.2022)

[5]Auto-Annotate Tool: https://github.com/mdhmz1/Auto-Annotate/ (дата обращения 30.11.2022)

Также Auto-Annotate применяет обученные модели для автоматической разметки. Пользователь может размечать данные двумя способами:

- с помощью нейросети, обученной на датасете COCO[6], в котором отсутствуют аннотированные дорожные знаки.
- с помощью нейросети, которую пользователь обучит сам на выбранном датасете.

Таким образом, для автоматической разметки дорожных знаков на изображениях пользователь должен самостоятельно обучить нейросеть.

2.1.3. V7 Darwin

V7 Darwin[7] — платный веб-инструмент автоматической разметки. Инструмент работает с видео, с изображениями, с текстом. Darwin использует обученные модели для автоматического аннотирования данных. Также компания V7 предоставляет возможность обучения нейросети внутри платформы на выбранных пользователем датасетах.

2.2. Алгоритмы обнаружения дорожных знаков

Существующие методы решения задачи детектирования дорожных знаков можно разделить на пять категорий [3]:

- методы на основе цвета;
- методы на основе формы;
- методы на основе цвета и формы;
- методы на основе машинного обучения;
- методы на основе LIDAR.

В данном разделе будут рассмотрены метод на основе машинного обучения, метод на основе цвета и формы.

[6]Более подробно о датасете: https://cocodataset.org/ (дата обращения 30.11.2022)
[7]Официальный сайт: https://www.v7labs.com/ (дата обращения 01.12.2022)

2.2.1. Метод Виолы-Джонса

Метод Виолы-Джонса [6] — алгоритм обнаружения объектов на изображениях в реальном времени, предложенный Паулом Виолом и Майклом Джонсом в 2001 году. Изначально алгоритм был разработан для детектирования лиц [7]. Но в настоящее время модификации метода Виолы-Джонса применяются для обнаружения дорожных знаков, автомобилей и множества других объектов. Алгоритм обнаружения дорожного знака, описанный в статье «Boosted road sign detection and recognition» [9], является модификацией метода Виолы-Джонса. В алгоритме используются интегральное представление изображения, признаки Хаара для обучения классификаторов, алгоритм машинного обучения Adaboost, каскад сильных классификаторов.

Интегральное представление изображения вычисляется по формуле:

$$S(\mathrm{x}, \mathrm{y}) = \sum_{i=0}^{\mathrm{x}} \sum_{j=0}^{\mathrm{y}} I(i, j)$$

где I изображение, $I(i, j)$ — яркость пикселя (i, j), интегральное представление изображения $S(x, y)$, содержащее в себе сумму яркостей пикселей изображения I, накопленных от $(0, 0)$ до пикселя (x, y). Интегральное представление позволяет быстро вычислять значение признаков изображения.

Признак Хаара представляет собой разность сумм пикселей двух смежных областей внутри прямоугольника, который может быть любого размера и занимать различные положения на изображении. Признаки Хаара показаны на рисунке 1.

AdaBoost — это алгоритм машинного обучения, который строит сильный классификатор путем выбора слабых классификаторов. Классификатор называется сильным, если допускает малое количество ошибок классификации, слабый классификатор наоборот допускает большое количество ошибок классификации. Каскад сильных классификаторов повышает качество и скорость детектирования объектов.

Несмотря на высокую точность и скорость алгоритма, у метода

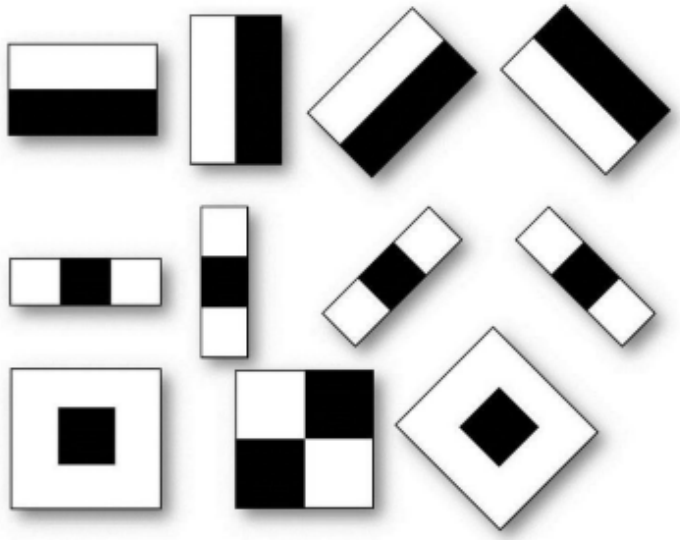

Рис. 1: Признаки Хаара

Виолы-Джонса есть и недостатки: чувствительность к освещению, длительное время обучения каскадов классификаторов.

2.2.2. Detection, Tracking and Recognition of Traffic Signs from Video Input

Алгоритм обнаружения дорожных знаков, описанный в статье «Detection, Tracking and Recognition of Traffic Signs from Video Input» [2] использует знания о цвете, форме. На основе данных двух свойств выделяется четыре категории европейских знаков: инструктивные (синий круг), запрещающие (красный круг), предупреждающие (желтый треугольник) и информативные (синий квадрат). Детектор использует каскад классификаторов, обобщенное преобразование Хафа, введенное Лоем и Барнсом [5]. С помощью каскада классификатора выделяются области интереса (ROI). В каждой области интереса присутствует один из трех цветов (синий, красный, желтый). Далее выделяются края. После выделения краев применяется соответствующий детектор многоугольников Лоя и Барнса, основанный на преобразовании Хафа. Например, для карты краев области интереса с выделенным желтым цветом используется детектор треугольников.

Для оценки точности алгоритм был применен к 96 видео, содержа-

щим в совокупности 146 знаков (из них 48 уникальных). Видео были записаны в дневное время в городских, сельских районах и на автомагистралях. Точность детектирования дорожных знаков составила 93%. В статье точность детектора определялась количеством детекций дорожных знаков, позиция знака в кадре не использовалась для оценки качества детектора.

Однако у алгоритма есть недостатки. Одним из которых является то, что детектор работает с изображениями в цветовом пространстве RGB, каналы которого чувствительны к изменениям освещения. Также для обучения классификаторов должна быть разнообразная выборка дорожных знаков (изображения дорожных знаков в любое время года, в любую погоду). Помимо этого в данной статье точность алгоритма проверялась на небольшом количестве дорожных знаков.

3. Реализация

3.1. Используемые инструменты

3.1.1. C++

Для реализации инструмента был взят язык C++. C++ является высокопроизводительным языком программирования. Также на C++ реализована библиотека OpenCV.

3.1.2. OpenCV

Для работы с изображениями была взята библиотека OpenCV. OpenCV — библиотека для работы с алгоритмами компьютерного зрения, машинным обучением и обработкой изображений. OpenCV является бесплатной, кроссплатформенной библиотекой с открытым исходным кодом, активно поддерживаемой разработчиками.

3.2. CVAT

Для интеграции детектора дорожных знаков был выбран open-source инструмент разметки CVAT[8]. Инструмент имеет подробную документацию как и для пользователя, так и для контрибьютора. Также в CVAT используются модели глубокого обучения, которые реализованы как serverless функции и существует возможность встроить в инструмент разметки свою модель[9].

3.3. Nuclio

Nuclio[10] – open-source serverless фреймворк, используемый для автоматизации развертывания приложений на основе обработки данных,

[8]CVAT: https://github.com/opencv/cvat (дата обращения 13.05.2022)

[9]Документация: https://opencv.github.io/cvat/docs/manual/advanced/serverless-tutorial/ (дата обращения 13.05.2022)

[10]Nuclio: https://github.com/nuclio/nuclio (дата обращения 13.05.2022)

для исполнения ресурсоемкого кода на сервере. Данная платформа используется в CVAT для развертывания serverless функций в nuclio контейнере.

3.4. Алгоритм обнаружения дорожных знаков

В текущей реализации алгоритм обнаружения дорожных знаков использует знания о цвете и форме знака. В основном российские дорожные знаки имеют форму треугольника, круга, прямоугольника, квадрата. Также российские знаки можно разделить по цвету: на красные, синие, желтые дорожные знаки. Поэтому было принято решение реализовывать алгоритм обнаружения дорожных знаков таким образом: сперва кадр бинаризируется на основе сегментации по трем цветам (синий, красный, жёлтый). Далее на каждом бинарном изображении с одним из трех выделенных цветов применяется детектор многоугольнико. Идея использования информации о цвете и форме дорожных знаков для их обнаружения и была взята из статьи «Detection, Tracking and Recognition of Traffic Signs from Video Input» [2]

Алгоритм работает с видеокадрами. Сперва изображения бинаризируется на основе цветовой сегментации изображения. Для выделения красного цвета на изображении была реализована функция highlightRed(), для выделения двух других цветов – highlightColor(). Функции возвращают изображение с объектами соответствующего цвета. Пример результата работы функции highlightRed() показан на рисунке 2.

Далее к каждому изображению с определенным выделенным цветом применяется детектор многоугольников, обернутый в класс DetectorPolygons. Детектор обнаруживает треугольники, круги, прямоугольники. Пример результата работы алгоритма обнаружения дорожных знаков показан на рисунке 3.

3.5. Интеграция

Для интеграции детектора в CVAT были рассмотрены несколько способов: встроить детектор дорожных знаков вместо Intelligent-scissors

Рис. 2: Пример результата работы функции highlightRed()

Рис. 3: Пример результата работы алгоритма

или использовать serverless функции.

Процесс интеграции был значительно замедленен из-за неточной документации для запуска CVAT в режиме дебага [11], отсутствуют инструкции для скачивания некоторых библиотек, также для запуска докер контейнер использовалась неверная команда, впоследствии разработчики исправили данную ошибку. Также на WSL CVAT не запускался в режиме дебага.

Так как интерфейс CVAT написан на TypeScript, первоначально рас-

[11] Документация: https://opencv.github.io/cvat/docs/contributing/development-environment/ (дата обращения 13.05.2022)

сматривалась идея использовать Emscripten[12], но после изучения документации, данный вариант интеграции перестал быть главным. Далее был рассмотрен способ внедрения детектора с помощью аддонов для Node.js [13]. Аддоны позволяют вызывать модули, написанные на C++ из кода JavaScript. Но Node.js не может быть подключен к коду, исполняемому в браузере.

Далее было принято решение отправлять http-запросы детектору из кода Intelligent-scissors[14]. Аналогично веб-серверу для отправки координат результата работы трекера[15] был реализован http-сервер для детектора. Но из-за архитектуры CVAT не удалось отправить запрос из кода Intelligent-scissor.

Далее была изучена документация CVAT для serverless функций[16]. Первоначально рассматривалась идея развернуть детектор с помощью nuclio -контейнера[17], но после ознакомления с документацией было выяснено, что Nuclio[18] не поддерживает C++. Было принято решение отправлять http-запросы из nuclio- контейнера с YOLO v3. Но возникли сложности с тем, чтобы совместить функциональность YOLO v3 и детектор дорожных знаков. В связи с этим было принято решение создать nuclio-контейнер, в котором посылаются http-запросы при помощи Python на сервер. Но чтобы отправлять запросы из nuclio контейнера необходимо обращаться к ip адресу в локальной сети. Поэтому был написан python скрипт, который записывает в файл ip адрес компьютера в локальной сети. В свою очередь в nuclio-контейнере ip адрес читается из файла из монтированной папки. Также был написан bash-скрипт для запуска сервера.

---

[12]Документация: https://emscripten.org/ (дата обращения 13.05.2022)

[13]C++ addons: https://nodejs.org/api/addons.html (дата обращения 13.05.2022)

[14]Intelligent-scissors: https://opencv.github.io/cvat/docs/manual/advanced/ai-tools/#opencv-intelligent-scissors (дата обращения 13.05.2022)

[15]Веб-сервер для трекера Марии Житнухиной: https://github.com/osechkina-masha/adas_spbu/tree/pedestrian_tracking_4_sem/PedestrianTracking (дата обращения 14.05.2022)

[16]Документация https://opencv.github.io/cvat//docs/administration/advanced/installation_automatic_annotation/ (дата обращения 14.05.2022)

[17]Документация https://opencv.github.io/cvat/docs/manual/advanced/serverless-tutorial/ (дата обращения 14.05.2022)

[18]Документация https://github.com/nuclio/nuclio (дата обращения 14.05.2022)

4. Апробация

4.1. Условия апробации

Для проверки работоспособности инструмента было проведено пользовательское тестирование. Работа инструмента была оценена по шкале удобства использования системы[19]. Шкала позволяет численно оценить удобство использования системы от 0 до 100 баллов на основе опросника, содержащего 10 вопросов о простоте, об удобстве использования системы.

Инструмент был протестирован научным руководителем и консультантом В результате был получен средний балл 78.75 по System Usability Scale. Стоит отметить, что средняя оценка по данной методике является 68 баллов [8].

4.2. Тестирование детектора

Для оценки качества работы детектора были применены следующие метрики: IoU, precision, recall.

Intersection over Union (IoU) [4] измеряет степень перекрытия между предсказанными и настоящими объектами, выражаемую в виде отношения площади пересечения к площади объединения. Чем ближе значение IoU к единице, тем лучше модель справляется с задачей обнаружения объектов.

Для вычисления precision, recall, необходимо поделить объекты на следующие классы:

- TP (верно-положительные) – это количество правильно обнаруженных объектов, обнаружен дорожный знак и это верно.
- TN (ложно-отрицательные) – это количество правильно определенных отрицательных объектов, обнаружен другой объект и это верно

[19]System Usability Scale: https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html (дата обращения 05.12.2022)

- FP (ложно-положительные) – это количество ложных срабатываний, обнаружен дорожный знак и это неверно.
- FN (ложно-отрицательные) – это количество пропущенных объектов, обнаружен другой объект и это неверно.

precision показывает, как много из обнаруженных объектов действительно являются дорожными знаками.

$$precision = \frac{TP}{TP + FP}$$

recall показывает, какую долю дорожных знаков из всех знаков нашел алгоритм.

$$recall = \frac{TP}{TP + FN}$$

Среднее значение IoU при оценке детектора дорожных знаков составило приблизительно 0.597.

Среднее значение precision составило 0.72. Среднее значение recall – 0.4.

Заключение

За осенний семестр были выполнены следующие задачи:

1. Ознакомиться с уже существующими инструментами разметки данных.
2. Изучить существующие подходы к задаче обнаружения дорожных знаков.
3. Реализовать алгоритм детектирования дорожных знаков.
4. Реализовать полуавтоматический инструмент для создания эталонных данных для дорожных знаков на основе созданного алгоритма обнаружения дорожных знаков.
5. Провести пользовательское тестирование.

Реализация детектора доступна на Github[20]. Реализация интеграции детектора в CVAT[21]. Имя пользователя: rongirl.

[20] Репозиторий проекта:
https://github.com/osechkina-masha/adas_spbu/tree/trafficSignDetection

[21] Репозиторий проекта:
https://github.com/rongirl/cvat_with_traffic_sign_detector

Список литературы

[1] A. Ziebinski R. Cupek D. Grzechca, Chruszczyk L. Review of advanced driver assistance systems (ADAS). — 2017. — URL: https://aip.scitation.org/doi/abs/10.1063/1.5012394 (online; accessed: 2022-10-08).

[2] Andrzej Ruta Yongmin Li Xiaohui Liu. Detection, Tracking and Recognition of Traffic Signs from Video Input. — 2008. — URL: https://ieeexplore.ieee.org/document/4732535 (online; accessed: 2022-12-04).

[3] Chunsheng Liu Shuang Li Faliang Chang Yinhai Wang. Machine Vision Based Traffic Sign Detection Methods: Review, Analyses and Perspectives. — 2019. — URL: https://ieeexplore.ieee.org/document/8746141 (online; accessed: 2022-11-27).

[4] Hamid Rezatofighi Nathan Tsoi JunYoung Gwak Amir Sadeghian-Ian Reid Silvio Savarese. Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression. — 2019. — URL: https://ieeexplore.ieee.org/document/8953982 (online; accessed: 2023-05-28).

[5] N. Barnesi G. Loy D. Shaw A. Robles-Kelly. Regular polygon detection. — 2005. — URL: https://ieeexplore.ieee.org/document/1541332 (online; accessed: 2022-12-04).

[6] Paul Viola Michael Jeffrey Jones. Robust Real-time Object Detection. — 2001. — URL: https://www.researchgate.net/publication/215721846_Robust_Real-Time_Object_Detection (online; accessed: 2022-12-02).

[7] Paul Viola Michael Jeffrey Jones. Robust Real-Time Face Detection. — 2004. — URL: https://www.researchgate.net/publication/220660094_Robust_Real-Time_Face_Detection (online; accessed: 2022-12-02).

[8] Sauro Jeff. Measuring Usability with the System Usability Scale (SUS). — 2011. — URL: https://measuringu.com/sus/ (online; accessed: 2023-05-30).

[9] Sin-Yu Chen Jun-Wei Hsieh. Boosted road sign detection and recognition. — 2008. — URL: https://ieeexplore.ieee.org/document/4621071 (online; accessed: 2022-12-02).