Интеллектуальный детектор для системы управления транспортными объектами города

Абдулнагимов А.И.
Доцент, к.т.н.
Институт информатики, математики и робототехники
Уфимский университет науки и технологий
Уфа, Россия
e-mail: abdulnagimov.ai@ugatu.su

Антонов В.В.
Профессор, д.т.н.
Институт информатики, математики и робототехники
Уфимский университет науки и технологий
Уфа, Россия
e-mail: antonov.v@bashkortostan.ru

Максимов М.С.
Студент
Институт информатики, математики и робототехники
Уфимский университет науки и технологий
Уфа, Россия
e-mail: maks6863@gmail.com

Муллагулов А.Э.
Студент
Институт информатики, математики и робототехники
Уфимский университет науки и технологий
Уфа, Россия
e-mail: albertmullaguloff@yandex.ru

Аннотация[1]

В работе исследованы архитектуры глубокого обучения YOLOv5-v10 для разработки компактного детектора для идентификации и классификации транспортных объектов дорожного движения с функциями подсчета трафика, оценкой их скорости движения и координат перемещения. Для повышения точности расчетов используются алгоритмы компенсации дисторсии видеокамеры, алгоритмы преобразования координат камеры в мировую систему координат с использованием методов прямого геометрического преобразования и Чанга. Предложены алгоритмы оценки скорости и координат передвижения транспортных средств. Для сглаживания скорости исследованы два метода - простой фильтр Калмана и последовательное применение взятия медианы и усреднения. Решена проблема переназначения номеров транспортных объектов в связи со сбоями детекции в реальном времени за счет использования вектора скорости объекта. Проведена оценка точности распознавания и классификации объектов, оценка погрешности вычисления скорости и координат передвижения объектов. Представлена реализация на аппаратной платформе NVIDIA Jetson AGX Xavier. Применение данного детектора планируется в специально оборудованных автомобилях (дорожных лабораториях) для оценки плотности потока в автоматизированном режиме, а также с возможностью установки на столбы оживленных перекрестков в защищенном боксе для оценки плотности потока и оценки аварийный ситуаций в online режиме с передачей информации на сервер.

Ключевые слова: *обнаружение транспортных объектов, машинное зрение, YOLO, оценка скорости, координаты движения*.

1. Введение

Интеллектуальная транспортная система современного мегаполиса сегодня использует инновационные технологии и устройства для регулирования транспортных потоков, обеспечения эффективного и безопасного движения наземного транспорта [1,2]. С развитием технологий Интернета вещей (IoT), искусственного интеллекта (ИИ) и мобильных сетей пятого и шестого поколений (5G/6G) открываются новые горизонты для создания интеллектуальных транспортных систем. Такие технологии обеспечивают возможность создавать “умные” системы управления транспортными потоками в части регулирования потока автомобилей, обнаружения опасностей и предотвращения

[1]**Труды X Международной научной конференции "Информационные технологии интеллектуальной поддержки принятия решений", 12-14 ноября, Уфа-Баку-Чандигарх, 2024**

аварийных ситуаций, а также позволяют оптимизировать стоимость содержания дорог за счет автономного подсчета трафика на различных участках дорог.

В связи с этим возникает необходимость разработки интеллектуальных детекторов, которые выполняли функции обнаружения и классификации транспортных средств (ТС), ведут подсчет количества ТС по полосам движения, оценку плотности транспортного потока, его прогнозирование, классификацию «пробок» и аварийных ситуаций; оценивают скорость ТС, передают информацию на сервер. Вся эта информация может использоваться как для моделирования трафика в сложных дорожных сетях, интеллектуального регулирования светофорами и др.

Сегодня применение компьютерного зрения в виде нейронных сетей (НС) позволяет создавать интеллектуальные системы детекции и классификации объектов дорожного движения, определения положения, оценки расстояния между ними, автоматического сбора статистики на проезжей части, а также учета маршрута движения транспортных средств [3-6].

Обзор литературы и интернет источников показывает, что исследования и разработки в данной области сегодня очень востребованы и уже начинают конкурировать друг с другом по наличию интеллектуальных функций. Например, программное обеспечение AVEDEX [7] разработано для автоматического анализа автомобильного и пешеходного трафика по видеоизображению. Классификация транспорта осуществляется в соответствии с ГОСТ 32965-2014. AVEDEX работает с широким спектром видеокамер, при разных оптических схемах и ракурсах, включая вид с видеорегистратора и беспилотных летательных аппаратов. Программа определяет среднюю скорость потока и сигнализирует о заторах на дороге, не требует пусконаладки системы специалистами.

Другое решение Smart Traffic System [8] для подсчета количества ТС по направлениям движения, определения классификации типов ТС, расчета интенсивности, фиксации ДТП. Данное решение обладает кроссплатформенностью с возможностью развертывания его как на серверных мощностях, так и на встраиваемых устройствах с нейроускорителями. Smart Traffic System имеет удобный редактор позволяет самостоятельно размечать зоны детекции в виде полигонов без участия разработчиков.

Активно развиваются компактные аппаратные платформы для машинного обучения, например, такие как NVIDIA Jetson. В работе [9] представляется обзор работ по оценке и оптимизации приложений НС на платформе Jetson. Рассматривается как аппаратная, так и алгоритмическая оптимизация, выполненная для запуска алгоритмов НС на Jetson, демонтируются приложения алгоритмов. Приводится анализ платформ для исследования способов внедрения систем в микрокомпьютеры. Тяжелые архитектуры, требуют вычислительных ресурсов и оперативной памяти, их недостаток влияет на скорость обработки данных. Более того, сложные по времени алгоритмы тратят заряд аккумуляторов, подключенных к оборудованию. Параметры для анализа производительности определяются как потребление (GPU, CPU, RAM, потребляемая энергия). В работе [10] рассмотрена 2D – CNN сеть для задачи классификации с целью тестирования на различных микрокомпьютерах. Таким образом, исследование направлено на достижение высокой точности, высокой скорости работы алгоритмов при минимальных требованиях к оборудованию в приложениях глубокого обучения.

Кроме того, важным направлением для нашей задачи является использование NVIDIA DeepStream, которая предоставляет мощные средства для разработки приложений на основе потокового видео и глубокого обучения. NVIDIA DeepStream - это платформа для высокопроизводительной обработки видеопотоков в реальном времени [11]. Она включает в себя инструменты для захвата видео, декодирования, предварительной обработки, инференса с использованием моделей глубокого обучения, отслеживания объектов, постобработки и отображения результатов.

В текущей работе представлена разработка компактного детектора для идентификации транспортных объектов дорожного движения с функциями подсчета трафика, оценки скорости движения и координат перемещения. Данная разработка является продолжением исследований авторов [12-14]. Применение данного детектора планируется в специально оборудованных автомобилях (дорожных лабораториях) для оценки плотности потока в автоматизированном режиме, а также с возможностью установка на столбы оживленных перекрестков в защищенном боксе для оценки плотности потока и аварийных ситуаций в online режиме с передачей информации на сервер.

2. Подготовка данных и методы их преобразования

Для обучения детектора использовалось более 3500 размеченных фотографий с камеры городского видеонаблюдения по классам: автобусы, автомобили, пешеходы, грузовики, фургоны.

Так как городские камеры имели искажения – дисторсию, которая не позволяла строить четкие границы для вычисления скорости и координат перемещения ТС, в работе использовался метода Нелдера-Мида [15], который минимизировал аберрацию видеокамеры, вызывающую нарушение геометрического подобия между объектом и его изображением. На рисунке 1*а* показаны искривленная разделяющая (двойная сплошная) полоса потоков автомобилей и искривленный пешеходный переход.

На рисунке 1*б* эти искажения были компенсированы за счет устранения дисторсии, что вызвало соответственно искривление на границах изображений.

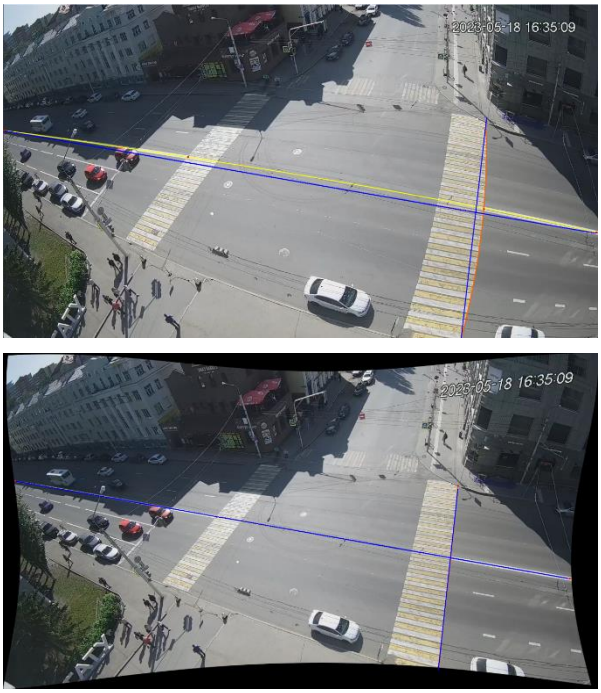

Рис. 1. Устранение дисторсии оптической системы видеокамеры

Далее применялись алгоритмы преобразования координат камеры в мировую систему координат с использованием метода прямого геометрического преобразования (direct linear transformation) и метод Чанга [16]. Метод метода прямого геометрического преобразования позволил перейти из одной системы плоскостей проекций в другую за счет создания линейных уравнений и помещения этих уравнений в матрицы таким образом, чтобы было возможным нахождение «неизвестного» уравнения. Суть метода Чанга заключается в незначительной модификации метода прямого геометрического преобразования. Преобразование точки из координат камеры в мировую систему координат выглядит следующим образом (1):

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix} \begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}, \quad (1)$$

где $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$ – это двумерная однородная точка изображения, $\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ r_{31} & r_{32} & t_z \end{bmatrix}$ – матрицы внутренних и внешних параметров камеры, $\begin{bmatrix} X_w \\ Y_w \\ 1 \end{bmatrix}$ – однородная Внутренняя матрица камеры имеет два разных фокусных f_x, f_y расстояния и оптические центры c_x, c_y камеры. Матрица внешних параметров имеет параметры вращения (ориентации) камеры r_{ij} и вектор перемещения t. Полное описание данных алгоритмов приведено в ранних работах авторов [12,13].

Далее на основе описанных алгоритмов выстраиваются опорные точки – угловые границы перекрестка, а также значения середин между ними. Иллюстрация опорных точек в координатной системе камеры представлена на рис. 2.

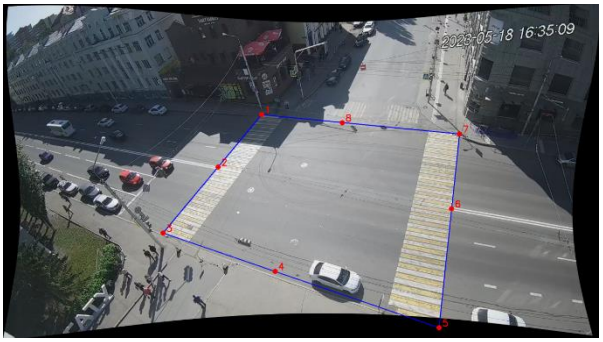

Рис. 2. Иллюстрация опорных точек в координатной системе камеры

3. Исследуемые архитектуры YOLO

Для создания детектора использовались архитектуры в виде нейронных сетей YOLO [17] (You Look Only Once), которые сегодня являются эффективными алгоритмами детекции объектов в реальном времени. YOLO разделяет изображение на сетку, где каждая ячейка предсказывает границы объектов и их классы, обрабатывая изображение целиком за один проход, что делает её быстрой по сравнению с многоэтапными методами.

Были исследованы архитектуры YOLO 5s, 8n, 9c и 10n, общие характеристики которых приведены в таблице 1. Приведено сравнение по параметрам:

- **mAP^{val}50-95** - средняя точность, усредненная по пороговым значениям IoU от 50% до 95%;
- **Speed CPU ONNX (ms)** - среднее время обработки одного кадра моделью, запущенной на центральном процессоре (CPU) с использованием ONNX Runtime;
- **Speed A100 TensorRT (ms)** - среднее время обработки одного кадра моделью, запущенной на графическом процессоре NVIDIA A100 с использованием TensorRT;
- **Params (M)** - количество параметров модели (в миллионах);

- **FLOPs (B)** - число операций с плавающей запятой в миллиардах (B).

Таблица 1. – Характеристики моделей Yolo на предварительно обученном наборе данных MS COCO

Модель	mAP^{val} 50-95	Speed CPU ONNX (ms)	Speed A100 TensorRT (ms)	Params (M)	FLOPs (B)
YOLO5s	43.0	120.7	1.27	9.1	24.0
YOLO8n	37.3	80.4	0.99	3.2	8.7
YOLO9c	53.0	-	-	25.5	102.8
YOLO10n	38*[2]	1.84*[3]		2.3	6.7

По моделям v9 и v10 отсутствует информация по Speed ONNX и Speed A100. Приведены похожие значения из официальных источников моделей [18].

YOLOv5 - первая модель, написанная на PyTorch. В нее добавлены модули Focus и CSPNet для улучшения производительности, что позволило модели достигать 140 FPS (кадров в секунду) на Tesla A100 [18]. YOLOv8 основана на YOLOv5 с улучшениями в головной модели, Feature Pyramid Network и Path Aggregation Network [19]. Модель достигает скорости до 150-160 FPS. YOLOv8 nano весит менее 5 МБ, что делает её подходящей для устройств с ограниченными ресурсами. YOLOv9 версия включает новаторские концепции, такие как программируемая градиентная информация (PGI) и обобщенная эффективная сеть агрегации слоев (GELAN), что способствует повышению эффективности и точности в задачах обнаружения объектов [20]. Архитектура YOLOv10 включает новаторский подход с согласованными двойными назначениями, что позволяет обучаться без необходимости в NMS (Non-Maximum Suppression), значительно снижая задержку вывода, при этом сохраняя конкурентоспособную производительность [21].

Для обучения детектора использовался узел суперкомпьютер УУНиТ с GPU Tesla A100, а также облачные мощности Google Colab (T4 GPU). Сравнительный анализ по метрике качества ранжирования mAP и времени обучения представлен в таблице 2.

Сравнительный анализ таблицы 2 показывает, что лучшие результаты по точности получены с архитектурами YOLOv5s и YOLOv8s, а по количеству параметров модели (ее размеру) и скорости инференса (см. таблицу 3) была выбрана архитектура YOLOv8n для интеграции в Jetson. В связи с необходимостью выполнении детекции в реальном режиме времени скорость инференса играет важную роль в скорости обработки данных и выдаче результата.

Таблица 2. – Сравнительный анализ обученных моделей

Модель	Batch size	Кол-во эпох	Время обучения в colab	Время обучения на суперкомпьютере	mAP
YOLOv5s	16	40	~90 мин.	-	0.81
YOLOv5s	16	150	-	54 мин	0.818
YOLOv8n	16	35	~75 мин	-	0.776
YOLOv8n	16	150	-	49 мин	0.809
YOLOv9c	16	30	~70 мин	-	0.789
YOLOv9t	16	150	-	98 мин	0.792
YOLOv10n	32	40	~80 мин	-	0.768
YOLOv10n	32	150	-	48 мин	0.786

Таблица 3. – Оценка моделей по скорости инференса

Модель	Скорость инференса, мс
YOLOv5s	223
YOLOv8n	112

4. Алгоритмы оценки скорости и координат передвижения ТС

Для решения проблемы переназначения id одного и того же ТС в связи с непостоянством фиксации объектов в online режиме, используются методы сохранения координат bounding box ТС на предыдущих кадрах с оценкой вектора их скорости для прогнозирования перемещения (рис. 3). Алгоритм предполагает, что ТС перемещается в том же направлении и с той же скоростью, что и ранее. Пусть положение объекта A описывается координатами $A(x, y)$ и имеется определенное положение одного и того же объекта на предыдущем $A(x_1, y_1)$ и текущем кадрах $A(x_2, y_2)$, тогда вектор скорости: $V_A = (x_2 - x_1, y_2 - y_1)$.

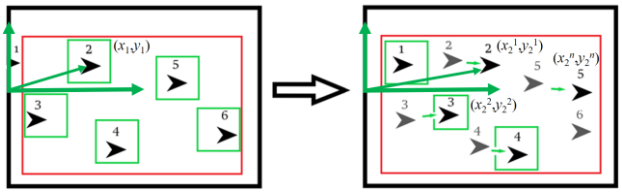

Рис. 3. Алгоритм сохранения номера ТС с учетом предыдущего состояния и вектора скорости

Предполагаемые координаты определенного ранее объекта определяются формулой:

$$A(x_i, y_i) = A(x_j, y_j) + (i - j) * V_A, \quad (j < i),$$

где i - номер текущего кадра, j - номер кадра, на котором последний раз был обнаружен объект.

[2] Average precision (AP) - метрика, которая рассчитывается как средняя точность на всех релевантных позициях в определенном списке.

[3] Latency, ms (задержка, мс) - это метрика времени, которое требуется для передачи данных по сети. Измерена с помощью TensorRT FP16 на графическом процессоре T4.

Скорость ТС считается относительно пройденного расстояния в пикселях в пределах 3 кадров для каждого объекта с учетом времени, проходящего за один кадр:

$$v_{км/ч} = \frac{d(C_1, C_2)}{10000} \cdot \frac{1}{3600 \cdot \text{FPS}},$$

где $d(C_1, C_2)$ - евклидово расстояние между объектом на текущем и предыдущем кадрах/

Для сглаживания скорости ТС и устранения погрешностей детекции нейросети были апробированы фильтр Калмана и взятие медиан с последующим их усреднением:

$$\frac{\sum x_{median(i)}}{n}.$$

Последний способ оказался более успешным, т.к. фильтр Калмана дает либо большое запаздывание при большем коэффициенте с учетом предыдущего сигнала, либо не может удалять из значения скорости большие выбросы, получаемые от определения скорости на текущем шаге.

Данный подход проверен с реальным автомобилем (id=5 на рис. 4), который проехал на контрольном участке со скоростью 57 км/ч. Работа алгоритма показала среднее отклонение в 5.14 км/ч (рис. 5).

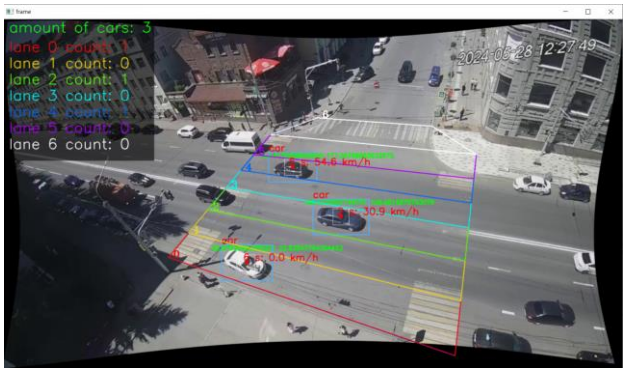

Рис. 4. Оценка скорости тестового автомобиля

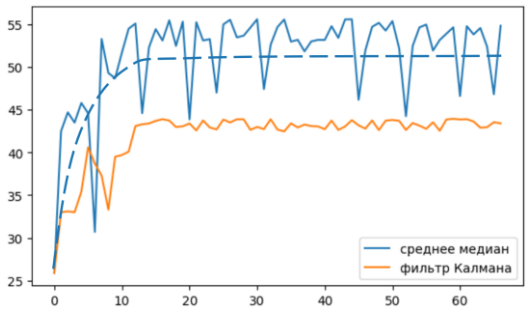

Рис. 5. Сравнение алгоритмов расчета скорости ТС

Для расчета координат перемещений ТС рассчитаны ширина и длина сторон перекрестка равные 29.4 метра и 24.4 метра соответственно (см. рис. 2), поставлены опорные точки для формирования координатной системы перекрестка: (0, 244), (0, 122), (0, 0), (147, 0), (294, 0), (294, 122), (294, 244), (147, 244). Графическое представление угловых точек из координатной системы перекрестка представлено на рисунке 7.

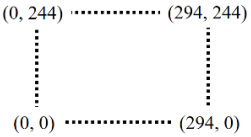

Рис. 6. Формирование координатной системы перекрестка для расчета перемещения ТС

Размеры перекрестка, а также координаты широты и долготы получены при помощи сервиса Яндекс Карты. Для решения системы уравнений с подставленными в нее значениями опорных точек использовался метод сингулярного матричного разложения (SVD) [22], который реализован в библиотеке NumPy. Погрешность вычисления координат перемещения ТС составила также 10%.

Стек технологий использующийся в детекторе представлен на рисунке 7. На Jetson, посредством перевода модели в формат TensorRT, а также задействование GPU, получилось достичь средней скорости обработки одного кадра 50 ms. Подключены библиотеки OpenCV, Pytorch, Pandas, NumPy, DeepSort. Проведенные эксперименты показали, что модель YOLOv8 успешно адаптирована для работы на платформе NVIDIA Jetson Xavier, обеспечив производительность для on-line детекции автомобилей с точностью 80% по метрике mAP50-9.

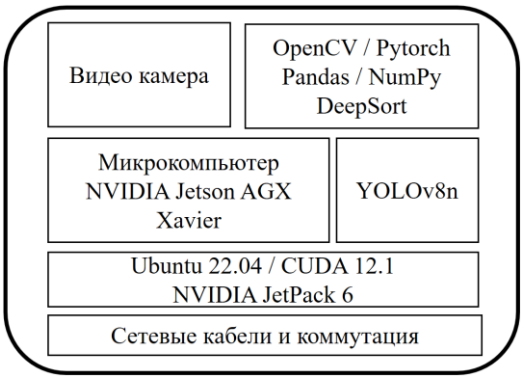

Рис. 7. Аппаратно-программная реализация на базе микроконтроллера NVIDIA Jetson

4. Заключение

Разработан интеллектуальный детектор для системы управления транспортными объектами города на базе NVIDIA Jetson AGX Xavier. Используется архитектура YOLOv8n для реализации функций идентификации и классификации транспортных объектов дорожного движения. Разработаны алгоритмы оценки скорости движения ТС и координат их перемещения. Для сглаживания скорости исследованы два метода - простой фильтр Калмана и последовательное применение взятия медианы и усреднения. Решена проблема переназначения номеров транспортных

объектов в связи со сбоями детекции в реальном времени за счет использования вектора скорости объекта. Оценка точности распознавания и классификации объектов составляет 80%, оценка погрешности вычисления скорости и координат передвижения объектов - 10%. Применение данного детектора планируется в дорожных лабораториях для оценки плотности потока в автоматизированном режиме, а также с возможностью установки на столбы оживленных перекрестков в защищенном боксе для оценки плотности потока и оценки аварийный ситуаций в online режиме с передачей информации на сервер.

Список используемых источников

1. Mohr D., Pokotilo V., Woetzel J. Urban transportation systems of 25 global cities. Elements of success // McKinsey & Company – 2021.
2. Ling S., Ma S., Jia N. Sustainable urban transportation development in China: A behavioral perspective // Frontiers of Engineering Management. 9(1), 2022. P.16–30. https://doi.org/10.1007/s42524-021-0162-4
3. Маннанов Р.В. Технологии компьютерного зрения в дорожной деятельности // Журнал «Дороги и мосты». – 2020. – С. 11–22.
4. Лукин В. Компьютерное зрение для аналитики дорожного движения: 3 успешных кейса // Журнал «Системы безопасности» №6/2020 – 2020. – С. 110.
5. Chebykin I.A. Automating road traffic monitoring using computer vision. // World of transport and transportation. 18(6), 2020. P. 74–87.
6. Соколов А.Е., Система технического зрения для учета движения на перекрестках // Вестник Новосибирского государственного университета. Серия: Информационные технологии. – 2012. – С. 87–92.
7. Сайт компании AVEDEX. URL:https://avedex.net/ (дата обращения 01.10.2024)
8. Сайт компании CodeInside. URL: https://traffic.codeinside.ru/ (дата обращения 01.10.2024)
9. Mittal S. A survey on optimized implementation of deep learning models on the NVIDIA jetson platform // Journal of Systems Architecture. 2019. Vol. 97. P. 428442.
10. Süzen, A.A., Duman, B., Şen, B. Benchmark analysis of jetson tx2, jetson nano and raspberry pi using Deep-CNN // 2020 International Congress on Human-Computer Interaction, Optimization and Robotic Applications (HORA). IEEE, 2020. P. 1–5.
11. Официальная документация DeepStream. URL: https://developer.nvidia.com/deepstream-sdk (дата обращения 01.10.2024).
12. Lopukhova E., Abdulnagimov A., Voronkov G., Grakhova E. Machine Learning-Driven Calibration of Traffic Models Based on a Real-Time Video Analysis // Applied Sciences (Switzerland), 2024, 14(11), P. 4864.
13. Abdulnagimov, A., Klyavlin, N., Alektorov, G. Traffic Object Detection System Based on YOLOv5 for V2V Communications // Proc. of 2023 International Russian Automation Conference, RusAutoCon 2023, 2023. P. 1022–1027.
14. Abdulnagimov, A., Lopukhova, E., Alektorov, G., Klyavlin, N. Intellectual lidar-based object classification for V2V communication technology implementation // ACM International Conference Proceeding Series, 2022. P. 672–678.
15. Zhang Z. A flexible new technique for camera calibration // IEEE Transactions on pattern analysis and machine intelligence. 2000. V. 22. №. 11. P. 1330-1334.
16. Documentation on scipy.optimize.minimize // Electronic resource. URL: https://docs.scipy.org/doc/scipy/reference/optimize.minimize-neldermead.html#optimize-minimize-neldermead/ (дата обращения 01.10.2024).
17. Redmon, J. et al. You only look once: Unified, real-time object detection // Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. P. 779-788.
18. Документация Ultralytics. URL: https://docs.ultralytics.com/ (дата обращения 01.10.2024).
19. Yaseen, M. What is YOLOv8: An In-Depth Exploration of the Internal Features of the Next-Generation Object Detector / arXiv, 2024. https://arxiv.org/html/2408.15857v1
20. Chien-Yao Wang, I-Hau Yeh, Hong-Yuan Mark Liao. YOLOv9: Learning what you want to learn using programmable gradient information / arXiv, 2024. P. 1-18. https://arxiv.org/abs/2402.13616
21. Ao Wang, Hui Chen, Lihao Liu, Kai Chen, Zijia Lin, Jungong Han, Guiguang Ding, YOLOv10: Real-Time End-to-End Object Detection / arXiv, 2024. P. 1-21. https://arxiv.org/abs/2405.14458
22. Singular Value Decomposition (NumPy Library) // URL: https://numpy.org/doc/stable/reference/generated/numpy.linalg.svd.html (доступ 01.10.2024).