Д. К. Зитцер, А. Я. Хатамов
Тюменский государственный университет, г. Тюмень
УДК 004.94

АВТОМАТИЗИРОВАННАЯ СИСТЕМА УПРАВЛЕНИЯ ДОСТУПОМ ТРАНСПОРТНЫХ СРЕДСТВ НА ПАРКОВКУ

Аннотация. В статье представлена возможность автоматизации доступа транспортных средств на парковку, путем распознавания автомобильных номерных знаков. Представляются способы и подходы по распознаванию номерных знаков, оценка и сравнение точности с аналогами.

Ключевые слова. Распознавание автомобильных номерных знаков, автоматизация парковки, нейронные сети.

Введение. С уверенностью можно сказать, что в третьем десятилетии XXI века, использование нейронных сетей для автоматизации задач бизнеса стали быстро растущей тенденцией в области информационных технологий. Целью публикации является поиск новых способов и методов создания модуля высокоточного распознавания автомобильных номеров, предварительно создав отлаженный конвейер по сбору и дообучению моделей нейронной сети. Создание такого модуля позволит использовать его в большом количестве сценариев с участием автомобилей (например: анализ загруженности улицы, платные дороги и парковки, организация беспрепятственного въезда на закрытые территории автомобилей экстренных служб и т. д.). В качестве бизнес-задачи был выбран сценарий «Автоматизации парковки предприятия».

Поскольку задача распознавания автомобильных номеров является одной из самых распространенных в компьютерном зрении, к настоящему времени были сформированы и протестированы различные подходы ее решения. Например, существует алгоритм увеличения точности распознавания, в случае, когда неверным оказывается только один символ распознанного номера [1]. Суть алгоритма заключается в получении дополнительной информации при распознавании номера, а точнее, данные о каждом распознанном символе (насколько он похож с каждым символом алфавита по меркам алгоритма), дополнительно обработав эту информацию,

можно дополнить базу распознанных номеров так называемыми “претендентами”, которые позволят получать больше совпадений при поиске в базе, при этом не меняя составных элементов системы распознавания [1]. Использование алгоритма преобразования Хафа для обнаружения номерного знака [2], сегментация текста и применение метода машинного обучения SVM (Support Vector Machine) для распознавания текста номерного знака [3]. Так как сегментация символов является самым распространенным способом распознавания текста, есть фундаментальные способы сегментации текста, такие как, использование заранее заданного шаблона, построение горизонтальной проекции средней интенсивности, проведение контурного анализа [4].

Процесс распознавания номерного знака условно можно разделить на следующие этапы:

1. Обнаружение и выделение номерного знака в кадре.
2. Предобработка и трансформация («выравнивание»).
3. Распознавание текста.

Технические характеристики локального и облачного стенда разработки указаны в табл. 1.

Таблица 1

Технические характеристики стендов разработки

Наименование	*Процессор (CPU)*	*Оперативная память*	*Видеокарта*
Acer Nitro 5 (локальный)	Intel Core i5 9300H	8GB	Nvidia GTX1650
Google Colab (облачный)	AMD EPYC 7B12	13GB	Nvidia Tesla T4

Проблема исследования. Основная проблема — сомнительная точность аналогов и большое количество условий при распознавании номерного знака (размер номерного знака, угол отклонения, высота установки камеры и т. д.). В связи с данной проблемой возникает потребность в создании высокоточного и универсального модуля распознавания автомобильных номерных знаков.

Материалы и методы.

Обнаружение номерного знака

В компьютерном зрении есть 2 основных способа обнаружения (далее «детекция») — нахождение ограничивающей рамки и сегментация (рис. 1) [5].

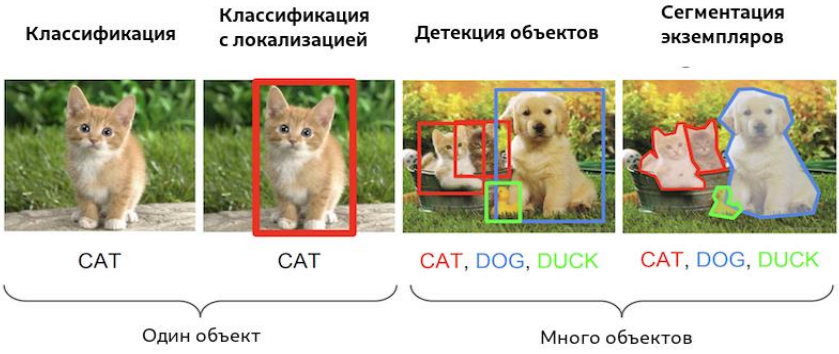


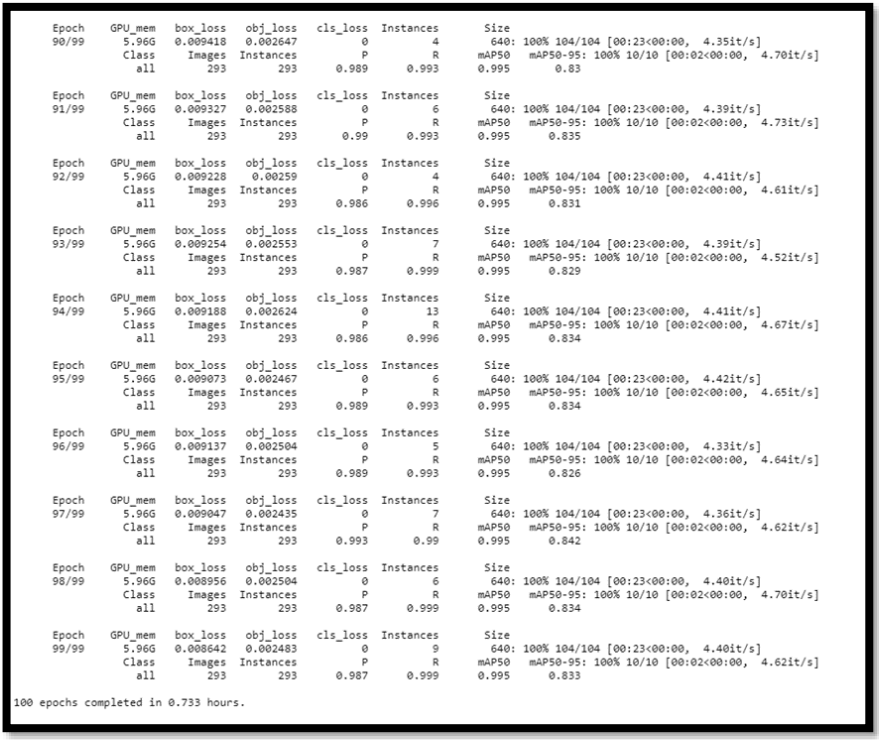
Рис. 1. Детекция объектов на изображении

Для обнаружения номерного знака более предпочтительным является детекция на основе ограничивающей рамки, так как сегментация — очень затратный процесс вычислительных процессов и нам необязательно знать точные границы номерного знака, ведь у нас далее будет возможность находить крайние точки номерного знака, благодаря обучению нейронной сети для поиска этих точек и опираясь на них проводить необходимые трансформации.

Т. к. предполагается использовать машинное обучение в данной и последующих задачах, а конкретно, обучение с учителем, нам необходимо выбрать архитектуру нейронной сети, собрать и подготовить набор данных. В качестве нейронной сети, которую предстояло обучить, было выбрано Open Source решение — YOLOv5, которое отличается приемлемой точностью и производительностью (еще один важный критерий выбора именно этой модели будет описан далее). Самый длительный процесс — сбор данных, он осуществлялся с помощью мобильной камеры и картинок из открытых источников данных (например, Platesmania [6]). В итоге данного шага

было собрано более 6000 картинок с автомобильными номерами и приведен к единому формату (название картинки — <номер_автомобиля>.jpg). Следующим шагом явилась разметка набора данных (т. е. указание «верных» ответов на картинке), с помощью инструмента MakeSense [7] (Не единственное решение в данной задаче). Итогом данного шага явился набор данных, состоящий из картинок и «правильных» к ним ответов (к каждой картинке соответствует текстовый файл с именем картинки). Перед запуском обучения необходимо разделить наш набор данных на обучающую и тестовую выборку. Делается это путем разделения набора данных на две части, в моем случае данные были разделены в соотношении 90% / 10%. Хочется подчеркнуть, что для обучения всех нейронных сетей была использована облачная среда разработки Google Colab, которая на бесплатной основе предоставляет доступ к большим вычислительным ресурсам. результат визуализации работы детектора и логи обучения можете увидеть на рис. 2 и 3 соответственно.

Рис. 2. Визуализация работы детектора



```
      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      90/99      5.96G   0.009418   0.002647          0          4        640: 100% 104/104 [00:23<00:00,  4.35it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.70it/s]
                   all        293        293      0.989      0.993      0.995       0.83

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      91/99      5.96G   0.009327   0.002588          0          6        640: 100% 104/104 [00:23<00:00,  4.39it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.73it/s]
                   all        293        293       0.99      0.993      0.995      0.835

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      92/99      5.96G   0.009228    0.00259          0          4        640: 100% 104/104 [00:23<00:00,  4.41it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.61it/s]
                   all        293        293      0.986      0.996      0.995      0.831

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      93/99      5.96G   0.009254   0.002553          0          7        640: 100% 104/104 [00:23<00:00,  4.39it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.52it/s]
                   all        293        293      0.987      0.999      0.995      0.829

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      94/99      5.96G   0.009188   0.002624          0         13        640: 100% 104/104 [00:23<00:00,  4.41it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.67it/s]
                   all        293        293      0.986      0.996      0.995      0.834

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      95/99      5.96G   0.009073   0.002467          0          6        640: 100% 104/104 [00:23<00:00,  4.42it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.65it/s]
                   all        293        293      0.989      0.993      0.995      0.834

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      96/99      5.96G   0.009137   0.002504          0          5        640: 100% 104/104 [00:23<00:00,  4.33it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.64it/s]
                   all        293        293      0.989      0.993      0.995      0.826

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      97/99      5.96G   0.009047   0.002435          0          7        640: 100% 104/104 [00:23<00:00,  4.36it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.62it/s]
                   all        293        293      0.993       0.99      0.995      0.842

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      98/99      5.96G   0.008956   0.002504          0          6        640: 100% 104/104 [00:23<00:00,  4.40it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.70it/s]
                   all        293        293      0.987      0.999      0.995      0.834

      Epoch    GPU_mem   box_loss   obj_loss   cls_loss  Instances       Size
      99/99      5.96G   0.008642   0.002483          0          9        640: 100% 104/104 [00:23<00:00,  4.40it/s]
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 10/10 [00:02<00:00,  4.62it/s]
                   all        293        293      0.987      0.999      0.995      0.833

100 epochs completed in 0.733 hours.
```

Рис. 3. Логи обучения

Выравнивание номерного знака

На первый взгляд задача кажется банально простой, но на ее решение ушла большая часть времени разработки. Ниже описаны методы, которые были призваны решить данную задачу, увы, но без успеха:

1. Контуры. Алгоритм этого подхода заключался в следующем: исходная картинка переводится в оттенки серого, накидывается небольшое размытие, для того чтобы избавиться от шумов на картинке, с помощью алгоритма Canny [8], находятся края объектов, которые далее преобразуются в контуры (рис. 4).

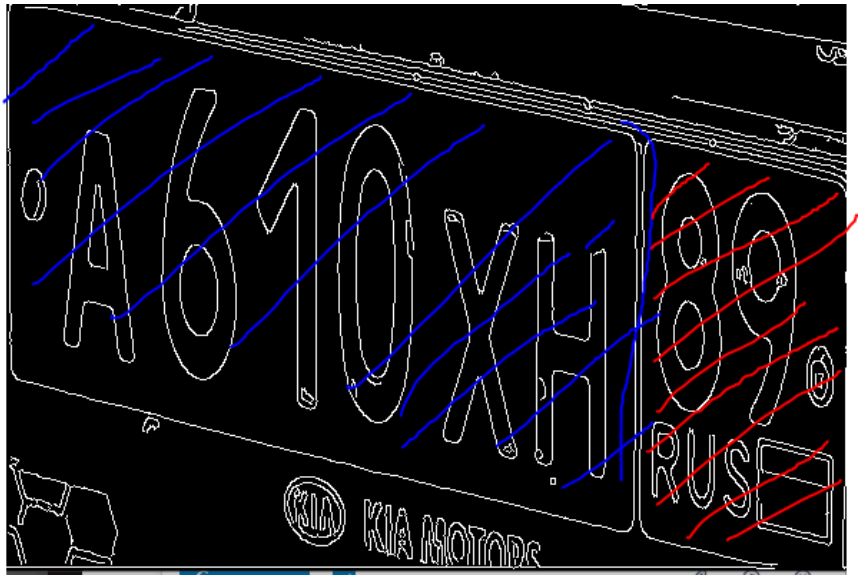


Рис. 4. Контуры на номерном знаке

Дальше находим самый большой контур, который и является номерным знаком. Недостаток данного алгоритма — на картинке мы явно можем наблюдать, что основная часть номера и часть номера, где указан регион, находятся на разных контурах. Можно было бы предположить, что самый большой контур — основная часть номера, а второй по величине контур — это часть, в которой указывается регион. Но проведя исследования на нескольких сотнях картинок, было выявлено, что не всегда второй по площади контур, будет являться частью номерного знака. Данный алгоритм прекрасно подойдет для номерных знаков, у которых нет разделения сплошной черной полосой (например, номера ЕС).

2. Определение номера по цвету, построение маски. После детекции номера на кадре, вырезаем эту область. Известно, что у номерного знака фон одноцветный и логично было бы предположить, что в прямоугольной области, где повторяются пиксели одинако-

вого цвета и является самими номерным знаком. Был написан алгоритм, который позволял бы находить 3 самых распространенных цвета на кадре. Следующий шаг заключался в том, чтобы с помощью HSV-маски, выделить область, где находятся эти самые часто встречающиеся пиксели. Далее взять 4 крайние точки этой маски и выровнять номерной знак (рис. 5).

Рис. 5. Визуализация алгоритма, с самыми часто встречающимися пикселями

Недостатком данного алгоритма явилось то, что цвет фона номерного знака может совпадать с цветом самого автомобиля и тогда края номерного знака будут находиться в самых крайних точках фрагмента с номерным знаком.

3. Text Detector CRAFT [10]. Суть этого алгоритма заключалась в том, чтобы найти на фрагменте с номерным знаком абсолютно весь текст, с помощью нейронной сети CRAFT, получить координаты текста и найти 4 крайние точки номерного знака (рис. 6).

Рис. 6. Text Detector CRAFT

Этот метод не дал результатов, потому что, как видно на рис. 6, выделено 4 отдельные зоны с текстом и, если рекламная надпись на рамке номерного знака будет протягиваться от начала номера и до самого его конца, появятся лишние области с текстом, которые нежелательны для этого алгоритма. В силу ограничений по времени не удалось написать универсальный алгоритм, который бы принимал все координаты этих четырехугольников и выдавал 4 точки (левая верхняя, правая верхняя, нижняя левая и правая нижняя).

4. Использование сегментации. В качестве модели для сегментации использовался MASK R-CNN. Т. к. для сегментации необходима другая разметка данных, пришлось еще раз обратиться к сервису MakeSense [7]. Разметил небольшой набор данных и обучил модель также в Google Collab. Точность была приемлемой, результаты визуализации на рис. 7.

Рис. 7. Сегментация номерного знака

Данный подход имел два существенных недостатка. Самый главный — это производительность, а также установка большого количества зависимостей. При таких манипуляциях, время, затрачиваемое на 1 картинку, составляло 1.3–1.9 секунды, что в наших реалиях очень много. Второй недостаток — расположение пикселей в маске. Модель на выходе давала примерно 2000 точек, которые и объединялись в маску. Весь подвох заключался в том, что всего лишь 1 пиксель, который выступал на несколько пикселей от самых крайних, портил весь алгоритм нахождения крайних точек номерного знака автомобиля (рис. 8). Учитывая, что начало координат на картинке

находится в левом крайнем углу, найдем правую верхнюю точку номерного знака. Для этого нам при максимальном значении X (правый край), необходимо найти минимальный Y (верхняя точка). В таком случае получим точку, отмеченную красным цветом на рис. 8.

Рис. 8. Проблема сегментации

Но все равно, самой главной причиной, по которой было принято решение отказаться от использования данного метода — производительность и смысла в дальнейших исследованиях по этому руслу, выявлено не было.

Раз мы можем обучить нейронную сеть находить номерной знак на кадре, то почему бы не обучить ее чтобы она искала 4 крайних точки номерного знака?. Ответ на данный вопрос был ключом к решению поставленной задачи. Набор данных использовался тот же, что и для обучения детектора, за небольшим изменением того, что картинки теперь содержали только область с номерным знаком. Была проведена разметка с единственным классом и получен один файл (количество строк равно количеству картинок в наборе данных), в котором «ответы» хранились в формате:

<Название картинки>, <TopLeft>, <TopRight>,
<BottomRight>, <BottomLeft>

В качестве модели выбрана модель EfficentNetV2s на архитектуре Keras, для «переоборудования» от обычного детектора, который выдает 2 координаты, в сеть, которая способна выдавать 4 координаты, необходимо изменить количество нейронов на последнем слое (рис. 9).

```
bboxHead = Dense(128, activation="relu")(flatten)
bboxHead = Dense(64, activation="relu")(bboxHead)
bboxHead = Dense(32, activation="relu")(bboxHead)
bboxHead = Dense(8, activation="sigmoid")(bboxHead)
```

Рис. 9. Слои нейронной сети

Используя функцию из библиотеки OpenCV (Стандарт работы с изображениями в компьютерном зрении) perspectiveTransform и передав в нее координаты точек номерного знака меняем перспективу так, чтобы номерной знак нормализовался (рис. 10).

Рис. 10. Результат выравнивания номерного знака

Распознавание текста

Последним шагом в распознавании номерного знака является распознавание текста. Перед использованием доступных Open Source решений, была попытка написания собственного «распознавателя» текста. Идея заключалась в том, чтобы использовать детектор YOLOv5, с 22 классами (количество уникальных символов на номерном знаке) и детектировать символы на фрагменте с номерным знаком, расположив их слева направо (рис. 11).

Рис. 11. Распознавание текста с помощью модели детекции

Полученная точность оказалась неприемлемой. Причина заключалась в недостаточном наборе данных. Дело в том, что такого рода разметка занимает в десятки раз больше времени, чем разметка но-

мерной рамки (т. к. разметка номера подразумевает выделение одного прямоугольника и такого же количества описания, а для разметки, которую необходимо провести для обучения собственного «распознавателя» необходимо отдельное выделение каждого символа (8-9 на номерном знаке) и такое же число описаний) и увеличение набора данных обернулось бы неделями или месяцами только разметки данных, при этом заранее сложно сказать, пришли бы мы к успеху или нет. Еще необходимо было обеспечить примерно равное количество объектов каждого класса. Например, если на всех номерах нашей выборки буква “А” встречается — 400 раз, то количество остальных символов тоже должно находиться в этом диапазоне. В ином случае модель будет легко распознавать букву “А”, но с распознаванием других символов могут возникнуть большие проблемы, что недопустимо в решении такой задачи. Ведь один неверно распознанный символ означает, что это совсем другая машина. Поэтому было принято решение использовать PaddleOCR в качестве модели распознавания текста. Paddle OCR [9] — проект от разработчиков PaddlePaddle, с открытым исходным кодом, позволяющий детектировать области с текстом, а потом извлекать его. Сложность заключается в ориентации текста, будь текст перевернут или под наклоном, данная нейронная сеть могла бы и не справиться. Но т. к., у нас есть решение, которое позволяет выравнивать номерной знак автомобиля, это не является помехой.

Для обучения этой модели был подготовлен набор данных из 54 000 картинок, в которых отражались уже «выровненные» номерные знаки. Обучение в 150 эпох длительностью в 20 часов дало хорошие результаты. Результат запуска модели на тестовой картинке представлен на рис. 12.

Рис. 12. Запуск модели распознавания текста

Результаты. Для проверки точности распознавания всего конвейера был сформирован набор, состав (рис. 13) и содержание которого отображено в табл. 2.

Таблица 2

Набор данных

	Изображений	*Номеров*	*Критерии*
Легкие кейсы	635	686	• разборчивые • дневные • в фокусе
Сложные кейсы	522	637	• смазанные • перекрытые • двухрядные • грязные • повернутые • коммерческие • ночные
Всего	1157	1323	

Рис. 13. Состав тестового набора данных

В сравнении участвовали «гиганты» среди Open Source решений по распознаванию номерных знаков. Результаты отражены в табл. 3. Также хочется отметить, что в таблице представлено 2 версии моей системы распознавания, которые отличаются объемом датасета, временем обучения и принципом предобработки.

Таблица 3

Сравнение точности распознавания

Продукт	*Правильно найденные*	*Неправильно найденные*	*Не найденные*	*Распознанные правильно*	*Распознанные с ошибкой*	*Не распознанные*	*Среднее количество операций*
Tevian [11]	1110	25	213	967	76	67	1.54
Nomeroff Net [12]	1082	29	241	873	172	37	2.5
АС управления доступом транспортных средств на парковку V1	1102	35	221	728	269	105	1.85
АС управления доступом транспортных средств на парковку V2	1174	47	149	895	182	97	1.5

«Правильное найденные» — обнаруженные номерные знаки, которые совпадают с правильными ответами.

«Неправильно найденные» — обнаруженные номерные знаки, которые не указаны, как верные.

«Не найденные» — номерные знаки, присутствующие в правильных ответах, но не обнаруженные детекторами.

«Распознанные правильно» — номерные знаки, с правильно распознанным текстом.

«Распознанные с ошибкой» — номерные знаки, с ошибкой в распознавании текста.

«Не распознанные» — номерные знаки, которые в распознанном тексте имели уверенность ниже порогового значения.

«Среднее количество операций» (Расстояние Левенштейна) — минимальное количество односимвольных операций (а именно вставки, удаления, замены), необходимых для превращения одной последовательности символов в другую [13].

ПРОИЗВОДИТЕЛЬНОСТЬ

На первой итерации разработки конвейера по распознаванию автомобильных номерных знаков количество кадров в секунду (далее FPS) составляло 10, чего недостаточно для обработки видео в реальном времени. Первым шагом оптимизации стало исключение повторного распознавания номера в кадре. Для этого была использована вертикальная линия, при пересечении которой и распознавался номерной знак, а также SORT-трекер [14]. Алгоритм заключался в следующем:

1) Номерной знак детектируется в кадре, трекер присваивает уникальный id, добавляется в список номеров в кадре с состоянием — «Не распознан».

2) При движении автомобиля пересекается вертикальная линия, установленная пользователем.

3) Происходит распознавание номерного знака и в списке номеров в кадре, номерной знак с данным id помечается как «Распознанный».

4) Если номерной знак с данным id остается в кадре, он не будет больше передаваться для выравнивания и распознавания текста, т. к. его статус — «Распознанный».

5) Спустя 5 кадров (регулируемый параметр) после того, как номерной знак исчез из кадра, данный id удаляется из списка номеров в кадре.

Данный алгоритм позволил реализовать как логику подъезда автомобиля к шлагбауму, так и повысить производительность до 15 FPS.

Далее описываются методы увеличения производительности, которые на данный момент не реализованы, но находятся на стадии планирования. Квантизация — это процесс уменьшения битности вычислений в нейронной сети. По умолчанию большинство нейронных сетей используют 32-битные вычисления с плавающей точкой (float32), и, соответственно, такие же 32-битные веса и активации. Если заменить их на 8-битные целочисленные операции (int8), можно сократить требования к памяти в 4 раза, а время инференса более чем в 4 раза. Последнее достигается благодаря использованию векторных и матричных операций, которые в случае 8 бит могут принимать в 4 раза больше аргументов за раз [15].

Преобразование в формат OpenVINO IR [16] — является одной из причин выбора описанных выше архитектур нейронных сетей. OpenVINO — набор инструментов, позволяющий оптимизировать и обеспечить просто исполнение моделей на устройствах от Intel. Еще одним неоспоримым плюсом данного решения является унифицированный интерфейс подготовки входных данных, запуска модели и обработки выходных данных. OpenVINO позволяет запускать модели как в синхронном, так и в асинхронном режиме. Основное преимущество асинхронного запуска заключается в том, что, когда устройство занято выводом, приложение может параллельно выполнять другие задачи (например, заполнение входных данных или планирование других запросов), а не ждать, пока текущий вывод завершится [16]. Данная процедура позволяет увеличить производительность почти в 2 раза (рис. 14).

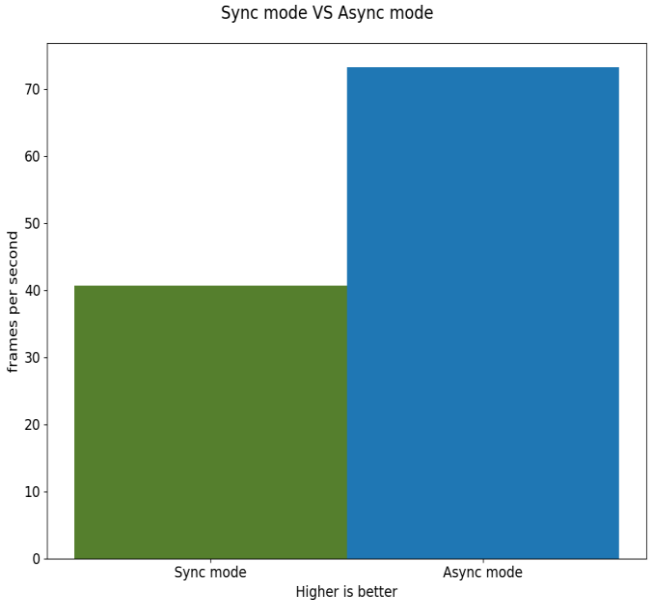


Рис. 14. Синхронный и асинхронный запуск с OpenVINO

УСТАНОВКА КАМЕР ВИДЕОНАБЛЮДЕНИЯ

Для определения высоты и угла установки камер видеонаблюдения был проведен ряд исследований. Исследование проводилось в следующих условиях (рис. 15):

- к стене прикреплен номерной знак;
- на расстоянии 2 м от стены, на штативе была установлена мобильная камера;
- с шагом изменения в 6 градусов были сделаны снимки номерного знака;
- проверялись 3 уровня высоты (номерной знак выше камеры на 1 м, ниже на 1 м и на одном уровне с камерой).
- Результаты показали, что приемлемые результаты достигаются в случае, когда камера стоит на одной уровне с номерным знак и когда камера выше номерного знака (рис. 16).

Рис. 15. Исследование угла распознавания текста

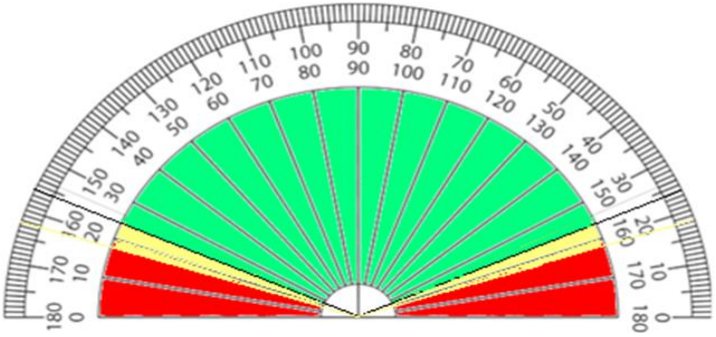


Рис. 16. Углы распознавания текста. Камера выше номерного знака на 1 м

Зеленым цветом отмечены углы, при которых текст распознается корректно, желтым — ошибка в 1 символе, красным — неприемлемая точность. Исходя из этих данных будет установлена камера видеонаблюдения (рис. 17).

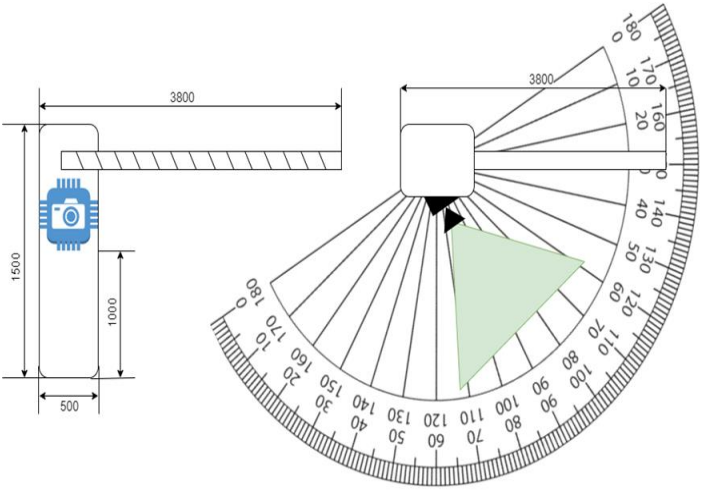


Рис. 17. Установка камеры видеонаблюдения

УПРАВЛЕНИЕ ШЛАГБАУМОМ

После распознавания номерного знака и проверки доступа транспортного средства на территорию парковки, необходимо отправить сигнал на шлагбаум, который будет поднимать стрелу. Для управления стрелой необходимо сомкнуть/разомкнуть цепь на шлагбауме, сделать это возможно с помощью WiFi-реле. Рассмотрим данный процесс на примере шлагбаума Came Gard 8 и WiFi-реле Sonoff Basic R3 (рис. 18).

Рис. 18. Sonoff Basic R3 / Came Gard 8

Для подачи команды на открытие необходимо проделать следующие шаги:

1. Подключить Sonoff Basic R3 к сети, в которой находится компьютер-отправитель сигнала. Это можно сделать с помощью мобильного приложения, следуя инструкциям из официальной документации.

2. Необходимо узнать IP-адрес данного Wi-Fi-реле Sonoff Basic R3.

3. Отправить запрос на Wi-Fi-реле, используя MQQT. MQTT — это протокол, которым устройства могут обмениваться сообщениями между собой по интернету. Он используется для связи между устройствами, которые работают на разных платформах и имеют различные ограничения по ресурсам и пропускной способности сети.

4. Отправляем запрос на языке Python, используя библиотеку paho-mqtt. Ниже представлен псевдокод, отправки запроса на Wi-Fi-реле:

Функция ОтправитьКоманду(Тема, Сообщение):

Создание объекта MQTT клиента;

Подключиться к Sonoff Basic R3, используя полученный IP (шаг 2);

Опубликовать сообщение по указанной теме;

Отключиться от Sonoff Basic R3.

Пример вызова данной функции:

ОтправитьКоманду("came-gard/command", "open")

Данная реализация не является единственной и в зависимости от протокола передачи сообщений, небольшие фрагменты данного решения могут быть изменены.

ИНТЕРФЕЙС УПРАВЛЕНИЯ СИСТЕМОЙ

На фреймворке Django был реализован веб-интерфейс с функциями:

- Установка вертикальной прямой, при пересечении которой распознается номерной знак.
- Регистрация, запись и хранение информации об операциях (въезд/выезд) на парковке.

- Экспорт истории операций.
- Управление списком разрешенных для въезда автомобилей.
- Просмотр камер видеонаблюдения.

Заключение. В результате работы разработан модуль распознавания автомобильных номеров и связана с информационной системой посредством HTTP-протокола.

Представленный в статье результат разработки может быть улучшен за счет увеличения набора данных и времени обучения моделей нейронной сети. Необходимо построить автоматический конвейер сбора и подготовки данных, автоматизировать дообучение моделей и тестирование точности на вышеописанном наборе данных.

СПИСОК ЛИТЕРАТУРЫ

1. Востриков М. С. Повышение достоверности результатов поиска автомобильных номеров с использованием модифицированного алгоритма распознавания государственных регистрационных знаков / М. С. Востриков, К. Л. Тассов // Cyberleninka: [сайт]. — URL: https://cyberleninka.ru/ article/n/povyshenie-dostovernosti-rezultatov-poiska-avtomobilnyh-nomerov-s-ispolzovaniem-modifitsirovannogo-algoritma-raspoznavaniya (дата обращения: 29.05.2023). — Текст: электронный.
2. Елизаров А. И. Методика построения систем распознавания автомобильного номера / А. И. Елизаров, А. В. Афонасенко // Cyberleninka: [сайт]. — URL: https://cyberleninka.ru/article/n/metodika-postroeniya-sistem-raspoznavaniya-avtomobilnogo-nomera (дата обращения: 29.05.2023). — Текст: электронный.
3. Кирпичников А.П. Автоматическое распознавание автомобильных номеров / А.П. Кирпичников, С. А. Ляшева, А. В. Обухов, М. П. Шлеймович // Cyberleninka: [сайт]. — URL: https://cyberleninka.ru/article/n/avtomaticheskoe-raspoznavanie-avtomobilnyh-nomerov (дата обращения: 29.05.2023). — Текст: электронный.
4. Обухов А. В. Методы автоматического распознавания автомобильных номеров / А. В. Обухов, С. А. Ляшева, М. П. Шлеймович // Cyberleninka: [сайт]. — URL: https://cyberleninka.ru/article/n/metody-avtomaticheskogo-raspoznavaniya-avtomobilnyh-nomerov (дата обращения: 29.05.2023). — Текст: электронный.
5. Задача нахождения объектов на изображении // neerc.ifmo : [сайт]. — URL: https://t.ly/91Jrr (дата обращения: 29.05.2023). — Текст: электронный.

6. Platesmania: Фотографии автомобилей, мотоциклов, прочих транспортных средств и номеров: [сайт]. — URL: https://platesmania.com (дата обращения: 29.05.2023).
7. Makesense: [сайт]. — URL: https://www.makesense.ai (дата обращения: 29.05.2023).
8. Canny Edge Detection // OpenCV: [сайт]. — URL: https://docs.opencv.org/3.4/da/d22/tutorial_py_canny.html (дата обращения: 29.05.2023). — Текст: электронный.
9. Github: PaddleOCR: [сайт]. — URL: https://github.com/PaddlePaddle/PaddleOCR (дата обращения: 29.05.2023).
10. Github: CRAFT text detector: [сайт]. — URL: https://github.com/fcakyon/craft-text-detector (дата обращения: 29.05.2023).
11. Tevian: AI-распознавание людей, объектов и событий: [сайт]. — URL: https://tevian.ai/ru/product/transport-recognition (дата обращения: 29.05.2023)
12. Github: [сайт]. — URL: https://github.com/ria-com/nomeroff-net (дата обращения: 29.05.2023)
13. Расстояние Левенштейна // Wikipedia: [сайт]. — URL: https://ru.wikipedia.org/wiki/Расстояние_Левенштейна (дата обращения: 29.05.2023)
14. Высокопроизводительный SORT-трекер // Medium: [сайт]. — URL: https://medium.com/inside-in-sight/high-performance-python-sort-tracker-225c2b507562 (дата обращения: 29.05.2023). — Текст: электронный.
15. Google Drive: Квантизация нейронных сетей: [сайт.] — URL: https://drive.google.com/file/d/1uBhIYXIjajhtyQhHlKIaZ2MNWELCb2io/edit (дата обращения: 29.05.2023). — Текст: электронный.
16. OpenVINO IR: OpenVINO documentation: [сайт]. — URL: https://docs.openvino.ai/2022.3/openvino_ir.html (дата обращения: 29.05.2023). — Текст: электронный.