SMART VEHICLE RECOGNITION AND SCHOOL PICKUP EFFICIENCY ENHANCING SYSTEM USING ARTIFICIAL INTELLIGENCE AND IOT SYSTEM

Nina Zhang [1], Austin Amakye Ansah [2]

[1] Fulton Science Academy, 3035 Fanfare Way, Alpharetta, GA 30009
[2] The University of Texas at Arlington, 701 S. Nedderman Drive, Arlington, TX 76019

ABSTRACT

StudentConnect is a system designed to streamline and improve the efficiency of student pickups at schools by automating parent arrival notifications. The system integrates two main components: PlateConnect, a license plate recognition system, and the StudentConnect mobile app [1]. PlateConnect uses a Raspberry Pi, Pi camera, YOLOv8, and OpenCV to capture and recognize vehicle license plates in real time [2]. Recognized plates are cross-referenced with a Firebase database to identify parents and their wards. Upon a match, an arrival notice is automatically generated and uploaded to Firebase, allowing school administrators to access real-time notifications via the StudentConnect app [3]. Parents can also view their notifications or manually notify the school of their arrival. Despite challenges such as limited hardware processing power and environmental factors affecting accuracy, the system significantly enhances pickup efficiency and safety. Future improvements include upgrading hardware and utilizing cloud services to optimize performance during peak times.

KEYWORDS

Raspberry PI, Student Management, Computer Vision, OCR

1. INTRODUCTION

In many schools, the process of managing student pickups is often inefficient and prone to delays. Typically, parents arriving to collect their children must signal their presence to school staff by waving papers or signs displaying their child's name. In some cases, staff members must manually call out the names of students to match them with their arriving parents, creating bottlenecks and increasing the chance of confusion, especially during peak times. This manual approach is not only time-consuming but also leaves room for human error, leading to potential mismatches, missed pickups, or extended waiting periods for parents and children.

Additionally, this traditional method lacks a reliable system for tracking when and which parents have arrived, making it difficult for schools to efficiently monitor and manage student handovers. In larger schools or during high-traffic periods, this can become overwhelming for staff, reducing the overall effectiveness of the pickup process [4]. Moreover, the lack of automation in this system presents challenges for security, as there is no automated method of confirming a

David C. Wyld et al. (Eds): SIPO, BDHI, NET, SOEA, CSML, AISCA, MLIOB, DNLP – 2025
pp. 01-10, 2025. CS & IT - CSCP 2025 DOI: 10.5121/csit.2025.150401

parent's identity through their vehicle, leaving schools reliant on visual cues and verbal confirmations.

To address these issues, a more efficient, automated solution is needed to streamline parent pickups, reduce errors, and enhance the overall safety of students. By leveraging technology such as license plate recognition and real-time notification systems, schools can modernize the pickup process, offering a smoother experience for parents and staff while ensuring the safe and timely release of students.

2. CHALLENGES

In order to build the project, a few challenges have been identified as follows.

2.1. Install Necessary Libraries

One challenge encountered during the development of the system was installing the necessary libraries, such as YOLOv8, on the Raspberry Pi. The Raspberry Pi restricts modifications to the system Python, which made it difficult to install specific packages and dependencies required for YOLOv8 and OpenCV [5]. Managing these dependencies without directly altering the system's Python environment required finding workarounds, such as using virtual environments or alternative Python installations. This added complexity to the setup process, as maintaining compatibility with the hardware and ensuring smooth performance of the plate recognition system was essential for real-time operation.

2.2. Get Character Detection Accuracy

Another challenge in the development of this system was getting character detection accuracy within levels that would be usable in production. Using PyTesseract initially, not only was OCR slow with the additional overhead caused by YOLO inference, but tesseract accuracy was very poor, especially for a crucial task like license plate reading where accuracy matters the most [6]. Tesseract is mostly optimized for moderate accuracy tasks like reading text for later editing.

2.3. Connection

Connecting to the raspberry pi would mean disconnecting the current monitor being used and connecting the keyboards and peripherals to the pi. This is a time consuming task and it is easier to connect to it using VNC or SSH, but VNC is preferred since there is also the need to view the raspberry pi as an operating system. The main issue with connecting to the pi with ssh or VNC was that the wrong hostname was being used and there wasn't a clear way of checking which one was the right hostname.

3. SOLUTION

The StudentConnect system integrates two primary components: PlateConnect for license plate recognition and the StudentConnect app for managing notifications. PlateConnect is deployed on a Raspberry Pi using a Pi camera to capture vehicle license plates. It leverages YOLOv8 and OpenCV for real-time object detection and optical character recognition (OCR) of license plates [7]. Once a plate is recognized, the PlateConnect system cross-references the captured data with the Firebase database to identify registered plates. If a match is found, an arrival notice is automatically generated, associating the parent with their respective wards (children) based on pre-existing relationships stored in the database.

The arrival notice is uploaded to Firebase in real time, making it immediately available to school administrators through the StudentConnect mobile app. Administrators can switch to an admin mode in the app to view all active notices, including the associated children and their parents' arrival status. Parents can log in to the same app to view their own arrival notices or manually notify the school by pressing an "I'm here" button if their plate was not automatically detected.

The Python-based PlateConnect system continuously monitors the Firebase database to detect and process newly captured license plates, ensuring seamless operation during peak pick-up hours. The Firebase backend serves as a secure repository for user data, plate details, and notifications, allowing synchronized access across devices for both parents and administrators. This setup significantly improves the school's ability to manage student pickups by providing timely and accurate information about parent arrivals.

The StudentConnect system comprises two key components: the PlateConnect license plate recognition system and the StudentConnect mobile application, working together to streamline student pickup operations. PlateConnect is deployed on a Raspberry Pi, equipped with a Pi camera for capturing images of vehicle license plates. Using the YOLOv8 object detection model and OpenCV's Optical Character Recognition (OCR), PlateConnect identifies license plates in real time as vehicles arrive at the school.

Upon successful plate recognition, the system compares the captured plate number with records stored in a Firebase database. The database holds parent-vehicle associations and their corresponding wards (children). When a match is found, the system generates an arrival notice that is automatically uploaded to Firebase. This notice indicates which parent has arrived and the students they are responsible for, enabling school staff to quickly coordinate student pickups. Administrators access the system through the StudentConnect mobile app, which provides real-time access to all arrival notices. In admin mode, staff can view a complete list of parents who have arrived and the corresponding students. Parents can also log in to the app to see their arrival status and children's information. If the system fails to recognize a parent's vehicle, the parent can manually notify the school using an "I'm here" button in the app.The system's use of Firebase ensures that both arrival notices and user information are stored securely in the cloud, allowing real-time synchronization across devices [8]. By automating vehicle recognition and integrating a notification system, StudentConnect significantly improves the efficiency and safety of school pickup processes.

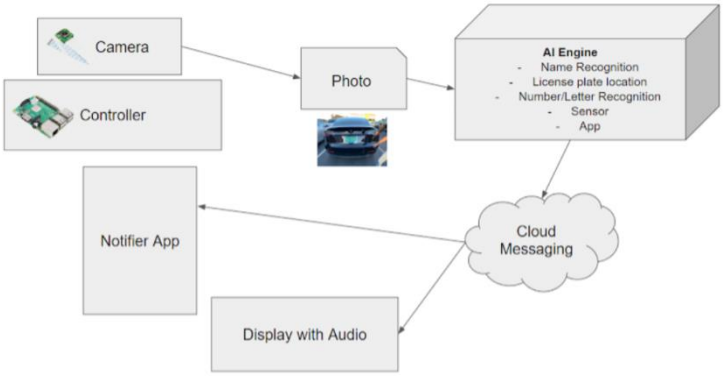

Figure 1. Overview of the solution

The first page that users encounter in the app is a dashboard to see which students have arrived. This also ensures shows whether they have checked in or not. The homepage displays the wards for the parent. If their parent's license plate was successfully scanned, they would see their child's information pop up on their screen. If the system failed at scanning their plate (for example, if they have a vanity license plate), they have the option of manually checking in their child.

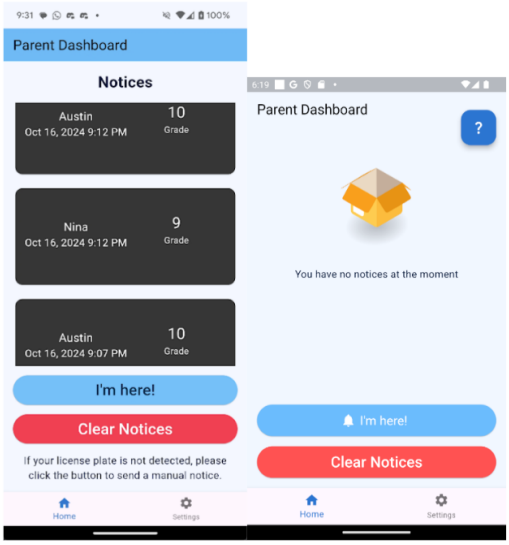

Figure 2. Screenshot of the APP

```
@override
Widget build(BuildContext context) {
  return Scaffold(
    body: Container(
      child: notices.isNotEmpty
          ? ListView.builder(
              itemCount: notices.length,
              itemBuilder: (context, index) {
                return ListBox(
                  name: notices[index].ward_name,
                  time: notices[index].arrival_time.toString(),
                  grade: notices[index].student_grade,
                ); // ListBox
              },
            ) // ListView.builder
          : Container(
              child: Center(child: Text("You have no notices today."))), // Contai
    ), // Container
```

Figure 3. Screenshot of code 1

Listbox is a custom card widget implementation made easy thanks to flutter's Widget api. It allows for the storing of more information than a usual ListTile. The List box was designed to store just the name, time of arrival, and grade of the student. This is enough information to identify the ward properly.

By default, notices are shown from the bottom up, so the latest arrivals are shown last, but the data in the database is sorted in order of arrival time. The UI is also in order of arrival time, but the listview reverses the order to show the latest last and scroll to the bottom.

The license detector module is a python class that runs in a pipeline manner. The procedure follows a four stage pipeline that is described by four states: Plate bounding box detection, Region cropping and filtering, text extraction, and parent notice creation.

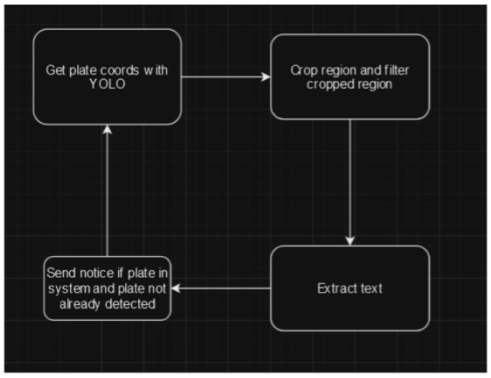

Figure 4. Screenshot of the procedure

LicensePlateDetector is a multi-function class that is used to run the main license extracting processes in the system. It designed such that libraries can be swapped out easily. For instance, the system needs to be tested on both windows or linux, and the raspberry pi, to make development quicker since the raspberry pi can be slow [9]. PiCamera2 and OpenCV can be interchanged so that the code also runs on the raspberry pi. PaddleOCR and PyTesseract can also be swapped. The current system usually attains good performance on stock videos.

Figure 5. Screenshot of plates

The process_frame function takes a frame from the camera or video feed, preprocesses the image to make text extraction easier, then uses yolo to obtain the bounding box coordinates for all license plates detected in the frame.

```python
class LicensePlateDetector:
    def is_valid_plate(self, text):
        return any(re.match(pattern, text) for pattern in patterns)

    def process_frame(self, frame):
        """Main processing pipeline"""
        if frame is None:
            return None, []

        original_frame = frame.copy()
        frame, gray, binary = self.preprocess_image(frame)
        detections = self.detect_license_plates(frame)

        results = []
        for detection in detections:
            x, y, w, h = detection['bbox']
            plate_img = gray[y:y+h, x:x+w]

            if plate_img.size == 0:
                continue

            plate_text = self.extract_plate_text(plate_img)

            if plate_text:
                results.append({
                    'text': plate_text,
                    'bbox': detection['bbox'],
                    'confidence': detection['confidence']
                })
```

Figure 6. Screenshot of code 2

Lastly, the license plate extraction is done to get the actual plate contents. Optionally, the OpenCV library is used to graphically display where the plates are in the frame using their bounding box coordinates [10].

The third vital component for this project was the feedback page. This is utilized to collect recommendations from the users of the app like school admins and parents. This helps with the further improvement of the system and its integration into the school as a vital component of ward pick ups.

Figure 7. Screenshot of settings

The feedback feature works by utilizing the system's email handling server to navigate to any mail app installed. When the 'send' button is tapped, the initial email body and subject is submitted through StudentConnect before being sent to the mail app. If an email is successfully sent without issues, then a snackbar showing the message status is displayed.

```
const SizedBox(height: 24.0),
ElevatedButton(
  onPressed: () async {
    FlutterEmailSender.send(Email(
        body:
            "Message: ${messageController.text}\n",
        subject: "[StudentConnect] Support & Feedback",
        recipients: [
          "studentconnecthelp123@gmail.com"
        ])).then((value) {
      ScaffoldMessenger.of(context)
          .showSnackBar(const SnackBar(
              content:      Austin A., 1 week ago
                  Text("Message sent")));
      if (!mounted) return;
      Navigator.pop(context);
    });
  },
  child: const Text("Send"),
  )
 ],
),
```

Figure 8. Screenshot of code 3

4. EXPERIMENT

4.1. Experiment 1

We sought to increase license plate detection accuracy by testing the plate reading pipeline with PyTesseract and PaddleOCR in separate test runs.

To assess the accuracy of Pytesseract and PaddleOCR, a dedicated test module was developed and executed on a dataset of 10 diverse images. We hypothesized that PaddleOCR would demonstrate superior performance compared to Pytesseract. The evaluation involved running an identical processing pipeline for both libraries, with each being tested in separate runs. The resulting datasets were compiled and analyzed, utilizing Matplotlib for visualization. This structured approach enabled a clear comparison of the two libraries, highlighting their strengths and weaknesses in license plate text extraction.

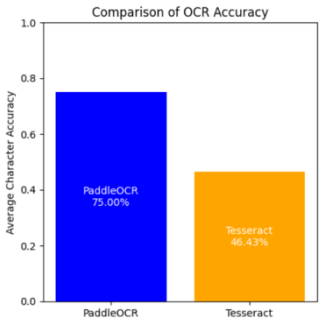

Figure 9. Figure of experiment 1

Extracting license plate text on still targets like a single image will sometimes result in the wrong lines on the plate being detected since the detection is done in one pass. As compared to video frames, the pipeline always detects the correct license plate at least once, and this is enough to send a notice. With a still image test, the chances of proper extraction are slightly skewed. From the results, paddle correctly extracted 75% of the characters read correctly, whereas tesseract only successfully did this 46% of the time on the plates passed to it.

During testing of the system, tesseract was shown to perform poorly on license plates, often reading certain characters wrong, like A as 4. Paddle performed much better on all targets.

With Paddle, 8/10 files were read successfully, whereas tesseract could only correctly scan 4/10 of the files with the current pipeline.

4.2. Experiment 2

Since the raspberry-pi 4b used in the project has no gpu and only comes with an on-board CPU, it is important to test the performance of both models and weigh their pros and cons.

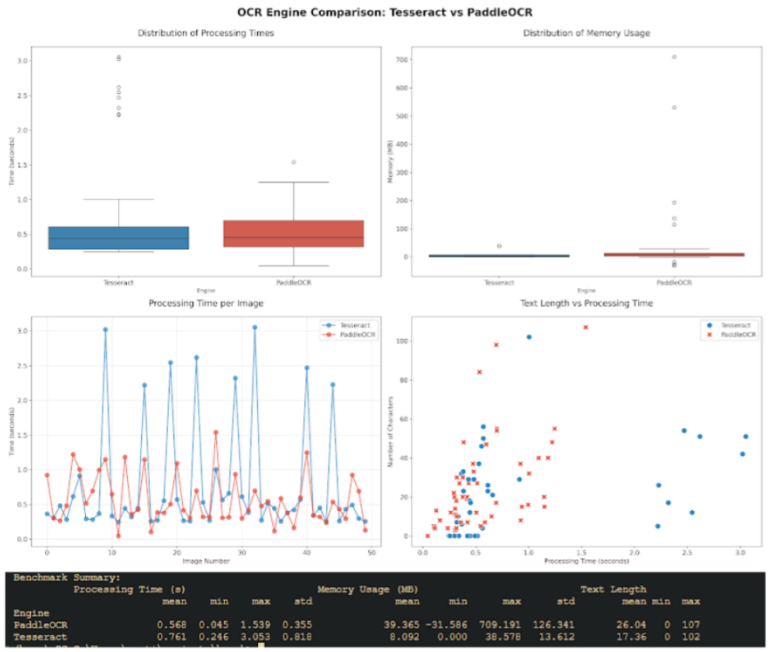

Figure 10. Figure of experiment 2

The results show that Paddle has a higher inference speed (lower time taken) than Tesseract on most tasks.

Both models are relatively fast on a small character size, but Tesseract appears to be faster in some cases. Paddle uses much more memory than Tesseract.

When it comes to memory usage, PaddleOCR is shown to draw more power during inference, but the time it takes for Paddle to infer text compared to Tesseract far outweighs that drawback.

5. RELATED WORK

This paperdescribes an implementation of a marksheet parser using PaddleOCR[11]. Their initial PyTesseract implementation was slow, leading to them using PaddleOCR for optical character recognition instead. Using the bounding boxes provided by the output from Paddle, they were able to figure out which fields went to certain boxes. For instance, grades would have boxes with certain widths and could be treated as such.

In this article, the authors use Flutter and Firebase to build a qrcode based parking check-in system[12]. Users scan a qr code with the app to verify that they have parked, and their parking status is stored in firebase. Parking attendants are then able to see which people have parked or

left the parking lot. They use firebase to store check-in data much like how the studentconnect system creates notices after scanning a license plate.

Theyemploys the same method for capturing license plate information[13]. First a YOLO pipeline is used to capture the license plate bounding box, then gaussian thresholding is used to filter the image before passing it to paddleOCR for text extraction. The difference is that they use their own pre-trained model for YOLO inference. PaddleOCR works out of the box like in the studentconnect system.

6. CONCLUSIONS

One limitation of the StudentConnect system is its reliance on the Raspberry Pi, which has limited processing power. This can lead to delays in real-time license plate recognition, particularly when handling multiple vehicles during peak pickup times [14]. Additionally, environmental factors such as poor lighting or adverse weather can impact the accuracy of the Pi camera and the YOLOv8 model, leading to potential misreads of license plates.

To improve system performance, future iterations could incorporate a more powerful hardware setup or offload some of the processing to cloud-based services, enabling faster and more accurate plate recognition [15]. Enhancements to the camera, such as adding infrared or higher-resolution options, could also improve performance in low-light conditions. Moreover, expanding the system to support multi-camera setups could allow for greater coverage and more efficient handling of high traffic. Regular updates to the YOLO model and OCR algorithm could further boost recognition accuracy.

REFERENCES

[1] Riaz, Waqar, et al. "YOLO based recognition method for automatic license plate recognition." 2020 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA). IEEE, 2020.

[2] Revathi, A. S., and Nishi A. Modi. "Comparative analysis of text extraction from color images using tesseract and opencv." 2021 8th International Conference on Computing for Sustainable Global Development (INDIACom). IEEE, 2021.

[3] Καπετάνος, ΓεώργιοςΙ. Combining MLOps and low-power networking in edge computing applications. BS thesis. 2023.

[4] Ross, Zev, et al. "Noise, air pollutants and traffic: Continuous measurement and correlation at a high-traffic location in New York City." Environmental research 111.8 (2011): 1054-1063.

[5] Kim, Hyun-Tae, and Sang-Hyun Lee. "A study on object distance measurement using opencv-based yolov5." International Journal of Advanced Culture Technology 9.3 (2021): 298-304.

[6] Mori, Shunji, Ching Y. Suen, and Kazuhiko Yamamoto. "Historical review of OCR research and development." Proceedings of the IEEE 80.7 (1992): 1029-1058.

[7] Mithe, Ravina, Supriya Indalkar, and Nilam Divekar. "Optical character recognition." International journal of recent technology and engineering (IJRTE) 2.1 (2013): 72-75.

[8] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.

[9] Richardson, Matt, and Shawn Wallace. Getting Started with Raspberry Pi: Electronic Projects with Python, Scratch, and Linux. Maker Media, Inc., 2014.

[10] Bradski, Gary. "Learning OpenCV: Computer vision with the OpenCV library." O'REILLY google schola 2 (2008): 334-352.

[11] Bagaria, Sankalp, and S. Irene. "A Novel Implementation of Marksheet Parser Using PaddleOCR." arXiv preprint arXiv:2407.11985 (2024).

[12] Agarwal, Nirbhay, et al. "Authentication Of Vehicle In Parking Using Flutter And Firebase." Journal of Pharmaceutical Negative Results (2023): 5545-5553.

[13] Sarkar, Oshmita, et al. "Automatic Number Plate Character Recognition using Paddle-OCR." 2024 International Conference on Innovations and Challenges in Emerging Technologies (ICICET). IEEE, 2024.

[14] Khawas, Chunnu, and Pritam Shah. "Application of firebase in android app development-a study." International Journal of Computer Applications 179.46 (2018): 49-53.

[15] Kim, Kap Kee, et al. "Learning-based approach for license plate recognition." Neural networks for signal processing X. Proceedings of the 2000 IEEE signal processing society workshop (Cat. No. 00TH8501). Vol. 2. IEEE, 2000.