Efficacy of Basic Construction Vehicle Monitoring (CVM) Systems

Alexander Martin

Table of Contents:

Introduction:

The research outlined in this paper investigates the ability of basic Construction Vehicle Monitoring (CVM) systems to provide practical insight into operator efficiency and safety. Additionally, this paper explores factors that contribute to a CVM system's performance as well as measures the difficulty of developing a basic CVM system.

Due to the large size of many construction sites, it can be difficult to monitor the movement of construction vehicles. This can be a compounding issue, to the extent that some contractors "don't even know where their equipment is located" on the site. It is important to know how vehicles move around a site in order to calculate the productivity of each vehicle. If construction sites were to implement CVM systems, it could enable construction project managers to more accurately estimate metrics of productivity by providing real data about each vehicle's movement. Additionally, the safety of vehicle operators could be evaluated with this data. Vehicle speed and acceleration can be measured with CVM systems, which may offer perspective into how safely vehicles are maneuvered and operated around the site.

To conduct the research, a basic CVM system was assembled. The CVM system was composed of a prototyping computer board, a camera module, and a vehicle sensing program. Each component was selected for high accessibility for contractors seeking to assemble their own CVM system. Vehicles were to be recognized by the CVM system with unique QR codes printed visibly upon their tops. Each QR code was to represent an equipment number to ensure compatibility with vehicle identification standards many contractors and rental companies already practice. The CVM system was installed above a simulated construction site to easily monitor model vehicles without any potential view obstructions.

Assembling the CVM System:

The chosen prototyping computer board was the Raspberry Pi Zero WH (Zero). The Zero is well regarded as a low-cost and high-capability computer board for a wide range of projects, with direct support for camera modules.[1] The chosen camera module was the Raspberry Pi Camera Module 2 (Camera), featuring a fixed-focus lens and direct compatibility with the Zero.[2] A 15 Pin to 22 Pin cable adapter was required to connect the Camera to the Zero. The Zero and the camera module were ultimately selected for their built-in compatibility together (see ***Figure 1***). A microSD card was also required to store the operating system and other relevant files. The vehicle sensing program was written in the Python programming language, due to its ease of use and its direct compatibility with both the Zero and Camera when using a Raspbian-based operating system.[3] [4] The chosen Raspbian-based operating system for the Zero was a version of Bookworm OS because of its support for newer Python versions and its compatibility with both the Zero and the Camera. Each of the hardware components are inexpensively available for purchase at online marketplaces. Bookworm OS and Python are each available for free on their respective foundations' websites (Raspberry Pi Foundation, Python Software Foundation).

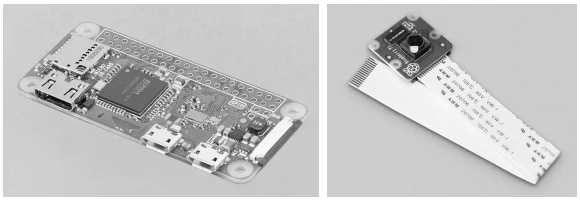

Figure 1: The Zero (left) and the Camera (right)

The Raspberry Pi Zero WH and the Raspberry Pi Camera Module 2 were both constituents of the CVM system, chosen because of their direct compatibility.

To further simplify the ease of programming (beyond selecting a beginner-friendly programming language), generative AI was asked to design the vehicle sensing program. AI needed three attempts to construct a functioning program. To better tailor the data output from the AI-written program, one modification was personally performed to the program after the AI's third attempt in lieu of risking further issues with more attempts (see ***Figure 2***). However, this modification was not necessary; data could still have been manually recorded from the program's realtime output. The program written by AI required only one dependency (Pyzbar) which was easily installed using a Linux-based command (apt install python3-pyzbar). After installation, the program functioned without issue. Per the program, the position of each vehicle was reported in pixels in the Cartesian X and Y directions relative to the CVM system. If desired, the position could be easily converted to a more common metric used in construction (feet, yards, miles, etc). For this research, however, it was decided that such conversions were unnecessary and beyond the scope of testing the CVM system's efficacy. The time of detection for each vehicle was measured by the Zero's onboard clock in the local timezone.

```
83                    print(f"DETECT {time.strftime('%H:%M:%S',
time.localtime(now))} ID={q} X={cx:.1f} Y={cy:.1f}")
84            # tiny sleep to avoid starving CPU if necessary
85            time.sleep(0.005)
```

```
83                    print(f"DETECT {time.strftime('%H:%M:%S',
time.localtime(now))} ID={q} X={cx:.1f} Y={cy:.1f}")
84                    with open(f"{q}.txt", "a") as file:
85                        file.write(f"ID={q} X={cx:.1f} Y={cy:.1f} TIME=
{time.strftime('%H:%M:%S', time.localtime(now))}\n")
86
87             # tiny sleep to avoid starving CPU if necessary
88            time.sleep(0.005)
```

Figure 2: Snippets of code from Text Compare Tool of the unmodified vehicle sensing program (top) and the modified vehicle sensing program (bottom) [5]

A line of code personally added to the AI-written program specifies an additional way for data to be recorded from the program. This is the only modification made to the AI-written program.

It was decided that due to cost and space constraints, the CVM system would be tested in a scaled-down simulation of a construction site — it would monitor vehicles of several inches in length from a vertical distance of one foot away. However, in practice, the CVM system should be able to be scaled up for an actual construction site (or a section of a site thereof) by focusing the Camera's lens to monitor life-size vehicles from any given elevation.

Once the components were connected and installed, a simple experiment was conducted to ensure that they were functioning as expected. A QR code was displayed at varying distances from the CVM system to test the Camera's fixed-focus lens and the program's capability of perception. After much fine tuning of the lens, the program was able to accurately recognize the QR code and describe its movement from one foot away (see ***Figure 3***). Focusing the Camera was difficult, requiring multiple adjustments to the lens before it could effectively perceive a QR code (see ***Figure 4***).

Figure 3: The experimental setup for testing the functionality of the CVM system's components

A phone on a stand (left) displays a QR code for the CVM system (right) to detect. A laptop (back right) logs output from the program. The Camera's lens was focused until the program could recognize the QR code from one foot away.

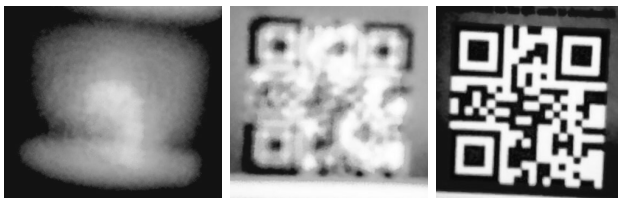

Figure 4: Camera perception after focusing the lens once (left), six times (middle), and thirteen times (right)

After thirteen attempts of focusing the lens, the CVM system was able to effectively recognize QR codes.

Testing the CVM System:

After the CVM system was determined to be functional, the CVM system was installed one foot above a simulated construction site (see ***Figure 5***) and model vehicles (see ***Figure 6***) were moved beneath it. Each model vehicle was about 4 inches long and 1 inch wide, and featured a unique QR code on its top. To further test the perception of the CVM system, the sizes of the QR codes on many of the model vehicles were varied (by nearly as much as two times). Model vehicles were moved both in random patterns and in mock scenarios to best model all construction activities. Scenarios included loading trucks with excavated materials (with both excavators and front-end loaders), emergency field fueling, and multi-vehicle operations (see ***Figure 7***). When a vehicle was detected, the program recorded its time of detection and its position in a file corresponding to that vehicle's equipment number. In addition, the program outputted these measurements for easy review (see ***Figure 8***).

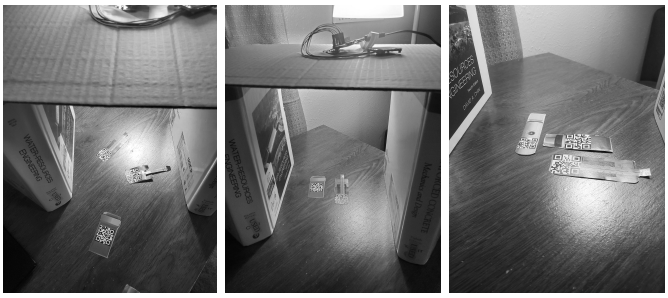

Figure 5: Installation of the CVM system

The CVM system was installed (propped upon two books and a cardboard sheet) one foot above a simulated construction site (the area of the tabletop between the two books). A lightbulb illuminated the simulated construction site to represent light from the sun or construction lamps. Model vehicles were moved beneath the CVM system in the simulated construction site.

Figure 6: Model vehicles used to test the CVM system [6]

Model vehicles were used in place of life-size vehicles. Each model vehicle featured a unique QR code.

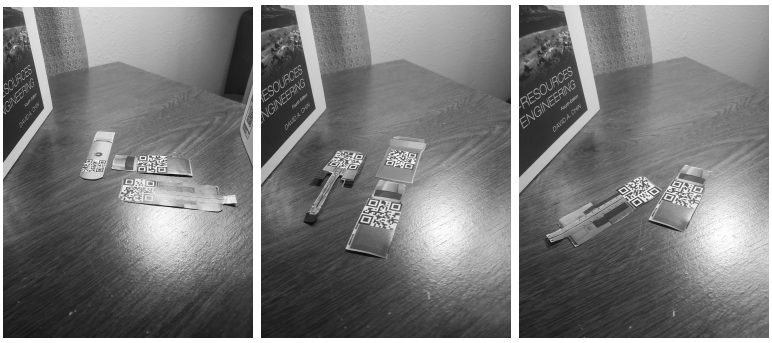

Figure 7: Acting scenarios with model vehicles

Model vehicles moved around the simulated construction site engaging in various construction activities.

```
DETECT 20:35:25 ID=444 X=419.0 Y=371.0
DETECT 20:35:49 ID=444 X=306.2 Y=371.0
DETECT 20:35:52 ID=444 X=306.2 Y=370.5
[20:36:02] frame #31 shape=(480, 640, 4) fps=0.3
DETECT 20:36:11 ID=444 X=218.0 Y=374.8
DETECT 20:36:14 ID=444 X=217.8 Y=375.0
DETECT 20:36:17 ID=444 X=217.2 Y=374.8
```

Figure 8: A snippet of output from the vehicle sensing program

When a vehicle was detected, the program outputted measurements of the vehicle's position (pixels) and the current time (hours:minutes:seconds).

Analyzing Performance of the CVM System:

The CVM system demonstrated general capability of detecting vehicles. In addition, it was able to detect multiple vehicles moving around the site simultaneously! However, it struggled to detect vehicles with smaller QR codes, often failing to detect them entirely. Glare from the simulated sunlight/lamplight reflected into the Camera's lens, also impacting the readability of QR codes by the CVM system in certain locations (see ***Figure 9***).

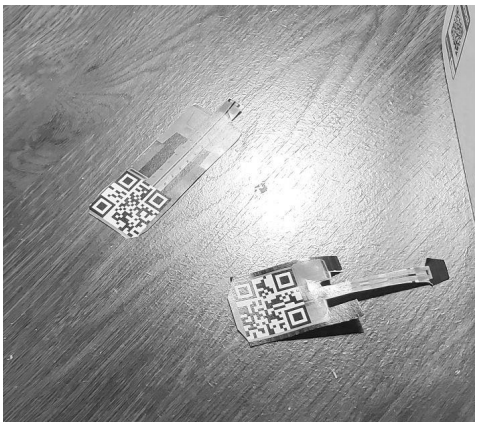

Figure 9: Glare from the simulated sunlight/lamplight

Substantial glare from the simulated sunlight/lamplight impacted the ability of the CVM system to recognize QR codes.

The CVM system processed visual input at a moderate speed. Per the program, a video stream was initiated to continuously collect visual input. Once initiated, individual frames from the stream were captured and analyzed. From capture to finishing analysis, the CVM system processed frames at nearly 15 Frames Per Second (FPS). The Camera itself is capable of achieving much higher FPS; delays in processing emerged from the analysis of each frame. [7] Some dependencies used to analyze each frame (such as Pyzbar) can reduce the overall speed of the program.[8]

Despite any limitations in the CVM system's processing speed, it was still able to record enough measurements to model each detected vehicle's movement with sufficient definition. Enough position data was collected within an appropriate span of time to effectively determine vehicle movement in the X and Y directions (see ***Figure 10***) and to depict movement around the site graphically (see ***Figure 11***).

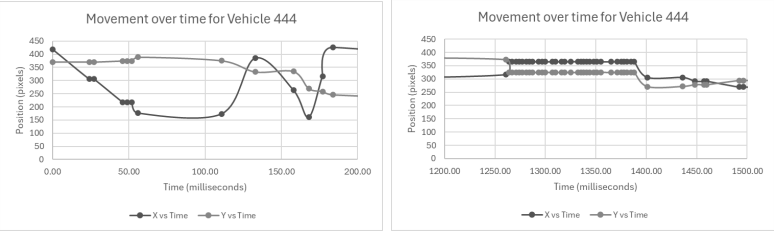

Figure 10: Movement of Vehicle 444 in each direction

The recorded data from the CVM system has enough definition to accurately model each vehicle's movement. Shown above are two excerpts from the recorded data for Vehicle 444, each depicting movement in the X and Y directions over a span of time.

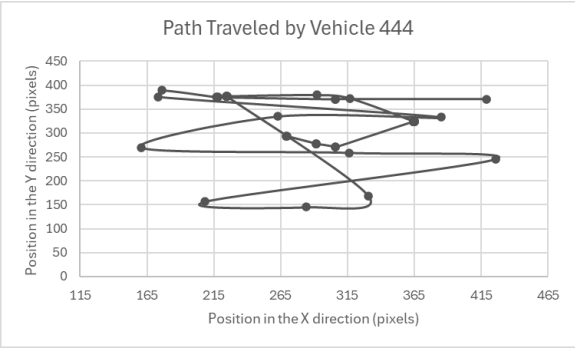

Figure 11: The path travelled by Vehicle 444

The recorded data from the CVM system can be applied to determine the path of a vehicle. Depicted above is the path traveled by Vehicle 444 during the experiment, as recorded by the CVM system.

Conclusions:

In general, under ideal conditions, the CVM system proved to be effective. It detected vehicles moving around the site and recorded data about how the vehicles were operated. After collecting data, the vehicle's position, speed, and acceleration could be calculated to yield numerous metrics about operator use of the vehicle. In addition, this data could be used to determine where vehicles are located on the site!

Several factors did impact the efficacy of the CVM system. Firstly, the CVM system struggled to recognize smaller QR codes. On a real construction site, differing vehicle shapes and sizes might necessitate smaller or larger QR codes; the CVM system might only intermittently recognize vehicles or only recognize vehicles with an ideally sized QR code. Glare

from the simulated sunlight/lamplight also affected the system's recognition of vehicles. This issue might be more problematic on real construction sites, affecting the recognition of any vehicle regardless of the size of its QR code. Lastly, the range of view of the Camera was limited by the focus of the lens. On a construction site, to provide practical data, the Camera's elevation and focus should be optimized.

For a contractor developing a similar CVM system without computer-technical knowledge, a satisfactory system shouldn't be too difficult to produce. Firstly, each hardware component can be easily sourced from online marketplaces and is directly compatible with the other components. Secondly, the programming of the CVM system is largely accomplished by AI; very little programming (if any) is required to write a functional program. Some difficulties may arise from prompting the AI for the necessary characteristics of the program, testing the program, and requesting changes where needed. Assistance from AI and online forums could provide help for writing and testing the program. Additional difficulties may arise when setting up the Zero and navigating Bookworm OS; Bookwork OS can be used with either a command line interface or with a desktop interface, and can even be used without a monitor screen! To avoid any potential challenges, non-technical contractors should consider using a monitor screen with the desktop interface of Bookworm OS.

For large construction sites, to capture all construction activity while best optimizing the elevation and focus of the Camera, it may be preferable to simultaneously use multiple CVM systems around the entire site. While multiple independent CVM systems could be installed and would return practical data, it might be most advantageous to utilize a shared storage system between each CVM system and to record the coordinates of each vehicle with a coordinate system relative to the entire network of CVM systems. This could streamline analysis of the data by reducing duplicate and simplifying position tracking.

References:

1. Raspberry Pi Foundation. n.d. “Raspberry Pi Zero W.” Raspberry Pi. <https://www.raspberrypi.com/products/raspberry-pi-zero-w/> (accessed Nov. 22, 2025).

2. Raspberry Pi Foundation. n.d. “Camera Module v2.” Raspberry Pi. <https://www.raspberrypi.com/products/camera-module-v2/> (accessed Nov. 23, 2025).

3. Python Software Foundation. n.d. “Getting Started.” Python.org. <https://www.python.org/about/gettingstarted/> (accessed Nov. 20, 2025).

4. Pounder, L. 2022. “How To Use Picamera2 to Take Photos With Raspberry Pi.” Tom’s Hardware. <https://www.tomshardware.com/how-to/use-picamera2-take-photos-with-raspberry-pi> (accessed Dec. 5, 2025).

5. Text-Compare. n.d. “Text Compare Tool.” Text-Compare. >https://text-compare.com/> (accessed Nov. 22, 2025).

6. Freepik. 2019. “Construction Machines Top View (Vector Graphic).” Freepik. <https://www.freepik.com/free-vector/construction-machines-top-view_4266194.htm> (accessed Nov. 14, 2025).
   Image designed by Freepik (www.freepik.com)

7. Raspberry Pi Forum. 2018. “Camera Module v2 with Pi Zero W.” Raspberry Pi Forums. <https://forums.raspberrypi.com/viewtopic.php?t=212518> (accessed Nov. 23, 2025).

8. Dynamsoft. 2018. “Python Barcode Recognition with ZXing and ZBar.” Dynamsoft Codepool. <https://www.dynamsoft.com/codepool/python-zxing-zbar-barcode.html> (accessed Nov. 23, 2025).