Computer Vision Applied to Vehicle Flow Monitoring for a Smart Campus

Maria Isabela Monteiro Servino
School of Technology
UNICAMP
Limeira, Brazil
m221700@dac.unicamp.br

Luis Fernando Gomez Gonzalez
Institute of Computing
UNICAMP
Campinas, Brazil
gonzalez@unicamp.br

Rodrigo Luiz Ximenes
School of Technology
UNICAMP
Limeira, Brazil
ximenes@unicamp.br

Talía Simões dos Santos Ximenes
School of Technology
UNICAMP
Limeira, Brazil
talia@unicamp.br

***Abstract*—In the face of the ever-growing number of registered vehicles in Brazil, the country has encountered significant challenges, such as traffic congestion and parking scarcity. Similar challenges have been experienced by the Campus I of Limeira, which faces an increasing demand for parking spaces. In this context, a pioneering project is proposed for Limeira's Campus I, using computer vision for vehicle monitoring. This project enables real-time flow tracking and statistical data collection. The interdisciplinary initiative aims to enhance campus connectivity, space optimization, and mobility and set the stage for future innovations.**

Keywords—monitoring; vehicles; computer vision.

I. Introduction

With the continuous increase in the number of registered vehicles in Brazil, the country has been facing significant challenges, such as traffic congestion and parking shortages [1], [2]. Similar challenges are being encountered by Limeira's Campus I, where there is a growing demand for parking spaces. In pursuing solutions to urban problems, universities have embraced the concept of Smart Campuses, with Unicamp Campinas serving as an example by implementing projects to modernize the campus and enhancing the community experience [3].

In this context, a pioneering project is proposed for Limeira's Campus I, leveraging computer vision [4] for vehicle monitoring. This approach enables the tracking of vehicle flow and the collection of statistical data. This multidisciplinary initiative aims to foster connectivity, optimize space utilization, enhance campus mobility, and pave the way for future innovations.

II. Methodology

Various architectures were tested to implement the proposed solution. Initially, ESP-32 CAM boards were employed as low-cost cameras connected via Wi-Fi, transmitting images to a virtual machine on the Oracle Cloud [5] and a HaarCascade model in OpenCV was utilized for vehicle detection [6]. However, this approach proved slow due to image upload times and faced data protection and personal privacy regulations (LGPD) challenges. An alternative proposal involving Raspberry Pi boards also fell short of expectations, as it couldn't achieve acceptable frame rates for detection [7] and incurred higher costs. As the definitive approach, an Intelbras VIP S4020 V2 IP camera, presented in Figure 1, was chosen in collaboration with the campus's Regional Administration Department (SAR), along with a desktop computer.

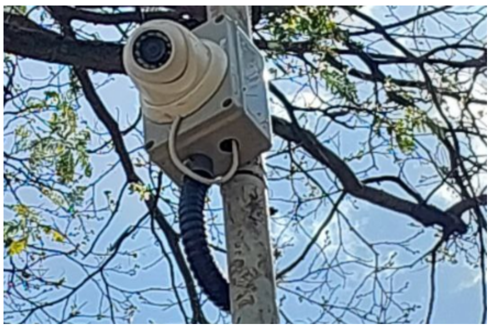

Fig. 1. Locally installed camera.

The camera was installed on a post overlooking the campus's main entrance and connected using a locally installed network cable for video transmission. Processing occurs on a local computer, enabling vehicle detection at an adequate frame rate exceeding 5 frames per second (fps). This setup allows real-time updates of the vehicle count on campus. The computer is placed at the main entrance gatehouse, where staff responsible for monitoring and gate operation are located, as depicted in Figure 4, situated between the vehicle entry and exit points.

The communication between the computer and the camera is achieved through the Real-Time Streaming Protocol (RTSP) [8]. Image connection and processing are carried out using Python 3 and the OpenCV library, including vehicle detection. The library also performs image transformations, displaying relevant detection information and metrics. Additionally, the Numpy and PyQt5 libraries are utilized for mathematical calculations and creating a graphical interface showcasing the image manipulated by OpenCV.

The YOLOv4-tiny neural network model is employed for vehicle detection due to its speed and high accuracy in identifying various object classes such as cars, motorcycles, and bicycles [9], [10]. Figure 2 illustrates the architecture diagram for image processing.

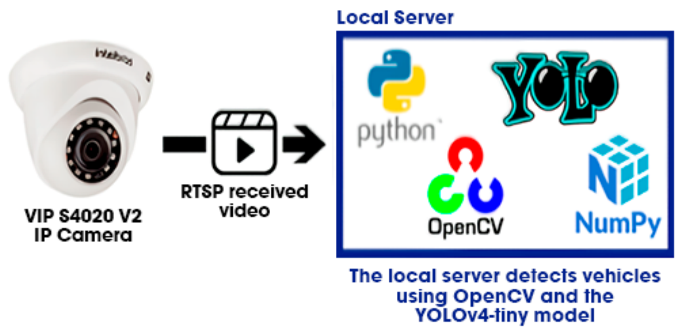

Fig. 2. Architecture diagram.

The implemented code follows a flow that begins by creating a window with two widgets using the PyQt5 library: an image displaying the latest processed frame and a timer initiating the processing of a new frame according to the camera's framerate. As specified by the model, processing areas of 416 x 416 pixels are established. Within these areas, regions to be ignored are marked in black to conserve processing power. Subsequently, these areas are resized, and object detection is performed, as observed in Figure 3. Only detections of cars, motorcycles, and bicycles with confidence above a threshold are considered. The Non-Maximum Suppression (NMS) algorithm is also employed to eliminate unwanted or duplicate detections.

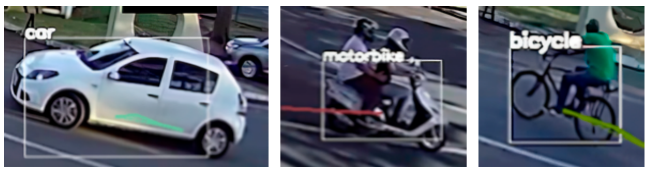

Fig. 3. Vehicle detection with Non-Maximum-Suppression.

Vehicle centroids are calculated and stored in a dictionary. With each frame, the detected centroids are associated with the dictionary based on the shortest distance [11] relative to the centroid of the previous frame. A new vehicle is considered and added to the dictionary with a new associated key if the distance exceeds a threshold.

Using the last eight centroids of each vehicle, line segments are formed and checked for intersections with delimited areas for vehicle addition or removal. If an intersection [12] occurs, the vehicle count is updated, and the vehicle is ignored in subsequent detections to prevent duplicates. Vehicles without new centroids detected for over 30 frames are removed to conserve memory.

Each vehicle's trajectory is defined by concatenating all line segments of the vehicle, denoted as $C_{Vi}C_{V(i+1)}$ where C_{Vi} represents the centroid at index *i* of the vehicle *V*.

Figure 4 presents the application interface with the vehicle count displayed in the upper-left corner as "C" for cars, "M" for motorcycles, and "B" for bicycles. Yellow rectangles outline the detection regions sent to the neural network, while a black region is defined to prevent vehicle detection on the highway near the campus.

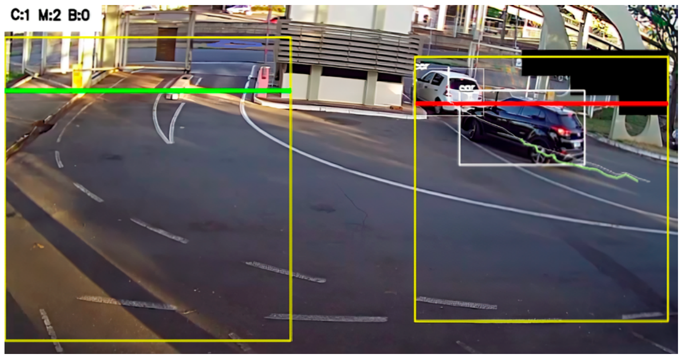

Fig. 4. Code execution.

The green and red lines denote detection areas for incrementing or decrementing vehicles. The light green and white lines represent the car trajectories delimited by the white rectangles. These lines assume random colors to describe each vehicle's trajectory. When they cross the red or green lines, they turn white, indicating the vehicle was detected, counted, and will be excluded from subsequent detections.

In addition to the vehicle detection algorithm, a web application was developed to display the number of vehicles present and parking space availability on campus. Hosted on an external web server, the application utilizes PHP, HTML, CSS, and JavaScript, along with an SQLite database. Figure 5 illustrates the complete architecture diagram of the application, emphasizing local image processing, data transmission to a centralized web server, and end-user access to this information.

The local server sends POST requests to the web server through a REST API, inserting records for each detected vehicle, including the date, time, vehicle type (car, motorcycle, or bicycle), and detection type (entry or exit). The community can then access real-time information about available parking spaces on the campus by accessing a publicly available web application.

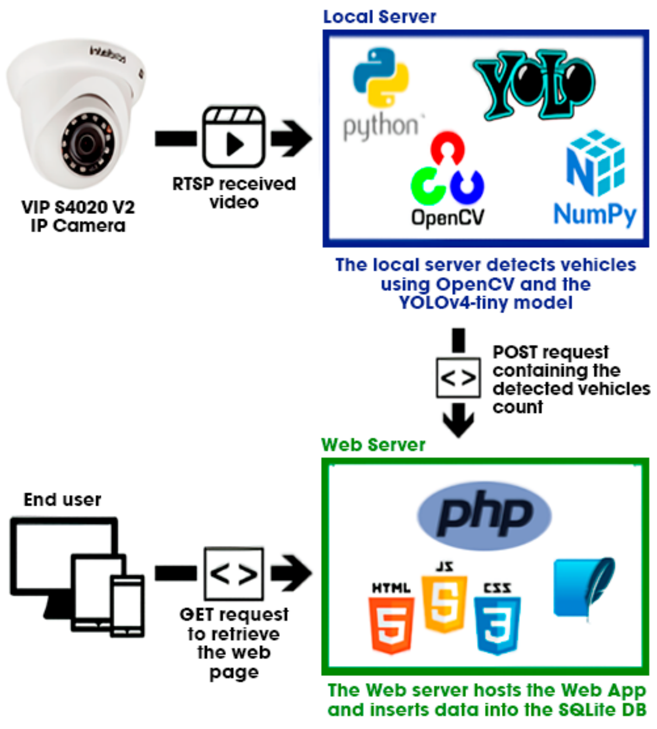
Local Server
python
YOLO
OpenCV
NumPy
VIP S4020 V2
IP Camera
RTSP received video
The local server detects vehicles using OpenCV and the YOLOv4-tiny model
POST request containing the detected vehicles count
Web Server
End user
php
HTML
JS
CSS
GET request to retrieve the web page
The Web server hosts the Web App and inserts data into the SQLite DB

Fig. 5. Complete application architecture.

The web application's user interface, depicted in Figure 6, provides end-users with the available parking spaces for cars, motorbikes, and bicycles, along with the total count. The boxes indicating free parking spaces change color according to their availability. If over 40% of spaces are free, the color is green; for 10% to 40% free, it's yellow; and if below 10%, it's red. Beneath the parking space details, an hourly histogram illustrates the daily vehicle flow across the entire campus, encompassing all vehicle types and parking areas.

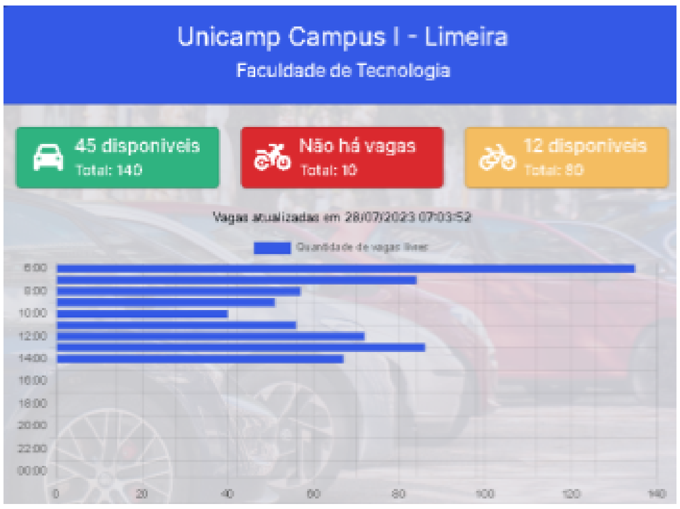

Fig. 6. Web application interface.

III. Results and Discussion

Throughout the application's conception, from architectural modifications to model optimizations, the complexity of the problem became evident. Balancing processing speed with the necessary precision level and the available computational resources proved challenging. After the model's development, a comparison was conducted between manual vehicle counting and the count derived from the vehicle detection algorithm. This was done using a framerate of 8 frames per second (FPS) on a 7th generation Core I5 7200U notebook with 8GB of RAM. The obtained results were consistent for the analyzed video time samples.

The detection accuracy is estimated to surpass 90% in a definitive deployment scenario. This estimate takes into account the various unpredictable variations in time, lighting, and vehicles. All the hours of video analyzed by the application represent only a subset of the potential scenarios encountered in which success and effective performance were achieved.

With the knowledge and autonomy gained during development, the potential for optimizing this detection model is evident. Opportunities include exploring alternative neural network models, adjusting image processing to facilitate detection, optimizing the centroid search algorithm, or even replacing the method of line intersection detection with polynomial interpolation aided by linear systems to verify intersections.

IV. Conclusions

It is anticipated that this pioneering approach at Limeira's Campus I will contribute to the creation of an environment tailored to the needs of the community members, enhancing mobility efficiently, safely, and sustainably. Additionally, it aims to engage faculty and students in developing a conducive space for embracing intelligent solutions in support of a Smart Campus that positively impacts the community. This initiative simultaneously showcases our capacity for scientific and technological development.

With the continuation of efforts and the pursuit of future optimizations, the expectation is for the vehicle traffic monitoring and management system to achieve higher levels of speed and reliability. This will further enhance the application's autonomy and potential utilization at other entrances of Campus I Limeira. Potential expansions to Campus II or even for other applications are also within reach through the release of the code in an open-source model.

Acknowledgment

The authors thank the National Council for Scientific and Technological Development (CNPq) for the financial support. and also André Cardoso for his support with the installation of the camera used in this work.

References

[1] IBGE – INSTITUTO BRASILEIRO DE GEOGRAFIA E ESTATÍSTICA. Vehicle Fleet. Rio de Janeiro: IBGE, 2013. Available at: https://cidades.ibge.gov.br/brasil/pesquisa/22/28120. Accessed on May 9, 2022.

[2] M.Y.I. Idris, Y.Y. Leng, E.M. Tamil, N.M. Noor and Z. Razak, 2009. Car Park System: A Review of Smart Parking System and its Technology. Information Technology Journal, 8: 101-113. Available at: https://scialert.net/abstract/?doi=itj.2009.101.113. Accessed on May 7, 2022.

[3] Neves, A. R. M.; Sarmanho, K. U.; Meiguins, B. S. The Role of the University in the Construction of Smart and Human Cities. Revista Electronica de Sistemas de Informação, [s. l.], May-Aug 2017. Available at: https://www.proquest.com/openview/6dc5b34d62727ad904a4837ebadcc168/1?pq-origsite=gscholar&cbl=178195. Accessed on May 8, 2022.

[4] Milano, D.; Honorato, L. B.. Computer Vision. Computer Vision, [s. l.], 2010. Available at: https://www.academia.edu/download/35825905/2010_IA_FT_UNICAMP_visaoComputacional.pdf. Accessed on May 8, 2022.

[5] Choose Flexible Forms of Virtual Machines. Available at: https://www.oracle.com/br/cloud/compute/virtual-machines/. Accessed on December 20, 2022.

[6] OPENCV. OpenCV: Cascade Classifier. Available at: https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html. Accessed on January 4, 2023.

[7] Q-ENGINEERING. YoloV4 Raspberry Pi 4. Available at: https://github.com/Qengineering/YoloV4-ncnn-Raspberry-Pi-4. Accessed on January 16, 2023.

[8] WIKIPEDIA CONTRIBUTORS. RTSP. Available at: https://pt.wikipedia.org/wiki/RTSP. Accessed on January 16, 2023.

[9] BOCHKOVSKIY, A.; WANG, C.-Y.; LIAO, H.-Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. April 22, 2020.

[10] YOLOv4 Tiny Object Detection Model. Available at: https://roboflow.com/model/yolov4-tiny. Accessed on January 28, 2023.

[11] WIKIPEDIA CONTRIBUTORS. Euclidean distance. Available at: https://en.wikipedia.org/wiki/Euclidean_distance#Squared_Euclidean_distance. Accessed on February 12, 2023.

[12] WIKIPEDIA CONTRIBUTORS. Line–line intersection. Available at: https://en.wikipedia.org/wiki/Line%E2%80%93line_intersection. Accessed on February 20, 2023.