Machine Learning and Artificial Intelligence
J.-L. Kim (Ed.)

doi:10.3233/FAIA230775

Smart Mountain: A Solution Based on a Low-Cost Embedded System to Detect Urban Traffic in Natural Parks

Paulo COSTA [a], Eduardo PEIXOTO [a,c], and Davide CARNEIRO [b,1]
[a] *Escola Superior de Tecnologia, Instituto Politécnico do Cávado e do Ave (IPCA)*
[b] *CIICESI, Escola Superior de Tecnologia e Gestão, Instituto Politécnico do Porto*
[c] *Centro Algoritmi, Universidade do Minho, Braga, Portugal*

Abstract. We live in an era in which the preservation of the environment is being widely discussed, driven by growing concerns over climate issues. One major factor contributing to this situation is the lack of attention societies give to maintaining high sustainability levels. Data plays a crucial role in understanding and assessing sustainability impacts in both urban and rural areas. However, obtaining comprehensive data on a country's sustainability is challenging due to the lack of simple and accessible sources. Existing solutions for sustainability analysis are limited by high costs and implementation difficulties, which restrict their spatial coverage. In this paper, we propose a solution using low-cost hardware and open-source technologies to collect data about the movement of people and vehicles. This solution involves low-cost video-based meters that can be flexibly deployed to various locations. Specifically, we developed a prototype using Raspberry Pi and YOLO which is able to correctly classify 91% of the vehicles by type, and 100% of the events (entering of leaving). The results indicate that this system can effectively and affordably identify and count people and vehicles, allowing for its implementations namely in remote sensitive areas such as natural parks, in which the access of people and vehicles must be controlled and monitored.

Keywords. Video-based counters, Smart Data, Sustainability, Raspberry PI, Yolo

1. Introduction

As people increasingly seek natural and greener spaces, namely for for holidays or leisure activities, urban mobility[1] extends to rural areas[2], posing a potential threat to these often sensitive ecosystems. At the same time, while Smart Cities programs have been implemented in many cities worldwide, rural areas have seen limited initiatives[3,4]. To monitor and control the expected impact of this shift, smart monitoring solutions can play a crucial role [5,6].

However, the lack of systematic and consistent monitoring processes has hindered the development of sustainable policies. Namely, insufficient data on these specific environments poses challenges for local entities and private companies alike, as they require accurate data to justify investments, make informed decisions, and assess policy impacts.

[1]Correspondence: dcarneiro@estg.ipp.pt

In this study, we propose an inexpensive vehicle counting system specifically tailored for natural parks and similar environments. Rather than developing advanced solutions, our goal is to use inexpensive hardware and open-source solutions, that can be effectively adopted by municipalities and natural parks management. The system enables the implementation of numerous counting points, revolutionizing activity detection and providing sustainability indicators for rural planners and society.

2. Measuring mobility in real-time in the rural environment

In this research, we investigate how to measure the overall mobility activity in natural parks so that it can be monitored in real-time. The collected data can also be used to control accesses, namely in sensitive areas, and to support the design of better policies that can take into consideration people's movement patterns and the local characteristics.

The system is implemented using a Raspberry Pi: a low-cost, low-power, small-size, single-board computer. The version in use for the implementation of this work is the Raspberry Pi 3B, which has a 64-bit Quad Core CPU at 1.2GHz, 1 GB of RAM, Wi-fi, Bluetooth and other useful inputs.

Several interconnected components were implemented (Figure. 1). A camera that is connected to the Raspberry Pi, which in turn runs the traffic detection software that includes an AI model to detect people and vehicles, as well as all the inherent logic necessary for a detection and subsequent accurate and effective tracking.

This detection service runs in three distinct threads: one for reading the images sent by the camera, another for processing the images read and calculating the positions of objects, and the last for drawing the results obtained and keeping a buffer accessible by the HTTP server for video streaming by MJPEG.

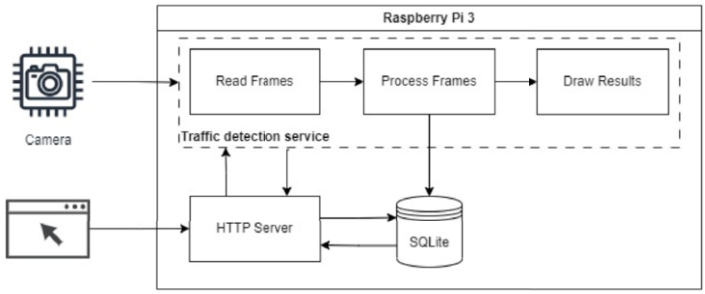

Figure 1. High-level diagram of the system.

The visualization of the results, as well as the configuration of the parameters of the detection service, is made available through a dashboard accessible from the HTTP server that acts as an intermediary between the system and the end user, allowing their interaction through a small API.

In this architecture, there are two key functionalities: 1) object detection and tracking; and 2) the decision component of accounting for an inflow or outflow. These functionalities are implemented using YOLO and OpenCV.

YOLO (You Only Look Once) is a family of algorithms for object detection that efficiently identify and localize objects in an image or video in real-time by dividing the

image into a grid and predicting bounding boxes and class probabilities simultaneously [7].

OpenCV (Open Source Computer Vision Library) is a popular open-source computer vision and image processing library that provides a wide range of tools and functions for tasks such as image/video manipulation, object detection, and feature extraction [8].

The system works as follows. First, a specifically trained YOLO model is loaded, after which the system is ready to operate. For each image obtained from the video, the model is used to detect and localize the different objects. For each detected object the system returns its class, the confidence, the bounding box, the position in which the object is first detected and whether the object is crossing an entry/exit line.

The object tracking functionality is responsible for identifying a same object as unique along the frames. This is implemented through a combined use of the IoU (Intersection over Union) of each object and the distance of their centroids. IoU is a method that evaluates the percentage of overlap between 2 rectangles, in this case the current position of the object and the position of the last frame in which it was detected. Centroid distance is a measure that tells the distance between the centroids of two rectangles, again comparing the current position of the object with the position of the last frame in which it was detected.

By combining these two concepts, we explore the advantages of both: while the centroid distance is a fast and efficient approach for objects in linear motion, the IoU helps to solve more complex situations, such as object occlusions. This results in a robust and accurate tracking, capable of handling a variety of challenging scenarios that may arise.

Regarding the functionality of accounting for an entry or exit of an object on the monitored area, it first requires the the definition of the entry and exit lines in a configuration file. Then, the following steps are followed (Figure. 2 is used as a support for this explanation).

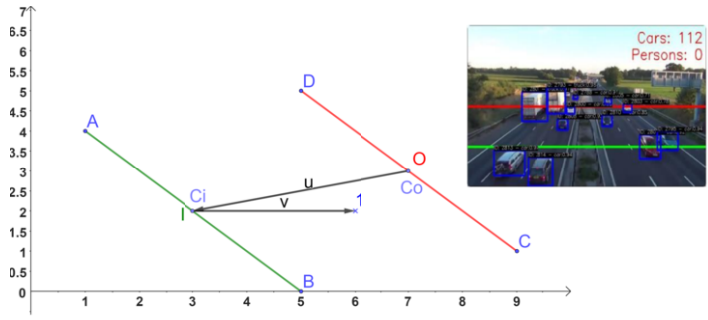

Figure 2. Chart demonstrating the calculation of positions. (Green line - entry line; Red line - exit line; 1 - traveling object; Ci - centroid of the entry line; Co - centroid of the outgoing line; U - direction vector; V - vector between object and line for which the position is calculated)

First, the direction vector is calculated, using the equations depicted in Equation. 1. For the scenario given in Figure. 2, the direction vetor would have the value $\left(\frac{-4}{\sqrt{17}}, \frac{-1}{\sqrt{17}}\right)$.

$$Unit\ Vetor = \begin{cases} x = \frac{\Delta x}{m} \\ y = \frac{\Delta y}{m} \end{cases} \qquad centroid = \begin{cases} x = \frac{xi + xf}{2} \\ y = \frac{yi + yf}{2} \end{cases}$$
$$magnitude\ (m) = \sqrt{\Delta x^2 + \Delta y^2} \tag{1}$$

After calculating the direction vector, the pool of detections is iterated, processing the respective positions, which fall into one of 2 cases: either the object is detected in the bounding area for the first time, or it is detected outside of the bounding area for the first time. As previously mentioned, the position where the object was detected for the first time is first validated, which allows us to decide which case to deal with. This condition is verified, taking into account the coordinates of the object, in relation to both lines, and it must be found between them. This means before the entry line and after the exit line, and can be calculated as depicted in Equation. 2.

$$Vetor = \begin{cases} x = \Delta x \\ y = \Delta y \end{cases}$$
$$Scale\ Product = ax * bx + ay * by$$
$$Angle\ Between\ Vetors\ (\Theta) = \cos^{-1}\left(\frac{Scale\ Product}{ma * mb}\right) \tag{2}$$

After obtaining the vector between the centroid of the object and the centroid of the line for which we want to know its position, in this case the entry line, we calculate the angle between this same vector and the direction vector, initially obtained, if this angle is greater than $\frac{\pi}{2}$, we can conclude that the object is before the intended line, otherwise it is after it. At this point, the system is able to conclude the position of the object, relative to any of the lines, which allows us to count the entry and exit of objects in the monitored area, or to track their movement.

3. Results

This section describes the results obtained from the implementation of a prototype of the proposed system. The main parameters used are described in Table 1.

Table 1. Main configurations used.

Configuration	Value
Confidence threshold	0.5
Non-maximum threshold	0.3
Input size	416 x 416
IoU threshold	0.5
Maximum frames not detected	30
Maximum distance threshold	50

For the purpose of this paper, the system and the model were tested with a short video clip containing cars, buses and trucks. Specifically, 5 cars entered the monitored region and 2 existed, 1 bus entered the region, and 2 trucks entered and 1 existed. The

events observed in the video were as follows: car entry, bus entry, truck entry, car exit, four car entries, one car exit, one truck entry and one truck exit.

We tested three different version of YOLO models (v3, v4 and v7). While v7 had the fastest prediction times, v3 was slightly better in terms of predictive accuracy. Given the need for conciseness, we limit this analysis to v3. Specifically, Table 2 details the detections made by the model, the confidence level and the actual events.

Table 2. Predicted vs. observed detections by a specifically trained YOLOv3 model.

Predicted Class	Confidence	Actual class	Event	Actual Event
Car	0.90	Car	Entry	Entry
Bus	0.99	Bus	Entry	Entry
Truck	0.97	Truck	Entry	Entry
Car	0.96	Car	Exit	Exit
Car	0.97	Car	Entry	Entry
Car	0.97	Car	Entry	Entry
Truck	0.84	Car	Entry	Entry
Car	0.98	Car	Entry	Entry
Car	0.98	Car	Exit	Exit
Truck	0.96	Truck	Entry	Entry
Truck	0.96	Truck	Exit	Exit

Two main aspects can be evaluated from these results. The first concerns the ability of the model to correctly classify the objects in the video. In the case of the model described, it was able to correctly classify 91% of the objects in the video. The second is the correct classification of the events (i.e. entry and exit). In this case, the approach implemented, which combines IoU with centroid distance, was able to correctly identify 100% of the events.

Asides from collecting data from the identified vehicles, the system also provides estimates of the emissions in the area. This kind of information can then be used by policy-makers and other actors to evaluate the impact of mobility on a certain area, and to devise better policies. The prototype of the frontend developed also allows the user to easily change some of the most important configurations (Figure 3).

Figure 3. Prototype of the frontend developed.

4. Conclusions and Limitations

In this paper we detailed the implementation of a traffic detection system, designed with the aim of facilitating and improving the management of natural parks in what concerns the access of people and vehicles, and their impact. It is nowadays clear that Artificial Intelligence and data and image processing techniques play a crucial role in solving these problems in a cost-effective way.

The proposed and implemented solution, although relatively simple, proved to be efficient and capable of dealing with the complexity of the problem addressed, proving the robustness, flexibility and validity of the chosen technologies. It is able to monitor the traffic in a certain area autonomously, providing sustainable indicators and counters in real time, supporting more informed decision-making processes. Moreover, the whole system can fit into a small box containing only the Raspbery Pi and a camera.

The main limitation of the work is that it was only briefly validated, and using existing video streams of traffic. So in future work we will still implement this system on an actual part, in a real setting, in which there is interest in monitoring and controlling access of certain types of vehicles or of people. This will allow to do an on-site validation, in real conditions.

Still, given the results, we are confident that the model will generalize well to that specific setting and perform as necessary for the intended tasks.

References

[1] M. Miskolczi, D. Földes, A. Munkácsy, and M. Jászberényi, "Urban mobility scenarios until the 2030s," *Sustainable Cities and Society*, vol. 72, p. 103029, 2021.

[2] S. Porru, F. E. Misso, F. E. Pani, and C. Repetto, "Smart mobility and public transport: Opportunities and challenges in rural and urban areas," *Journal of traffic and transportation engineering (English edition)*, vol. 7, no. 1, pp. 88–97, 2020.

[3] L. Butler, T. Yigitcanlar, and A. Paz, "Smart urban mobility innovations: A comprehensive review and evaluation," *Ieee Access*, vol. 8, pp. 196 034–196 049, 2020.

[4] L. Camarero and J. Oliva, "Thinking in rural gap: mobility and social inequalities," *Palgrave Communications*, vol. 5, no. 1, 2019.

[5] C. Toma, A. Alexandru, M. Popa, and A. Zamfiroiu, "Iot solution for smart cities' pollution monitoring and the security challenges," *Sensors*, vol. 19, no. 15, p. 3401, 2019.

[6] S. Paiva, M. A. Ahad, G. Tripathi, N. Feroz, and G. Casalino, "Enabling technologies for urban smart mobility: Recent trends, opportunities and challenges," *Sensors*, vol. 21, no. 6, p. 2143, 2021.

[7] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of yolo algorithm developments," *Procedia Computer Science*, vol. 199, pp. 1066–1073, 2022.

[8] S. Gollapudi and S. Gollapudi, "Opencv with python," *Learn Computer Vision Using OpenCV: With Deep Learning CNNs and RNNs*, pp. 31–50, 2019.