

# Traffic Control System Development Based on Computer Vision

Diogo Eugênio da Silva Cortez, Itamir de Morais Barroca Filho(✉), Everson Mizael Cortez Silva, and Gustavo Girão

Digital Metropolis Institute, Federal University of Rio Grande do Norte Natal, Lagoa Nova, Natal, Brazil
{itamir.filho,girao}@imd.ufrn.br
http://www.imd.ufrn.br, http://www.ufrn.br

**Abstract.** In 2020, the number of Brazil inhabitants was approximately 212 millions, whereas the number of vehicles jumped from 104 to 107 millions. It is observed, therefore, that there is 1 vehicle for every 2 inhabitants. That same year, the population in urban areas went to 80%. Managing traffic in big cities is becoming a huge challenge. Traffic lights operating with Fixed Time Signal to control vehicle flux are no longer efficient in all traffic scenarios. Technological advances in Computer Vision, moving objects detection and classification techniques and the demanding of little computer power to manage these tasks allowed the development of TEOP, a CV-based traffic control system. This low cost solution was implemented to advance the Fixed Time Signal system, cameras and logical network infrastructure already in use in Brazilian cities, transforming regular traffic lights into smart traffic lights (STL). An application was developed and installed in a computer to capture images of traffic, count vehicles and calculate time needed for them to pass through. Raspberry controlled traffic lights. In comparison to regular traffic lights, STLs improved traffic flow by 33%, allowing a wait of just 4 s in cases where there is no vehicle flow at the competing traffic light and reducing the crossing time of 46 vehicles from 152 to 116 s when there were 12 vehicles on the competing side, a significant gain. It was also capable of reporting traffic jams and creating a database that could be used for decision-taking by agencies responsible for each route.

**Keywords:** Smart traffic management system · OpenCV · YOLO-Tiny · SSD · MobileNetV2

## 1 Introduction

According to 2020 data shown in [1] the vehicle fleet in Brazil and some of its states such as Rio Grande do Norte reaches approximately half the number of their respective populations. Most of these vehicles are registered in the metropolitan regions of the state.

The disorderly growth of urban areas, that are now housing 80% of Brazilians, was accompanied by continuous growth in vehicles as well. Excess traffic on streets are not only harmful, however, to life quality but also implies financial losses [2].

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022 Workshops, LNCS 13381, pp. 323–339, 2022.
https://doi.org/10.1007/978-3-031-10548-7_24

The traffic light was first implemented in London, 1968, as a response to increasing traffic accident occurrences at the time. It remains the main traffic regulator to this date [3]. According [4] a traffic efficient system, including better time management for traffic lights, could reduce pollution in urban areas. In many parts of the world, vehicle traffic is the main cause for air pollution [5]. This happens because of the increase in gas usage when there is deficiency in traffic light programming.

Smart Traffic Lights (STLs) can use sensors or cameras to calculate the density of vehicles that go through it. With data in hands, it is possible to adequate light time and in turn provide better traffic flux. It is also possible to adjust dates, times and other parameters. Traffic connectivity with the traffic control unit allows configuration and monitoring of traffic light networks.

If adjusted according to the route's demands, smart traffic lights employ better traffic scenarios, while also being adaptable to other positive scenarios [6].

According to [7] one efficient traffic light control solution is the car flux automatic detection. According to literature review made by these authors, smart traffic lights are one of the most propelled nowadays to optimize traffic systems, but its high cost prevents mass implementation, especially in small towns.

According to [17] two and a half million dollars were spent on equipment and infrastructure for the installation of 88 traffic lights, an average of twenty six thousand dollars per traffic light.

Nowadays, many states and cities have high quality camera monitoring systems to help provide safety and control traffic [8]. In many cases, the systems have optic fiber infrastructure and/or wireless technology to allow high-definition image transferring.

The smart traffic light is highly recommended because it is able to improve traffic flow, according to the data collected by the sources mentioned above, so it is recommended to use the solution proposed in this work together with the implementation of the green wave to avoid excess traffic. traffic at adjacent traffic lights.

As it is a strategic issue of great relevance to cities, it is recommended to implement information security policies together with the solution proposed in this work. According to [10], it is not only present in hardware and software implementation, but information security in organizational units must also include infrastructure, logistics and human resources.

According to literature revision made by the authors, the majority of papers analyzed suggest traffic light control based on density of vehicles that traffic under it. Some of these studies found gaps that could make them unfeasible since they rely on camera images or the internet to generate time management and controlling traffic lights. Part of these papers shows the traffic light could freeze in the last time setting made, providing red light for an undetermined time. In other cases, traffic control systems would not allow broader parameter settings.

This paper intends to create a smart traffic control system using Computer Vision (CV). In order to do that, the traffic light and CV control algorithm will work on a PC computer, traffic lights will be controlled (turned on and off) through a single board unit with Raspberry Pi 3. Computer Vision would then count vehicles crossing the traffic light; in these instances, data collected is used to identify traffic jams. Besides, the system would also have redundancy for fixed time mode for instances where camera

issues are identified. Other aspects of this solution are its low implementation costs, since it relies on already-installed traffic systems, cameras and other logic network infrastructure on multiple Brazilian cities, low energy usage and high efficiency. More differentials of the proposed system allow more than 50 adjustments to be configured for each traffic light allowing to customize each intersection to obtain maximum fluidity and improvements in vehicle counting, in addition to adapting working independently of cameras and the Internet, as it detects problems on these items and changes its working mode automatically.

This article was organized in the following structure: the second section presents the related works, the third section describes the methodology used and, in Fig. 3, shows the simplified operation of the algorithm used, the fourth section presents the tests, the fifth section presents the limitations found and finally, the contributions and future works are arranged in the sixth section.

## 2 Related Works

In order to improve traffic flow, according to literature revision, many analyzed studies suggest controlling traffic lights based on how many vehicles go through it. According to [9], who proposed traffic light control based on vehicle density, this technique allowed a 35% improvement in traffic flux when compared to pre-existing Fixed Time System traffic lights. Some works on the topic will be listed below:

In works of [11], they proposed a Distributed Traffic Control System. In this system, each traffic light has one camera pointed at each direction of the road. Employed were two techniques to count objects: the first one called border detection technique, used to count just vehicles, and the second, YOLO, would classify them. All systems were implemented using a Raspberry Pi 3. With the goal of verifying which techniques would come closer to real counting of objects in roads, a comparison was made between them. Towards the end of testing, border detection technique showed mistakes and remained distant from the quantity of objects passing through. On the other hand, YOLO showed closer results to what was expected. Authors say that Distributed Traffic Control System will give successful results, avoiding major delays and lowering traffic time.

On the other hand [12], proposed a Traffic Vigil System for Vehicle Counting. This system has each traffic light equipped with a camera pointed at each road direction. Background subtraction, detection and tracking through blobs were used to detect and count objects on the road. The system was implemented in a type PC computer, and experimental results showed that it could provide real-time useful information for traffic vigilance. This paper does not inform us if the traffic lights control is in use.

The proposal of [13], was a solution for real-time monitoring, tracking and counting of vehicles in the city of Jelgava, Latvia. This system has each traffic light equipped with a camera pointed at each road direction. Background subtraction, detection and tracking through blobs were used to detect and count objects on the road. The paper does not specify which hardware was used to count objects. Traffic light control is not the goal of this paper and, according to its authors, the system shows good performance and acceptable precision. In parts of the test, precision reached a 97% hits level for regular traffic conditions. Tests also showed that this system proved to be inadequate for traffic jam days.

The work of [14], proposes an ARM-based real-time Traffic Classification System. In this system, each traffic light is equipped with a camera pointed at each road direction. It suggests linking a single board computer based on ARM Cortex-A8 to each camera that will process images, count objects and send data to a remote server. The counting algorithm was Harris' border detection. Experiments showed that the proposed system could process real-time images four times quicker than other OpenCV-based implementations. Traffic light control was not the aim of this paper.

Whereas [15], proposed a System to detect Physical Characteristics of Objects in Traffic. However, the study does not inform how many cameras were used in this system. A PC computer was used to process images. Traffic light control was not the aim for this study. Tensorflow was also used to train learning models. Java and OpenCV were used to classify objects. Success rate in detecting size and object color was below 70%, although object classification rate was above 90%.

At last [16], proposed an Intelligent Control System for Traffic Intersections. In this study, Raspberry Pi 3 was used to capture and process traffic images., as well as controlling traffic lights. The entirety of this study works based on Raspberry Pi 3 B+. The Tensorflow with SSD and MobileNetV2 were used in model training and object detection. Developers made bench tests in small scale and according to these tests, the system functioned as expected, reducing waiting time in traffic for a minimum.

## 3 Methodology

After completing the bibliographic review, it was noted that there are numerous works with solutions to improve traffic flow based on the CV. Despite that, there are gaps that need filling, such as the traffic lights relying exclusively on data generated by camera images to work or, exclusive internet dependence to receive data and control traffic lights. In these instances, the traffic light might be disrupted; it can freeze in its last generated time setting, with the time of 0, 1 or 2 s for the red light, meaning the traffic light would always be red until the problem is solved. This issue was resolved in two steps: first, monitoring the health of cameras and internet was implemented in order to detect problems in these items. Second, if camera or internet problems are identified, the traffic light returns to work at fixed times until the problem is resolved.

During the planning of the system suggested by this work, it was observed that there are numerous possibilities for configuring the operating parameters of a traffic light. According to our bibliographical review, analyzed papers implemented a small part of these parameters, making it impossible for users to have access and alter them.

Therefore, the development of the Traffic Engineering Operations Center (TEOP) was proposed, which would create, set and manage traffic lights through WEB – smart traffic lights or fixed time signal traffic lights, which operate controlling vehicles using a predefined time stamp. Besides registration and system setting of a city's traffic light, TEOP can also work as an interface for the traffic light's state. By warning the Traffic Controller (TC) about the situation of each traffic light that comprises this system, either informing if they are working properly or if there is a communication problem between them or the cameras, or if there is congestion and their classification. TEOP will allow the Traffic Controller access to more than 50 traffic light configuration patterns, some

examples of traffic light customization are: creating regions of interest (ROI), vehicle count line, size of detectable vehicles, fixed light time settings, time (in seconds) for each vehicle to pass the intersection, operation that can be legacy mode (fixed time operation), Photo mode (counting vehicles waiting for the green light to turn on and calculating the time for their passage) or Photo with Video mode (which allows counting the vehicles that manage to pass the intersection pointing out traffic jams depending on the number of vehicles that can pass), among dozens of other parameters.

And it will also allow the TC to train their own models using images from objects that usually pass through the traffic lights.

### 3.1 Selected Computer Vision Techniques

Every image processing, whether through detection or object classification, will work using the OpenCV 4 framework, which will load network classification settings for multiple objects through YOLOv3-Tiny, YOLOv4-Tiny and MobileNetV2. Alternatively, it will execute the detection algorithm by removing image background, MOG2. Algorithm that will work inside this framework is described as follows: MOG2 – according to [13], this algorithm detects motion based on background removal and shows great performance and acceptable precision, reaching 97% hits. YOLOv3, YOLOv4 – these algorithms detect and classify objects. According to [11], tests made with different condition images gave us an average precision around 91%. MobileNetV2 with SSDLite – just like YOLO, this algorithm detects and classify objects, and according to [16], MobileNetV2 with SSDLite is 20 times more efficient and 10 times smaller than YOLOv2, and using COCO dataset, with success rate comparable between them. For this work, the success rate is given by the number of vehicles correctly counted.

Aiming at providing hardware variety, the proposed solution allows the user to choose between CV technologies for live video according to their reality, between MOG2, MobileNetV2 with SSDLite, YOLOv3-Tiny and YOLOv4-Tiny.

### 3.2 Used Hardware

The computer model used in tests was the Dell 7050 notebook. This model was equipped with an Intel Core i5 7300HQ 2.5 GHz processor with integrated video and dedicated video Nvidia GTX 1050, 16 GB RAM and SSD A2000 PCIe NVMe. Besides the computer, which executes image capturing and processing and then controls the traffic lights, Raspberry Pi 3 was also used to serve as intermediate between the computer and traffic lights, and will execute commands to turn on and off traffic lights using times stamps calculated by the computer.

### 3.3 System Operation

Figure 1 shows the visual diagram with the working system. Circle number 1 identifies the system composed by the server that allows TEOP, web register system and traffic lights management. Circle number 2 highlights, organized in gray rectangles, some functions available on TEOP for users, such as traffic light register, operation mode

selection, traffic light state follow up, network follow up and congestion follow up. It will also be possible to open and accompany calls to resolve traffic light issues, and the user could train the model using images from objects that traffic through the traffic light in question. Circle 3 shows the computer with the image processing algorithm and traffic light control, using the network to connect and capture images from the camera on circle 3a, which receives the images, processes them and obtains vehicle quantity while waiting for the traffic light to turn green. With this number, it calculates the time to open the traffic light and sends this information to Raspberry Pi 3, identified in circle 4, that will turn on and off lights according to information sent to the computer.

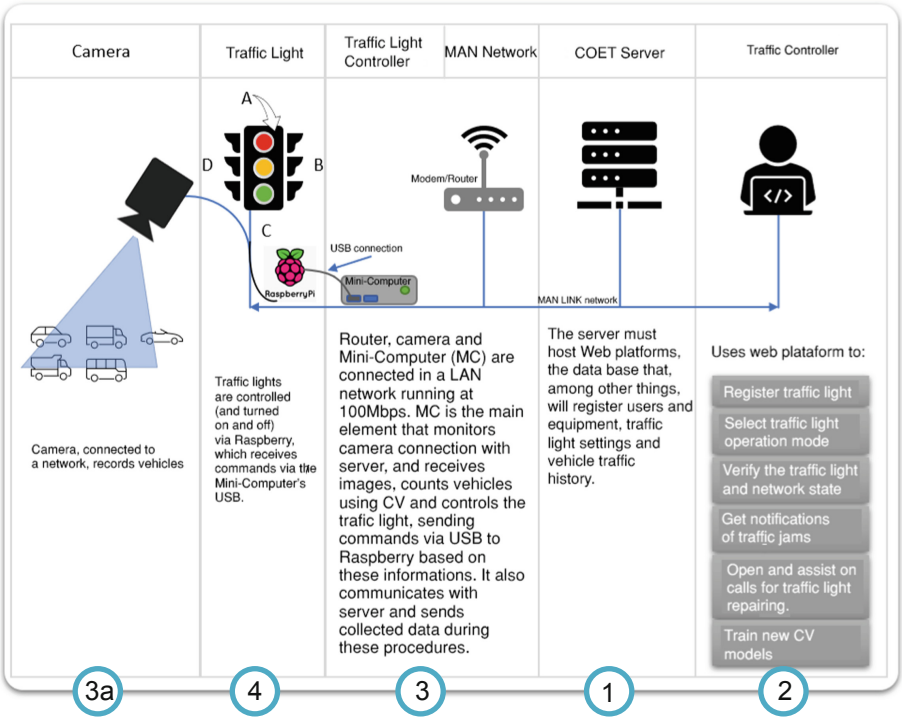


**Fig. 1.** Visual diagram of the working system

### 3.4 Preparation of Images Used in Training

In this work, models trained by the developers of YOLOv3-Tiny, YOLOv4-Tiny and MobileNetV2 were used, containing 80, 80 and 90 types of objects respectively. These models were used by the authors to create new models with six types of objects found in traffic, such as: people, bicycles, cars, motorcycles, buses and trucks. The objective was to compare models provided by developers with models trained by the authors and verify if there is any advantage or disadvantage in using a smaller number of objects.

The COCO 2014 database with more than 80 thousand images was used, and the author developed a tool separating the first 1 thousand objects of each of the six

types mentioned. Later, the tool edited text documents referring to selected images and removed objects that would not be used. At this moment, there were 1 thousand objects of each class that were used in a percentage of 70% for training, 20% for tests and 10% for validation.

### 3.5 Model Training

After selecting the images, the learning transfer from YOLOv3-Tiny and YOLOv4-Tiny, which are simpler versions with fewer parameters of YOLO (recommended for CV on mobile and embedded devices), to the customized models used in the project, was carried out.

Then, the same procedure was performed using the MobileNetV2 algorithm. In this case, it was necessary to convert images to the Tensorflow TFRecord format. Training information is displayed in the Table 1 below:

**Table 1.** Information on utilized models.

| Source | Algorithm | Classes | Dataset | Images per class | Training Time | mAP% | Training loss | Validation loss | Iterations |
|---|---|---|---|---|---|---|---|---|---|
| **Devs.** | yolov3-tiny | 80 | - | - | - | 50% | | - | - |
| **Devs.** | yolov4-tiny | 80 | COCO 2017 | - | - | 50% | | - | - |
| **Authors** | yolov3-tiny | 6 | COCO 2014 | 1.000 | 9h 14min | 52% | C:0.0% | - | 12.000 |
| **Authors** | yolov4-tiny | 6 | COCO 2014 | 1.000 | 7h 50min | 70% | C:0.0% | - | 12.000 |
| **Devs.** | SSD MobileNetV2 | 90 | COCO 2018 | - | - | - | - | - | - |
| **Authors** | SSD MobileNetV2 | 6 | COCO 2014 | 1.000 | 7h 38min | 50% : 95% | 0.6% | 0.5% | 30.000 |

TEOP also allows its users to train their own models, implementing even more options in the system. Available algorithms in the training function are YOLOv3-tiny, YOLOv4-tiny and MobileNetV2 with SSD. The user has an option for using images of objects captured in the traffic light using the capture tool available in the Computer Vision Improvement screen, as shown on Fig. 2:

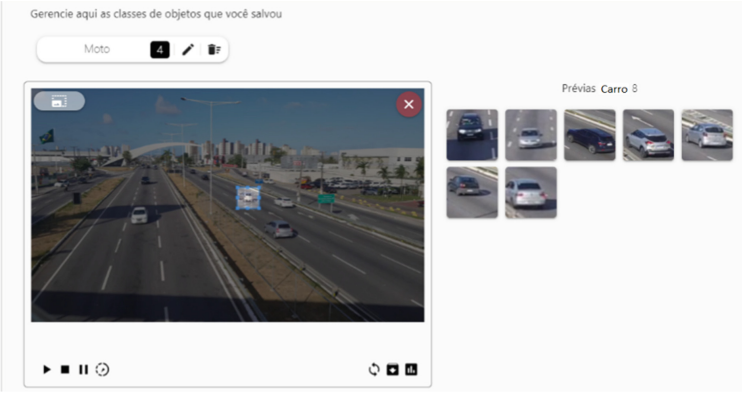


**Fig. 2.** Object capture image tool.

Figure 3 shows the workflow of the traffic light control algorithm that will be executed on a PC computer, installed next to the traffic light that will process the images, as mentioned in 3.2, calculate the time required for the crossing of vehicles, using the formula described in 3.6, and send the information to turn the traffic lights on and off, via USB connection, to the Raspberry Pi. This dynamic was divided into two streams: 1 and 2. They start equally and differentiate from step three. The algorithm is described below:

After the algorithm starts (step 1), the system asks which method will be used to load traffic light settings and starting times of Fixed Time Signal traffic lights. By pressing letter L, settings and times will be loaded in a text file already downloaded and saved in a local folder. If pressed M, the user will type times manually and settings will be loaded from a local file. If chosen A key, regular predefined times already in the algorithm and its settings will be loaded from the local file. If the S key is chosen, times and settings will be loaded on the server.

Afterwards, the algorithm goes to step 2. It is important to point out that each time it reaches this step, the algorithm researches in its database new settings. This allows the traffic controller to make real-time settings such as modifying regions of interest, real-time vehicle counting lines and even fixed time signals in traffic lights.

In step 3, the traffic light functioning mode is loaded. If it is a Fixed Time Signal traffic light, flux 1 starts and the traffic light will work with predefined times repetitively until a new model is selected. This mode is identical to what regular traffic lights operate, with advantages such as the possibility of changing time signals according to controllers demands, as well as problem monitoring in the traffic light, camera and network connection.

Step 4, connection tests with server and camera are made. From the server side, every 30 s a test will also verify if traffic lights are accessible. If found any issues, a call will be automatically opened informing the possible issue found. In flux 1 tests are made and information is kept in variables until reach step 7, where everything is sent to the server. In flux 2, if success is obtained with camera testing, it follows Picture Mode (PM) or Picture and Video Mode (PVM). If not, fixed times chosen in step one are loaded and PM and PVM will wait until the problem is solved.

On step 5, the traffic light starts working by lighting in the loaded time signals. In flux 1, it will work repetitively with pre-programmed times until another working mode is chosen. In flux 2, each time a yellow light goes on, the camera moves towards an area in which the red light is still lit. It captures a picture of every traffic light working this way. Vehicles are then counted, and the time they need to pass through the traffic light is generated. Still in flux 2, if the working mode is picture and video mode, the camera will capture pictures of traffic lights with green lights on, and the algorithm will start counting vehicles.

Step 6 is applied only on flux 2, and it repeats the step 5 routine for this flux. On step 7, collected info on step 4 is then sent to the server. The server will periodically consult this information and, upon any indication of error, it will open a call automatically based on collected data. After step 7, the traffic light finishes its cycle and goes back to step two. This procedure is repeated nonstop. Figure 3 shows this working dynamic.

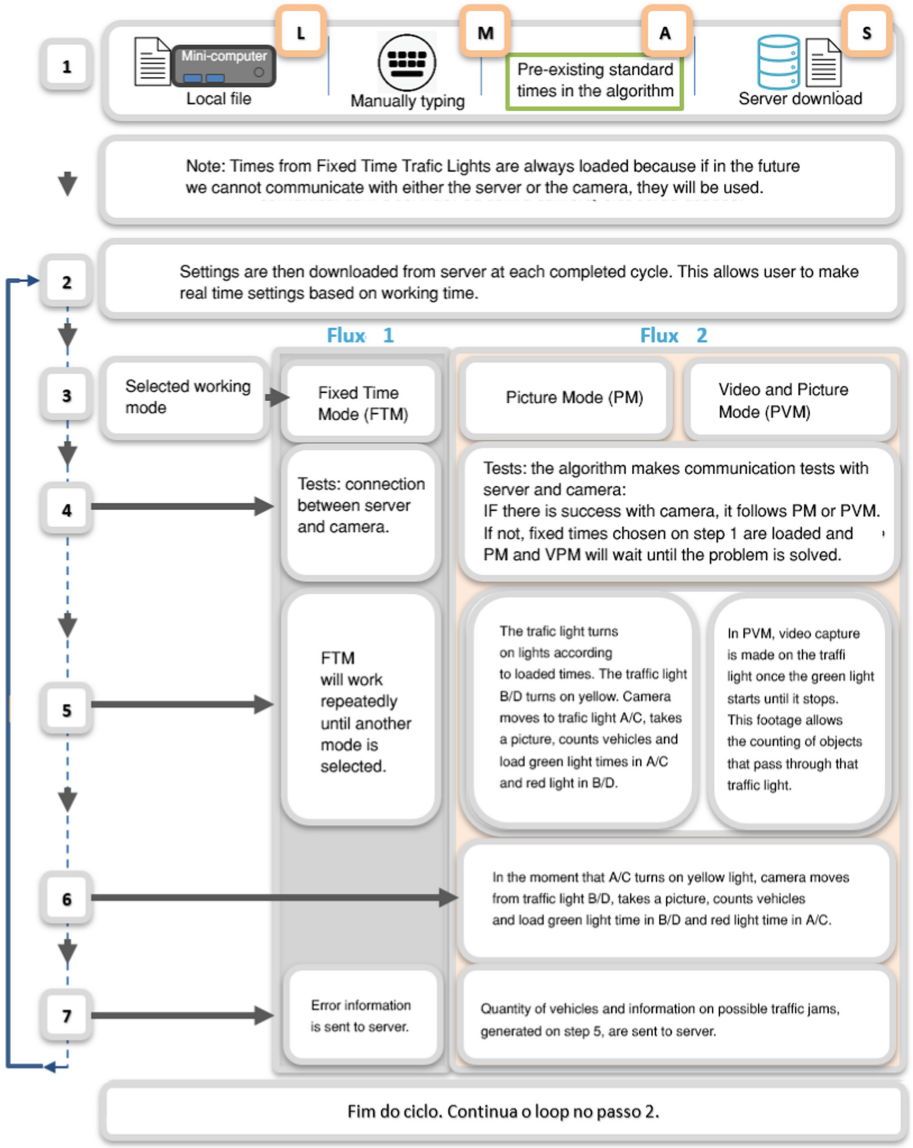


**Fig. 3.** Fluxogram showing the simplified working of the traffic light control algorithm

### 3.6 Object Count and Error Correction on Object Counting

Object counting, whether in photos or video, is done only in the region of interest (ROI) selected by the user via the web in the traffic light general settings screen. Exclusively in the case of video mode, after selecting the ROI and on the same screen of the Web system, the user selects a counting line that has the trigger function, performing the

count as soon as the vehicle makes contact with the line. Figure 4 shows example of video mode counting.

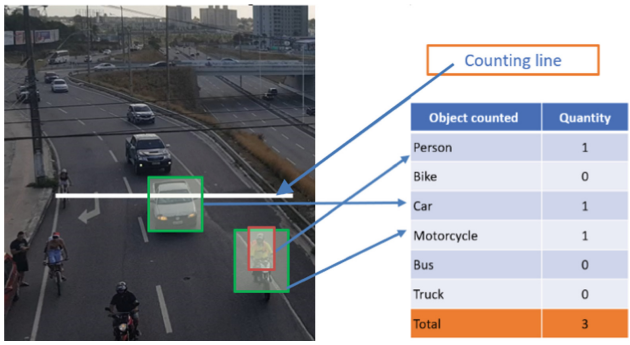


**Fig. 4.** Counting line

After creating the ROI and the counting line, objects are counted through motion detection, using MOG2, or object classification with YOLO-Tiny and MobileNetV2.

The counting can happen in two moments depending on the traffic light operation mode, if in Photo mode, the image capture will be done before the traffic light opens, the objects will be counted through YOLOv3-Tiny or YOLOv4-Tiny, which can be chosen by the user on the traffic light general settings screen. After counting the vehicles waiting for the traffic light to turn green, the time that sets the traffic light will be calculated based on the formula:

$$\text{Green light} = (\text{VWTLG} * \text{NSDVC})/\text{NC}$$

Where, vehicles waiting for the traffic light to turn green (VWTLG), number of seconds defined for a vehicle to cross (NSDVC) and number of carriageways (NC). There is no need to correct vehicle counting errors using Photo mode. The second moment, where there is a need to count vehicles, is in Photo with Live Video mode - in this case - the counting correction techniques will be necessary and shown below.

At this moment, some issues might happen, such as multiple counts of the same object or lack of counting. Table 2 below shows the rate of multiple reads:

**Table 2.** Multiple read errors

| Modelo | Objetos detectados | Taxa acertos (%) | Deixou de contar | Erro de leituras múltiplas | Observações |
|---|---|---|---|---|---|
| | Dia | | | | |
| MOG2 Téc. 1 | 53 | 49,23 | 33 | 21 | Resolução da detecção: 1280x720 |
| Yolov3 80 Classes | 63 | 90,77 | 6 | 4 | Resolução da detecção: 416x416 |
| Yolov3 6 Classes | 65 | 93,85 | 4 | 4 | |
| Yolov4 80 Classes | 65 | 96,92 | 2 | 2 | |
| Yolov4 6 Classes | 66 | 96,92 | 2 | 3 | |
| SSD Mobilenetv2 90 Classes | 62 | 93,85 | 4 | 1 | Resolução da detecção: 300x300 |
| SSD Mobilenetv2 6 Classes | 25 | 38,46 | 40 | | |

To solve the multiple count problem, each object was given a unique ID that allowed its tracking. The idea is for the ID to remain the same until the object disappears from view, but usually the applied technique will not work as expected in most cases, but did resolve the issue – with a success rate higher than 90%.

To resolve the issue of missing object count, a margin of error has been added to the count line. This meant adding more vertical pixels to the count line. Users can choose pixel quantity using the general traffic light settings screen. If the object is within this interval, it will be counted. After performing tests, it was found that this margin of error should not be large, to avoid multiple counts, and should not be too small, otherwise it will not count some objects.

## 4 Tests

This work does not intend to reach high levels of object classification. This can happen, however, depending on several factors analyzed in this section that can lead to further improvements. The main goal here is to count traffic objects and try to come closer to the real life number of objects. Therefore, it was not taken into account whether the classified objects correspond to the real objects. The traffic light studied in this work works based on the number of vehicles that pass through it, in this sense it is extremely important to obtain results closer to reality to optimize the work of the traffic light, thus achieving its objective of increasing traffic flow. In addition, it is of great importance to keep the system running for as long as possible through problem detection and alerting techniques, as well as follow-up of call openings.

### 4.1 Test General Settings

All tests were performed on a DELL 7050 notebook that was in the role of the mini computer processing the images and sending the commands to the Raspberry Pi to control the traffic lights. Notebook settings were detailed in Sect. 3, item 3.2. Tests with MOG2 were done using native video resolution; Tests with YOLO-Tiny had a resolution of 416 $\times$ 416 and tests with SSD MobileNetV2 had a resolution of 300 $\times$ 300. When necessary, the analyzed image was resized, making it larger or smaller, according to the resolution previously described for each algorithm. For example, if the original image was 240p and YOLO was chosen, the image was resized to 416 $\times$ 416.

As the video tests and photos were stored inside the notebook, the frames were limited to 25 fps because the videos were recorded in this setting. This limited the algorithm functioning but did not affect object detection. Tests used images captured at around 3pm. Tests that analyzed pictures were made with YOLO-Tiny since this algorithm is responsible for counting vehicles on pictures. The developer trained models based on algorithms YOLO-Tiny and MobileNetV2, these models names are: YOLOv3 6 classes, YOLOv4 6 classes and SSD MobileNetV2 6 Classes. These models were trained using only object classes found in traffic. This was detailed in Sect. 3, item 3.4. Preparation of images used in training.

For the execution of the tests, the CV algorithm was chosen via the Web platform. The application used was the traffic light control and in it the functions of counting

frames per second (FPS) and showing the images of vehicles at the traffic light were temporarily enabled. In this sense, it was possible to count the vehicles and the FPS for each algorithm used.

### 4.2 On Detecting Objects in Pictures

During the tests carried out on photos, it was observed that the model trained by the authors, based on YOLOv4-tiny, obtained better results than expected in relation to the others, reaching 90% hits in detectable objects eligible (by the authors) in photos in 360p resolution as shown in Table 3. This mode, named Yolov4 6 classes, classified twice as many objects than other algorithms, as shown on Table 4. It was also the best at analyzing resolutions.

**Table 3.** Test 2 results with 360p pictures

| Model | Object Detected | Success Rate |
|---|---|---|
| | Day | (%) |
| **Yolov3** 80 Classes | 6 | 60 |
| **Yolov3** 6 Classes | 5 | 50 |
| **Yolov4** 80 Classes | 7 | 70 |
| **Yolov4** 6 Classes | 9 | 90 |

**Table 4.** Test 1 results with 360p pictures

| Model | Object Detected | Success Rate |
|---|---|---|
| | Day | (%) |
| **Yolov3** 80 Classes | 2 | 40 |
| **Yolov3** 6 Classes | 1 | 20 |
| **Yolov4** 80 Classes | 2 | 40 |
| **Yolov4** 6 Classes | 4 | 80 |

The model trained by the author, named Yolov3 6 Classes, had the lowest hits among others tested, despite utilizing the same image quantities and same settings as Yolov4-Tiny. This shows that Yolov4-Tiny improved its learning and detection in contrast to its previous version.

### 4.3 On Detecting Objects in Videos

During the video tests, it was observed that the model trained by the authors, based on YOLOv4-tiny, was better than the others, reaching 92% hits in eligible detectable objects (by the developer). Table 5 has more in-depth data. This model, named Yolov4 6 classes, classified approximately 28% more than YOLOv3 80 Classes and 67% more than YOLOv3 6 Classes, as shown on Table 5.

SSD MobileNetV2 had the worst results on video 1, but on video 2 it was among those with the best classification rate, as shown in Table 6. It obtained better speed among object classification algorithms evaluated.

Video 1 had the worst classification rate due to vehicle-camera distance, which highlights that distance and recording angle influence object detection.

Average object detection through MOG2 was above 90% on video 1, but had the biggest multiple reading rate, reaching 9%.

Generally, object classification algorithms such as YOLO-Tiny and MobileNetV2 had a great classification rate among all video image resolutions, reaching hits levels of 96% with lower multiple reading objects rate.

### 4.4 On Mean Average Precision (mAP) and their Relation with Resolution and Image Attribution

It was clear that during tests, the relation between object classification, by either YOLO-Tiny or MobileNetV2, is proportional to picture quality and resolution. The bigger the image resolution, both photo and video, and better the image quality attributes such as bits per second rate (kbps) when it comes to video, the better the object classification rate. Under 240p resolution, most algorithms could not detect any object.

On specific videos, the fact they have 25 frames per second gives the algorithm more chances to try to classify objects, and the problem related in the previous paragraph is not as big. However, it reflects object classification error rates. Motorcycles are rarely classified as motorcycles, and in these cases, they were classified either as cars or as people when it was possible to detect either the motorcycle or the person.

**Table 5.** Test 1 result executed in 720p vídeo resolution

| Model | Average FPS | Success Rate (%) | Observations |
|---|---|---|---|
| **MOG2** Téc. 1 | 21,44 | 90,20 | Detected resolution: 1280x720 |
| **Yolov3** 80 Classes | 15,7 | 64,71 | Detected resolution: 416x416 |
| **Yolov3** 6 Classes | 16,88 | 25,49 | |
| **Yolov4** 80 Classes | 13,83 | 78,43 | |
| **Yolov4** 6 Classes | 14,78 | 92,16 | |
| **SSD Mobilenetv2** 90 Classes | 21,21 | 41,18 | Detected resolution: 300x300 |
| **SSD Mobilenetv2** 6 Classes | 28,11 | 3,92 | |

**Table 6.** Test 2 result executed in 720p vídeo resolution

| Model | Average FPS | Success Rate (%) | Observations |
|---|---|---|---|
| **MOG2** Téc. 1 | 18,95 | 49,23 | Detected resolution: 1280x720 |
| **Yolov3** 80 Classes | 15,84 | 90,77 | Detected resolution: 416x416 |
| **Yolov3** 6 Classes | 16,33 | 93,85 | |
| **Yolov4** 80 Classes | 14,29 | 96,92 | |
| **Yolov4** 6 Classes | 14,75 | 96,92 | |
| **SSD Mobilenetv2** 90 Classes | 20,26 | 93,85 | Detected resolution: 300x300 |
| **SSD Mobilenetv2** 6 Classes | 27,03 | 38,46 | |

### 4.5 On Analyzed Techniques Performance

The techniques used to count objects in this article have been divided into two steps. First step: working through MOG2, does movement detection and will always have a bigger success rate than the second category that does detection and identifies kinds of objects. Due to rapid MOG2 processing, it was necessary to create a delay mechanism to maintain the frame rate in 25 fps, otherwise the video would be too fast and rapidly perpass 150 and 200 fms, making video analysis and algorithm counting almost unviable.

In the second step, where algorithms are able to classify objects, video tests showed that MobileNetV2 was overall faster, obtaining better frame rates and easily reaching 25% advantage over YOLOv4-Tiny, despite the fact YOLO-Tiny versions used bigger video resolution than MobileNetV2, as shown in item 4.1. However, YOLOv4-Tiny version was better than MobileNetV2 since it identified more objects in all tests. YOLOv3-Tiny also shared the same advantage over MobileNetV2 in mostly all tests.

### 4.6 Test Results After System Implementations

Below are the results of a test done to quantify the maximum time a vehicle waits for a traffic light to open when the other road is empty. Next, the traffic light efficiency test in Smart mode will be described:

**Waiting Time Test**

The traffic light general settings screen gives us the option to set up a minimum time the traffic light with less than a four seconds timer will get. It is possible, in this instance, that the traffic controller enters a value such as zero so that the green light in that traffic-less traffic light does not light, prioritizing the traffic light that has traffic. This solution might cause an issue where all traffic lights get values lower than four seconds. In this scenario, traffic lights would get red lights constantly until vehicles start arriving. A complementary mechanism would verify if the master traffic light has a timer with less than four seconds, and if that is the case, it will receive 8 s while the other gets the minimum time chosen by the traffic controller.

After tests carried out with this mechanism, it was proved that the waiting time of vehicles arriving at the main traffic light is equal to the time given to the yellow traffic light of the other traffic lights where there is no flow of vehicles. The maximum waiting time for those arriving at non-main traffic lights is 8 s plus yellow traffic light time, at times when there are no vehicles at main traffic lights, this solves one of the main problems of fixed time traffic lights that commonly turn on the traffic light green for 1, 2, 3 or more minutes, even when there is no traffic.

**Traffic Flux Test**

The main purpose of our work is to control the traffic light according to how many vehicles are waiting for the green light. To validate the proposal, some tests were carried out using photos with vehicles waiting for the green light of the traffic light. The solution proposed by this work was made for two-time traffic lights and the following test was performed on a cross-shaped road, where the first time was identified as A-C and the second as B-D (in Table 7), with two lanes on each way. For simplicity, the yellow light time and the minimum time given for the red light, mentioned in item 4.5, when there is no flow of vehicles, were not considered. Table 7 shows the test results:

**Table 7.** Comparison between fixed time semaphore and smart semaphore

| Item | Traffic light | Vehicles waiting for the green light | Estimated average time for a vehicle to cross | N. of lanes per lane | Total time for crossing all vehicles | Fixed Time Semaphore set to 60s (FTS) | Smart Traffic light |
|---|---|---|---|---|---|---|---|
| | | (N. vehicle | x seconds) | / lanes | = time + FTS | Flow: 30 vehicles per minute | |
| 1 | A-C | 46 | 4s | 2 | 92s + 60s | 152s | 60s + 24s + 32s = 116s |
| | B-D | 12 | | | 24s + 60s | 84s | 84s |
| 2 | A-C | 60 | | | 120s + 60s | 180s | 60s + 20s + 60s = 140s |
| | B-D | 10 | | | 20s + 60s | 80s | 80s |
| 3 | A-C | 40 | | | 80s + 60s | 140s | 80s |
| | B-D | 0 | | | 0s | 0s | 0s |
| 4 | A-C | 8 | | | 16s | 16s | 16s |
| | B-D | 60 | | | 120s + 60s | 180s | 60s + 16s + 60s = 130s |
| 5 | A-C | 14 | | | 28s | 28s | 28s |
| | B-D | 50 | | | 100s + 60s | 160s | 60s + 28s + 40s = 128s |

Table 7 shows the comparison between the fixed-time traffic light (FTTL) and the smart traffic light (Si) proposed by this work. Item 1 shows that Si was more efficient and managed to provide flow to the 46 vehicles using 36 s less than the FTTL. In item 4, the time savings were even greater, being 50 s compared to the FTTL.

In item 3, in addition to the gain in relation to time, there was also 1 cycle less of the traffic light opening since B-D did not open, when the algorithm verified that there were no vehicles waiting for the green light, the traffic light did not open. In this case, there was a gain of 60 s by Si in relation to the functioning of the FTTL.

Examples with three opening and closing cycles were used in these tests. They prove that the smart traffic light promotes considerable time savings, reaching an improvement of 33% in relation to the fixed time traffic light, as mentioned in item 3. In general terms, the results were within the expected reaching close to 35%, results obtained by (KANUNGO, SHARMA and SINGLA, 2014).

## 5 Limitations

Considering this paper intends to propose a solution based on working traffic lights, camera and network in Brazil, where they use a single PTZ camera, the functioning mode was a bit limited. With a single camera, it is not possible to simultaneously observe traffic in every road to compare and see which has more traffic in order to prioritize it. This could make the algorithm more efficient and simple. Another limitation imposed by a single camera was that once the red light is on, if there were multiple cameras it would be possible to record the flow of vehicles under all traffic lights. However, it was only possible to choose one side with the green light, and that was the one that had the most vehicles counted before the green light came on.

Regarding the evaluation of the system, it is necessary to do more case studies to bring more data about the benefits, problems and challenges of using the proposed solution. The system evaluation limited itself in previously recorded video and picture analysis since it was not possible to use traffic lights in a real environment.

## 6 Contributions and Future Works

Because of the work presented in this paper, the following contributions are numbered:

1) The development of an intelligent traffic control system based on computer vision was proposed. With this system it is expected to easily promote the implementation of intelligent traffic lights for local and remote operation, improve the flow of traffic and detect problems quickly and automatically. Currently, multiple cities rely on conductors and citizens to report issues, causing delays on repairing processes.
2) Enable the implementation of a smart traffic lights control system by using camera and network infrastructure that is already in use by states and cities, drastically reducing implementation costs in comparison to other marketing proposals.
3) The web platform, TEOP, responsible for controlling the system and traffic light control algorithm, will be available to download.
4) Allowing temporary traffic light use in the following instances:

Mobile traffic lights to meet temporary events demands;
Viability tests while installing new traffic lights and;
Temporary substitution in places where traffic lights are necessary, but there were problems that shortly made the fixed time traffic light stop working.

Regarding future work, there is still room for implementations and improvements to reach more efficient results.

1. Implementing the green wave to allow vehicles to go through a bigger adjacent traffic lights number, keeping a constant medium velocity and avoiding encounters with a red light. This is the more interesting scenario to work with a smart traffic light to avoid jams on traffic lights located immediately after the smart traffic light.
2. Considering that the system includes regular Brazilian traffic lights, it is possible to expand the search to adapt the control algorithm to other types of traffic lights so that it can operate in more models.
3. To make the traffic flow even more efficient, it is recommended to implement a new functionality using the data obtained by the PVM, which counts the vehicles under the green traffic light. Based on this data, it is possible to predict opening and closing times, suggesting times for the traffic light to work more and more accurately.
4. Regarding improvement on detection and counting, there is still space to improve mean Average Precision (mAP) on vehicle classification and MOG2. This will make the system work more efficiently and, besides, it can lower multiple detection error rates and uncounted vehicles. Tests in other climate conditions created by the user, based on objects captured by the traffic light camera, could verify if there are any improvements on object detection using this approach.
5. For the system to be fully functional in a production environment, it is necessary to interconnect the PC computer with the Raspberry so that the Raspberry can execute the commands, coming from the PC, to turn the traffic lights on and off. Following this path, it is also necessary to allow communication between the traffic light control algorithm and the PTZ camera using the ONVIF protocol, which is standard. It works by sending out commands as codes to connected cameras, allowing the moving of cameras to traffic lights that will open, in turn leading to system improvement.
6. Another improvement indicated for future work is the creation of a model based on the set of images generated in the traffic light. It is possible that the models created using these photos achieve a better average hits than those achieved in this work.

## References

1. IBGE. Frota de veículos no Brasil (2020). https://cidades.ibge.gov.br/brasil/pesquisa/22/28120?ano=2020. Acesso em 12 ago. 2020 (BAUZA; GONZALVES; SANCHEZ-SORIANO, 2010)
2. Krauss, M.: Automação de sistema semafórico. 67 f. Trabalho de Conclusão de Curso (Tecnologia em Automação Industrial). - Universidade Tecnológica Federal do Paraná. UFTPR, Curitiba (2014)
3. Martins, J.A.: Transporte, Uso do Solo e Autossustentabilidade: Teoria e Prática para a Previsão de Impactos sobre a Qualidade do Ar. 1996, 250f. Tese (Engenharia de Transportes). UFRJ, Rio de Janeiro (1996)

4. Silva, G.C.: Tráfego, monóxido de carbono e ruído em áreas urbanas: o caso de Florianópolis. 1998. 126 f. Dissertação (Engenharia Civil). UFSC, Florianópolis (1998)
5. G1. Distrito Federal. Semáforos inteligentes devem ser implantados em fevereiro no DF, diz Detran. G1, Distrito Federal, 08 November 2017. https://g1.globo.com/distrito-federal/noticia/semaforos-inteligentes-devem-ser-implantados-em-fevereiro-no-df-diz-detran.ghtml. Acesso em 29 maio. 2021
6. Castro, F.S., et al.: Identificação e Rastreamento de veículos utilizando fluxo ótico. Colloquium Exactarum **7**(2), 73–88 (2015)
7. Valente, J.: Aumento do monitoramento traz debate sobre modernização e privacidade. Agência Brasil. Brasília, 12 September 2019. https://agenciabrasil.ebc.com.br/geral/noticia/2019-09/aumento-do-monitoramento-traz-debate-sobre-modernizacao-e-privacidade. Acesso em 01 maio 2021
8. Kanungo, A., Sharma, A., Singla, C.: Smart traffic lights switching and traffic density calculation using video processing. In: Recent Advances in Engineering and Computational Sciences (RAECS), pp. 1–6 (2014)
9. Marciano, J.L.P.: Segurança da informação: uma abordagem social. 212 f. Tese (Doutorado em Ciência da Informação)-Universidade de Brasília (2006)
10. Bhatnagar, S., Bhatnagar, Y.: Distributed Trafic Control System (2019)
11. Ardhapure, M.R., et al.: Smart traffic signal management (2018)
12. Komasilovs, V., et al.: Traffic Monitoring System Development in Jelgava City, Latvia. VEHITS (2018)
13. The, P.N., et al.: Real time ARM-based traffic Level of Service classification system. In: 2016 13th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON). IEEE (2016)
14. Altundogan, T.G., Karakose, M.: Image processing and deep neural image classification based physical feature determiner for traffic stakeholders. In: 2019 7th International Istanbul Smart Grids and Cities Congress and Fair (ICSG). IEEE (2019)
15. Cismaru, S.-I., et al.: Experimental study regarding an intelligent control system for traffic light intersections. In: 2020 International Symposium on Power Electronics, Electrical Drives, Automation and Motion (SPEEDAM). IEEE (2020)
16. Sandler, M., et al.: MobileNetV2: Inverted Residuals and Linear Bottlenecks, 13 January 2018
17. Aquino, C.: Dos Barris à Paralela: 88 Smart traffic lights will reduce traffic time. Correio 24 horas. Salvador, 08 may 2017. https://www.correio24horas.com.br/noticia/nid/dos-barris-a-paralela-88-semaforos-inteligentes-vao-diminuir-tempo-no-transito/. Acesso em 31 May 2021