UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

A LOW-COST EMBEDDED VEHICLE COUNTING AND CLASSIFICATION SYSTEM BASED

Josué Fernando Navarro Pabón

Ingeniería Electrónica y Automatización

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Electrónica

Quito, 24 de diciembre de 2020

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

A LOW-COST EMBEDDED VEHICLE COUNTING AND CLASSIFICATION SYSTEM BASED

Josué Fernando Navarro Pabón

Nombre del profesor, Título académico **Diego Benitez, PhD.**

Quito, 24 de diciembre de 2020

© DERECHOS DE AUTOR

Nombres y apellidos:	Josué Fernando Navarro Pabón
Código:	00136095
Cédula de identidad:	1719251173
Lugar y fecha:	Quito, 24 de diciembre de 2020

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

En este documento se resume la experimentación realizada en base al entrenamiento e implementación de un sistema de detección y conteo de vehículos en tiempo real y de bajo coste. Para esto el sistema ha sido implementado sobre una Raspberry Pi 3b+, un minicomputador de bajo costo programable (sistema embebido), además de añadir poder de procesamiento por medio de un procesador NCS2 de Intel. El objetivo del proyecto ha sido buscar la forma de optimizar la velocidad de procesamiento sin comprometer la exactitud con la que el sistema realiza la detección y el conteo. Se analiza también la eficiencia de dos algoritmos de seguimiento de objetos, los cuales son el algoritmo de seguimiento de centroides y el algoritmo basado en filtro de Kalman. En las conclusiones del proyecto se menciona la necesidad de migrar el sistema de conteo de vehículos a un minicomputador más potente, además de tomar consideraciones sobre el hardware como la refrigeración. Finalmente, se hace mención del trabajo a futuro que se debe realizar sobre el proyecto, lo que incluye la implementación del sistema de conteo para controlar semáforos de forma inteligente por medio de ecuaciones en diferencias.

Palabras clave: Aprendizaje de máquina, aprendizaje profundo, sistema embebido, Raspberry, redes neuronales, algoritmo de seguimiento, seguimiento de centroides, filtro de Kalman.

ABSTRACT

This document summarizes the experimentation carried out based on the training and implementation of a low cost and real time vehicle detection and counting system. For this, the system has been implemented don a Raspberry Pi 3b+, a programmable low-cost minicomputer (embedded system), adding processing power through an Intel NCS2 processor. The objective of the project has been to find a way to optimize the processing speed without compromising the accuracy with which the system performs detection and counting. The efficiency of two object tracking algorithms is also analyzed, which are the centroid tracking algorithm and the algorithm based on the Kalman filter. In the conclusions of the project, the need to migrate the vehicle counting system to a more powerful minicomputer was mentioned, in addition to taking considerations on hardware such as cooling. Finally, mention is made of the future work that must be done on the project, which includes the implementation of the content system to control traffic lights intelligently through discrete difference equations.

Key words: Machine Learning, Deep Learning, embedded system, Raspberry, neural networks, tracking algorithm, centroid algorithm, Kalman filter.

Contents

List of Tables

List of Figures

1 INTRODUCTION

Nowadays in modern society, one of the main ingredients that allows us to get involved in the understanding of the mobility phenomena and urban culture in general, is without a doubt the car. It is known as a material used for transportation; but throughout human history its geniality has surpassed this main concept, making it transcending into more than just a tool. It is a symbol of economic status, technological capacity and many other ideological notions, that wrapped together compose the possession of a car in a daily necessity greater than just its functional appeal of conveyance. Naturally, this increase in the demand, clearly encouraged by vehicle manufacturers, gave birth to a market that is one of the economical engines of the top nations in the planet. A clear example of the previous statement is the progressive growth of the vehicle market of Ecuador in 2020; showing sales reaching 68.280 new units until June; which contrasts when comparing the 43.600 units sold in the previous year (Ament 2015).

As a direct consequence of the previously mentioned socio-economic environment, from the second half of the XX century until modern times, urban development and mobility policies in the vast majority of modern cities have been focusing on acquiring cheaper and legal vehicles; jointly increasing the necessary infrastructure for the growth of the vehicle fleet. Unfortunately, the constant increase in the number of vehicles on the roads and not having the corresponding development in the communication between them, low toxic fuels, safety road education, etc., create various negative effects such as pollution due to vehicle emissions, risk for cyclist and pedestrians when circulating, among others. The increase in the number of vehicles produces a gradual and progressive reduction in the average traffic speed, reaching in the worst-case scenario long waiting periods mainly manifested in the crossings of 2 or more roads. Produced principally due to the obvious need of vehicles to circulate in certain moments by the same area. These agglomeration phenomena are commonly known as vehicular congestion. During a certain period of vehicular congestion, almost all vehicles move at really low speed with high engine revolution conditions, which end up in greater pollution concentration as well as a deteriorated travel experience for the passengers, loss of labor productivity in response of the longer required travel time and many other harmful consequences, both for the environment and the population in general.

The use of traffic lights is the most common traffic control system used nowadays. Their beginnings date back to the late XVII century, but in modern times they are design with automatic switching lights based in a manually programmable timer via a single control board linked by physical wires (Ament 2015). There are in existence different versions of traffic lights with synchronized light switching, which are used in concurrent roads to allow more fluent periods of movement while other versions that are built with a central remote control light switching system, where the switch intervals are calculated by computers that use data such as the approximate traffic flux in roads at a determinate schedule. The nature and characteristics of the software that is used to calculate the light changing sequences in this type of traffic lights is commanded by the hardware used in the central control unit, which may be a commercial system designed specifically for this purpose or a system adapted from general.

Due to inherent environmental and socio-economic effects and safety implications, traffic control systems physical and operational characteristics are regulated by government entities in some countries. In the USA, the National Electrical Manufacturers Association (NEMA) is the organization who is in charge of defining the accepted standards that connectors and operation ranges should have. These regulations defined by NEMA are used as bases for the regulation in most of western hemisphere countries (NEMA 2020).

In an urban road system, the traffic control on concurrent roads is essential to guarantee safety in the transit of vehicles and pedestrians. When the road system presents high vehicular load, this condition gets worse by the waiting period that vehicles must experience on certain necessary crossings that allow alternate passage from different directions. This waiting period, when controlled by indicator lights (traffic lights) is generally inefficient due to, among other reasons to:

- Keeping vehicles from moving on one of the lanes while there are no vehicles circulating on the other one.
- Keeping similar crossing times in roads with low and high traffic flux.
- Stopping vehicles to allow pedestrians to cross, even though there are not any of them waiting.

- The change in traffic conditions for people and vehicles during different times of the day and different days of the week.

The traffic lights controlled by a conventional timer present a light switching rate that allows the increase or decrease in the flow of people and vehicles in concordance with the stipulations that the programmer considers of greater or lower importance. For example, seeking a bigger flow in one of the roads, due to its high vehicle concurrence. However, the most common programming allocates an equivalent passage time for each of the concurrent roads; in order to allow an average wait time for all roads on the cruise without the need of reprograming the timer when the traffic load conditions change. The use of remote computerized control systems is less widespread, mainly caused by considerations of the high economic costs generated by the specialized infrastructure requirements for implementation and capable maintenance staff.

The maximum traffic load relief capacity of a road is determined by its inherent physical characteristics and none of the existing traffic control systems can increase traffic flow beyond that. However, it is imperative to optimize the control mechanisms to maximize the potential of the relief capacity capabilities of the systems. Based on the previous statement, the characteristics we are seeking to improve the traffic lights control systems are:

- Continuous detection of the number of vehicles traveling in each direction of the concurrent roads in crossroads with adaptive light switching for variable traffic conditions.
- The capability of not stopping vehicles in one of the roads when no other vehicle is crossing the other way on a concurrent road.
- Competitive implementation and maintenance costs, that allows the system to be used in low-income communities.

To solve the continuous vehicle detection and vehicle counting problems, in this investigation paper it is proposed a possible solution based on image recognition systems using computerized vision open-source libraries OpenCV (OpenCV-team 2020), developed on Linux operative system for embedded systems developed in turn with open-source tools

project development from Linux distributions. The architecture used to this approach is YOLOv4 tiny (Bochkovskiy et al. 2020), a version of YOLOv4 focused on processing speed and optimized for embedded systems.

The objective of developing the system on a embedded system is to give an alternative to other high cost electronic devices such as infrared sensors, loop inductors, or piezo-electric sensors. Apart from a low cost, the use of the operative system and the open-source code libraries will allow to have tools developed and maintained by thousands of programmers all around the world, avoiding incurring in large licensing costs. The final objective of the project is to design an intelligent traffic control system (ITCS), which consists of an embedded computerized system that, through a low-cost camera, detects moving vehicles in real time, in order to carry out the count of vehicles that have traveled in a certain time. Open-source software offers ITCS users, both in private entities and public institutions, not only immediate economic advantages due to its inherently free nature, but it also allows initial implementations and future maintenance of the system, being done by staff with just generalized technical knowledge, to be possible and affordable.

2 RELATED WORKS

The document *Computer Vision Classifier and Platform for Automatic Counting* (Espinoza et al. 2017) describes a collective monitoring platform for data collection. The embedded system used in the video analysis is a Raspberry Pi, which is responsible for the detection and counting of vehicles; The system is designed to work on real time. However, its main function is to collect data to generate a free access database. This is the reason why the videos are not analyzed at the exact moment in which they were recorded. The videos end up being saved to discern a strict frame to frame analysis. In contrast with the proposed work, the objective of the embedded system is to give an approximation of the vehicle quantity on the real time instant; in other words, it must be able to give information at the moment were vehicles are detected. Document *RT-VC: An EICIENT Real-Time Vehicle Counting Approach* (Alghyaline et al. 2019) describes a vehicle detection and counting system, in this case using the Kalman

filter. This type of filter is a prediction algorithm commonly used in tracking to determine the next possible position of a moving object. Jointly with the Kalman filter, the Hungarian algorithm is used. The algorithm that was previously mentioned is a multiple-indicator probabilistic algorithm that takes various indicators (within them the Kalman filter prediction), to take the decision of relating the same vehicle in two different frames. This document is being used as a reference to use YOLO as a better choice to work out the vehicle identification. However, the algorithms used for tracking, have been reconsidered, because they are not intended for embedded systems.

3 MATERIALS AND METHODS

3.1 Hardware

3.1.1 Embedded system.

An embedded system is a computational system aimed for the execution of limited dedicated functions. In contrast with the personal computers (PC), that are designed to cover a wide range of necessities, the embedded system are designed to be focused on specific necessities covering. It used in this project is the low-cost computer Raspberry Pi 3 B+ model (Raspberry Pi Foundation n.d.).

3.1.2 Neural Computing System.

A computing neural system is a low consume, deep inference learning development system tool. This allows the development of artificial intelligence (AI) applications. In this project the Neuronal Compute Stick 2 (NCS2) from Intel is used, which it is known for its compatibility with a great range of last generation devices thanks to its USB format. The NCS2 contains the Movidius Myriad X (MMX) processing unit from Intel, which has 16 processing cores and a deep neuronal networks hardware accelerator (Intel 2020). Its main goal within the project is to improve and accelerate the Deep Learning (DP) capacity of the host device, in this case, the Raspberry Pi 3+. Due to the MMX processor; the NCS2 is capable of reaching a performance

comparable to the one that a desktop processor can achieve. The calculations that the program will perform to solve the neural networks will be computed by the NCS2, by doing it in such way that it does not occupy a large percentage of the processing power from the computer. For this reason, the NCS2 is implemented into this system.

3.2 Convolutional Neural Network

Convolutional neural networks (CNN) are defined as a type of deep learning algorithm that takes an input image and assigns different priority levels to various characteristics of it. Its architecture resembles the connectivity patterns that are seen in neurons of the human brain. A CNN is able to capture space-time dependencies in an image, through the application of filters. These type of architecture fits better to image data-sets because it reduces the number of parameters involved while taking into account that weights of the neurons are reusable.

Object recognition systems have three characteristics that must be satisfied to consider a successful detection. The first one refers to image classification, which means that it must identify what is in the image and its associate parameter is called class. The second one refers to object localization, identifying in which part of the the image the object is located; over the image this is showed as bounding boxes. These two aspects that were previously mentioned are related only with a single object. Being objects detection the last aspect, this one requires the possibility of finding all the objects in one image, including the previous aspects of classification and localization.

There are few different algorithms focused on object detection. However, two important groups can be identified: algorithms focused on classification and algorithms based on regression. Algorithms focused on classification are implemented in two stages. In the first stage, the image is divided into regions to later be analyzed using CNNs while the second stage identifies objects in each region.

On the other hand, algorithms based on regression try to predict classes and bounding boxes over the whole image (without sectionize it), to later analyze it using CNNs in those regions that showed the presence of any object, no matter what. Subsequently, the algorithm gives classes to the selected regions using probability. This type of algorithms is faster than

the ones based on classification, but this increase on speed produces a decrease in accuracy. These algorithms are specially used in real-time applications.

You Only Look Once (YOLO) is an open-source system dedicated to the object's detection in real time and algorithms based on regression. Its main characteristic is to require only one CNN to detect objects in images; this kind of detection systems are known as one stage systems. YOLOv4 has been developed by Bochkovskiy et al. 2020 and has undeniable improvements both in processing speed, measured in frames per second (FPS), and in accuracy. The main changes of YOLOv4 respect to its previous versions is the improvement in its ability to work on a single graphic processing unit (GPU), in addition to the application of techniques such as Bag of Freebies and Bag of Specials that improve network performance. As part of this system's family, YOLO tiny is a variant of the YOLO's algorithms optimized for embedded systems; YOLOv4 tiny sacrifices accuracy for speed.

3.3 Object detection

Object detection architecture requires three phases to complete the process: Backbone, Neck, and Dense Prediction. Backbone, is the features-extraction architecture, where networks and algorithms that use convolutions and other mathematics operations extract characteristic parameters from images. Darknet53, an open-source neural network framework (NNF), has 53 convolutional layers that make it more accurate but much slower. Darknet53 is used as Backbone taking advantage of the fact that it is more efficient when it is trained with a low number of objects; around ten objects for maximum efficiency. Neck phase has the purpose of add extra layers between backbone and dense prediction phases. These parameter aggregation methods are used to get higher accuracy. YOLOv4 uses modified versions of Path Aggregation Network (PANet) and Spatial Attention Module (SAM) algorithms (Świeżewski 2020). Finally, Dense Prediction phase is the part of the system used to locate bounding boxes and classify what's inside each box.

YOLOv4 tiny, as a one stage systems, first applies the convolutional network over the whole image. This is possible because YOLO algorithms allows to use a set of features (extracted at backbone phase) that represent all the registered objects, avoiding looking for

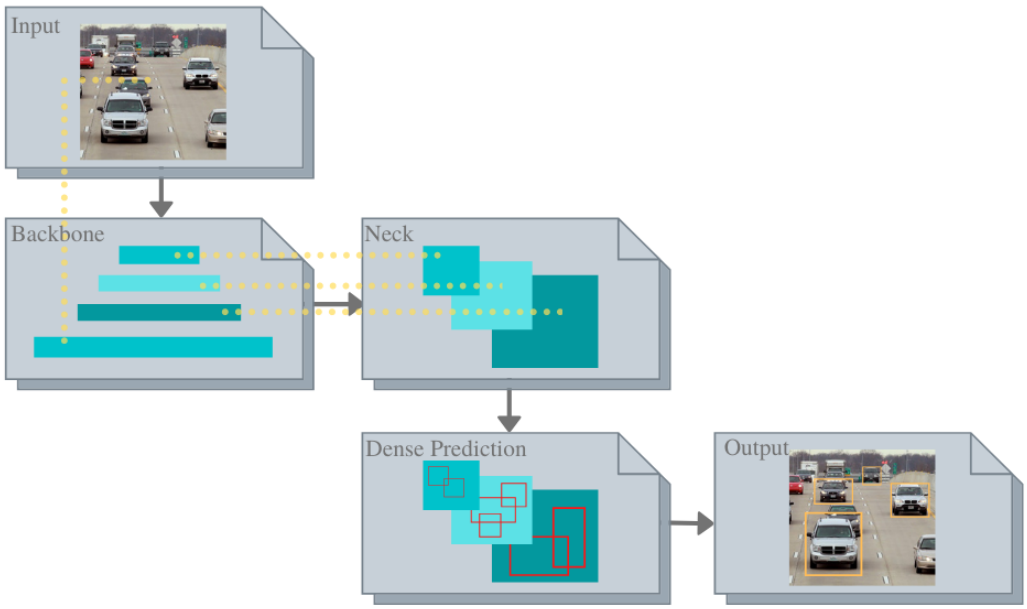

Figure 1: Object detection architecture for one stage system

every object over the whole image. After finding the areas with possible objects to detect, and having identified which objects are in them, the estimation of accuracy is calculated using the Intersection over Union criterion (IoU).

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \tag{1}$$

IoU is a metric that evaluate the similarity between the object detected bounding box and the predicted one (Rosebrock 2016). The overlap area is the section both bounding boxes share with each other, and the Union area is the total area used by both. The relation between both areas are represented in the equation (1) and a visual example is in Figure 2.

Figure 2: Intersection over Union

3.4 Object tracking

Tracking is the process of identifying the same object in different frames, in other words, the recognition of an object that moves in time. For a neuronal network, knowing if an object in two or more frames is the same or another one, is normally impossible. That is why felling to more predictable probabilistic methods to achieve that the system will be able to decide; is needed.

The Kalman filter is a predictive mathematical model that, within object tracking, makes it possible to approach the next position of a moving object from the information obtained of previous positions. Because this filter is a recursive algorithm, it is a reliable option in real-time applications. By itself, the Kalman filter does not allow to conclude if two objects in different frames are the same one, so it is complemented with the Hungarian algorithm. The Hungarian algorithm is a probabilistic method that gives numerical values to certain parameters extracted from the same system. Among the parameters used there are: the IoU, the Euclidean distance between centroids, the shape size and convolution cost. This algorithm collects all this information and gives it different values, to finally compute the probability that the objects in different frames are the same object.

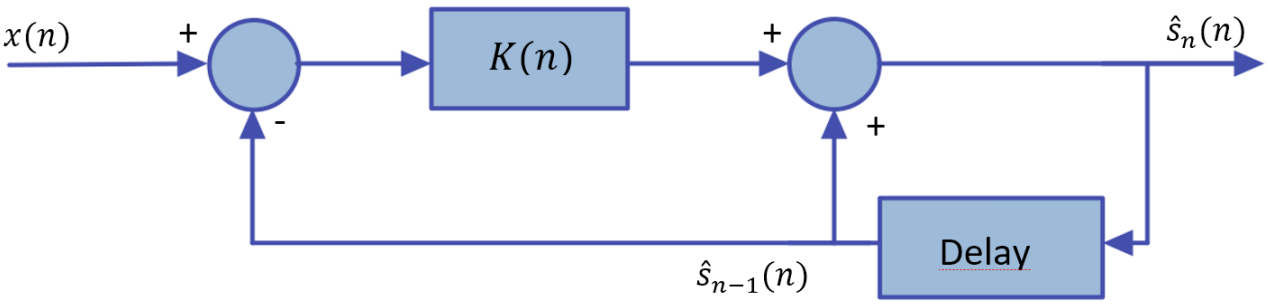

Figure 3: Block Diagram for Kalman filter

Even tough the combination of the Kalman filter and the Hungarian algorithm are an option that ensures high precision within vehicle tracking, like its seen in the documentation of Chen et al. 2012, this also represents an additional operational cost due to their characteristic of not being designed to put up with embedded systems. The use of some of these algorithms causes FPS to decrease.

The centroid tracking algorithm (CTA), suggested in Heredia and Barros G. 2019 is an alternative with a lower operational cost. This algorithm works by granting a numbered centroid to each object that is detected on the screen. By delimiting a maximum reference distance (in pixels) when moving to the next frame, a centroid that is within the delimited area is searched. If a centroid within the delimited area is found, it is the same centroid; otherwise, it is assumed that the object disappeared. This algorithm is simple and does not requires difficult calculations, but it is prone of doing mistakes, for example, while having two objects very close to each other.

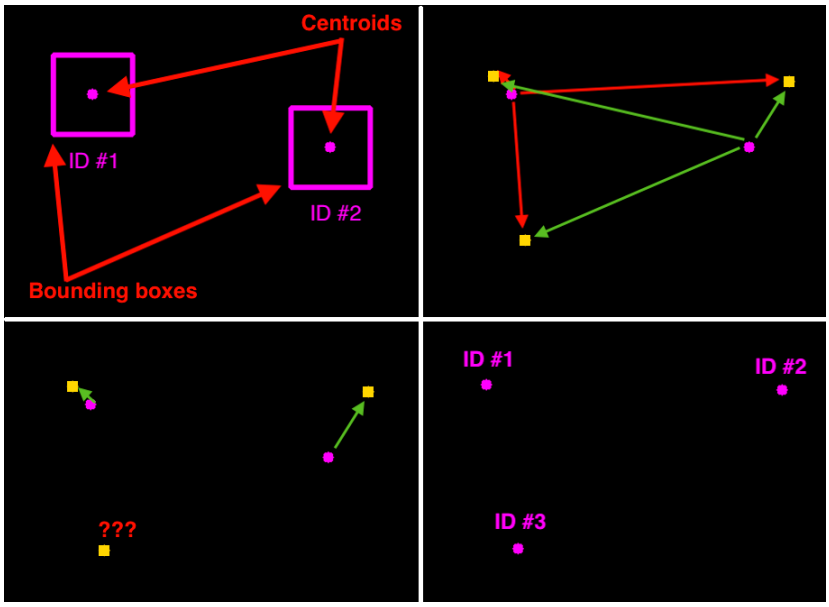

Figure 4: Frame explication of CTA

Regardless of the tracking method, vehicle counting must be done by using a reference zone. As soon as it detects the crossing of the object, a certain reference passes and the counter increases by one unit. The importance of the tracking process in this stage, resides is capability of avoiding counting the same vehicle as two on different frames. The one most prone to making mistakes at this stage is the centroid tracking algorithm.

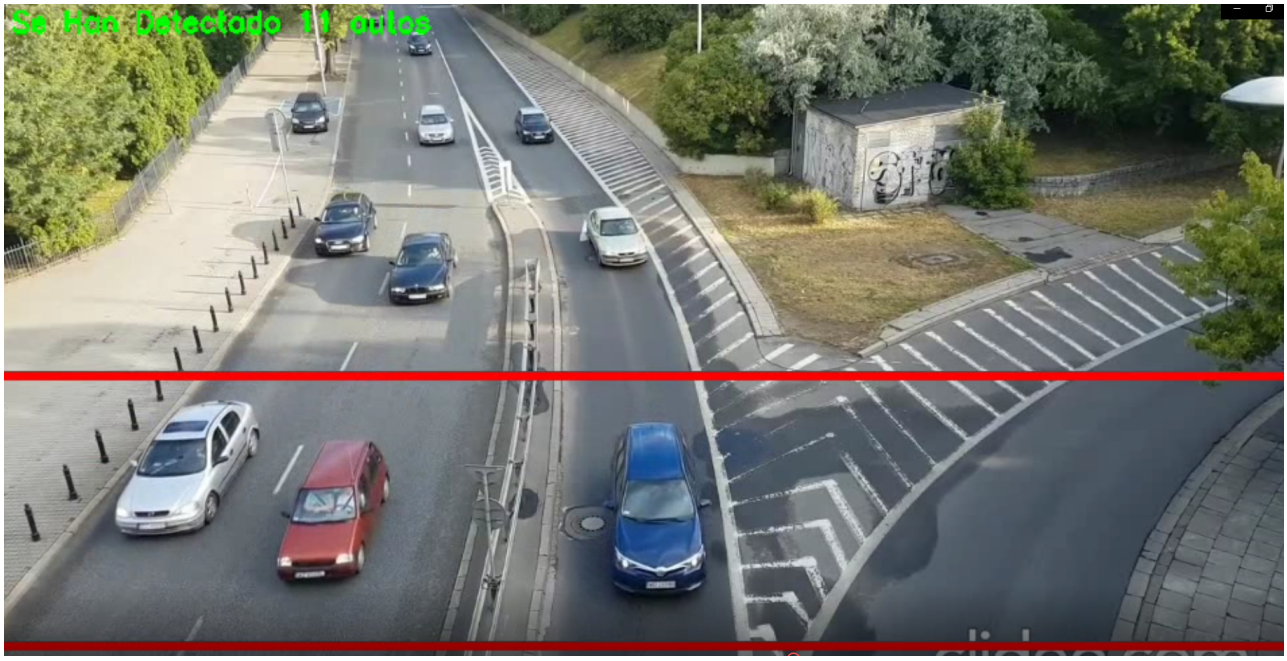

Figure 5: Reference line for counting vehicles

4 PROCESS AND RESULTS

4.1 Training

With the objective of increasing the system's accuracy, a vehicle focused training based on the Darknet53 model was made. The image data-set is obtained from Google Open Source (Google 2020b), which is a data-set that contains more than 9 million of different images from various categories, all of them with their respective correctly segmented bounding boxes. Around 1000 images were extracted from this data-set, which included various vehicle types such as cars, buses, trucks and vans. Afterwards, the images are entered into Roboflow (Roboflow, Inc. 2020), a website that is responsible for transforming the set of correctly identified images into a trained computer vision model. The web gives the possibility of creating models for different systems, including YOLO Darknet. From this website the training files and labels were extracted. The last one containing the recognition pattern for the object's identification.

After this, the training phase, which does not need to be executed in the embedded system, begins. Instead, It was used the Google Colab (Google 2020a) (GC) alternative, which allows the system to create a Notebook with a virtual console that provides a greater GPU machine

for a period of 12 hours. For training, an NVIDIA Tesla P100 card was used and the GC Notebook was set to train the pre-trained Darknet53 model.

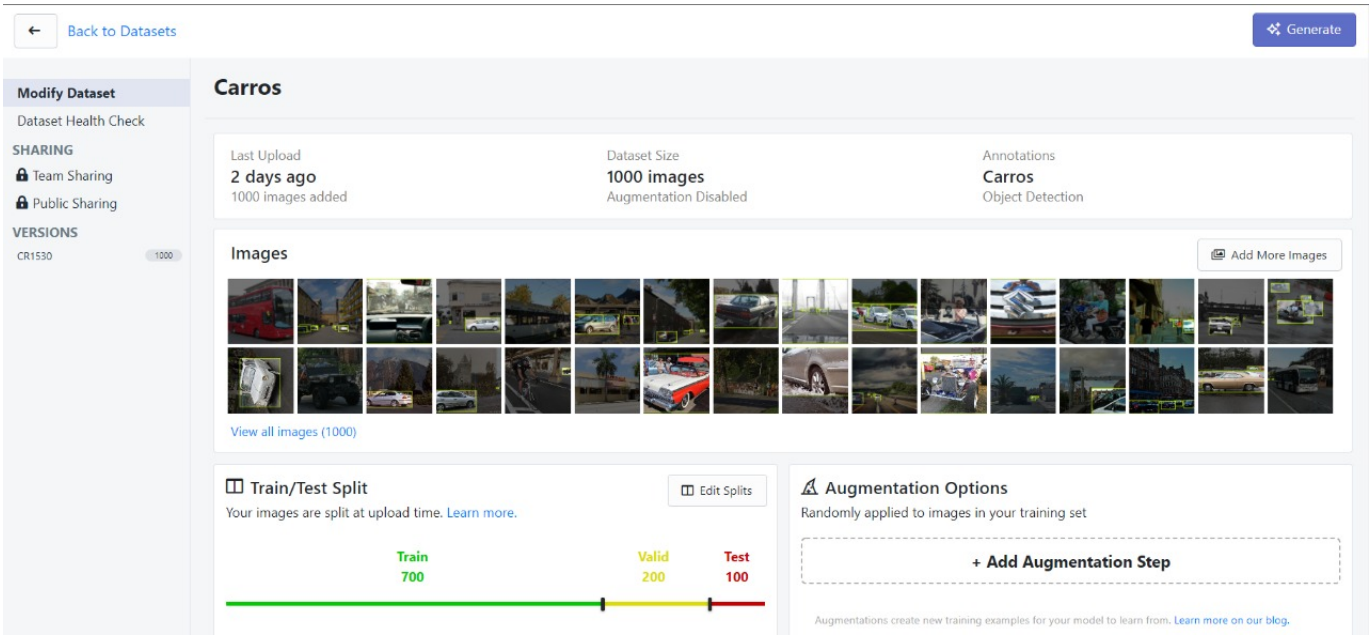

Figure 6: Google Colab training

Overfitting commonly occurs in this type of process. It is generated when the system or the neuronal network is trained with a high precision standard, making it unable to recognize slightly different objects. In order to analyze and avoid overfitting, a training of 8000 iterations was done, in other words the system makes logical connections a total of 8000 times. These 8000 iterations are divided into groups of 1000, testing the system for each of these. Thanks to this, overfitting was detected in around 6000 iterations . For each process, the division of the dataset was 700 images for training, 200 for validation (cross validation set) and 100 for testing, where the number of correctly detected vehicles was compared with the real number of cars to determine the accuracy factor. Of all the suggested models, the one with the highest accuracy was used.

The training process provides all the necessary files that will be used in the implementation within the embedded system. These are the configuration (.cfg) and the probability files (. weights). GP allows to make testing on the same virtual console of the notebook. Over 5000 iterations , predictions started to have recognition problems.

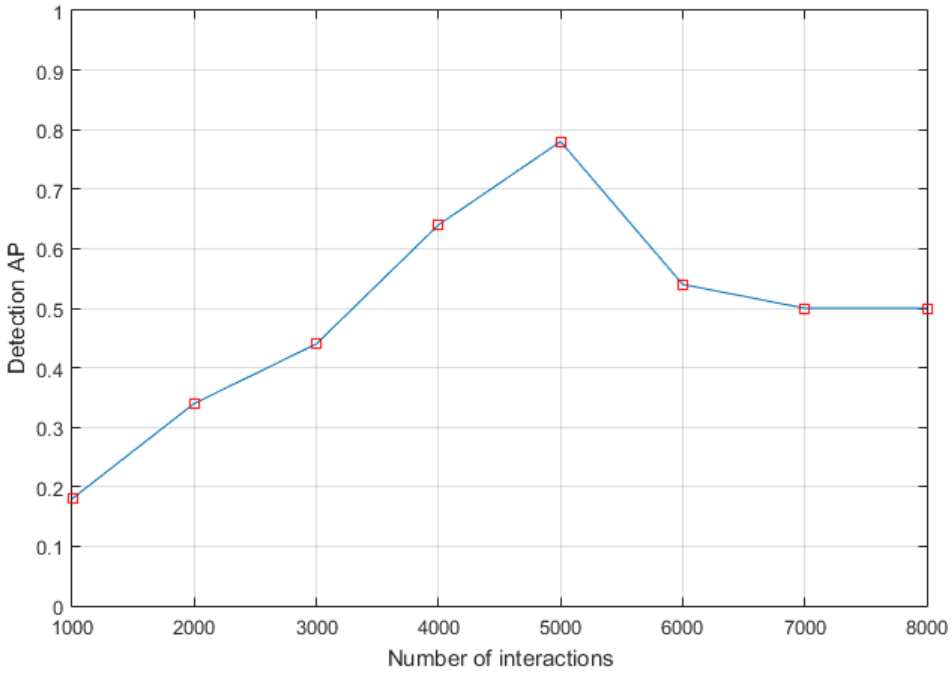

Figure 7: Accuracy (AP) vs number of iterations

a) b) c)

Figure 8: iterations results for: a) 1000 iterations , b) 5000 iterations and c) 8000 iterations

4.2 Implementation

Already within the embedded system, the native programming language for which the system is optimized is Python. The first lines of code focus on calling libraries needed for the implementation process, including OpenCV. Afterwards, the MMX processor is called, which is the one used by the NCS2. Thanks to this, most of the processing load is redirected to the external processor, improving the capacity of the Raspberry. OpenCV makes the extraction of frames easier, in addition of having functions related to: rescaling, extraction of parameters, mapping and function graphing on frames.

The system is able to take the first frame dimensions and perform a resizing. The work scaling was defined as 960x450, this being the most balanced scale between model recognition and work area; a smaller area means less work processing. This standard of image scaling is based on the Heredia and Barros G. 2019 documentation.

The frames are entered one by one into the recognition system, where the YOLOv4 tiny system is in charge of identifying objects by comparing the image with the provided information from Darknet53. at the end of the execution from the analysis process, YOLOv4 returns a group of vectors that has the following structure for each detected object:

$$[L, x, y, w, h, c] \tag{2}$$

Where L corresponds to the class that has been detected in other words, the corresponding label. The values of x and y, correspond to the centroid coordinates of the detected object. The values of w and h, correspond to the height and width of bounding box that contains the object (these values are measured having the centroid as reference). The value c, corresponds to the accuracy determined by IoU.

The OpenCV library contains functions that allow the bounding boxes to be drawn by the use of these parameters. By discretion of the user, the respective labels and the accuracy value can also be printed.

In the case of the CTA algorithm for the tracking part, the operation is simple. Each object is assigned a numbered centroid and is memorized as a vector. In the next frame, the

calculation of the Euclidean distance between each vector centroid and the new centroids found in the subsequent frame is performed by the system. If it is true that any of the new centroids is located within the area of one of the memorized centroids, it is said to be the same. If any centroid does not find another one to be assigned, it is assumed that the object is no longer in the image, leaving the used number free.

In the case of the Kalman filter and the Hungarian algorithm, the operation is more complex. The Python implementation is based on the Laaraiedh n.d. document. On the other hand, OpenCV provides functions related to the Kalman filter (*KalmanFilter ()*).

Using the results released by this function together with the calculation of other estimators such as the IoU or the same Euclidean distance, the probability of a relationship between an object in two different frames is calculated. Videos from the YouTube platform were used for the performance tests. The videos with their original names can be found in Table No1. In addition, the time ranges used in each video are also included.

Table 1: Videos used from Youtube

	Videos (Youtube names)	**Time Used**
Video 1	M6 Motorway Traffic	0:00 - 3:00
Video 2	Relaxing highway traffic	0:00 - 4:00
Video 3	M25 Motorway Traffic UK HD - rush hour - British Highway traffic, congestion	2:30 - 3:30
Video 4	West Side Highway Traffic HD Close up in Manhattan, New York	0:30 - 1:30

In the videos that were previously mentioned and, in the time-intervals described, the vehicles have been manually counted to know the real count value. Subsequently, the Python code was implemented with each one of the videos using both tracking algorithms. Due to the presence of accounting issues, the accuracy value was not calculated so a value of percentage error was obtained. The results can be seen in Table No2.

Table 2: Comparison between CTA and Kalman Filter

Videos	Vehicles Number	CTA			Kalman Filter		
		FPS	Vehicles Detected	Error	FPS	Vehicles Detected	Accuracy
Video 1	225	7.63	230	0.02	5.62	224	0.004
Video 2	128	6.89	121	0.054	5.77	126	0.015
Video 3	63	6.73	68	0.078	5.57	63	0
Video 4	119	6.33	117	0.016	5.14	119	0

5 CONCLUSIONS AND RECOMMENDATIONS

5.1 Conclusions

The architectures, neural networks and algorithms used were chosen with the processing speed of the embedded system in mind. The project, as it is focused on a low-cost detection of vehicles in real-time, has some performance issues mainly because the Raspberry Pi minicomputers do not have a powerful GPU.

Working in parallel with an NCS2 to redirect work processing is a viable option. However, when the system has to perform more actions, it still lacks of enough processing speed. This lack of processing power manifests in the hardware; the embedded system overheats despite of having heat sinks. This means that for an application like this to have better performance, an improved cooling system is needed.

Ideally, the Kalman filter and the Hungarian algorithm should be used for car tracking, because they generate greater accuracy in car counting under different angle views. The CTA is a tracking algorithm designed for embedded systems as it requires less processing power. Despite this, the FPS difference between the two algorithms is just 2 frames, which means that both end up not reaching the desired expectations.

5.2 Recommendations

It is recommended to relocate the project to another embedded system that has a native GPU that works on vehicle counting and detection. The objective of the previous statement is to achieve a FPS rate higher than 20, with which more accurate results can be obtained. The

change to the NVIDIA Jetson minicomputers as a plausible alternative is recommended. At the same time, it is also recommended to work with a suitable cooling system, because overheating reduces equipment lifetime.

It is recommended to use a better data-set for the viewing angle that the real-time system will have. The used data-set had a large number of images taken from the side, back and front, but the angle view used was chopped, which caused detection errors.

5.3 Future work

For future work it is expected to successfully relocate the project into a more powerful embedded system that still meets the low-price standards. Having perfected and optimized the vehicle detection and counting, it is intended to use a system that allows traffic lights to be controlled by the use of the embedded system and the camera of the system itself.

The control of traffic lights must be done throughout the study of discrete differential equations. These are the ones that describe systems in motion in a discrete way (equivalent to differential equations for continuous time). A discrete differential equation system with at least one equation per intersection is required. The equations must associate the number of stopped vehicles on one road and the vehicle rate that pass on the other (equivalent to the derivative in continuous time).

References

Alghyaline, S., El-Omari, N. K. T., & Al-Khatib, R. M. (2019). *Rt-vc: An efficient real-time vehicle counting approach*. https://www.researchgate.net/publication/332631701_RT-VC_AN_EFFICIENT_REAL-TIME_VEHICLE_COUNTING_APPROACH

Ament, P. (2015). *Traffic light. the great idea finder*. http://www.ideafinder.com/history/inventions/trafficlight.htm

Bochkovskiy, A., Wang, C.-Y., & Liao, H.-Y. M. (2020). Yolov4: Optimal speed and accuracy of object detection. https://doi.org/https://arxiv.org/abs/2004.10934

Chen, Z., Ellis, T., & Velastin, S. A. (2012). *Vehicle detection, tracking and classification in urban traffic.* https://doi.org/10.1109/itsc.2012.6338852

Espinoza, F., Barros G., G., & Barros, M. J. (2017). *Computer vision classifier and platform for automatic counting: More than cars.* https://doi.org/10.1109/etcm.2017.8247454

Google. (2020a). *Colaboratory*. https://colab.research.google.com/notebooks/intro.ipynb

Google. (2020b). *Open images dataset*. https://opensource.google/projects/open-images-dataset

Heredia, C., & Barros G., J. (2019). Svm and ssd for classification of mobility actors on the edge. https://doi.org/10.1007/978-3-030-36211-9_1

Intel. (2020). *Intel neural compute stick 2*. https://ark.intel.com/content/www/us/en/ark/products/140109/intel-neural-compute-stick-2.html

Laaraiedh, M. (n.d.). Implementation of kalman filter with python language. https://arxiv.org/ftp/arxiv/papers/1204/1204.0375.pdf

NEMA. (2020). *The association of electrical equipment and medical imaginf manufacters*. https://www.nema.org/

OpenCV-team. (2020). *Opencv*. https://opencv.org/about/

Raspberry Pi Foundation. (n.d.). *Raspberry pi 3 model b+*. https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

Roboflow, Inc. (2020). *Roboflow app*. https://app.roboflow.com/login

Rosebrock, A. (2016). *Intersection over union (iou) for object detection*. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/

Świeżewski, J. (2020). *What is yolo object detection?* https://appsilon.com/object-detection-yolo-algorithm/