Article

Deep Learning-Based Garbage Bags and Potholes Detection Model Using Raspberry Pi

Juan Felipe Palacios †, Santiago Vitery † and Luis Felipe Giraldo *

Department of Electric and Electronic Engineering, Universidad de Los Andes, Bogotá D.C., Colombia; jf.palacios@uniandes.edu.co (J.F.P.); s.vitery@uniandes.edu.co (S.V.)

* Correspondence: lf.giraldo404@uniandes.edu.co

† These authors contributed equally to this work.

Abstract: Given the current process for garbage collection and road maintenance, due to gaps in the pavement, in addition to the non-compliance of citizens with the norms established by entities for these services; city streets are becoming dirtier and less passable, affecting transportation. The main problem is that the entities in charge of these tasks do not have daily updated information. In the proposed article, a model for the detection of garbage bags and holes based on artificial vision and deep learning is proposed, which collects geographic information from garbage bags and holes present in the streets of a city. From this information a heat map is generated, which can be provided to the companies in charge of cleaning and maintaining the streets, and contribute to the progress towards a smart city. The behavior of the model has been explored and tested using a Raspberry Pi in real time, and the model has been shown to be fully functional and efficient. The overall performance of the proposed model has been achieved in terms of accuracy, precision and F1-score as 83%, 91% and 82% respectively.

Keywords: Smart Cities; Convolutional Neural Network; Raspberry Pi; Garbage bags; Potholes; Deep Learning.

1. Introduction

With the recent increase in the urban population, technology has been one of the fundamental pillars to improve the quality of life of citizens. And this not only helps at home with electronic devices and elements that help in people's day-to-day lives, it is also present as a much larger entity in cities. However, the increase in population also brings problems to cities, such as higher unemployment rates, deforestation and reduction of green areas, increased energy consumption and pollution [1]. A smart city is a city that uses technology and innovation to promote sustainable development in a more efficient way. Concepts such as the Internet of Things, Big Data, Industry 5.0 ,take part in the formation of a smart city [2]. It is then where technology becomes an important factor in finding solutions for problems in cities.

One of the existing problems in large cities is the poor condition of the streets, which limits the traffic of people and vehicles. From sidewalks with excess garbage bags, road signs in poor condition, damaged utility poles; to roads with potholes, worn areas, etc. This makes the mobilization of citizens impossible [3]. To counteract this problem, technological solutions have been proposed to locate these objects so government entities act in the shortest possible time to solve it. These solutions focus mostly on computer visual detection. However, not many systems are known to solve these problems [4]. Very little has been done to develop automatic systems for recognizing objects that obstruct the streets. On the other hand, traditional methods for street cleaning and maintenance take a long time [5]. Although some of these tasks are carried out on a daily basis, an entire city cannot be covered in just one day. In addition, the method is to make a trajectory through different parts of the city. If something happens in the area where the trajectory has already been carried out, the garbage, debris and damage remain immobile until the next day of that same trajectory. This means that the streets are accumulated with

obstacles restricting the passage of people. Therefore, the use of technology to solve these types of problems should be supported. However, no databases have been found with both the objects to be studied. Then, it is necessary to start creating these resources so that the implementation of computer visual detection is a tool to work hand in hand with the methods already used.

In this article, an implementation of a deep convolutional neural network model is proposed to detect and classify garbage bags and holes in city streets. In addition, the model will be used to identify the geographical position of those garbage bags and holes in the city. This, in order to generate a heat map of the city, where you can see the streets and areas where there are more of these two variables. This model can potentially improve the pavement repair and waste collection system in smart cities. To design such a model, a database with images of real scenarios was first created to train the deep learning model. The images were taken in two cities in Colombia using a Raspberry Pi camera module. Then, the data was labeled and the model of a convolutional neural network was trained. Through experimental results, it was found that this type of model is efficient and fully functional for the problem studied. Next, the neural network model was implemented on a Raspberry Pi and the computational performance of the identification system was evaluated in real time [6]. After the data collection, a geographical heat map of the cities was made, identifying the areas with garbage bags and holes in a followed trajectory. Finally, the results obtained are concluded and discussed, in addition to giving possible future work that can be achieved in the project.

2. Materials and Methods

To generate the object detection model, images of the streets of two Colombian cities were collected using a Raspberry Pi camera module. For data collection and image processing, a Python script was coded in version 3.7. From the data acquisition, the images in which garbage bags and / or potholes were identified were selected and separated. Subsequently, the images were labeled to generate the dataset that was used to train the convolutional neural network. The model was then loaded onto the Raspberry Pi and a second data collection was performed, but this time only the time when a garbage bag or pothole was detected is taken. Afterwards, the time of the trajectory was related to the time in which garbage bags and potholes were detected, to identify their geographical position. Finally, with another Python script, the heat maps of the cities were made. A detailed explanation of each of the steps is provided below.

2.1. Prototype

To acquire the images, a 5 megapixel Raspberry version 1.3 camera module was used, connected to a 64-bit Raspberry Pi model 4 with 2Gb of RAM and an ARMv7 processor. The images that were obtained had a dimension of 480 x 640 pixels in RGB format. The Raspberry Pi was connected to the car's power outlet via a USB adapter. In order to get best vision angle, the camera was supported behind the central rear view mirror (figure 1) so that it had a frontal and centered view of the street.

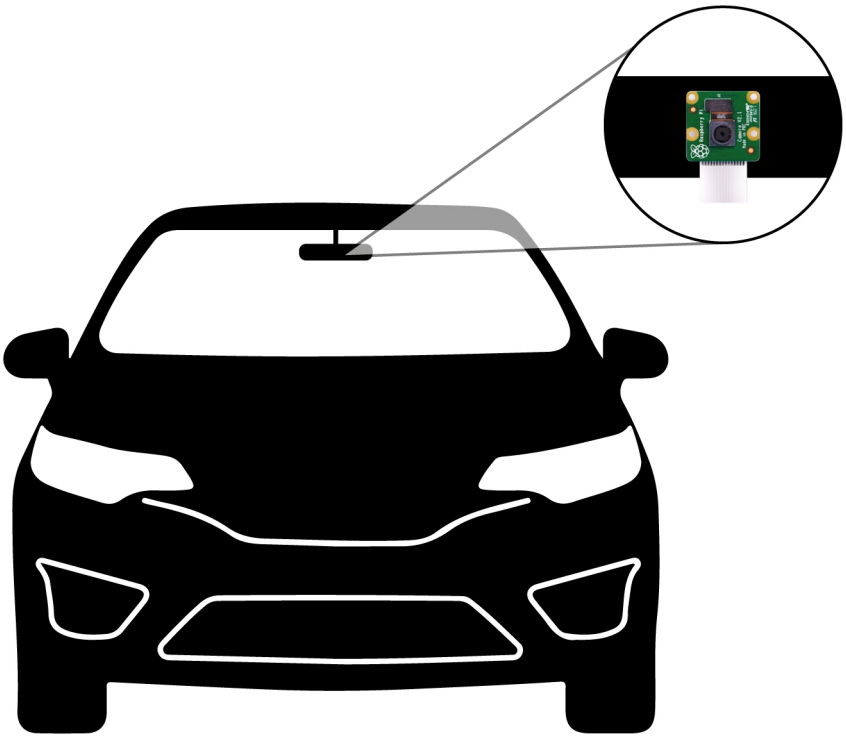

Figure 1. Location of the camera in the car.

The cost of the prototype is approximately $90 USD, which includes the cost of the Raspberry ($60 USD), the camera module ($10 USD) and other items such as USB-C cable, microSD and adapter for the car outlet.

2.2. Database Collection

Images were collected from the streets of the cities of Neiva, Huila and Pasto, Nariño. Data acquisition was done for approximately 2 hours, and around 100,000 images were obtained. These images were saved in the Raspberry microSD and later, they were transferred to a computer for their due processing.

First, all the images were cropped to have a square shape, 480 x 480 pixels in size. Afterwards, images that had garbage bags and / or potholes were selected. From this selection, about 6,000 images were obtained for the training of the convolutional neural network. The images were tagged using the IBM *Cloud Annotations* tool [7]. After labeling the images of the two cities, a total of 6,340 images were obtained with 10,259 garbage bags and 1,419 holes. Because a neural network for detecting objects in images will be retrained, the final amount of images is considered acceptable to obtain a functional and efficient result. Examples of labeled images are shown in the figures 2 and 3.

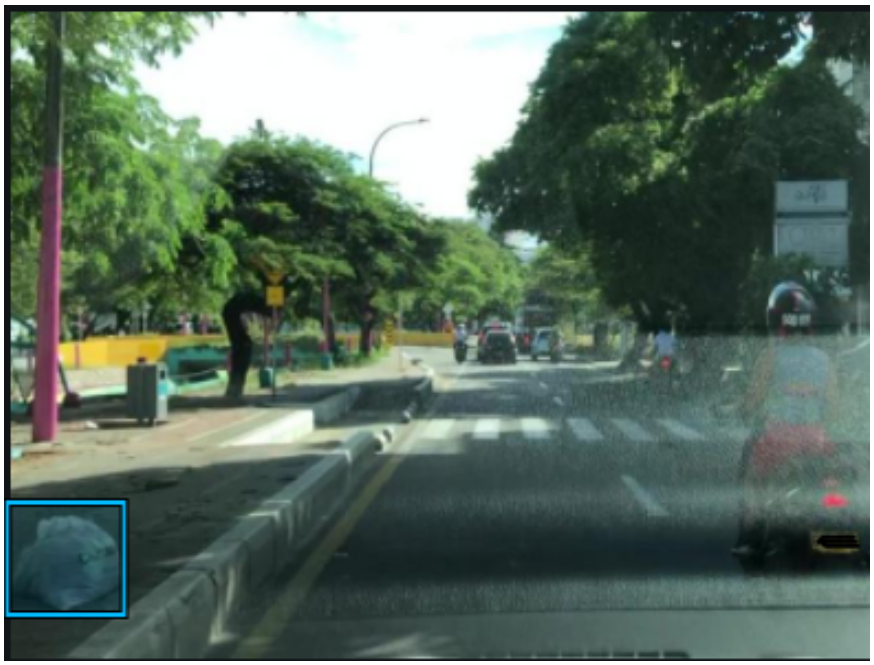

Figure 2. Example of trash bag label. The blue box corresponds to a trash bag.

Figure 3. Example of pothole label. The orange box corresponds to a hole in the street.

2.3. Object detection model

With the database already established and organized, the object detection models were retrained. Two architectures focused on mobile devices were selected, Yolo v4-tiny [8] and EfficientDet-Lite0 [9]. The two models use an image in RGB format as input and return the coordinates of the boxes where a garbage bag or pothole is detected, with the probability of detection of each one.

To perform fine-tuning, the dataset was divided into 80% for training and 20% for validation. The two models were trained until overfitting was evident. This was done by changing the number of training steps and checking the accuracy in the validation set.

Finally, the model was selected taking into account the Accuracy, Precision and F1-Score metrics.

2.4. Implementation

In this part, the object detection model was implemented on the Raspberry Pi. The Raspberry is expected to process the data in real time locally, and deliver only the result of the inferences made. This is so that each iteration only saves the number of detections of each class in a plain text file, without saving the associated image. In the figure 4 a flow diagram with the proposed algorithm for object detection is presented.

It is important to note that a version of Tensorflow compatible with the Raspberry Pi processor must be installed. In this case, the version installed was the one corresponding to the armv7l processors.

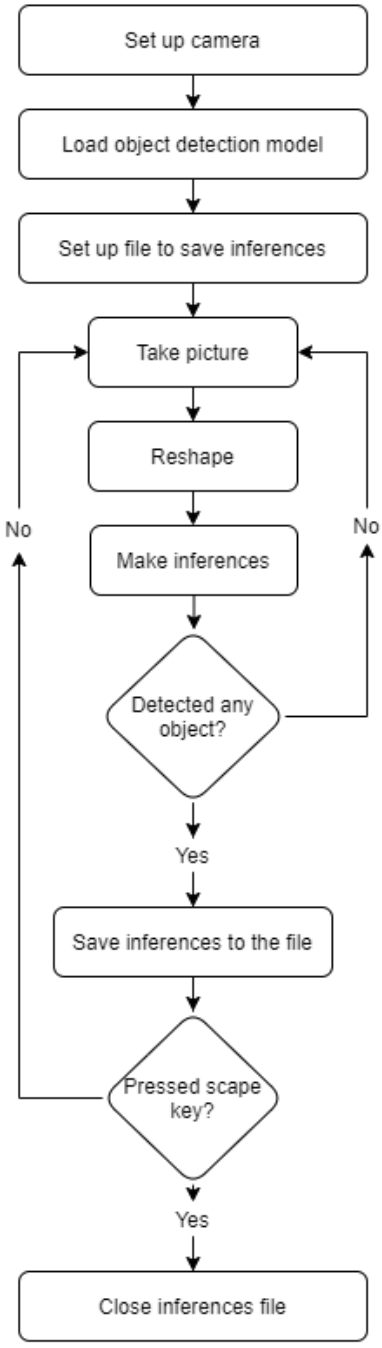

Figure 4. Flowchart of the implemented algorithm.

3. Results

In this section, the experimental results of all phases of the project are presented.

3.1. Object detection model

The models were trained offline using the computational resources provided by Google-Colab. This is because this system provides its users with the use of GPUs to accelerate the training process of neural networks. If this same process were done on the Raspberry, the training time would be drastically increased, due to the computational qualities of the Raspberry. YOLO's model training was through Darknet, while EfficienDet's was through TensorFlow. From the trained models, the Accuracy, Prescision and F1-Score metrics were compared, and thus choose the best one for the solution of this problem. The metrics of the validation set obtained after training the models are presented in the table 1.

Table 1. Accuracy, Precision and F1-Score of the trained models.

Model	Accuracy	Precision	F1-score
YOLOv4	83.78%	91%	0.82
EfficientDet	71.51%	72%	0.69

It can be seen that the metrics of the YOLOv4 model exceed those of the EfficentDet model. The values obtained in the validation of the YOLOv4 model were quite good and are comparable with a project that only detected garbage bags [10], so this model was considered adequate and efficient for solving the problem, and implement it on the Raspberry Pi. On the other hand, Darknet requires an integrated GPU to function, so the Raspberry cannot run the trained model. Given this, the model was exported to a version of TensorFlow so that the Raspberry can run it. To do this, TensorFlow version 2.3.0 was installed on the Raspberry.

Before implementing the algorithm on the Raspberry, the model was qualitatively tested using some images taken with a mobile camera. The figures 5 and 6 show examples of detections of both classes.

Figure 5. Example of detection. The garbage bags detections are shown in red boxes.

Figure 6. Example of detection. The potholes detections are shown in green boxes.

There are some objects that are detected depending on the distance to the camera. For example, the figures 7 and 8 show the same garbage bag on the left side of the street, but the model did not detect it when the object was further away. This example exposes the importance of taking photographs close enough to ensure that the model detects objects.

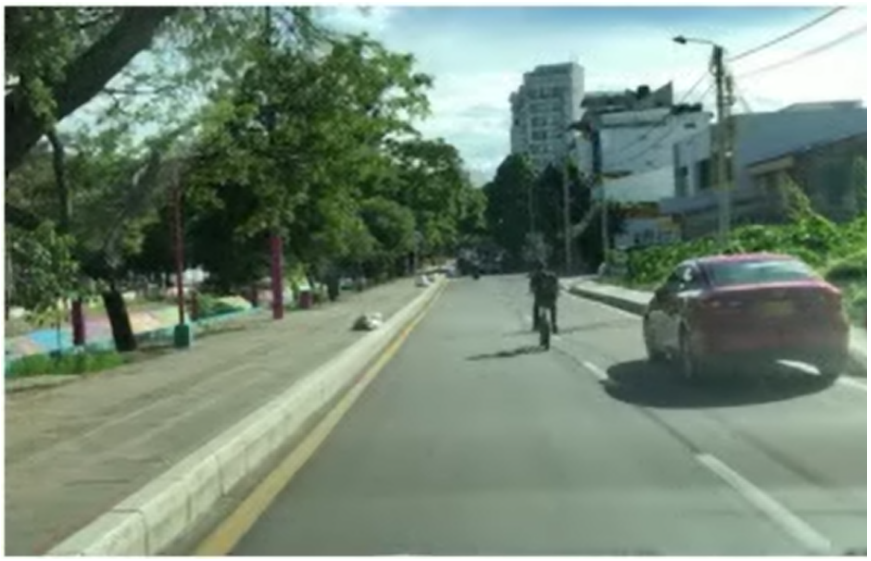

Figure 7. Example of an image without detections.

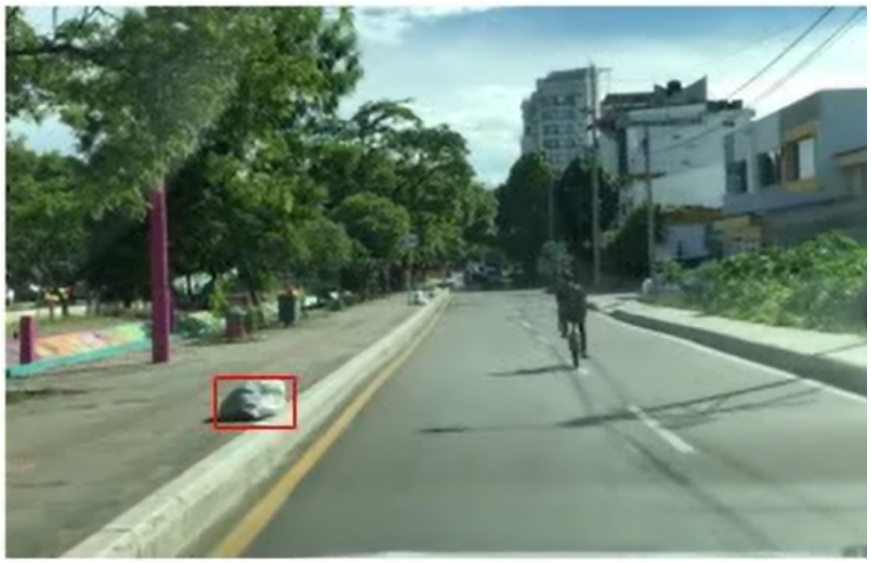

Figure 8. Example of detection. The garbage bags detections are shown in red boxes.

Finally, there are some cases of false positives from both garbage bags and potholes. The figure 9 illustrates three false positives of garbage bags that are actually bollards. The reason could be the similarity of shape, size, color and light contrast, which makes them look like garbage bags. On the other hand, the figure 10 shows a false positive of a garbage bag on the left side of the image and a false positive of a pothole on the right side. Again, the reason seems to be the similarity of shape and color for the two cases; resulting in false positives. The model detects the underside of utility poles as a garbage bag, and shadows from trees, bushes, or people, as a pothole. Some objects detected as false positives are particular objects that appear only in one of the cities, so this problem might be corrected in a different way in each city. This situation could show a lack of images containing objects similar to the problem classes in the database. On the other hand, these detections were made with a threshold of 0 to 100 %, that is, the model was being tested to detect everything that it considered garbage bags and potholes. From this, it was decided to reduce the threshold so that the model would only detect if there is a probability between 50 and 100 of estimation, to reduce the cases of false positives.

Figure 9. Example of false positives. The garbage bags detections are shown in red boxes, the potholes detections are shown in green boxes.

Figure 10. Example of false positives. The garbage bags detections are shown in red boxes, the potholes detections are shown in green boxes.

3.2. Implementation of the model on the Raspberry Pi

As mentioned before and expressed in the flow diagram, in each iteration of the implemented algorithm, the image acquisition is performed, the image is processed through the object detection model, and finally, the inferences are saved with the number detections per class in a plain text file. During the data acquisition with the model implemented in the Raspberry, the processing times of each part of the algorithm were taken. These values are presented in the figures 11 and 12.

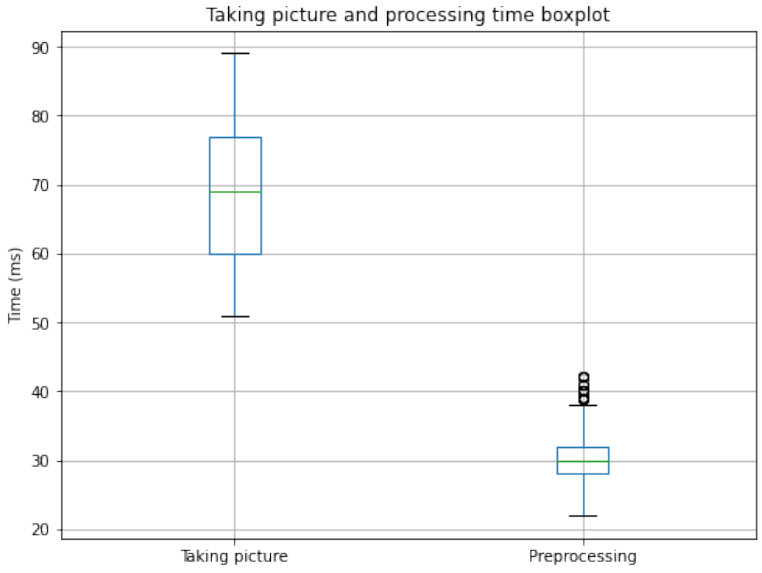

Figure 11. Time Boxplot of taking picture and preprocessing time.

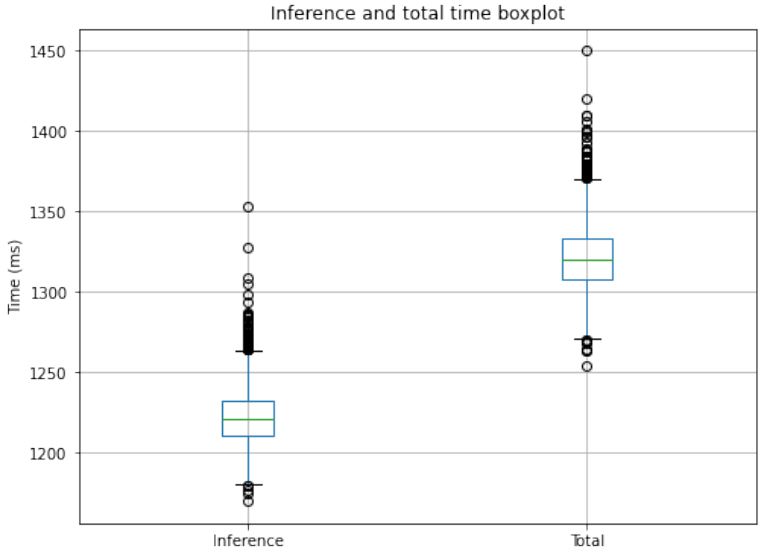

Figure 12. Time Boxplot of inference and total time.

Table 2 shows the mean value of each part of the iteration, in addition to the CPU and RAM usage of the Raspberry during detection.

Table 2. Mean computational costs of the prototype.

Use of CPU	89,5%
Use of RAM	556 Mb
Taking picture time	69ms
Preprocessing time	30ms
Inference time	1220ms
Total time	1320ms

From the previous figures, it can be seen that the vast majority of the time of each iteration is used in detecting objects, which lasts more than one second. Taking into account that the data acquisition is carried out in a car and that the speed can vary, according to the road conditions and the driver, it is necessary to establish a relationship between the detection process time and the distance between each pair of detections. The relationship can be found by following the next equation:

$$d = v * t \tag{1}$$

where d is the distance between detections, v is the speed of the car and t is the average time of each iteration. Taking the distance between detections in meters, the speed of the car in kilometers per hour and replacing the time with $1.32s$ we obtain the following function:

$$d = \frac{1.32}{3.6} * v \tag{2}$$

The figure 13 shows the graph of the previous function for the speed range between $5km/h$ and $40km/h$

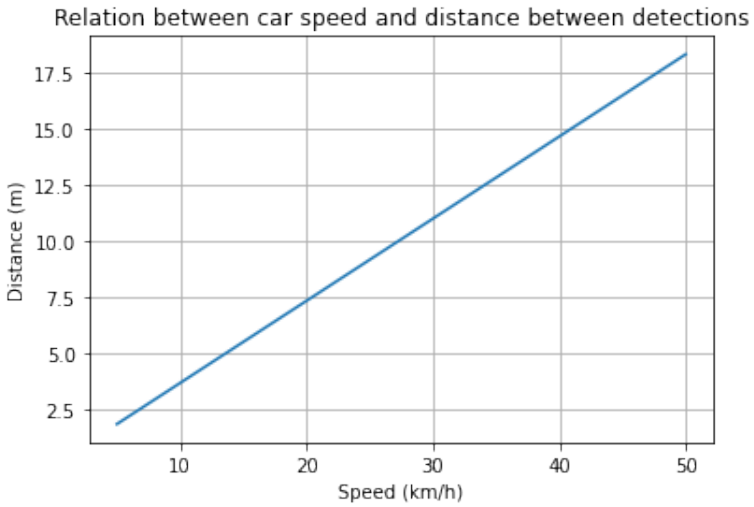

Figure 13. Distance between detections as a function of the car speed.

From the graph it can be concluded that, when driving at a moderate speed, the camera can capture all the objects present on the path followed. For example, if the prototype moves with a speed of 30 km/h, which is the speed limit at residential zones, the space between detection will be about 11 m. This is a sufficient distance to capture all objects on the path followed. Therefore, the processing time is small enough to implement the prototype in normal situations.

3.3. Heat map

Finally, from the plain text file, generated by the data acquisition script with the model; the specific time at which a detection was made and the class classification is obtained, that is, whether a garbage bag or a pothole was detected. As the Raspberry does not have GPS, in order to relate the geographical position of the detections, while the data was acquired, the trajectory was also recorded with a cell phone. In this way, the time was related to the time elapsed in the video, and from the video the geographical position at that moment was identified in Google Maps. This was done manually and a file was generated with the geographical coordinates of each detection in order to perform the heat map.

The data of the detections was used to compute a probability distribution using a python script. It allows to represent the density of the detections with density of colors. The maps were plotted using the library cartopy. The resulting heat maps for the cities of Neiva and Pasto are presented in figures 14 and 15 respectively.

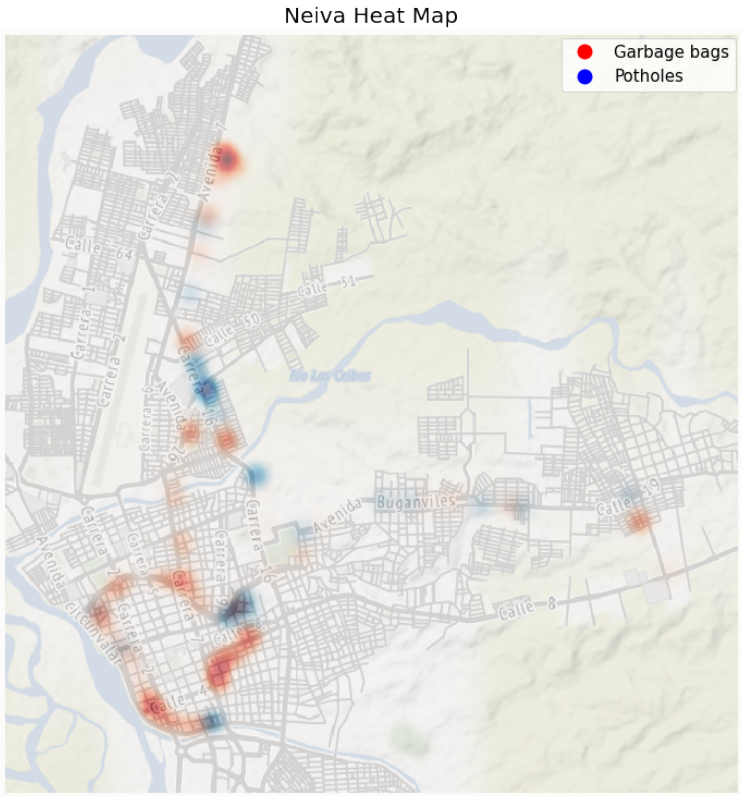

Figure 14. Heat map of the city of Neiva identifying the areas with garbage bags and potholes.

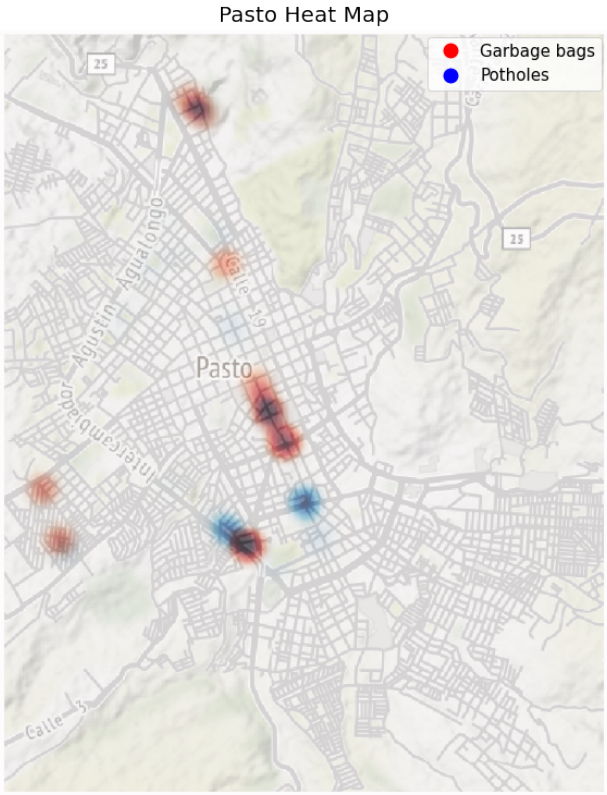

Figure 15. Heat map of the city of Pasto identifying the areas with garbage bags and potholes.

From these figures, it can be concluded that the model worked correctly. Both the density behavior for the garbage bags and for the potholes can be observed. The maps identify the streets and areas of the cities that have a large number of garbage bags and potholes, which makes it possible for these maps to be used by companies in charge of street maintenance and cleaning, to act quickly and pick up the trash bags and restore the pavement.

The heat maps were coded so that it is possible to visualize exactly which specific streets are affected, this was done by reducing the size of each density. Finally, these maps only show an example of what can be achieved in the future, since only a specific route was made and data was not taken from all the streets of the cities.

4. Conclusions

A model was designed for the identification of garbage bags and potholes in the streets of a city. The process resulted from a database collected in two different Colombian cities with a low-cost prototype, a deep learning model of which the validation was successful in a real situation, and a proposal of heat maps to help public entities to contribute to the problem of street cleaning and maintenance. The performance of the convolutional neural network turned out to be effective for real-time problem solving. The model was tested in two different cities, with positive results in both. However, from the different conditions of the streets in both cities, it is concluded that to improve the model those unique aspects of each city must be taken into account, so that the viability of the model increases in a specific city and reduce the amount of false positives. But for now, the implementation and generalization of the model is applicable to any city giving satisfactory results. On the other hand, the Raspberry Pi was shown to be a suitable tool, in terms of processing time, to implement a object detection device. In addition to being a device with good computational capacity at a low cost and small size. Finally, the objective of the project was achieved with excellent results, providing a model that can positively affect smart cities to reduce problems with transportation and the quality of the streets.

Future work includes reducing the complexity of the model to decrease processing times in the Raspberry, and therefore the distance between each detection. The lighter models also allow the prototype to be implemented in simpler and cheaper processors, which would be an advantage in creating a lower cost prototype for commercialization. On the other hand, it would be useful to add more classes to the network, such as damaged road signs or graffiti. Since the model in this project is trained by supervised learning, increasing the database will improve network performance, avoiding overfitting. In addition, to make the model more automatic, a GPS module should be incorporated into the Raspberry so that the geographic location data is acquired in the data collection and not in a post-processing of data. Next, it would be feasible to put several prototypes at the same time to increase the data collected in real time, and that the area reached by the model in the city is greater. Each prototype would be connected to a central node that receives all the information, to carry out a general processing of the acquired data. Or, all the data can be sent to a master computer by means of some wide area network modulation technique, such as LoRa. Finally, the detections could also be acquired as a time series. This could help reduce false positives by filtering detection spikes and only consider detections that remain constant for a spam of time.

References

1. S. Uniyal, “Human overpopulation: impact on environment,” Megacities and Rapid Urbanization: Breakthroughs in Research and Practice, IGI Global, 2020, pp. 20-30.
2. S. P. Mohanty, U. Choppali, and E. Kougianos, “Everything you wanted to know about smart cities: The Internet of things is the backbone,” IEEE Consumer Electronics Magazine, 2016, vol. 5, pp. 60–70.
3. M. Chong, “Dynamic capabilities of a smart city: An innovative approach to discovering urban problems and solutions,” Government Information Quarterly, 2018, vol. 35, no. 4, pp. 682-692.

4. S. Hossain, "Autonomous trash collector based on object detection using deep neural network," En TENCON 2019-2019 IEEE Region 10 Conference (TENCON). IEEE, 2019, pp. 1406-1410.
5. E. Betanzo-Quezada, "Evaluación de rutas de recolección de residuos sólidos urbanos con apoyo de dispositivos de rastreo satelital: análisis e implicaciones," Revista internacional de contaminación ambiental, 2016, vol. 32, no 3, pp. 323-337.
6. A. Gunnarsson, "Real time object detection on a Raspberry Pi," 2019.
7. IBM, "Cloud Annotations," 2008, Avaliable at https://cloud.annotations.ai/.
8. A. Bochkovskiy, C.Y. Wang, H.Y.M. Liao, "YOLOv4: Optimal Speed and Accuracy of Object Detection," 2020, Avaliable at https://arxiv.org/abs/2004.10934
9. M. Tan, R. Pang,Q.V. Le, "EfficientDet: Scalable and Efficient Object Detection," 2020, Avaliable at https://arxiv.org/abs/1911.09070
10. F. Forero, "Detección de Bolsa de Desechos sobre Andenes de Bogotá utilizando Redes Neuronales Convolucionales," 2018, Departamento de ingeniería eléctrica y electrónica, Universidad de los Andes.
11. J.F. Gaviria, A. Escalante, J.C. Castiblanco, N. Vergara, V. Parra, J.D. Zambrano, A.F. Zambrano, L.F. Giraldo, "Deep Learning-Based Portable Device for Audio Distress Signal Recognition in Urban Areas," Applied Sciences, 2020, vol. 10, no 21, pp. 7448.
12. D.K. Dewangan, S.P. Sahu, "Deep learning-based speed bump detection model for intelligent vehicle system using raspberry Pi," IEEE Sensors Journal, 2020, vol. 21, no 3, pp. 3570-3578.