applied sciences

MDPI

Article

Analysis of Statistical and Artificial Intelligence Algorithms for Real-Time Speed Estimation Based on Vehicle Detection with YOLO

Héctor Rodríguez-Rangel [1,†], Luis Alberto Morales-Rosales [2,*,†], Rafael Imperial-Rojo [1,†], Mario Alberto Roman-Garay [1,†], Gloria Ekaterine Peralta-Peñuñuri [1,†] and Mariana Lobato-Báez [3,†]

1 Technological Institute of Culiacan, Culiacan 80014, Sinaloa, Mexico; hector.rr@culiacan.tecnm.mx (H.R.-R.); rafael_imperial@itculiacan.edu.mx (R.I.-R.); mario_roman@itculiacan.edu.mx (M.A.R.-G.); gloria.pp@culiacan.tecnm.mx (G.E.P.-P.)
2 Faculty of Civil Engineering, Conacyt-Universidad Michoacana de San Nicolás de Hidalgo, Morelia 58004, Michoacán, Mexico
3 Higher Technological Institute of Libres, Libres 89930, Puebla, Mexico; mariana.lobato@upaep.edu.mx
* Correspondence: lamorales@conacyt.mx
† These authors contributed equally to this work.

Abstract: Automobiles have increased urban mobility, but traffic accidents have also increased. Therefore, road safety is a significant concern involving academics and government. Transit studies are the main supply for studying road accidents, congestion, and flow traffic, allowing the understanding of traffic flow. They require special equipment (sensors) to measure the car's speed. With technological advances, artificial intelligence, and videos, it is possible to estimate the speed in real-time without modifying the installed urban infrastructure. We need to employ public databases that provide reliable monocular videos to generate automated traffic studies. The problem of speed estimation with a monocular camera involves synchronizing data recording, tracking, and detecting the vehicles over the road considering the lanes and distance between cars. Usually, a set of constraints are considered, such as camera calibration, flat roads, including methods based on the homography and augmented intrusion lines, patterns or regions, or prior knowledge about the actual dimensions of some of the objects. In this paper, we present a system that generates a dataset from videos recorded from a highway—obtaining 532 samples; we separated the vehicle's detection by lane, estimating its speed. We use this data set to compare five different statistical methods and three machine learning methods to evaluate their accuracy in estimating the cars' speed in real-time. Our vehicle estimation requires a feature extraction process using YOLOv3 and Kalman filter to detect and track vehicles. The Linear Regression Model (LRM) yielded the best results obtaining a Mean Absolute Error (MAE) of 1.694 km/h for the center lane and 0.956 km/h for the last lane. The results were compared with several state-of-the-art works, having competitive performance. Hence, LRM is fast estimating speed in real time and does not require high computational resources allowing a future hardware implementation.

Keywords: statistical regression; kalman filter; vehicle tracking; vehicle speed estimation; YOLO

Citation: Rodríguez-Rangel, H.; Morales-Rosales, L.A.; Imperial-Rojo, R.; Roman-Garay, M.A.; Peralta-Peñuñuri, G.E.; Lobato-Báez, M. Analysis of Statistical and Artificial Intelligence Algorithms for Real-Time Speed Estimation Based on Vehicle Detection with YOLO. *Appl. Sci.* **2022**, *12*, 2907. https://doi.org/10.3390/app12062907

Academic Editors: Paweł Droździel, Radovan Madleňák, Saugirdas Pukalskas, Drago Sever and Marcin Ślęzak

Received: 31 January 2022
Accepted: 28 February 2022
Published: 11 March 2022

1. Introduction

The use of automobiles is essential in our daily lives. According to the organization Association for Safe International Road Travel (ASIRT), more than 1.3 million people die each year in traffic accidents. In addition, between 20 and 50 million people are injured, or disabled [1].

In the event of an accident, the most significant responsibility falls on the driver of the car [2]. Among the main factors that cause traffic accidents are speeding, distracted driving, obstacles on the road, poor signaling, state of the road infrastructure, and lighting.

According to data from the National Institute of Statistics and Geographic Information in Mexico (INEGI, 2011), between 1997 and 2009, accidents in the region increased by 72.7% in urban and rural areas [3].

An essential element when reviewing the cause of road accidents is speeding. For this reason, studies of vehicular traffic focus on reviewing the causes of this element in a road section. These studies require an effective speed monitoring system. In addition, control systems have been developed to assist the driver when driving on the streets, known as ADAS (Advanced Driver Assistance Systems).

Currently, a traffic study requires specialized equipment to generate the necessary data to analyze the situation and obtain possible solutions. Some of these devices are speed radars which use radio waves to calculate the time it takes for the wave to travel from the radar to the vehicle and back to the radar. While continuously using these devices is not feasible due to their high cost. Nevertheless, cities currently have a video surveillance system. It is possible to use this existing equipment in the town to determine various characteristics of vehicular flow such as speed, accidents, and others. Then, we can provide the traffic estimation to end-users, state security departments, planning departments, among others [4].

Intelligent traffic systems are divided into three types, depending on how they perform their vehicle detection and classification (i.e., Roadway-based, Over Roadway-based, Side Roadway-based). Roadway-based is an intrusive classification, as they need to be installed on the road. Leading to limit vehicular traffic during the period of installation of the devices. As example of this kind of classification are loop detectors [5], vibration sensors [6] and magnetic sensors [7]. Over Roadway-based, unlike previous devices, are installed in separate infrastructures, so the road is not modified. Among these devices are cameras [8], infrared sensors, ultrasonic sensors, satellites, and UAVs [9]. Side Roadway-based vehicle classifications do not invade the roadway and are installed on one side. Among them we have magnetic sensors [10], LIDAR [11], Wi-Fi [12], etc.

An important assumption in this work is that number of cameras installed worldwide has been growing in recent years. Estimating the speed limits allows preserving road safety. The most relevant parameter for traffic monitoring is the car's speed because when used with vehicle counting, it allows determining the behavior of motorized mobility in cities. Therefore, accurately estimating vehicle speed is an open problem for Intelligent Transport Systems (ITS). This task has been addressed by using sensors like radars, laser, and fixed infrastructure, but these sensors are affected by weather conditions and noises or occlusions. When cameras are considered to obtain the vehicle speed, we must consider the translation of a 3D plane into a 2D discrete plane. This intrinsic limitation results in a digital representation whose accuracy follows an inverse-square law; its quantity is inversely proportional to the square of the distance from the camera to the vehicle. When a monocular camera is used to obtain video images to estimate the distance of the vehicles, usually a set of constraints are considered, such as flat road, including methods based on the homography and the use of augmented intrusion lines, patterns or regions, or prior knowledge about the actual dimensions of some of the objects (e.g., the license plates or the size of the vehicles) [13].

The problem of carrying out the speed estimation with a monocular camera is that we have to synchronize data recording, tracking, and detecting the vehicles over the road considering the lanes and distance between cars, among other problems. To identify the location of vehicles on each frame, we must first distinguish objects from the background. This procedure is known as image segmentation. It can be accomplished by subtracting each frame from the background image, but it is highly resource-consuming and generates a noncontiguous object [14]. Nevertheless, new developments in object detection, such as YOLO [15], Yolov2 [16] and Yolov3 [17], have been used in several context with good accuracy, such as [18,19]. Yolo considers the frame object detection as a regression problem to spatially separated bounding boxes and associated class probabilities. A single neural network predicts bounding boxes and class probabilities directly from full images in one

evaluation. Since the whole detection pipeline is a single network, it can be optimized end-to-end directly on detection performance. [15]

Other works demonstrated that the use of consecutive [20–22] or nonconsecutive [23–25] images to estimate speed impacts considerably the accuracy. Hence, the problem is How to integrate all available measurements (instantaneous, mean, optimal, etc.) to better estimate the speed without affecting the final accuracy?

An optical approach has been used to determine the velocity of vehicles in movement with a monocular camera, as is the case of drones and autonomous vehicles. For instance, Hann Woei Ho et al. [26] propose an algorithm that estimates the distance and velocity of the car based on optical flow measured from a monocular camera and the knowledge of control inputs. They extended the Kalman filter to state the estimations and used them for landing control. The optical flow refers to the apparent visual motion of objects in a scene relative to an observer, allowing to determine how fast the camera moves and how close it is relative to the things it sees. In contrast, Maduro et al. [14] rectify image sequences captured by uncalibrated cameras; their method automatically estimates two vanishing points using lines from the image plane. The solution requires two known lengths on the ground plane and can be applied to fairly straight highways near the surveillance camera. Once the background image is rectified, it is possible to locate the stripes and boundaries of the highway lanes.

This paper focuses on estimating one moving vehicle's speed from side view video images obtained with an uncalibrated monocular camera. In our case, the problem of detecting the speed is carried out at the level of individual vehicles, detected using YOLOv3 [17], and we split the road into lanes. We remark that we only add parallel lines to limit the zone of the input and exit of the vehicle's detection and do not consider any other assumption such as ground flat nor prior knowledge about the actual dimensions of some of the objects. We summarize the contributions as follows:

- We create a set of samples (Dataset) captured with monocular cameras (cellphone cameras) from the side view of the road. The dataset includes different videos with two cell phones to consider the two operating systems on mobile phones on the market, android and iOs. We configured the cameras of the cellphones with the same properties; nevertheless, the cameras are not calibrated with a reference object.
- We compare five different statistical methods (Linear, Ridge, Lasso, Bayesian Ridge, Elastic Net Regressions) and three machine learning methods (Random Forest, Support Vector Machine Regressions, Artificial Neural Network) to evaluate their precision of estimation of cars' speeds in real time.
- The Linear Regression Model (LRM) yielded the best results obtaining a Mean Absolute Error (MAE) of 1.694 Km/h for the center lane and 0.956 Km/h for the last lane. The results were compared with several state-of-the-art works, having competitive performance. LRM is fast estimating speed in real time and does not require high computational resources allowing a future hardware implementation. Transport engineers could obtain traffic studies in road zones with limited urban infrastructure by using the LMR model inside cellphones.

The rest of the paper is organized as follows: Section 2 State of the Art describes some of the works previously developed. Section 3 Premilinars describes the statistical and machine learning methods. Section 4 Materials and Methods describe the equipment used and the process developed in this work. Section 5 Results mention the results obtained. Finally, Conclusion Section 6 enumerates the conclusions obtained from this work.

2. State of the Art

Today many works have been developed to determine the velocity of objects using different techniques and methods. The following represent the most representative works considered for the realization of this work. In Table 1 we show a comparison of papers that described the type of hardware used to obtain video images, one, two, or stereo vision. We describe the method used to obtain the speed estimation, classified according to physical

formulas, pixels proportion, or machine learning algorithms (e.g., YOLO v2, CNN). We also present the precision of each work, allowing us to observe the main differences and that the speed estimation is still an open problem to solve.

Schoepflin and Dailey in [27] and Anil Rao et al. in [28] used a video camera to identify the road lines, which were used as a reference point to estimate the distance traveled by the vehicles and thus calculate their speed.

Authors such as Kamoji et al. [21] uses a camera and the vehicle centroid as a reference point. On the other hand, Lee in [29] takes the vehicle centroid as a reference, but he uses a drone with two LiDAR systems to detect vehicles as they pass through two points to calculate the speed at which they pass.

Other authors use a video camera to process the images, focusing only on the vehicles' area. The complete process eliminates the entire background leaving only the vehicles [30–32].

Artificial intelligence algorithms were used in [15,22,33–38], ranging from the simplest ones such as multilayer perceptron and convolutional neural networks to networks such as Faster R-CNN, SqueezeDet, PWC-Net, AlexNet, and FlowNet.

The work of Redmon et al. [15] presents YOLO as a neural network able to identify multiple objects in an image with a single inference. This work is not focused on vehicular traffic analysis. However, in our work, we use YOLO v3 to detect a vehicle while estimating the vehicle's speed.

Table 1. State of the art summary. (* Machine learning works).

Author	Hardware	Method	Precision
Fernández et al. [39]	two Cameras	Physical Formula	<3 km/h
Yang et al., 2019 [40]	Stereo Cameras	Physical Formula	−1.6, +1.1 km/h
Yang et al., 2020 [41]	Stereo Cameras	Physical Formula	−0.72, +1.17 km/h
Yang et al., 2021 [42]	Stereo Cameras	Physical Formula	−0.9, +1.06 km/h
Luvizon et al. [43]	Camera	Physical Formula	0.59 km/h
Vakili et al. [23]	Camera	Physical Formula	1.32 km/h
Anil Rao et al. [28]	Camera	Pixels Proportions	3 km/h
Kamoji et al. [21]	Camera	Physical Formula	98% accuracy
Lee et al. [29]	Dron LiDAR	Physical Formula	1.31 km/h
Li et al. [30]	Camera	Physical Formula	2.3 km/h
Kurniawan et al. [31]	Camera	Projective transformation	97.01% No Shadow 83.86% Shadow
Jalalat et al. [32]	Camera	Subpixel Stereo Matching	±6%
Bell et al. [22] *	Camera GNSS IMU	YOLOv2 Faster R-CNN	2.25 km/h
Dong et al. [33] *	Camera	3D CNN	2.71 km/h

Table 1. *Cont.*

Author	Hardware	Method	Precision
Burnett et al. [34]	GPS LiDAR IMU Camera	SqueezeDet	Only Tracking
Kampelmuhler et al. [35] *	Camera	MLP	4.32 km/h
Song et al. [36] *	Camera	PWC-Net	1.728 km/h
Zhang et al. [37] *	Camera	Faster-RCNN CNN AlexNet	6.832 km/h
Loor et al. [38] *	Camera	FlowNet TimeNet SpeedNet CNN	3.6 km/h
Redmon et al. [15] *	Images	CNN	Only Detection

3. Preliminars

Different statistical (i.e., Linear, Ridge, Lasso, Bayesian Ridge, Elastic Net Regressions) and Machine Learning (i.e., Random Forest, Support Vector Machine Regressions, Artificial Neural Network) methods were used to realize this research work. This section will be divided into two parts (Statistical and Machine Learning) to address each method used to develop the work.

3.1. Statistical Methods

A statistical regression model seeks to find the relationship between a known variable x and an unknown variable y and determine the impact that variable y will have with the change in the value of variable x.

3.1.1. Linear Regression

The most common type of regression is linear regression [44]. We can model this type of regression on a straight line that models most of the sample data. Linear regression attempts to find a function representing the relationship between x and y. It predicts the value of y that will be the most important predictor of the relationship and turns out to be accurate for the known value of x. An example of this type of regression is showed in Figure 1.

Equation (1) shows us the general linear regression model, where we can observe the variables x and y, in addition to 2 coefficients a and b.

$$y = ax + b \tag{1}$$

where y is the dependent variable or the variable to be predicted, x is the independent variable or the variable we use to make a prediction. a is the slope of the line; it is known as the coefficient and is a kind of magnitude of change that passes through y when x changes. b is the constant to be determined; it is known as the intercept because when x is equal to 0, then $y = b$.

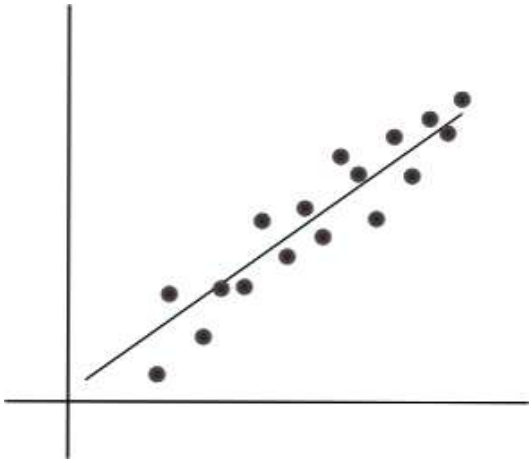

Figure 1. Linear regression example.

3.1.2. Ridge Regression

This type of regression [45] tries to eliminate overfitting by adding a penalty in the ordinary least squares adjustment. The penalty is known as l2; its effect is to reduce the values of the coefficients of the model as much as possible, without them reaching 0.

The hyperparameter α controls the model's penalization. If $\alpha = 0$, the Ridge regression is simply a linear regression. If α is large, all weights end up equal to zero and result in a flat line through the mean of the data (Figure 2).

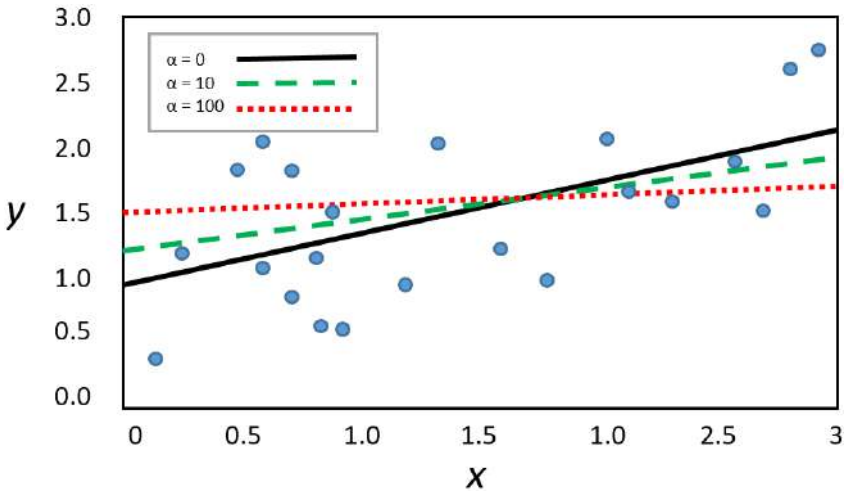

Figure 2. Ridge regression with different values of α.

Equation (3) describes the cost function for the Ridge regression. Equation (2) refers to Mean squared Error (MSE).

$$MSE = (\frac{1}{n}) \sum_{i=0}^{n} (y_i - x_i)^2 \quad (2)$$

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} (\theta_i)^2 \quad (3)$$

3.1.3. Lasso Regression

Like ridge regression, lasso [46] incorporates a penalty to avoid overfitting; it penalizes the sum of the squared coefficients. This penalty is known as l1 and has the effect of forcing the values of the predictor coefficients to tend to zero. Thanks to this effect, it manages to exclude the least relevant predictors.

Equation (4) refers to the Lasso regression function cost.

$$J(\theta) = MSE(\theta) + \alpha \sum_{i=1}^{n} \mid \theta_i \mid \tag{4}$$

3.1.4. Elastic Net

Elastic net [47] uses the l1 and l2 penalties used in ridge and lasso, respectively. The alpha parameter is the one that controls the influence of each of the penalties. Its values are in the range of [0, 1]. When $\alpha = 0$, ridge is applied, when $\alpha = 1$ lasso is applied. Thus, the combination of these 2 penalties generally yields good results.

Equation (5) refers to the Elastic Net regression function cost.

$$J(\theta) = MSE(\theta) + r\alpha \sum_{i=1}^{n} \mid \theta_i \mid + \frac{1-r}{2} \alpha \sum_{i=1}^{n} (\theta_i)^2 \tag{5}$$

3.1.5. Bayesian Ridge Regression

This type of regression [48] is used when the dataset does not have an acceptable quantity, or the data distribution is not optimal. Unlike other regression techniques, Bayesian regression draws its output from a probability distribution, where the output is drawn from a single value of each attribute. The output, y, is generated from a normal distribution (in which the mean and variance are normalized). The objective is to find the posterior distribution of the model parameters.

Equation (6) is the representation of the regression.

$$p(w|\lambda) = N(w|0, \lambda^{-1} I_p) \tag{6}$$

3.1.6. Machine Learning Methods

Machine learning methods for regression use statistical principles to find existing relationships between variables. Unlike statistical models, neural networks are used to generate a model capable of predicting results from previously unknown data through training.

3.1.7. Random Forest Regression

Random Forest regression [49] is a technique in the field of supervised learning that makes use of ensemble learning for regression. This ensemble learning technique consists of averaging the predictions of multiple machine learning algorithms to obtain a more accurate prediction.

Figure 3 shows a diagram of how the Random Forest method works. As can be seen, the method has several individual decision trees, where the prediction is the average of the result of each tree.

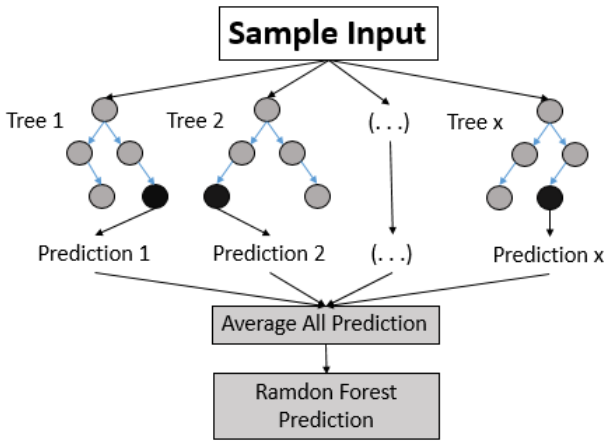

Figure 3. Representation of Random Forest Regression.

3.1.8. Support Vector Machine Regression (SVMR)

Support Vector Machine (SVM) is a set of supervised learning algorithms directly related to classification. An SVM [49] is trained to build the model that predicts the class of a new sample from a set of training data or samples and labeled classes.

SVMR uses the same principle as SVMs incorporating some changes, starting with the output that changes to Real number. As observed in Figure 4, SVMR defines a decision boundary (red lines) and defines a hyperplane within the boundary that fits the maximum number of points.

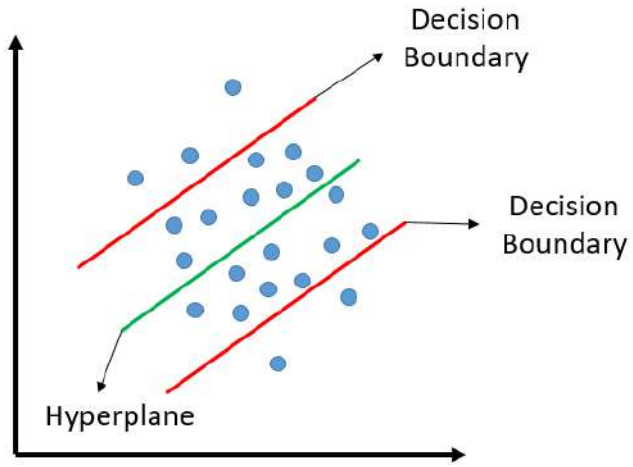

Figure 4. Support Vector Machine Representation.

3.1.9. Artificial Neuronal Network (ANN)

Artificial neural networks [50] are the most widely used method in machine learning to make predictions. Its operation is based on learning features from a set of input data through an iterative process called training. At the end of this process, the result is a model capable of inferring values from data that has not been previously analyzed.

Figure 5 shows a diagram of a multilayer perceptron network with three layers—the input layer with a neuron for each system's input. The hidden layer is formed by an interconnected set of neurons with a defined activation function. Moreover, the output layer corresponds to the prediction.

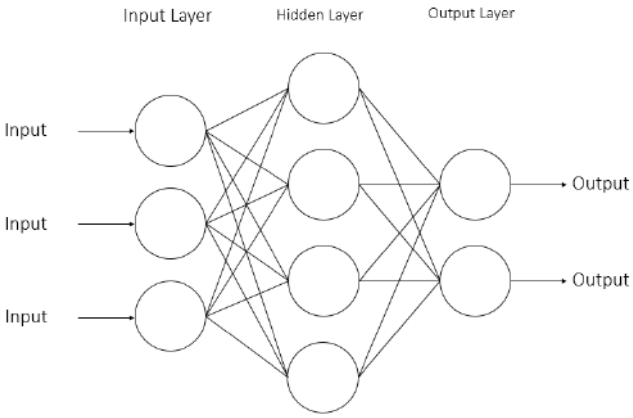

Figure 5. Artificial Neuronal Network Representation.

4. Materials and Methods

This section describes the devices used and the methods developed for the vehicle speed estimation from image sequences. We used a cell phone video camera to acquire the videos, a radar to measure the vehicle's speed, and a server with a GPU to compute the speed estimation.

We divided the methods explanations into two processes. The first one consists of obtaining the samples or dataset generation. The second one uses the generated dataset to estimate the vehicle speed from image sequences. Figure 6 shows the graphic diagram of both processes.

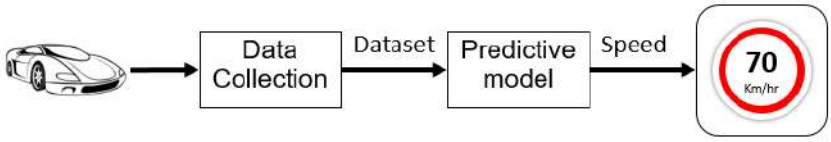

Figure 6. The graphic diagram of the two parts required to develop this work.

4.1. Materials

As described above, the data acquisition required a speed radar and a cell phone camera to capture the image sequences (video). The radar used is a Bushnell radar, which has an accuracy of $+/-$ 1.6 kilometers per hour (km/h). This radar sends a radio wave that bounces off the vehicle and is received in a different frequency. This difference makes it possible to calculate the vehicle speed. We used the cameras of two cellphones, a Xiaomi Redmi Note 7 and an iPhone X, to capture the videos, and we configured both cameras at 60 FPS and a Full HD resolution (1920 $\times$ 1080 pixels). The server is used to compute the video samples; it has the following characteristics: CPU: Intel Core i3-8100, Memory RAM: 32GB DDR4, GPU: GeForce RTX 2070 Super, under Linux Ubuntu 20.04.2 LTS. Figure 7 shows graphically the three components used in this research work.

Figure 7. Components used to estimate the vehicle's speed.

4.2. Methods

As mentioned before, this project aims to generate information about traffic flow from video sequences. We discovered that no dataset engaged the project's requirements when conducting research, which resulted in the need to develop a dataset.

The vehicle speed estimation was divided into two main phases. The first phase corresponds to generating the dataset (Data Collection) and the second phase uses this dataset to develop a *Predictive Model* to estimate the vehicle speed from image sequences.

For the dataset generation, it was necessary to go to a road physically. Once on the road, we acquired the image sequences and measured the vehicle speed along with the radar. The feature extraction process starts with the previous stage samples (image sequences captured). Then, using deep learning techniques results in a CSV (Comma-Separated Values) file with the essential features. Using the previously created dataset, we used and compared statistical regression and machine learning methods to estimate the vehicle's speed.

4.2.1. Data Collection

Deep learning bases its analysis on data, making inferences from the data. So it is of critical importance to have suitable samples for each problem. In this research, we needed to generate our samples, which were obtained in four stages.

The first stage corresponds to *sampling*, which involves going to a place with a constant flow of vehicles to obtain the most significant amount of data. The second stage is the samples *cleaning*, which seeks to consider only those vehicles whose speed was correctly measured. The third stage is the *Extraction*, where the data is extracted from the images sequences. The four stage is *Validation*, where the extracted data are corroborated to be a valid sample. These four stages result in a set of samples ready to be processed by the predictive model. This whole process is described in Algorithm 1.

Algorithm 1: Data Colletion

Data: Traffic video
Result: Csv File with features
Sampling (Camera, Radar) $\longrightarrow$ Samples;
while *Exist samples* **do**
  Cleaning *Samples* $\longrightarrow$ Clean_Samples;
  Extraction *Clean_Samples* $\longrightarrow$ Features;
  Validation *Features* $\longrightarrow$ Valid_Features ;
  Save_features *Valid_Features* $\longrightarrow$ csv_File
end

- Sampling: For sampling, two devices were required to extract the dataset that will later be used to train the predictive models. One of the devices is the Bushnell radar. At the same time, the other device is a video camera; both devices are described in the *Material Section*. The videos are taken at 60 frames per second, but it should be taken into account that having a low resolution may result in a loss of image quality and a smaller area than desired. While having low frames will cause loss of vehicles passing at a higher speed. On the other hand, increasing the resolution and frames per second causes the system to take longer to process the video.
  There is special care when positioning the camera since we do not want the experiments to be considered 2D speed calculations. Therefore, the camera was placed where vehicular traffic passes with a certain degree of inclination, without pointing the camera directly to the vehicles' side (Figure 8).
  Since the camera and the speed radar are devices that are not synchronized, it was necessary to implement a mechanism to associate the radar reading with the video recording.
- Cleaning: The radar device and the tracking system are not linked, so it is essential to combine the speed supplied by the radar with the information received by the tracking system (video camera). For this, a visual inspection of the samples taken is required. The first step is identifying the landmarks to reference vehicle entry and exit points. Using our exit landmark as a reference, we record the second when a vehicle passes through the exit landmark. Figure 8 shows an example of how the landmarks in the images were established as reference points for vehicle entry and exit. Also shown is the inclination of the camera used for sampling.
  In addition, we identify the lane in which the vehicles pass to separate them in the three street lanes, we represent with zero the first lane, one the central lane, and two the last lane, with this we seek to create three datasets as divided in the work of [35]
  This information obtained visually we save in a csv (comma-separated values) file, which will have the following three attributes:
  - The second that the object passes through the exit point.
  - The vehicle's speed.
  - Lane

Figure 8. Sample of camera position a x axes landmarks (entry and exit).

- Extraction: Specific characteristics of the previously cleaned videos are necessary to perform the speed estimation. These characteristics are the basis to carry out the estimation. The characteristics extracted from the videos are described in Table 2.

Table 2. Output attributes saved in the csv file

Attribute	Description
Output Angle	Angle from the input to the output.
Distance Traveled	Distance traveled in pixels from the input to the output.
Input Area	Area in pixels at the input.
Output Area	Area in pixels at the output.
FPS	Frames per seconds.
Time	Travel time from the entry to the exit point..
Lane	Street lane where pass the vehicle.
Speed	Speed detected by radar.
Identifier	Identifier to the created image.

The execution of the system is simple. We need the CSV file generated earlier with the speeds and limits that we also identified in the previous step. The system is in charge of reading the video using the OpenCV library, and it examines frame by frame the content. Different options were explored in the literature to choose the most appropriate network to detect vehicles, such as the SSD network [51] and its variants, Retina network [52], and its variants (ResNet), YOLO network [17], and its variants. In [17], performance comparisons of these networks were made. The results showed that the YOLOv3 network obtains a significantly lower response time of object detections (between 5 and, in some cases, 12 times lower). With a negligible precision difference, obtaining 55.3 versus 59.1 of mAP-50 (Mean average precision), but with milliseconds of 29 versus 172. We select the neural network YOLOv3 because it is a pretrained (with over 300,000 COCO dataset images) network and response time and accuracy obtained in previous comparative works. By using YOLOv3 [15] we can identify multiple objects in a single prediction; once it has identified all the vehicles, it draws the box corresponding to each one of them. This process allows the detection and identification of the objects of interest inside each frame. It was not necessary to apply data augmentation or similar techniques during feature extraction or model training. However, splitting the video into frames was required during the feature extraction process since the YOLOv3 network uses these frames for vehicle detection. The vehicles are tracked through a Kalman filter [53] to determine their location in the next frame. We choose this filter because it estimates a joint probability distribution over the variables for each timeframe. It has better performance when is used for linear or linearized processes and measurement systems than particle filter, which is more suitable for nonlinear systems. The system is in charge of preserving all the locations of the vehicles over time. Therefore, we can draw each vehicle's path inside the scenes and calculate the straight line corresponding to each trajectory. Figure 9 shows a detected vehicle in a white box, as well as a pair of lines, one yellow and

another red, that correspond to the tracking of the same one and the calculated straight line corresponding to the tracking.

Figure 9. Vehicle identified and tracked.

- Validation: The validation of the samples obtained is performed by visual inspection. The validation process starts by identifying the sample in the csv file using the unique identifier assigned to each sample. At the same time, the saved image corresponding to that sample is reviewed, and it is verified that the vehicle in question is entirely within the established delimiter (white box). This delimiter must completely cover the vehicle when passing through the entry and exit landmark. If the cover does not completely cover the vehicle, or the sample is taken moments before or after passing through a landmark, this sample is labeled as invalid. Otherwise, it is considered a valid sample. Figure 10 shows an example of a valid sample, and Figure 11 shows an example of an invalid one.

Figure 10. A valid sample.

Figure 11. An invalid sample.

4.2.2. Predictive Mode

Different statistical and machine learning methods were used to select the predictive model. The process for inferring data (in this case, speed) for both regression and machine learning models is similar. The process starts by separating the data (into training and validation sets) in both cases. This is usually done on a 70% (training)–30% (validation) basis. The training set is used to train (tune the model hyperparameters) the predictive model.

Once the model is trained, the validation set is used to perform the speed estimations. These estimations (accuracy) are used to validate the trained model. The model process is depicted in Figure 12.

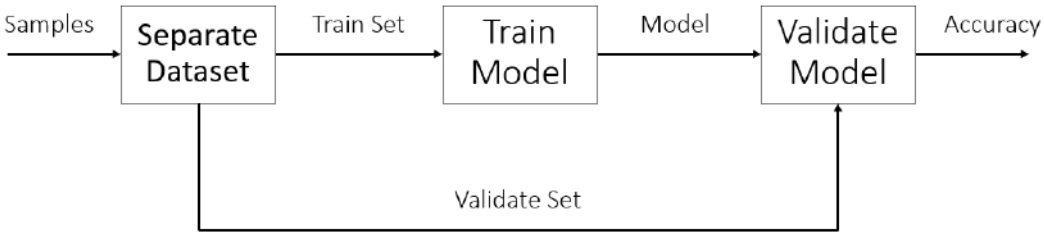

Figure 12. Flowchart for defining a predictive velocity model.

The validation process takes the trained model and receives input data from the validation set. The output of this model (speed estimation) is compared against the existing velocity captured. We call this difference the error, and the smaller the error, the better the learning model is. The evaluation metric (error) used for the vehicle's speed predictive model was Mean Absolute Error Equation (7), expressed meters/second.

$$MAE = (\frac{1}{n}) \sum_{i=0}^{n} | y_i - x_i | \quad (7)$$

In general, to obtain the output speed estimation, with the methods used in this work, we started operating as input data all the features described in Table 2, such as the output angle, distance traveled, input area, output area, time, and lane. Nevertheless, we obtained an average error of 2.16 km/h, representing a significant difference between the real and the estimated speed. Hence, we decided to reduce the number of input features and consider the distance (such as Loor et al. [38] and Dong et al. [33]) and time since with only these two features we obtained the best results.

5. Results

The results of this work will show the realization of the dataset (samples creation) and the performance of the learning models used for speed estimation.

5.1. Sampling

For sampling, a place was chosen in the city where the flow of vehicles is considered constant. This place was on the Las Torres road in front of Plaza San Isidro in the city of Culiacán, Sinaloa. We collected videos in the mornings between 10:00 and 12:00 h. In contrast, other samples were taken between 17:00 and 19:00 h. The samples indicated a greater vehicular flow in the afternoon.

This place has a flow from west to east, which is a flow from right to left for the camera. It can also be seen that the street through which the vehicles pass is not level. For the camera view, the vehicles pass from top to bottom. As mentioned in the Methodology, special care was taken when positioning the camera so that the cars do not pass parallel to the camera view. It was positioned pointing slightly up the street.

The implementation of the system resulted in a total of 29 processed videos, giving a total of 532 samples. For the experiments, the data were separated by lanes 0, 1, and 2, representing the first lane, middle lane, and last lane. It resulted in 8 samples for the first lane, 239 samples for the second lane, and 285 samples for the third lane.

5.2. Experimental Configuration

We used the configurations shown in Table 3 and a correlation between the distance and time to obtain the speed estimation accuracy of the models considered (five statistical methods and three machine learning methods), allowing us to present a comparison among them, see Tables 4 and 5.

We use the Scikit-Learn library [54] to carry out the model experimentations of Linear Regression (LR), Ridge Regression (CR), Lasso (Absolute Minimum, Selection, and Contraction Operator - Lasso), Bayesian Ridge Regression (BCR), Elastic Net Regression (Elastic Net), Random Forest Regression (RRF) and Support Vector Machine Regression (SVMR). The Scikit-Learn library determines how we can interact with the algorithm configuration, as shown in Table 3. Even when each variable has the same name, its functionality depends on how the algorithm uses this data to interpret and carry out its task; we can see further details in [54].

We use the Tensor Flow library [55] to implement the Multilayer Perceptron (MLP) neural network. The MLP network used in this work has two input neurons, five neurons in the hidden layer, and one neuron in the output layer.

We published the dataset and source code of the experimentation at https://github.com/garay54/Regression_Models, accessed on 30 January 2022.

Table 3. Parameters.

Model	Configuration
Linear Regresion	copy X = True fit_intercept= True n jobs = None normalize = False positive = False
Ridge Regression, Lasso Regression, Bayesian Ridge, Elastic Net, Random Forest, Suppor Vector Machine	alpha = 0.5 copy_X = True fit_intercept = True max_iter = None normalize = False random_state = None solver = auto tol = 0.001
MLP	Input = 2 Hidden = 5 Output = 1

Table 4. Comparative middle lane table of different learning methods results (speed) used. The error metric is the mean absolute error and is expressed in km/h.

	Statistical Models					Machine Learning Models		
	LR	**CR**	**Lasso**	**BCR**	**Elastic Net**	**RFF**	**SVMR**	**MLP**
SD	0.143	0.142	0.143	0.143	0.143	0.0141	0.163	3.397
MAE	1.694	1.694	1.699	1.695	1.701	1.828	1.960	2.767

Table 5. Comparative last lane table of different learning methods results (speed) used. The error metric is the mean absolute error and is expressed in km/h.

	Statistical Models					Machine Learning Models		
	LR	**CR**	**Lasso**	**BCR**	**Elastic Net**	**RFF**	**SVMR**	**MLP**
SD	0.081	0.081	0.082	0.081	0.094	0.094	0.141	1.843
MAE	0.956	0.965	0.975	0.956	0.997	1.087	1.087	3.174

5.3. Speed Estimation

After training each of the above models, the results show that the statistical methods perform better than the machine learning methods. The experiments indicated that the difference between the statistical methods is minimal. Tables 4 and 5 list the results

obtained by each of the proposed methods. It also shows the standard deviation of the errors obtained. As mentioned above, the metric used for the precision was the mean absolute error, and the scale presented is kilometers over an hour (km/h).

It is worth mentioning that no experiments were performed in the first lane due to the lack of samples corresponding to that lane. Tables 4 and 5 show a similar performance behavior in the statistical methods. They have a better performance than the random forest, support vector machines, and neural networks. It can be said that the method with the best performance was linear regression with a minimal difference. Equation (8) shows the equation of the linear regression with the resulting coefficients.

$$Speed = (0.00568) * Distance - (7.2141) * Time + (23.725) \quad (8)$$

where *Distance* and *Time* are the inputs of the estimation process. *Distance* is expressed in pixels and takes values from 701.38149 to 1206.1857 pixels; *Time* is expressed in seconds and takes values from 0.4164 to 2.0150 seconds.

5.4. Methods Comparison

In the Figure 13 we observe a comparison with some methods mentioned in Section 2 Loor et al. [38], Zhang et al. [37], Song et al. [36], Kampelmuhler et al. [35], Dong et al. [33], Bell et al. [22], Li et al. [30], Lee et al. [29], Anil Rao et al. [28], Vakili et al. [23], Luvizon et al. [43], Yang et al. [41] and Fernandez et al. [13]. This comparison was made between the accuracy reported in each work and the results shown in this article. These works use as a reference the license plate of the vehicle or characteristics that we can frequently find in a vehicle, such as lights. Thanks to these characteristics, it is possible to determine the trajectory and estimate the distance traveled. We observe that this work improves accuracy compared to most others without using an object reference to detect the vehicle and determine the speed. Only the work of Luvizon reported a better performance by a slight difference.

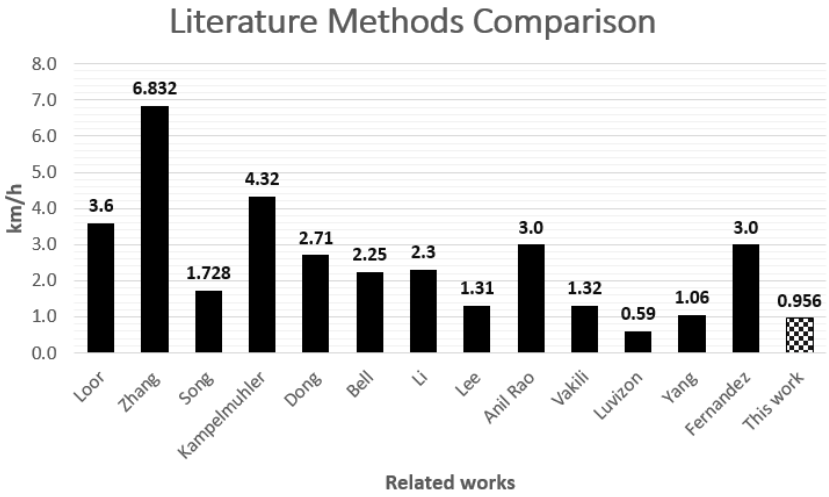

Figure 13. Methods Comparison.

Table 6 describes several aspects concerning the dataset and the input data used to extract features that allow the speed estimation. We contrast environmental considerations and limitations that can help decide when and why we prefer one solution over another. Likewise, Table 6 summarizes that further work is needed to fully solve the speed estimations problem without restrictions of the sensors, distance of the objects, calibrations, or high computational cost.

Table 6. Comparison of Input data and Considerations between state-of-the-art works and our proposal.

Autor	Dataset	Input Data	Features Extracted	Environment Considerations	Limitations
Fernandez et al. [13]	Doesn't Apply	Video images	License plate region	2 cameras calibration	Calibration of the cameras affect the results.
Yang et al. [41]	Does not apply	Video images	Limits of plate, logo and light of vehicle	Compare license plate, light, logo to matching vehicles	Needs two industrial cameras with high resolution.
Luvizon et al. [43]	Does not apply	Video images	Characters of license plate	Image rectificaton	If the license plate is not legible or too small it has detection problems.
Vakili et al. [23]	4 dataset	Video images	Coordinates of the license plate corners, number, and type and the color of the vehicle	Camera should have its back to the vehicles and higher height.	If the license plate is not legible or too small it has detection problems.
Anil Rao et al. [28]	Doesn't Apply	Video images	Pixels traveled by the vehicle	Uses homography to perspective transform	Camera calibration parameters must be manually adjusted and monitored.
Lee et al. [29]	Doesn't Apply	Input signal and output signal	Does not extract features	The high of the camera to avoid occlusions	It does not classify vehicles by type, it only makes its detections.
Li et al. [30]	Doesn't Apply	Video images	Distance	Better detections on day	Shadows affect final performance.
Bell et al. [22]	COCO dataset	Video images	Vehicle's coordinates	Distance is obtained by geometric estimation.	Change in size of detections affect velocity estimation.
Dong et al. [33]	VehSpeedDataset10 5332 short videos	RGB images and optical flow	Distance	Camera calibration using coordinate system	Lack of quality data.
Kampelmuhler et al. [35]	1074 sequences in freeway traffic	Video images	Vehicle Tracks, Depth and Motion	Does not have considerations	Processing time for feature extraction.
Song et al. [36]	KITTI	Video images	Coordinates of vehicle	Does not have considerations	Use a GPU for processing
Zhang et al. [37]	KITTI	Video images	Coordinates of vehicle	Does not have considerations	The detection thresholds must be manually calibrated.
Loor et al. [38]	KTH data set, KITTI	Video images	Distance on pixels	Does not have considerations	If the image moves, parked vehicles can be detected as moving.
This work	COCO dataset and Own dataset	Video images	Distance and time	Does not have considerations	The videos must be analize manually to separate the bad samples.

Table 7 describes the speed estimation process, explaining the form that the works detect vehicles and how, from this process, the speed estimation algorithms use the features extracted to produce the output speed. Vehicle detection has been divided into three methods: (1) object segmentation, (2) object detection, and (3) object tracking since we can carry out this process with combinations of algorithms focused on specific tasks. We remark that each work presents its way of using the extracted features impacting the performance.

Table 7. Comparison of speed estimation process between state-of-the-art works and our proposal.

Autor	Vehicles Detection			Speed Estation Algorithm	Performance
	Object Segmentation	Object Detection	Object Tracking		
Fernandez et al. [13]	Does not apply	MSER Detector, License plate	Does not use tracking	Uses the detection time of the cameras and the distance between detections.	<3 km/h
Yang et al. [41]	Does not apply	SSD (Single-Shot Multibox Detector) YOLOv4, license plate	LNCC+STIF s	Uses combination of logo, badge and light speeds, detects speed of each using distance and time.	0.9, +1.06 km/h
Luvizon et al. [43]	Character filtering	Text detection for license plate detection SNOOPERTEXT	Kanade–Lucas–Tomas (KLT), and Feature Transform (SIFT)	It converts pixels traveled by the vehicle to meters and uses the time it takes to cross the vehicle.	0.59 km/h
Vakili et al. [23]	Does not apply	The license plate	OpenALPR library	Using the geometric information of the system and the distance travelledby vehicles the speed is computed.	1.32 km/h
Anil Rao et al. [28]	Background Subtraction	Vehicle Centroid	Kalman filter, Hungarian Algorithm	Perspective Transform to obtein of distance traveled by the vehicle, then calculate the speed with the distance and time.	3 km/h
Lee et al. [30]	Does not apply	2 LiDAR sensors	Doesn't tracking	It uses the distance between 2 points and the time it takes for vehicles to cross those 2 points.	1.31 km/h
Li et al. [30]	Background Subtraction	vehicules	feature point of vehicle head (FPVH)	Use the tracking algorithm to obtain the distance traveled by the vehicle and then use the time.	2.3 km/h

Table 7. *Cont.*

Autor	Vehicles Detection			Speed Estation Algorithm	Performance
	Object Segmentation	Object Detection	Object Tracking		
Bell et al. [22]	Does not apply	YOLOv2 vehicles	Simple Online Realtime Tracking (SORT)	Performs pixel-to-distance transformation, and obtains the time the vehicle takes to travel the distance to obtain the speed.	2.25 km/h
Dong et al. [33]	Does not apply	Contours of the vehicle	Detected contours	It uses the calibration of the camera to obtain the distance traveled by the tracked vehicles, and the time to obtain the speed.	2.71 km/h
Kampelmuhler et al. [35]	Does not apply	Vehicles	Median Flow and MIL tracks	Uses a MLP model with the extracted features to obtain the Speed.	4.32 km/h
Song et al. [36]	Does not apply	PWCNet pretrained from FlyingChairs, vehicles	Does not use tracking	Speed estimation with additional geometrical and a temporal optical flow track.	1.728 km/h
Zhang et al. [37]	Does not apply	Region-based CNN (R-CNN), vehicles	Does not use tracking	Uses model of deep learning to obtain speed.	6.832 km/h
Loor et al. [38]	Does not apply	SpeedNet	Does not use tracking	Uses a FlowNet trained to predict speed of Vehicles on a video.	3.6 km/h
This work	Does not apply	YOLO v3	Kalman Filter	Coordinates are used to measure time and distance vehicles take to cross, then regression models are used to obtain the speed.	0.956 km/h

6. Conclusions

This work presents a system that creates its dataset to obtain the speed of a vehicle from image sequences. This dataset created served as a base for experiments with different statistical and machine learning methods.

We divide the speed estimation process into two stages. The first stage consisted of extracting the essential features from the videos of vehicular traffic. This stage used a video camera a radar to take de videos. We processed 29 videos (previously obtained) to obtain the 532 valid samples. This process combines the use of YOLO to detect multiple vehicles and the Kalman filter to predict the vehicle's trajectory. The second stage uses valid samples as an input of different statistical and machine learning methods to estimate the vehicles' speed. The system shows that it is relatively simple to obtain many parameters from a video by implementing existing computer vision algorithms and trained machine learning models.

One of the biggest challenges for implementing this system has been sampling since it is necessary to find an ideal place with enough vehicular flow to obtain as much data as possible. Another essential point to mention is the processing time of each sample. As presented above, it can take up to 33% more time than the original video in the worst case. Plus, the time required to clean and validate the samples adds more time to each video's processing.

The results obtained from the generated dataset with different statistical and machine learning methods show the accuracy of each method. The linear regression method had the best-performing accuracy, obtaining an accuracy of 0.956 km/h on the mid lane and 1.694 km/h on the third lane. However, the other statistical methods implemented showed similar performance. On the other hand, the machine learning methods used in this work had acceptable performance; however, they could not improve the performance achieved by the statistical methods.

We will explore the use of deep learning and transfer learning to the developed dataset to improve the results presented in this work. We will focus on solving limitations, such as complete system automation without human intervention since human interactions were necessary for the samples' generation and validation. In addition, we will include in the dataset of night videos to test our proposal's behavior and videos with a higher cars frequency. The information generated with the system, such as frequency and type of vehicles, could also be expanded to allow experts to improve traffic flow and reduce accidents or environmental impacts.

Author Contributions: Conceptualization, H.R.-R. and L.A.M.-R.; formal analysis, H.R.-R., M.L.-B. and L.A.M.-R.; investigation, H.R.-R., R.I.-R., G.E.P.-P. and L.A.M.-R.; methodology, H.R.-R. and M.L.-B.; sampling, R.I.-R.; software, R.I.-R.; validation, H.R.-R. and L.A.M.-R.; writing—original draft, H.R.-R., L.A.M.-R., M.A.R.-G. and G.E.P.-P.; writing—review and editing, H.R.-R., L.A.M.-R., M.A.R.-G. and G.E.P.-P. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by TECNM through the Research Project 9113ku and the Mexican National Council for Science and Technology (CONACYT) through the Research Project 613.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the study's design, in the collection, analyses, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

MAE	Mean Absolute Error
MSE	Mean Squared Error
INEGI	National Institute of Statistics and Geographic Information
ADAS	Advanced Driver Assistance Systems
ITS	Intelligent Transport Systems
YOLO	You Only Look Once
LR	Linear Regression
CR	Ridge Regression
Lasso	Least Absolute Shrinkage and Selection Operator
BCR	Bayesian Ridge Regression
RRF	Random Forest Regression
SVM	Support Vector Machine Regression
ANN	Artificial Neuronal Network
GPU	Graphic Processing unit
RAM	Random access memory
CPU	Central Processing Unit
HD	High definition
FPS	Photoframes Per Second
CSV	Comma-Separated Values

References

1. Zaki, P.S.; William, M.M.; Soliman, B.K.; Alexsan, K.G.; Khalil, K.; El-Moursy, M. Traffic signs detection and recognition system using deep learning. *arXiv* **2020**, arXiv:2003.03256.
2. Velázquez Narváez, Y.; Zamorano González, B.; Ruíz Ramos, L. Siniestralidad vial en la frontera norte de Tamaulipas. Enfoque en los procesos administrativos de control. *Estud. Front.* **2017**, *18*, 1–24. [CrossRef]
3. Carro-Pérez, E.H.; Ampudia-Rueda, A. Conductas de riesgo al conducir un automóvil en zonas urbanas del sur de Tamaulipas y la Ciudad de México. *CienciaUAT* **2019**, *13*, 100–112. [CrossRef]
4. Impedovo, D.; Balducci, F.; Dentamaro, V.; Pirlo, G. Vehicular traffic congestion classification by visual features and deep learning approaches: A comparison. *Sensors* **2019**, *19*, 5213. [CrossRef]
5. Coifman, B.; Neelisetty, S. Improved speed estimation from singleloop detectors with high truck flow. *Intell. Transp. Syst.* **2014**, *18*, 138–148. [CrossRef]
6. Jin, G.; Ye, B.; Wu, Y.; Qu, F. Vehicle Classification Based on Seismic Signatures Using Convolutional Neural Network. *IEEE Geosci. Remote Sens. Lett.* **2019**, *16*, 628–632. [CrossRef]
7. Balid, H.T.W.; Refai, H.H. Intelligent vehicle counting and classification sensor for real-time traffic surveillance. *Intell. Transp. Syst.* **2017**, *19*, 1784–1794. [CrossRef]
8. Bautista, C.M.; Dy, C.A.; Mañalac, M.I.; Orbe, R.A.; Cordel, M. Convolutional neural network for vehicle detection in low resolution traffic videos. In Proceedings of the 2016 IEEE Region 10 Symposium (TENSYMP), Bali, Indonesia, 9–11 May 2016; pp. 277–281. [CrossRef]
9. Liu, K.; Mattyus, G. Fast Multiclass Vehicle Detection on Aerial Images. *IEEE Geosci. Remote Sens. Lett.* **2015**, *12*, 1938–1942. [CrossRef]

10. Taghvaeeyan, S.; Rajamani, R. Portable Roadside Sensors for Vehicle Counting, Classification, and Speed Measurement. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 73–83. [CrossRef]
11. Lee, H.; Coifman, B. Using LIDAR to Validate the Performance of Vehicle Classification Stations. *J. Intell. Transp. Syst.* **2015**, *19*, 355–369. [CrossRef]
12. Won, M.; Zhang, S.; Son, S.H. WiTraffic: Low-Cost and Non-Intrusive Traffic Monitoring System Using WiFi. In Proceedings of the 2017 26th International Conference on Computer Communication and Networks (ICCCN), Vancouver, BC, Canada, 31 July–3 August 2017; pp. 1–9. [CrossRef]
13. Fernández Llorca, D.; Hernández Martínez, A.; García Daza, I. Vision-based vehicle speed estimation: A survey. *IET Intell. Transp. Syst.* **2021**, *15*, 8. [CrossRef]
14. Maduro, C.; Batista, K.; Peixoto, P.; Batista, J. Estimating Vehicle Velocity Using Rectified Images. In Proceedings of the VISAPP (2), Funchal, Portugal, 22–25 January 2008; pp. 551–558.
15. Redmon, J.; Divvala, S.; Girshick, R.; Farhadi, A. You Only Look Once: Unified, Real-Time Object Detection. In Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016; pp. 779–788. [CrossRef]
16. Huang, X.; Wang, X.; Lv, W.; Bai, X.; Long, X.; Deng, K.; Dang, Q.; Han, S.; Liu, Q.; Hu, X.; et al. PP-YOLOv2: A practical object detector. *arXiv* **2021**, arXiv:2104.10419.
17. Redmon, J.; Farhadi, A. Yolov3: An incremental improvement. *arXiv* **2018**, arXiv:1804.02767.
18. Roy, A.M.; Bose, R.; Bhaduri, J. A fast accurate fine-grain object detection model based on YOLOv4 deep neural network. *Neural Comput. Appl.* **2022**, *34*, 1–27. [CrossRef]
19. Yu, J.; Zhang, W. Face mask wearing detection algorithm based on improved YOLO-v4. *Sensors* **2021**, *21*, 3263. [CrossRef]
20. Kumar, K.K.; Chandrakant, P.; Kumar, S.; Kushal, K. Vehicle Speed Detection Using Corner Detection. In Proceedings of the 2014 Fifth International Conference on Signal and Image Processing, Bangalore, India, 8–10 January 2014; pp. 253–258. [CrossRef]
21. Kamoji, S.; Koshti, D.; Dmonte, A.; George, S.J.; Sohan Pereira, C. Image Processing based Vehicle Identification and Speed Measurement. In Proceedings of the 2020 International Conference on Inventive Computation Technologies (ICICT), Coimbatore, India, 26–28 February 2020; pp. 523–527. [CrossRef]
22. Bell, D.; Xiao, W.; James, P. Accurate Vehicle Speed Estimation from Monocular Camera Footage. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2020**, *V-2-2020*, 419–426. [CrossRef]
23. Vakili, E.; Shoaran, M.; Sarmadi, M. Single-camera vehicle speed measurement using the geometry of the imaging system. *Multimed. Tools Appl.* **2020**, *79*, 19307–19327. [CrossRef]
24. Dahl, M.; Javadi, S. Analytical modeling for a video-based vehicle speed measurement framework. *Sensors* **2020**, *20*, 160. [CrossRef]
25. Liu, C.; Huynh, D.Q.; Sun, Y.; Reynolds, M.; Atkinson, S. A Vision-Based Pipeline for Vehicle Counting, Speed Estimation, and Classification. *IEEE Trans. Intell. Transp. Syst.* **2021**, *22*, 7547–7560. [CrossRef]
26. Ho, H.W.; de Croon, G.C.; Chu, Q. Distance and velocity estimation using optical flow from a monocular camera. *Int. J. Micro Air Veh.* **2017**, *9*, 198–208. [CrossRef]
27. Schoepflin, T.; Dailey, D. Dynamic camera calibration of roadside traffic management cameras for vehicle speed estimation. *IEEE Trans. Intell. Transp. Syst.* **2003**, *4*, 90–98. [CrossRef]
28. Anil Rao, Y.; Kumar, N.S.; Amaresh, H.; Chirag, H. Real-time speed estimation of vehicles from uncalibrated view-independent traffic cameras. In Proceedings of the TENCON 2015-2015 IEEE Region 10 Conference, Macao, 1–4 November 2015; pp. 1–6. [CrossRef]
29. Lee, K.H. A Study on Distance Measurement Module for Driving Vehicle Velocity Estimation in Multi-Lanes Using Drones. *Appl. Sci.* **2021**, *11*, 3884. [CrossRef]
30. Li, S.; Yu, H.; Zhang, J.; Yang, K.; Bin, R. Video-Based Traffic Data Collection System for Multiple Vehicle Types. *IET Intell. Transp. Syst.* **2014**, *8*, 164–174. Available online: https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/iet-its.2012.0099 (accessed on 30 January 2022). [CrossRef]
31. Kurniawan, A.; Ramadlan, A.; Yuniarno, E.M. Speed Monitoring for Multiple Vehicle Using Closed Circuit Television (CCTV) Camera. In Proceedings of the 2018 International Conference on Computer Engineering, Network and Intelligent Multimedia (CENIM), Surabaya, Indonesia, 26–27 November 2017; pp. 88–93. [CrossRef]
32. Jalalat, M.; Nejati, M.; Majidi, A. Vehicle detection and speed estimation using cascade classifier and sub-pixel stereo matching. In Proceedings of the 2016 2nd International Conference of Signal Processing and Intelligent Systems (ICSPIS), Tehran, Iran, 14–15 December 2016; pp. 1–5. [CrossRef]
33. Dong, H.; Wen, M.; Yang, Z. Vehicle Speed Estimation Based on 3D ConvNets and Non-Local Blocks. *Future Internet* **2019**, *11*, 123. [CrossRef]
34. Burnett, K.; Samavi, S.; Waslander, S.L.; Barfoot, T.D.; Schoellig, A.P. *aUToTrack: A Lightweight Object Detection and Tracking System for the SAE AutoDrive Challenge*; University of Toronto: Toronto, ON, Canada, 2019; pp. 209–216. [CrossRef]
35. Moritz Kampelmuhler, M.G.M.; Feichtenhofer, C. Camera-Based Vehicle Velocity Estimation from Monocular Video. In Proceedings of the 23rd Computer Vision Winter Workshop, Cesky Krumlov, Czech Republic, 5–7 February 2018.
36. Song, Z.; Luand, J.; Zhang, T.; Li, H. *End-to-End Learning for Inter-Vehicle Distance and Relative Velocity Estimation in ADAS with a Monocular Camera*; Cornell University: Ithaca, NY, USA, 2020.

37. Yaqi Zhang, B.W.; Liu, W. *Vehicle Motion Detection Using CNN*; Stanford: Stanford, CA, USA, 2017.
38. Loor, C. *Visual Speedometer: Learning Velocity from Two Images*; University of Amsterdam: Amsterdam, The Netherlands, 2017.
39. Fernández-Llorca, D.; Salinas, C.; Jimenez, M.; Morcillo, A.; Izquierdo, R.; Lorenzo Díaz, J.; Sotelo, M.A. Two-camera based accurate vehicle speed measurement using average speed at a fixed point. In Proceedings of the 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil , 1–4 November 2016; pp. 2533–2538. [CrossRef]
40. Yang, L.; Li, M.; Song, X.; Xiong, Z.; Hou, C.; Qu, B. Vehicle Speed Measurement Based on Binocular Stereovision System. *IEEE Access* **2019**, *7*, 106628–106641. [CrossRef]
41. Yang, L.; Luo, J.; Song, X.; Li, M.; Wen, P.; Xiong, Z. Robust Vehicle Speed Measurement Based on Feature Information Fusion for Vehicle Multi-Characteristic Detection. *Entropy* **2021**, *23*, 910. [CrossRef]
42. Yang, L.; Li, Q.; Song, X.; Cai, W.; Hou, C.; Xiong, Z. An Improved Stereo Matching Algorithm for Vehicle Speed Measurement System Based on Spatial and Temporal Image Fusion. *Entropy* **2021**, *23*, 866. [CrossRef]
43. Luvizon, D.; Nassu, B.; Minetto, R. Vehicle speed estimation by license plate detection and tracking. In Proceedings of the ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing, Florence, Italy, 4–9 May 2014. [CrossRef]
44. Gutiérrez, E.; Vladimirovna, O. *Estadística Inferencial 1 para Ingeniería y Ciencias*; Grupo Editorial Patria: Ciudad de México, Mexico, 2016; pp. 271–348.
45. McDonald, G.C. Ridge regression. *WIREs Comput. Stat.* **2009**, *1*, 93–100. [CrossRef]
46. Hans, C. Bayesian lasso regression. *Biometrika* **2009**, *96*, 835–845. [CrossRef]
47. Zhang, Z.; Lai, Z.; Xu, Y.; Shao, L.; Wu, J.; Xie, G.S. Discriminative Elastic-Net Regularized Linear Regression. *IEEE Trans. Image Process.* **2017**, *26*, 1466–1481. [CrossRef]
48. Minka, T. Bayesian Linear Regression. Technical Report, Citeseer. 2000. Available online: https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.39.4002&rep=rep1&type=pdf (accessed on 30 January 2022).
49. Rodriguez-Galiano, V.; Sanchez-Castillo, M.; Chica-Olmo, M.; Chica-Rivas, M. Machine learning predictive models for mineral prospectivity: An evaluation of neural networks, random forest, regression trees and support vector machines. *Ore Geol. Rev.* **2015**, *71*, 804–818. [CrossRef]
50. Olabe, X.B. *Redes Neuronales Artificiales y sus Aplicaciones*; Publicaciones de la Escuela de Ingenieros; Escuela Superior de Ingeniería de Bilbao: Bilbao, Spain, 1998.
51. Fu, C.Y.; Liu, W.; Ranga, A.; Tyagi, A.; Berg, A.C. DSSD: Deconvolutional Single Shot Detector. *arXiv* **2017**, arXiv:1701.06659.
52. Lin, T.Y.; Goyal, P.; Girshick, R.; He, K.; Dollar, P. Focal Loss for Dense Object Detection. In Proceedings of the IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017.
53. Welch, G.; Bishop, G. An Introduction to the Kalman Filter. 1995. Available online: https://perso.crans.org/club-krobot/doc/kalman.pdf (accessed on 30 January 2022).
54. Kramer, O. *Scikit-Learn*; Springer: Berlin/Heidelberg, Germany, 2016; pp. 45–53.
55. Dillon, J.V.; Langmore, I.; Tran, D.; Brevdo, E.; Vasudevan, S.; Moore, D.; Patton, B.; Alemi, A.; Hoffman, M.; Saurous, R.A. Tensorflow distributions. *arXiv* **2017**, arXiv:1711.10604.