UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL
INSTITUTO DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

FELIPE LEIVAS

Vehicle Speed Estimation Based on License Plate Detection

Work presented in partial fulfillment
of the requirements for the degree of
Bachelor in Computer Science

Advisor: Prof. Dr. Rosito Jung Claudio

Porto Alegre
December 2021

“You miss 100% of the shots you don’t take

- Wayne Gretzky"

–Michael Scott

ABSTRACT

In this work, we present an approach for vehicle speed estimation using a flexible camera setup: the only requirement is a calibrated camera. Then we use the calibrated camera to record images of the vehicle on the road, and use a state-of-the-art object detector to identify if there is a vehicle in the image. For each vehicle we use a license plate detector to extract the corresponding pixels for the four corners of the license plate (LP), and we use the known dimensions of the LP and estimate the homography matrix to be able to obtain the real world coordinates for the LP. Then, we propose a two methods to better estimate the vehicle speed based on the tracking of the LP. We create a dataset to test the proposed method, and we show the results for each method proposed method. Our best method was able to estimate the speed of vehicles with an average error of 11.15%.

Keywords: Computer Vision. Vehicle Speed Estimation.

RESUMO

Neste trabalho propomos uma solução para estimação da velocidade de veículos usando uma configuração de câmera com apenas uma restrição: a câmera precisa estar calibrada. Após isso,usamos a câmera calibrada para gravar imagens de veículos nas vias, e usamos um detector de objeto estado da arte para identificar se existe um veículo na imagem. Para cada veículo que o detector de objetos detectar, usamos detector de placas de veículo para extrair os pixels correspondentes às quinas da placa, como sabemos as dimensões reais da placa, estimamos uma matriz capaz de obter as coordenadas de mundo da placa. Então propomos uma série de métodos para melhor estimar a velocidade do veículo com base no monitoramento da placa. Também criamos um dataset para podermos testar os métodos propostos. Também mostramos os resultados para cada método proposto. Nosso melhor método é capaz de estimar a velocidade dos veiculos com um erro médio de 11.15%.

Palavras-chave: Visão computacional. Estimador de velocidade de carro.

LIST OF FIGURES

Figure 1.1 Example of recorded frames by a hand-hold phone 11

Figure 2.1 Pinhole camera 12
Figure 2.2 Camera coordinate system 14
Figure 2.3 2D projection 15
Figure 2.4 RANSAC estimated model 19
Figure 2.5 Chess board pattern 20

Figure 3.1 Average object detection performance per year 22
Figure 3.2 YOLO method model 23
Figure 3.3 IWPOD-NET process 24

Figure 4.1 LP detection pipeline 29
Figure 4.2 LP coordinates 29
Figure 4.3 LP corner tracking (generated by multiple frames) 30
Figure 4.4 LP corner estimation inaccuracy 31
Figure 4.5 Fitted line for coordinates 32

Figure 5.1 Dataset video frame (cropped for better visualization) 35
Figure 5.2 Videos site 35
Figure 5.3 Absolute errors range (in Km/h) 38

LIST OF TABLES

Table 5.1 GT speed and estimated values using different methods 36
Table 5.2 Methods relative error (in%) 37
Table 5.3 Methods relative error in range 38

LIST OF ABBREVIATIONS AND ACRONYMS

LP	License Plate
LPR	License Plate Recognition
LPD	License Plate Detection
COP	Center of Projection
RANSAC	Random Sample Consensus
DL	Deep Learning
YOLO	You Only Look Once
CNN	Convolutional Neural Network
GT	Ground Truth

CONTENTS

1 INTRODUCTION....10
2 BACKGROUND....12
2.1 Pinhole Camera Model....12
2.2 Perspective Projection....13
2.2.1 Camera Transformations....13
2.2.2 3D to 2D Projection....14
2.2.3 Pixel Mapping....14
2.2.4 Final Projection equation....15
2.3 Estimation of the planar homography matrix....16
2.4 RANSAC Algorithm....18
2.5 Estimation of camera intrinsics....19
3 RELATED WORK....21
3.1 Object detection....21
3.1.1 Generic Object Detection....21
3.1.2 License Plate Detection....22
3.2 Vehicle Speed Estimation....24
4 THE PROPOSED APPROACH....27
4.1 Overview....27
4.2 Vehicle and License plate detection....28
4.3 Pose estimation....28
4.4 Speed computation....31
4.4.1 RANSAC-Based Method....31
4.4.2 Combinational Set Metrics....33
5 EXPERIMENTAL RESULTS....34
5.1 Dataset....34
5.2 Results....34
6 CONCLUDING REMARKS....39
6.1 Conclusion....39
6.2 Future Work....40
REFERENCES....41

1 INTRODUCTION

Video cameras are widespread in vehicle surveillance systems, intelligent transportation systems and intelligent vehicles (KHAN; ULLAH, 2019; LLORCA; MARTÍNEZ; DAZA, 2021). For vehicle surveillance and intelligent transportation systems, important applications are license plate recognition (LPR), detection of stolen vehicles, monitoring of traffic conditions, toll processing and law enforcement. For intelligent vehicles, embedded cameras are crucial to monitor the surroundings of the vehicle, allowing the development of driver assistance systems or even autonomous driving. In particular, vehicle speed estimation can be very useful for many purposes, such as law enforcement, collision anticipation and traffic forecast in intelligent traffic systems.

The cameras used on traffic-related applications can either be static or mobile, and might be calibrated or not, and having a calibrated camera allows us to explore more aspects of the image and gather more information from the scene. There are several methods for camera calibration, either for full calibration (intrinsics and extrinsics) (ZHANG, 2000) or only extrinsics (also called pose estimation) (SCHWEIGHOFER; PINZ, 2006). The core idea in camera calibration is to find a transformation that allows us to convert world coordinates into image pixels. This can be achieved using many methods, like using a synthetic calibration pattern to calibrate the camera (ZHANG, 2000) or to explore natural patterns such as shape/dynamics of pedestrians (FÜHR, 2017; TAKAHASHI et al., 2018), vehicles features (ZHANG et al., 2012; BHARDWAJ et al., 2018) or structural information provided by lane markings as done in (FUNG; YUNG; PANG, 2003; PAULA; JUNG; JR, 2014).

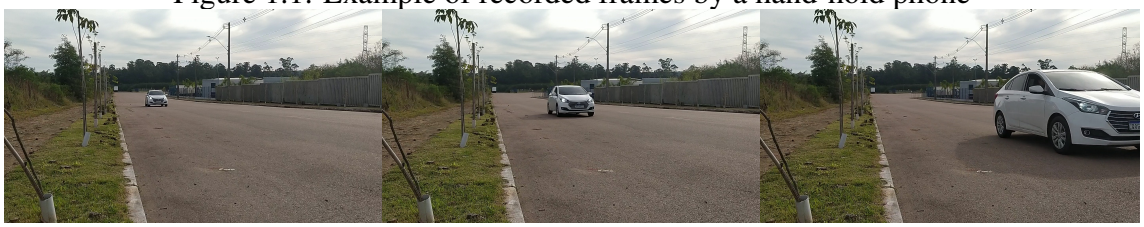
One common characteristic of traffic-related applications is the presence of license plates (LP), which are used to identify each vehicle and are adopted worldwide. In most countries, the LP dimensions of are standardized, which provides us a natural planar calibration pattern. However, using LPs as calibration patterns can also present some challenges, such as only having four correspondence points to calibrate the camera, which tends to introduce errors on the estimated pose. Also, detecting LPs in arbitrary camera setups is still a challenging task, but recent methods have shown promising results (SILVA; JUNG, 2021).

This work presents an approach for vehicle speed estimation using a “semi-static” camera, which means that the camera should have none or a negligible translation (in the order of a very few centimeters), but can have some rotational movement, like a person

standing and hand-holding a phone. Figure 1.1 represents a video in this setup. The core idea is to apply a vehicle detector combined with a flexible license plate detector (SILVA; JUNG, 2021) that provides the four corners of each detected LP, and estimate the relative camera pose assuming that the intrinsic parameters are known. When video sequences are available, we can track the pose of the LPs during a short period of time. In particular, we tackle the problem of estimating the speed of a vehicle,assuming the vehicle speed is constant and the vehicle has a linear trajectory when captured by a semi-static camera, which presents several potential applications.

The following chapters are organized as follows: Chapter 2 discusses the technical background relevant to our work. Chapter 3 explores related works in the literature and discusses similar approaches to the one proposed in this work. Chapter 4 introduces the proposed method for estimating the speed of vehicles. The created dataset and the results of the proposed approach are presented in Chapter 5. Finally, in Chapter 6 we present our conclusions and ideas for future work.

Figure 1.1: Example of recorded frames by a hand-hold phone



Source: Authors

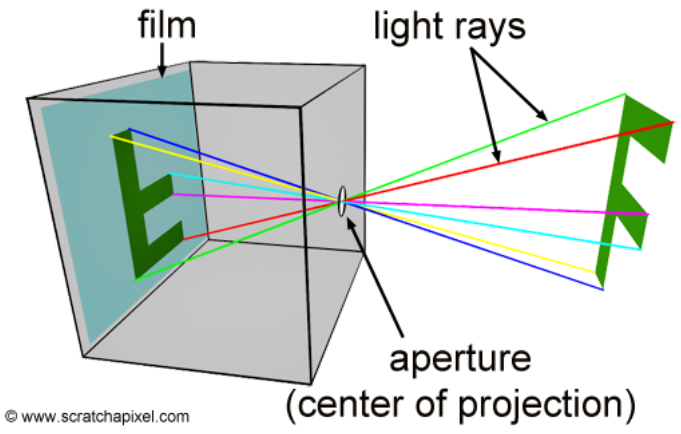
2 BACKGROUND

In this chapter, we explain the foundations needed to understand this work. Section 2.1 explains the pinhole camera model used throughout this project. Section 2.2 describes the basic idea of perspective projection, while Section 2.3 focuses on how homography matrix estimation can be performed. Section 2.4 briefly explains the RANSAC method, that is used as basis for one of our method to estimate the vehicle speed. Finally, Section 2.5 provides an overview about camera intrinsic parameters estimation.

2.1 Pinhole Camera Model

The pinhole camera model is composed by a black box, a film and a small aperture ("pin hole") on one side of the box. The box should not allow any light inside, besides the small aperture (ANDREW, 2001). The light that comes inside the box from the aperture hits the film directly, generating an inverted image as shown in Figure 2.1.

Figure 2.1: Pinhole camera



Source: (Scratchapixel, 2021)

In digital cameras, we use a lens instead of a pinhole to focus light into a semiconductor device (sensor) that records light electronically, then we need to compute this information to convert it into a digital image. The relationship between the 3D world coordinates and the corresponding pixel in the image are based on projections, which are explained in the next section.

2.2 Perspective Projection

The perspective projection process allows us to take a world coordinate and transform it into the corresponding image pixel, assuming a pinhole camera model. This is a three-step process (ANDREW, 2001).

The first step is coordinate transformation, in which we transform the coordinates from the world coordinate system into the camera coordinate system. Then, we project this new transformed coordinate into a plane that is orthogonal to the z axis of the camera. Finally, we discretize this information into pixels in order to obtain the final image. All these processes are explained in more detail in the following subsections.

2.2.1 Camera Transformations

Let us consider a world coordinate point $\boldsymbol{x_w} = [x_w, y_w, z_w]^T \in \mathbb{R}^3$, and a camera placed at position $\boldsymbol{x_0} = [x_0, y_0, z_0]^T \in \mathbb{R}^3$ which is rotated by a matrix R, as shown in Figure 2.2. The mapping of $\boldsymbol{x_w}$ onto the 3D camera coordinate system is given by

$$\boldsymbol{x_c} = R(\boldsymbol{x_w} - \boldsymbol{x_0}), \tag{2.1}$$

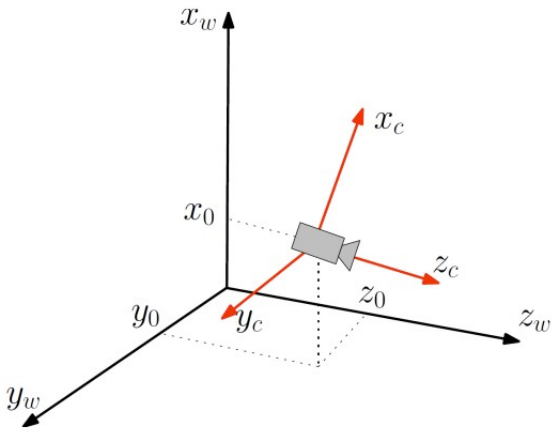
which is a combination of translation and rotation. To represent this joint transformation as a linear one, we use homogeneous coordinates (denoted by superscript h), as provided next:

$$\boldsymbol{x_c^h} = \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_c} \\ 1 \end{bmatrix}, \tag{2.2}$$

$$\boldsymbol{x_w^h} = \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \begin{bmatrix} \boldsymbol{x_w} \\ 1 \end{bmatrix}, \tag{2.3}$$

$$\boldsymbol{x_c^h} = \begin{bmatrix} R & -R\boldsymbol{x_0} \\ \mathbf{0}_{1\times 3} & 1 \end{bmatrix} \boldsymbol{x_w^h} = \begin{bmatrix} \boldsymbol{x_c} \\ 1 \end{bmatrix}. \tag{2.4}$$

Figure 2.2: Camera coordinate system



Source: Authors

2.2.2 3D to 2D Projection

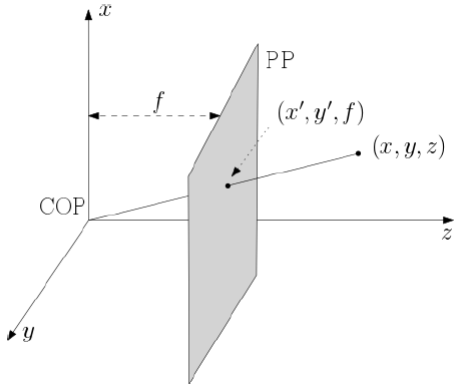
Let us consider a projection plane PP orthogonal to the z axis placed at $z = f$, in which we want to project $\boldsymbol{x_c^h}$. We need to find the intersection between the ray from $\boldsymbol{x_c^h}$ to the Center of Projection (COP) and PP. We can calculate the projection of $\boldsymbol{x_c^h}$ onto PP using triangle similarities, obtaining the projected point $[f\frac{x}{z}, f\frac{y}{z}, f]^T$. Since the last coordinate is fixed at $z = f$, a projection point can be characterized simply as $[x', y']^T = [f\frac{x}{z}, f\frac{y}{z}]^T$. Using homogeneous coordinates, we can define the projection matrix as:

$$\begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix}. \tag{2.5}$$

2.2.3 Pixel Mapping

Now we must discretize the spatial domain (image pixels) so we can map the projection plane to pixel coordinates. We map the COP to the image pixel with coordinates $[u_c, v_c]^T$, and with the s_u and s_v being the densities (pixels/linear measure), we map

Figure 2.3: 2D projection



Source: Authors

$[f\frac{x}{z}, f\frac{y}{z}]^T$ into a pixel coordinate $[u, v]^T$ through:

$$u = u_c + s_u \left(\frac{fx}{z} \right), \tag{2.6}$$

$$v = v_c + s_v \left(\frac{fy}{z} \right). \tag{2.7}$$

This process can also be written in terms of homogeneous coordinates:

$$\begin{bmatrix} u_s \\ v_s \\ c \end{bmatrix} = \begin{bmatrix} s_u & 0 & u_c \\ 0 & s_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ \frac{z}{f} \end{bmatrix} \Rightarrow \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \frac{u_s}{c} \\ \frac{v_s}{c} \end{bmatrix}. \tag{2.8}$$

2.2.4 Final Projection equation

Since all the transformations described so far can be represented by matrix multiplications, it is easy to find the final projection equation:

$$\boldsymbol{u^h} = \begin{bmatrix} s_u & 0 & u_c \\ 0 & s_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & -R\boldsymbol{x_0} \\ 0_{1\times 3} & 1 \end{bmatrix} \boldsymbol{x_w^h}, \tag{2.9}$$

then we can conclude that:

$$\boldsymbol{u^h} = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & -R\boldsymbol{x_0} \\ 0_{1\times 3} & 1 \end{bmatrix} \boldsymbol{x_w^h}, \tag{2.10}$$

where $f_u = fs_u$ and $f_v = fs_v$ are the focal lengths given in pixels. The first matrix represents the intrinsic parameters of a camera:

$$K = \begin{bmatrix} f_u & 0 & u_c \\ 0 & f_v & v_c \\ 0 & 0 & 1 \end{bmatrix}, \tag{2.11}$$

and, if we explicitly compute the multiplication of the other two matrices, we obtain

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & -R\boldsymbol{x_0} \\ 0_{1\times 3} & 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{21} & r_{31} & t_1 \\ r_{12} & r_{22} & r_{32} & t_2 \\ r_{13} & r_{23} & r_{33} & t_3 \end{bmatrix} = [R|\boldsymbol{T}], \tag{2.12}$$

which relates to the extrinsic camera parameters, in which R represents the camera rotation matrix and $\boldsymbol{T}$ represents the translation vector, or the 3D camera pose. Now, we can describe the process of mapping the coordinate $\boldsymbol{x_w}$ to a pixel coordinate ($\boldsymbol{u}$) for a given camera, with K as the intrinsic parameters, R as the matrix that describes the camera rotation, and $\boldsymbol{T}$ as the coordinates of the camera position, as:

$$\boldsymbol{u^h} = K[R|\boldsymbol{T}]\boldsymbol{x_w^h} = A\boldsymbol{x_w^h}. \tag{2.13}$$

2.3 Estimation of the planar homography matrix

Considering we are using a pinhole camera model, we can calculate the projection of any 3D coordinate to a pixel. Let $\boldsymbol{x}^h = [x, y, z, 1]^T$ and $\boldsymbol{u}^h = [u, v, 1]^T$ be the world homogeneous coordinate and the image pixels respectively. Let K be the intrinsic parameter matrix, $R = [\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}]$ be the rotation matrix, in which $\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}$ are the rotation vectors, and $\boldsymbol{T}$ be the translation vector that relates the world and image coordinate

system, we have:

$$A\boldsymbol{x}^h = s\boldsymbol{u}^h, \tag{2.14}$$

where

$$A = \begin{bmatrix} h_1 & h_4 & h_7 & h_{10} \\ h_2 & h_5 & h_8 & h_{11} \\ h_3 & h_6 & h_9 & h_{12} \end{bmatrix} = K \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_3 & \boldsymbol{T} \end{bmatrix}, \tag{2.15}$$

and s is a scaling factor.

If we consider that all of the 3D world points lie on the same plane, we can assume that $z = 0$ (with a suitable rotation matrix) and therefore our equations to map the world to the image point would be the following:

$$H\hat{\boldsymbol{x}}^h = s\boldsymbol{u}^h, \tag{2.16}$$

where

$$H = \begin{bmatrix} h_1 & h_4 & h_7 \\ h_2 & h_5 & h_8 \\ h_3 & h_6 & h_9 \end{bmatrix} = K \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{T} \end{bmatrix}, \tag{2.17}$$

and $\hat{\boldsymbol{x}}^h$ is similar to $\boldsymbol{x}^h$ but with the z coordinate discarded. H is the planar homography matrix, so it maps the world coordinate $[x, y, 0]^T$ to the image pixel $[u, v]^T$. If K is known (i.e., if the camera is calibrated), we can write

$$H\hat{\boldsymbol{x}}^h = s\boldsymbol{u}^h \Rightarrow K^{-1}K \begin{bmatrix} \boldsymbol{r_1} & \boldsymbol{r_2} & \boldsymbol{T} \end{bmatrix} \hat{\boldsymbol{x}}^h, \tag{2.18}$$

then, finally, we can conclude that:

$$\begin{bmatrix} \boldsymbol{r_1} & \boldsymbol{r_2} & \boldsymbol{T} \end{bmatrix} \hat{\boldsymbol{x}}^h = sK^{-1}\boldsymbol{u}^h = \begin{bmatrix} u^s \\ v^s \\ s \end{bmatrix}. \tag{2.19}$$

If we have a set of n correspondences $(\boldsymbol{x_i}, \boldsymbol{u_i})$ for $i = 1, ..., n$, we can write

$$\begin{bmatrix} r_{11} & r_{21} & T_1 \\ r_{12} & r_{22} & T_2 \\ r_{13} & r_{23} & T_3 \end{bmatrix} \hat{\boldsymbol{x}}_{\boldsymbol{i}}^{\boldsymbol{h}} \equiv \begin{bmatrix} u_i^s \\ v_i^s \\ s \end{bmatrix}, \tag{2.20}$$

where $\equiv$ denotes the equivalence of homogeneous coordinates. Dividing both sides by the third coordinate (to map homogeneous coordinates to 2D points) leads to the following two equations:

$$\frac{u_i^s}{s} = \frac{r_{12}x_i + r_{21}y_i + T_1}{r_{13}x_i + r_{23}y_i + T_3}, \tag{2.21}$$

$$\frac{v_i^s}{s} = \frac{r_{22}x_i + r_{22}y_i + T_2}{r_{13}x_i + r_{23}y_i + T_3}. \tag{2.22}$$

Combining Eqs. (2.21) and (2.22) and rewriting them in matrix form yields

$$M_i \boldsymbol{h} = 0, \tag{2.23}$$

where

$$M_i = \begin{bmatrix} -x_i & -y_i & -1 & 0 & 0 & 0 & u_i^s x_i & u_i^s y_i & u_i^s \\ 0 & 0 & 0 & -x_i & -y_i & -1 & v_i^s x_i & v_i^s y_i & v_i^s \end{bmatrix}, \tag{2.24}$$

$$\boldsymbol{h} = \begin{bmatrix} r_{11} & r_{21} & T_1 & r_{12} & r_{22} & T_2 & r_{13} & r_{23} & T_3 \end{bmatrix}^T. \tag{2.25}$$

We can obtain a $2n \times 9$ linear system by stacking matrices M_i in a row-wise manner. Since $\boldsymbol{h}$ has 8 degrees of freedom and each mapped point ($n \geq 4$) leads to two equations, we will require at least four mapped points to solve it. To avoid the trivial solution, we apply the singular value decomposition (SVD) and retrieve the least singular vector, as done in (DUBROFSKY, 2009). Also, note that such approach might generate vectors $\boldsymbol{r_1}$ and $\boldsymbol{r_2}$ that do not have unit norms, and hence a normalization procedure is applied after solving the system.

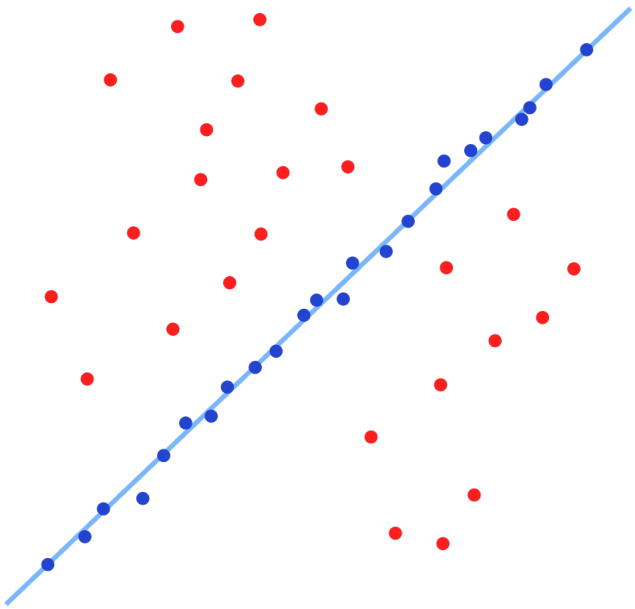
2.4 RANSAC Algorithm

The Random Sample Consensus (RANSAC) algorithm was proposed in (FISCHLER; BOLLES, 1981). It was designed to estimate the parameters of a mathematical model from a set of data that may contain outliers, where it tries to minimize the effect of those outliers on the final estimated solution.

The general idea for the algorithm is to first randomly sample a subset containing the smallest number of elements sufficient to determine the model parameters. Then, in a second step, the algorithm checks whether the other elements are considered outliers or if they fit the model, based on the computation of a suitable error and a preset threshold

value. At the end of this step we, have the set of inlier elements, outlier elements and the estimated model. The algorithm will iteratively repeat the two steps until the estimated model has enough inlier elements. In Figure 2.4 an example of line-fitting using RANSAC, where blue points were identified as inliers, red points as outliers, and the blue line is the fitted model. Although this is just a toy example, it provides the main concepts behind RANSAC, and will be used later for vehicle speed estimation through line-fitting.

Figure 2.4: RANSAC estimated model



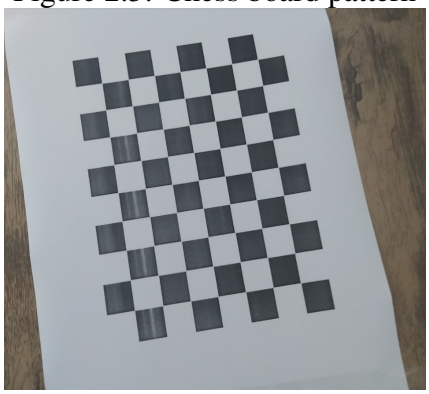
Source: (Wikipedia contributors, 2021)

2.5 Estimation of camera intrinsics

To extract the intrinsic parameters for a given camera, we need a set of 3D real world points and the corresponding 2D coordinates of these points in the image. A chessboard pattern, as shown in Figure 2.5, can be used to generate such set of 3D and 2D points. We take some sample images of this pattern from different positions and angles and, based on the correspondence on these images for the 3D and 2D points, it is possible to estimate the intrinsic parameters for the camera Zhang (2000).

For the 2D set of points we can use the internal corners of squares, simply locating where two black squares touch each other in the chess board. For the 3D set of points we can assume a fixed position (x, y) for the chessboard and $z = 0$, then, if we take different

Figure 2.5: Chess board pattern



Source: Authors

pictures of the chess board by moving the camera around it, the internal corners can be tracked as $(0,0),(1,0),(1,1),\ldots$ which denotes the location of points. After we map those 2D and 3D sets of points for each frame, we use an algorithm based on the different angles of each image and, based on how the intrinsic parameter matrix is the same for all images, we use a mathematical model to estimate it. This approach is better described on (ZHANG, 2000).

3 RELATED WORK

In this chapter, we discuss some existing object detection and speed estimation methods. In Section 3.1.1 we briefly elaborate about generic object detection, which is used in our pipeline for vehicle detection. In Section 3.1.2 we focus on LP detection, in particular, we describe a state-of-the-art LP detector that produces a quadrilateral representation of the LP that will be used along this work. In Section 3.2 we discuss vehicle speed estimation methods and two other works that have a setup similar to the one this work proposes and their limitations.

3.1 Object detection

In Section 3.1.1 we briefly review some generic object detectors, and detail a state-of-the-art approach that is used in our pipeline for vehicle detection. In Section 3.1.2, we focus on LP detection. In particular, we describe a state-of-the-art LP detector that produces a quadrilateral representation of the LP that will be used along this work.

3.1.1 Generic Object Detection

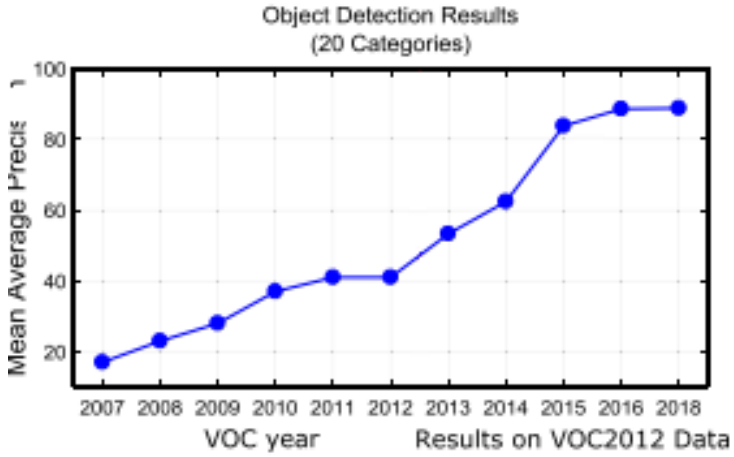
Object detection is a very important task in computer vision. Detecting instances of humans, animals, cars or even smaller objects, like suitcases or balls is lately being widely used in many real-word applications, like autonomous driving (BACH; STUMPER; DIETMAYER, 2018) and video surveillance (JIN; LI; KIM, 2017). As mentioned in Zou et al. (2019), object detection is the basis for many other computer vision tasks, like segmentation, image captioning, object tracking and others.

Usually there are two types of study groups for object detection (LIU et al., 2020). The first is general object detection, in which the goal is to be able to detect many types or classes of objects (people, cars, bikes, animals, etc) in the same framework. The other one is specific object detection, in which the goal is to identify a small number of classes (or maybe a single class), under a specific scenario.

As mentioned on (LIU et al., 2020), the advance of deep learning (DL) techniques in the last two decades has provided a significant improvement of object detection performance, as shown in Figure 3.1. The usage of DL allowed the models to learn complex,

subtle and abstract representations, improving the performance of a large range of problems such as visual recognition, object detection, speech recognition among others.

Figure 3.1: Average object detection performance per year



Source: (LIU et al., 2020)

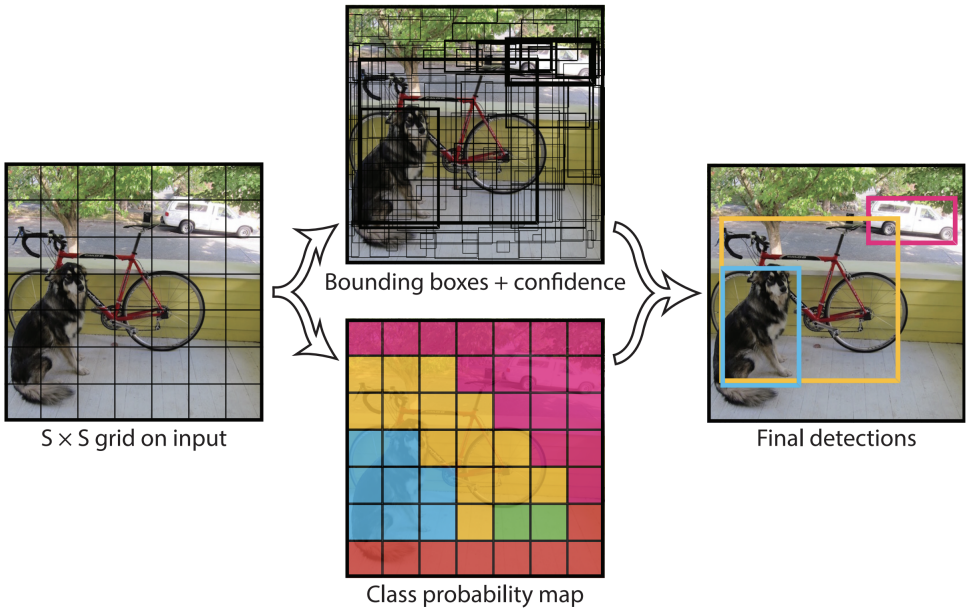
Among the several existing possibilities, we briefly revise the YOLO (You Only Look Once) family of object detection methods. The first version (REDMON et al., 2016) introduced a novel Convolutional Neural Network (CNN) architecture for generic object classification and detection to predict the bounding boxes and the class probabilities of those boxes in one evaluation (single-stage), as opposed to prior methods that first find potential bounding boxes (proposals) and then run a classifier on those boxes. The algorithm proposes to divide the input image into a square grid of size $S \times S$, and, for each grid cell, it predicts the bounding boxes, its confidence scores and the class probabilities. Subsequently, they calculate the class-specific score for each predicted box. Figure 3.2 shows how the model splits the image and does object detection in more details.

In this work, we use an improved implementation of YOLO proposed (YOLO v4) in (BOCHKOVSKIY; WANG; LIAO, 2020). Some of the improvements made to the original implementation were the replacement of the backbone and the improvement of the accuracy and performance of the network. More details can be found in (BOCHKOVSKIY; WANG; LIAO, 2020).

3.1.2 License Plate Detection

License Plate Detection (LPD) is a particular case of object detection, which has the goal of extracting vehicle LP information from an image. This can be useful for

Figure 3.2: YOLO method model



Source: (REDMON et al., 2016)

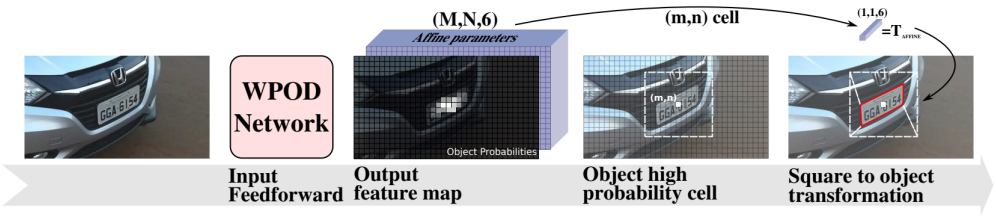
many applications, such as toll and park collection, road traffic surveillance and traffic law enforcement.

According to the survey article by Shashirangana et al. (2020), there are a few approaches for detecting the LP, such as edge-based, color-based, texture-based and the trending technique DL approaches. For the edge-based approaches, the idea is to assume that the LP is a rectangular object and explore known aspect ratios to detect it (LUO et al., 2009; SANYUAN; MINGLI; XIUZI, 2004). Color-based approaches rely on the difference between the LP and the vehicle color (YOHIMORI et al., 2004; JIA et al., 2005). Texture-based approaches, explore the presence of characters on the plate as a basis for the LPD (XU et al., 2005; CANER; GECIM; ALKAR, 2008). DL-based approaches, which are replacing most of the other LPD methods due to their high accuracy in Object detection, the idea is to train a model to learn how to detect a LP (LAROCA et al., 2018; SILVA; JUNG, 2021).

The success of state-of-the-art object detectors inspired many works consisting of training those object detectors, such as YOLO, to perform LPD. Laroca et al. (2018) proposed a method based on YOLO (REDMON et al., 2016) to extract the bounding box of LPs. But, for the approach proposed in this work, only a bounding box would not be enough, we would need the LP corners to be able to track it in order to estimate the vehicle speed. So we focus our analysis on Silva and Jung (2021), where they propose a model able to extract the corners of a LP.

Silva and Jung (2021) proposed a method to extract LP regions by providing a quadrilateral representation of the LPs. Their method consists of three steps: the first one is vehicle detection, which can be achieved by any generic object detector trained with vehicles. The authors used a detector based on YOLOv2, but in this work we explore YOLOv4, which showed better results for generic object detection. In the second step, each detected vehicle is fed to another network that identifies the presence of an LP, and returns a quadrilateral representation. They proposed a novel CNN called Improved Warped Planar Object Detection Network (IWPOD-NET), which takes the cropped image of the vehicle from the first step and extracts the corners of the LP and the affine matrix that allows us to rectify the LP into a horizontally and vertically aligned object. The process to extract the affine matrix and corners is illustrated in Figure 3.3. The third and final step is LP rectification and character recognition (which is not the focus of our work).

Figure 3.3: IWPOD-NET process



Source: (SILVA; JUNG, 2021)

3.2 Vehicle Speed Estimation

Estimating the vehicle speed can help improve traffic, enforce traffic laws, etc. As mention on Luvizon, Nassu and Minetto (2016) vehicle speed estimation systems are divided into two groups: intrusive and non-intrusive. Intrusive systems are usually based on inductive loop detectors, but they have complex installation and maintenance. Non-intrusive systems usually include laser meters and Doppler radars, which do not require installation, but have a higher cost and frequent maintenance. With the recent advances in computer vision techniques, many works put forward the usage of computer vision as a cheap form of non-intrusive system to estimate vehicle speed.

According to a recent survey (LLORCA; MARTÍNEZ; DAZA, 2021), there are few strategies to estimate a vehicle speed based on computer vision. The strategies are

spread into some categories, like camera calibration, camera setting, vehicle detection and tracking.

An accurate estimation of the intrinsic and extrinsic parameters is important to provide a good estimation of world coordinates. There are different types of camera calibration setups. They can be soft, meaning that the camera intrinsic parameters are estimated in a lab and the extrinsic are estimated manually (HUA; KAPOOR; ANASTASIU, 2018; BOUZIADY et al., 2018) or using an automated procedure (FILIPIAK; GOLENKO; DOLEGA, 2016; BHARDWAJ et al., 2018). They can also be a hard setup, in which the camera is installed and then the intrinsic and extrinsic parameters are jointly estimated (QIMIN et al., 2014; BARTL; HEROUT, 2019). There are also a few works in which the camera calibration is somehow neglected (LIN; LI, 2004; JING-ZHONG; XIAOQING, 2009).

The camera setting between works can be different: it can explore a fixed camera (YABO et al., 2016; FILIPIAK; GOLENKO; DOLEGA, 2016) or a moving one, such as attached to a drone (YAMAZAKI; LIU; VU, 2008; BASTOS et al., 2020). The camera distance to the road is also an important factor to classify works about speed estimation: the cameras can be classified as close (LUVIZON; NASSU; MINETTO, 2016; LIN; LI, 2004) or distant (YABO et al., 2016; CHENG et al., 2020).

How the vehicle is tracked also varies between existing approaches, and we can identify four groups; i) feature tracking (HUA; KAPOOR; ANASTASIU, 2018; BOUZIADY et al., 2018); ii) tracking the centroid of the detected vehicle (YABO et al., 2016); iii) tracking the entire region of the vehicle (HUA; KAPOOR; ANASTASIU, 2018); iv) tracking of the LP (FILIPIAK; GOLENKO; DOLEGA, 2016; LUVIZON; NASSU; MINETTO, 2016).

Our proposed technique can be classified as requiring soft camera calibration, fixed camera close to the road and vehicle tracking based on the LP tracking. In the following paragraphs we detail some techniques with similar setups as the one proposed in this work.

Famouri, Azimifar and Wong (2018) proposed a method to estimate the speed of a vehicle. They consider the center of the LP as the vehicle reference, and manually estimate the hypothetical plane (named motion plane) in which the LP moves. Then they map the LP position to the motion plane and estimate the distance over time and estimate the vehicle speed. A limitation of this technique is the necessity to have the manual step of determining the motion plane.

Luvizon, Nassu and Minetto (2016) proposed a method to estimate the speed of a vehicle based on a video input of the road. The video is captured by a fixed camera positioned above the road in a way that the rear and LP of the vehicle are clearly visible. The speed is calculated by tracking a few features of the vehicle over time. One limitation of this method is the necessity for the camera to be on a specific position for this method to work.

In this section we discussed how object detection can be used to extract the LP from vehicles. We also discussed two methods to estimate vehicle speed, both requiring some sort of camera setup knowledge. In the first one, the camera not only needs to be fixed, but we also need to do a manual step of estimating the motion plane. In the second approach, the camera needs to be fixed above the road and have a clear view of the LP and rear of the vehicle. In the next chapter, we propose a method to estimate the speed of a vehicle based on the LP pose estimation. The only setup we need is the camera to be calibrated (intrinsic parameter known) and the camera can be hand-held as long we can guarantee a negligible translation.

4 THE PROPOSED APPROACH

In this chapter, we describe the proposed pipeline to estimate the vehicle speed. In Section 4.1, we give a small overview of the proposed methods. In Section 4.2 we explain how we detect the vehicle and the LP, how we extract the corners of the LP, and how we can estimate the camera pose in relation to the LP corner using the homography matrix generated by the LP. In Section 4.3, we explain how we can track the vehicle from the estimation of camera position. In Section 4.4, we explain the proposed methods to estimate the vehicle speed after tracking it on the previous step.

4.1 Overview

We assume that the scene is captured by a given camera with known intrinsic parameters and negligible translation (either fixed or hand-held, possibly with small rotations). The main goal is to estimate the vehicle pose from a sequence of images and then estimate the speed of that vehicle. This sequence of images (frames) can either be a video or a burst of photos, as long as we can extract the correct timestamp from each frame, in this work we only used videos.

To estimate the speed of the vehicle, we need to track a specific point of the vehicle over time. As the LP is attached to the vehicle, we only need to track it instead of tracking the whole vehicle. If we are able to estimate the speed of the LP we can estimate the speed of the vehicle, as they are the same.

If we take the position of the LP and the timestamp for each frame, in theory, only two frames can determine the speed of the vehicle in that interval. But, in a real scenario, there are inaccuracies when estimating the pose of the vehicle, and using more frames tends to produce better speed estimates. Let A be the set of frames from which we can extract the LP of the vehicle, characterized by the image coordinates (pixels) of the corners of the LP. After we have those pixels, we can use the inverse of the intrinsics K^{-1} and estimate $\boldsymbol{r_1}$, $\boldsymbol{r_2}$ and $\boldsymbol{T}$, as described in Section 2.3. Then we can map pixel coordinates to the correspondent world coordinates. Then, we generate a set F containing tuples (t, x, y, z), where t represents the timestamp for that frame, and (x, y, z) represents the world coordinates for the left-top corner of the LP– this corner was chosen arbitrarily. Once we generate F, we explore different approaches to estimate the speed of the vehicle, and we demonstrate the results in the Chapter 5.

4.2 Vehicle and License plate detection

The first step in our approach is to extract the image pixels of the LP corners. As explained in Section 3.1.2, Silva and Jung (2021) proposed a CNN (IWPOD-NET) capable of extracting the four corners of a LP. Even though their main goal was to rectify the LP, we think we could leverage this functionality of IWPOD-NET and explore other goals, like calculating the speed of the vehicle. We used the pre-trained weights provided by the authors in <https://github.com/claudiojung/iwpod-net>. According to the authors, the model was trained with LPs from several countries/regions (Brazilian, Taiwanese, Chinese).

As mentioned in their paper, we need to pass as input for the IWPOD-NET a cropped image of the vehicle. So, for each frame we read, we first need to detect if there is a vehicle on that frame and, if so, generate a bounding box to crop that image selecting only the vehicle as the result.

As vehicles are common objects present in many datasets (such as COCO (LIN et al., 2014) for example), we decided not to train a model from scratch but to use a known model to perform the detection. More accurately, we used YOLO v4, an implementation of (BOCHKOVSKIY; WANG; LIAO, 2020), and adjusted a few parameters regarding the desired threshold and the classes we wanted the algorithm to detect– in this case we only want to detect vehicles.

The input of the model is the entire frame, and the output is the set of detected vehicles along with the corresponding bounding boxes. We crop the images based on the bounding boxes, leaving us with a smaller image containing mainly the vehicle in it. Then we take this cropped image and use it as the input image for IWPOD-NET, which returns the pixels of the LP corners. Since these coordinates are based on the cropped image, we need to translate them to the relative pixel on the original image. In Figure 4.1 we have a visual demonstration of our pipeline in the LP detection process.

4.3 Pose estimation

After we detect the corners of the LP in the image, we can map those points to the world coordinate system. In this work, we are only dealing with Brazilian LPs. As defined in (DIÁRIO OFICIAL DA UNIÃO, 2019), the dimensions for this LP are 400mm width and 130mm height. Assuming the left-top corner of the LP as the origin $(0, 0)$ of

Figure 4.1: LP detection pipeline

Source: Authors

our planar world coordinate system, the coordinates of the other corners follow directly from LP dimensions: $(0.4, 0), (0.4, 0.13), (0, 0.13)$, values are given in meters, as shown in Figure 4.2.

Given that matrix K is known and we have the four coordinates of the LP corners (as well as their mapped pixels), we use Eq. (2.23) and the approach described in the Section 2.3 to extract the values for $\boldsymbol{r_1}$, $\boldsymbol{r_2}$ and $\boldsymbol{T}$.

Figure 4.2: LP coordinates

Source: Adapted from (Carlos Teixeira, 2020)

As we know, $\boldsymbol{T}$ is the camera position in relation to the top-left corner of the LP. If we assume the camera position as the origin of the world coordinate system, then we could assume that $\boldsymbol{T}$ is actually the position of the top-left corner of the LP relative to the

camera position.

In this work, we assume that the camera should have a negligible translation, therefore we can assume that the camera position is the same over all frames. If we estimate $\boldsymbol{T}$ for all those frames, we have the position of the top-left corner in relation to the camera position for all frames. In Figure 4.3, we show the tracking of the top-left corner (red dot) on multiple frames.

Figure 4.3: LP corner tracking (generated by multiple frames)

Source: Authors

We convert each frame for which we were able to detect the LP corners into a tuple (t, x, y, z), where t represents the timestamp for that frame and (x, y, z) represents the world coordinates for the left-top corner of the LP, and create a set F that consists of all those generated tuples. Using this information, we can track the top-left LP corner position over time and estimate the speed of the vehicle, as explained in the next section.

4.4 Speed computation

Having the position of the LP corner over time enables us to calculate the speed of the LP, and, therefore, the speed of the vehicle. In the end of the step described in Section 4.3, we generate a set of tuples F containing the time and position of the LP corner in each detected frame. Ideally, we should be able to just select two of those tuples and calculate the vehicle speed. But due to some inaccuracy on the estimated positions of the LP corner, this is not enough. In Figure 4.4, we can see that the estimated corner does not match the actual corner of the LP, this is one of the reasons why we need a more robust method to estimate the vehicle speed. In this section we explore methods to use more than two tuples to mitigate the inaccuracy and generate a more precise estimation of the vehicle speed, assuming a constant speed during the tracked period.

Figure 4.4: LP corner estimation inaccuracy

Source: Authors

4.4.1 RANSAC-Based Method

This approach is based on the RANSAC method, explained in Section 2.4. The idea of this method is to take all of the values for the positions (x,y and z) and generate three sets, one for each coordinate. After we have a set for x, y and z, we apply the

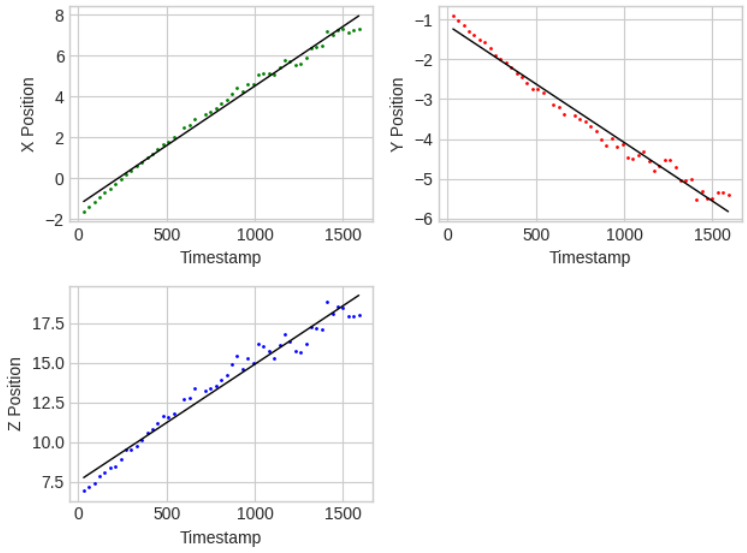
RANSAC algorithm to estimate three line equations that describe the position of each coordinate over time (assuming constant speed):

$$x = c_x t + b_x, \tag{4.1}$$

$$y = c_y t + b_y, \tag{4.2}$$

$$z = c_z t + b_z. \tag{4.3}$$

Figure 4.5: Fitted line for coordinates



Source: Authors

After we estimate the three equations independently (Figure 4.5 shows an example of the fitted line for each coordinate), we can estimate the position (P) of the LP corner for a given time as:

$$P(t) = (c_x t + b_x, \quad c_y t + b_y, \quad c_z t + b_z). \tag{4.4}$$

For a given t_1 and t_2, we can calculate the euclidean distance between $P(t_1)$ and $P(t_2)$, and divide this by the time elapsed between t_1 and t_2. This would give us the vehicle speed according to the estimated equations by our method and, therefore, the estimated speed by our method.

4.4.2 Combinational Set Metrics

In this approach, we take the set F and generate a new set F^p that is described by

$$F^p = \{S(a_i, a_j) | \forall (a_i, a_j) \in F^2 \text{ with } i > j\}, \tag{4.5}$$

where S is a function described in Eq. (4.6) that takes a pair of tuples (a, b) and calculates the vehicle speed related to them. The set F^p contains all pairwise speed estimates that can be obtained from the original set F.

$$S(a, b) = \frac{\sqrt{(a_x - b_x)^2 + (a_y - b_y)^2 + (a_z - b_z)^2}}{|a_t - b_t|} \tag{4.6}$$

The idea behind this strategy is to explore not only two adjacent tuples, but the combination of them all, having combinations with a small time window between them but also combinations that have a larger window. The size of F^p is the $\binom{n}{2}$, where n is the cardinality of F. Therefore, at the end of this step, we will have more data compared to just looking to the adjacent tuples.

After we generate the set F^p, we use a few metrics to estimate the final vehicle speed. The ones used were the mean (`Comb Mean`), median (`Comb Med`) and trimmed mean (`Comb Trim Mean`). The mean is a natural choice and should remove noisy estimates, but can be sensitive to outliers. The median provides an estimate that is robust to outliers, and the trimmed mean provides a combination of the mean and median: the median value is used to discard outliers, and the final estimate is the mean of the remaining values.

5 EXPERIMENTAL RESULTS

In this chapter, we demonstrate the performance of our methods over a dataset. In Section 5.1, we explain how and why we needed to create our own dataset to test the performance of our proposed methods. In Section 5.2, we demonstrate the results and analyze some of the metrics of our methods in order to identify the best candidate to estimate the vehicle speeds. In the tables below, all speeds are in Km/h (Kilometers per hour).

5.1 Dataset

Although the camera setup for this work is simple, we were not able to find any available dataset that would give us the necessary type of image and also provide the intrinsic camera parameters we need. Therefore, we decided to create a dataset from scratch. It consists of twenty-one videos of vehicles, taken as they were passing a speed trap that would show the actual vehicle speed. The ground truth (GT) speed for our dataset was the one detected by the speed trap. For example, in Figure 5.1 the detected speed is 43 Km/h.

The videos were taken from a handheld smartphone camera (Redmi Note 8) with a resolution of 1080×1920 pixels (Full HD) at 30 FPS. The person holding the camera was always standing up on the sidewalk and was about twenty meters away from the speed trap, as shown in Figure 5.1.

Because we were not able to find a speed trap that would show us the detected speed and also have a speed limit higher than 50, the speeds on our dataset were on a range of $[26, 46]$ Km/h. The videos were taken from two speed traps, both with a speed limit of 50 Km/h. The two sites where the videos were taken are shown in Figure 5.2. We used the method described in Section 2.5 to calibrate the camera, i.e., estimate the intrinsic parameters.

5.2 Results

We tested our methods against the created dataset. For each video we had the GT, and we used those values only for comparison. In Table 5.1 we show the estimated speed

Figure 5.1: Dataset video frame (cropped for better visualization)

Source: Authors

Figure 5.2: Videos site

Source: Authors

for each method and video. The detected speed column represents the GT for that video, the `Comb Mean`, `Comb Med` and `Comb Trim Mean`, represent estimated speeds by the combination mean, combination median and combination trimmed mean respectively, as explained in Sections 4.4.1 and 4.4.2. The best result for each video is highlighted in bold.

We consider two error metrics to evaluate the proposed methods: the maximum and average of the relative error. In Table 5.2, we present the relative errors, in percentage, for each video and proposed method. In the last two rows we present the maximum and average of the observed errors. The best result for each video is also highlighted.

Table 5.1: GT speed and estimated values using different methods

Video	Detected Speed	Comb Mean	Comb Med	Comb Trim Mean	RANSAC
video1	26	30.93	30.93	30.88	**30.56**
video2	30	47.78	40.07	43.01	**28.01**
video3	31	39.55	39.33	38.85	**36.95**
video4	32	34.58	31.88	**32.04**	30.20
video5	32	36.49	35.20	34.86	**34.40**
video6	33	38.27	37.52	37.45	**37.25**
video7	34	**31.91**	26.24	27.29	25.67
video8	34	84.89	39.85	50.12	**38.68**
video9	35	38.13	**37.36**	37.62	37.41
video10	35	**35.60**	33.40	33.22	32.52
video11	35	38.49	38.72	38.20	**37.27**
video12	36	45.86	45.10	45.16	**44.85**
video13	36	38.71	39.21	38.27	**36.73**
video14	36	40.71	37.87	37.89	**35.03**
video15	38	**36.67**	35.53	35.72	34.32
video16	38	42.35	42.02	41.82	**41.08**
video17	39	42.21	42.51	42.26	**40.96**
video18	43	53.84	51.67	51.78	**51.28**
video19	44	**43.02**	39.51	39.75	37.60
video20	44	101.54	56.50	65.03	**52.62**
video21	46	54.13	51.11	50.12	**45.76**

Source: Authors

In Table 5.3 we present the percentage of the estimates that were within a range of relative errors. We can see that the relative errors for the RANSAC method were spread throughout the first four ranges, and none was placed on a different range. When compared with the combination mean, we can see that even though the difference of average between the two approaches is just 3.17% the first approach has a higher number of samples in the first two ranges (0 to 10%), totaling 57.15% against 38.09% of the latter. Both combinational mean and trimmed mean performed poorly showing results in the 6^{th} and 7^{th} ranges, that were admitting the largest errors.

In Figure 5.3 we analyze the absolute error count (in Km/h) for the estimated speeds in ranges. We notice that all the proposed methods tend to estimate speeds higher than the actual vehicle speed. For all of their estimates, at least 75% of them were higher than the GT. According to Luvizon, Nassu and Minetto (2016), the USA standards for an acceptable measurement method must be within the range of $[-3\text{Km/h}, +2\text{Km/h}]$. The RANSAC approach was able to estimate a third of the samples (33.33%) within the range of $[-3\text{Km/h}, +2\text{Km/h}]$ of the actual speed, but it also presented an estimation that was 8.33 (Km/h) lower than the GT, which was the highest negative difference between all tested methods.

Table 5.2: Methods relative error (in%)

Video	Comb Mean	Comb Med	Comb Trim Mean	RANSAC
video1	18.98	18.97	18.76	**17.54**
video2	59.27	33.58	43.38	**6.63**
video3	27.57	26.86	25.32	**19.19**
video4	8.07	0.36	**0.11**	5.63
video5	14.04	9.99	8.93	**7.50**
video6	15.97	13.69	13.47	**12.88**
video7	**6.15**	22.84	19.73	24.50
video8	149.67	17.20	47.41	**13.76**
video9	8.95	**6.75**	7.48	6.89
video10	**1.71**	4.56	5.08	7.09
video11	9.97	10.63	9.13	**6.49**
video12	27.38	25.27	25.45	**24.58**
video13	7.54	8.93	6.30	**2.03**
video14	13.07	5.19	5.25	**2.69**
video15	**3.50**	6.50	6.00	9.68
video16	11.45	10.59	10.04	**8.11**
video17	8.24	9.00	8.37	**5.03**
video18	25.21	20.17	20.41	**19.26**
video19	**2.22**	10.20	9.66	14.55
video20	130.78	28.41	47.80	**19.59**
video21	17.68	11.10	8.95	**0.52**
Average	27.02	14.32	16.53	**11.15**
Max	149.67	33.58	47.80	**24.58**

Source: Authors

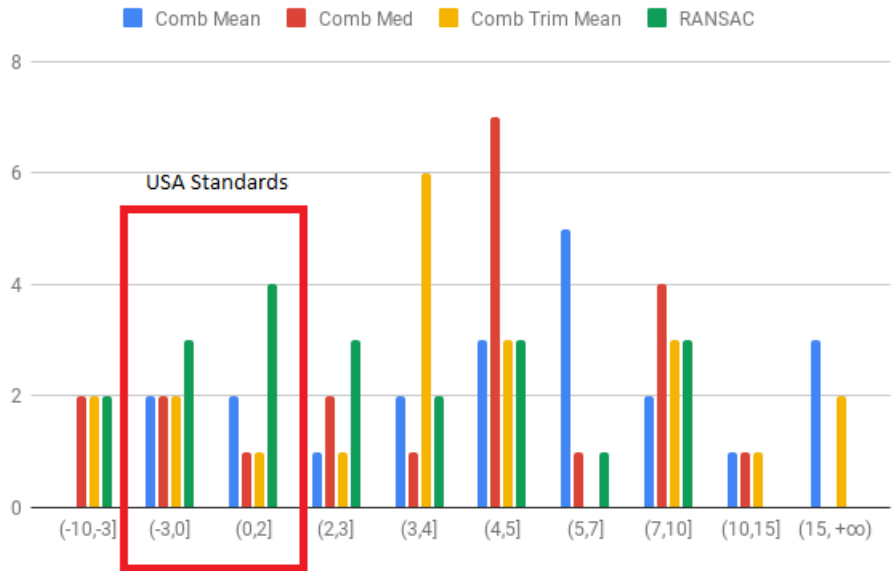
Although it is very different to compare different speed estimation approaches because they might differ in several aspects (assumptions, setup, dataset, etc.), we provide here a brief analysis about competitive approaches. For example, Luvizon, Nassu and Minetto (2016) reports 96% of the estimation within the range to be an acceptable measurement methods by the USA standards, opposed to 33.33% that our method was able to achieve, but our setup is less restrict than theirs, as we only need a calibrated camera, and they require a specific camera setup as detail on Section 3.2.

Table 5.3: Methods relative error in range

Range	Comb Mean	Comb Med	Comb Trim Mean	RANSAC
$0-5$	14.29	9.52	4.76	14.29
$5-10$	28.57	28.57	47.62	42.86
$10-15$	14.29	23.81	9.52	14.29
$15-25$	14.29	19.05	14.29	28.57
$25-40$	14.29	19.05	9.52	0.00
$40-100$	4.76	0.00	14.29	0.00
>100	9.52	0.00	0.00	0.00

Source: Authors

Figure 5.3: Absolute errors range (in Km/h)



Source: Authors

6 CONCLUDING REMARKS

In Section 6.1 we discuss about our conclusions and in Section 6.2 we detail about our plans to improve the proposed work.

6.1 Conclusion

In this work, we developed a method to estimate vehicle speed based on a sequence of images. The main idea is that the required setup to take those images would not be a complex one: we would only require a "semi-static" calibrated phone to record the vehicles on the side of a road. Based on the recorded images, we extract the LP positions over time, and generate tuples to store this information. Afterwards, we use those generated tuples and explore some methods to extract the vehicle speed.

After analyzing the results presented in Section 5.2, we can notice that, for this dataset, the proposed method that performed the best was RANSAC-based line fitting. It achieved an average relative error of 11.15% and a maximum error of 24.58%, which were the best average and maximum error among all of the methods proposed. The method that showed the worst results was the combinational mean, even thought for some cases (video7, video 10, video15, video19) it was the closest to the GT. However, when looking through the complete dataset, the overall estimated speeds were significantly wrong. The combination trimmed mean presented a better result than the combinational mean because a part of the outliers were discarded. The combinational median was the second best approach.

We believe the some of the reasons why some of our estimations have a larger error are:

1. the imprecision on the LP detection: the proposed solution highly depends on the accuracy of the LP detection. If the detected corners are not sufficiently accurate, this might have an impact on the final estimation.
2. camera movement not negligible as we anticipate: we assume that the translational motion of our videos (recorded with a person holding the camera) is neglectable, but this assumption might have been violated.
3. imprecision estimating the intrinsics values: pose estimation highly depends on the intrinsic parameters estimation, so if the parameters are not accurate, this can have

a high impact on the final estimation.

4. frames with a small resolution: if the resolution for each frame is a small one, we can lose some precision on the final estimation, as the car position can be very discretized and this can impact negatively on the final estimation.
5. violation of the speed constancy: our formulation assumes that the vehicle speed is constant during the capture, but might not hold in practice. Since the videos were captured close to a speed trap (so that we could obtain the GT speed), the drivers might slow down and hence violate the speed constancy assumption.

Finally, we can conclude that our technique seems to be a good vehicle speed detector, but it would not suit the purpose of law enforcement as the estimations can still have a significant margin for error.

6.2 Future Work

The presented results are promising, but they still show a significant inaccuracy. In this section we detail some improvements we believe could have a major impact on the estimations.

A key point in our technique is the the LP corners detection: as explained in Section 4.2 we used the pre-trained weights provided by the authors. One improvement we plan to make is to perform a fine tuning of the IWPOD-NET with Brazilian LPs only (including the new Mercosur plate models), as the provided weights were trained with LPs from others countries as well. We also plan to explore others methods to estimate the vehicle speed after we generate the set of tuples: one of these techniques is the weighted vector median, in which we would be able to use the direction of the movement of the LP to better estimate the speed.

REFERENCES

Carlos Teixeira. 2020. [Online; accessed 12-October-2021]. Available from Internet: <https://www.cnt.org.br/agencia-cnt/nova-placa-mercosul-entra-em-vigor->.

ANDREW, A. M. Multiple view geometry in computer vision. **Kybernetes**, Emerald Group Publishing Limited, 2001.

BACH, M.; STUMPER, D.; DIETMAYER, K. Deep convolutional traffic light recognition for automated driving. In: IEEE. **2018 21st International Conference on Intelligent Transportation Systems (ITSC)**. [S.l.], 2018. p. 851–858.

BARTL, V.; HEROUT, A. Optinopt: Dual optimization for automatic camera calibration by multi-target observations. In: IEEE. **2019 16th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)**. [S.l.], 2019. p. 1–8.

BASTOS, M. E. da S. et al. Vehicle speed detection and safety distance estimation using aerial images of brazilian highways. In: SBC. **Anais do XLVII Seminário Integrado de Software e Hardware**. [S.l.], 2020. p. 258–268.

BHARDWAJ, R. et al. Autocalib: Automatic traffic camera calibration at scale. **ACM Transactions on Sensor Networks (TOSN)**, ACM New York, NY, USA, v. 14, n. 3-4, p. 1–27, 2018.

BOCHKOVSKIY, A.; WANG, C.; LIAO, H. M. Yolov4: Optimal speed and accuracy of object detection. **CoRR**, abs/2004.10934, 2020. Available from Internet: <https://arxiv.org/abs/2004.10934>.

BOUZIADY, A. E. et al. Vehicle speed estimation using extracted surf features from stereo images. In: IEEE. **2018 International Conference on Intelligent Systems and Computer Vision (ISCV)**. [S.l.], 2018. p. 1–6.

CANER, H.; GECIM, H. S.; ALKAR, A. Z. Efficient embedded neural-network-based license plate recognition system. **IEEE Transactions on Vehicular Technology**, IEEE, v. 57, n. 5, p. 2675–2683, 2008.

CHENG, G. et al. Real-time detection of vehicle speed based on video image. In: IEEE. **2020 12th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA)**. [S.l.], 2020. p. 313–317.

DIÁRIO OFICIAL DA UNIÃO. **License Plate Resolution**. 2019. [Online; accessed 12-October-2021]. Available from Internet: <https://www.in.gov.br/web/dou/-/resolucao-n-780-de-26-de-junho-de-2019-179414765>.

DUBROFSKY, E. Homography estimation. **Diplomová práce. Vancouver: Univerzita Britské Kolumbie**, Citeseer, v. 5, 2009.

FAMOURI, M.; AZIMIFAR, Z.; WONG, A. A novel motion plane-based approach to vehicle speed estimation. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 20, n. 4, p. 1237–1246, 2018.

FILIPIAK, P.; GOLENKO, B.; DOLEGA, C. Nsga-ii based auto-calibration of automatic number plate recognition camera for vehicle speed measurement. In: SPRINGER. **European Conference on the Applications of Evolutionary Computation**. [S.l.], 2016. p. 803–818.

FISCHLER, M. A.; BOLLES, R. C. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. **Communications of the ACM**, ACM New York, NY, USA, v. 24, n. 6, p. 381–395, 1981.

FÜHR, G. Pedestrian tracking and collective behavior recognition. 2017.

FUNG, G. S. K.; YUNG, N. H. C.; PANG, G. K. Camera calibration from road lane markings. **Optical Engineering**, International Society for Optics and Photonics, v. 42, n. 10, p. 2967–2977, 2003.

HUA, S.; KAPOOR, M.; ANASTASIU, D. C. Vehicle tracking and speed estimation from traffic videos. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2018. p. 153–160.

JIA, W. et al. Mean shift for accurate license plate localization. In: IEEE. **Proceedings. 2005 IEEE Intelligent Transportation Systems, 2005.** [S.l.], 2005. p. 566–571.

JIN, C.-B.; LI, S.; KIM, H. Real-time action detection in video surveillance using sub-action descriptor with multi-cnn. **arXiv preprint arXiv:1710.03383**, 2017.

JING-ZHONG, W.; XIAOQING, X. A real-time detection of vehicle's speed based on vision principle and differential detection. In: IEEE. **2009 IEEE/INFORMS International Conference on Service Operations, Logistics and Informatics**. [S.l.], 2009. p. 493–496.

KHAN, S. D.; ULLAH, H. A survey of advances in vision-based vehicle re-identification. **Computer Vision and Image Understanding**, Elsevier, v. 182, p. 50–63, 2019.

LAROCA, R. et al. A robust real-time automatic license plate recognition based on the yolo detector. In: IEEE. **2018 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2018. p. 1–10.

LIN, H.-Y.; LI, K.-J. Motion blur removal and its application to vehicle speed detection. In: IEEE. **2004 International Conference on Image Processing, 2004. ICIP'04.** [S.l.], 2004. v. 5, p. 3407–3410.

LIN, T.-Y. et al. Microsoft coco: Common objects in context. In: SPRINGER. **European conference on computer vision**. [S.l.], 2014. p. 740–755.

LIU, L. et al. Deep learning for generic object detection: A survey. **International journal of computer vision**, Springer, v. 128, n. 2, p. 261–318, 2020.

LLORCA, D. F.; MARTÍNEZ, A. H.; DAZA, I. G. Vision-based vehicle speed estimation: A survey. **IET Intelligent Transport Systems**, 2021.

LUO, L. et al. An efficient method of license plate location. In: IEEE. **2009 First International Conference on Information Science and Engineering**. [S.l.], 2009. p. 770–773.

LUVIZON, D. C.; NASSU, B. T.; MINETTO, R. A video-based system for vehicle speed measurement in urban roadways. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, v. 18, n. 6, p. 1393–1404, 2016.

PAULA, M. B. de; JUNG, C. R.; JR, L. G. da S. Automatic on-the-fly extrinsic camera calibration of onboard vehicular cameras. **Expert Systems with Applications**, Elsevier, v. 41, n. 4, p. 1997–2007, 2014.

QIMIN, X. et al. A methodology of vehicle speed estimation based on optical flow. In: IEEE. **Proceedings of 2014 IEEE International Conference on Service Operations and Logistics, and Informatics**. [S.l.], 2014. p. 33–37.

REDMON, J. et al. You only look once: Unified, real-time object detection. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**. [S.l.: s.n.], 2016.

SANYUAN, Z.; MINGLI, Z.; XIUZI, Y. Car plate character extraction under complicated environment. In: IEEE. **2004 IEEE International Conference on Systems, Man and Cybernetics (IEEE Cat. No. 04CH37583)**. [S.l.], 2004. v. 5, p. 4722–4726.

SCHWEIGHOFER, G.; PINZ, A. Robust pose estimation from a planar target. **IEEE transactions on pattern analysis and machine intelligence**, IEEE, v. 28, n. 12, p. 2024–2030, 2006.

Scratchapixel. **Pinhole Camera**. 2021. [Online; accessed 14-October-2021]. Available from Internet: <https://www.scratchapixel.com/images/upload/cameras/pinholecam.png>.

SHASHIRANGANA, J. et al. Automated license plate recognition: a survey on methods and techniques. **IEEE Access**, IEEE, v. 9, p. 11203–11225, 2020.

SILVA, S. M.; JUNG, C. R. A flexible approach for automatic license plate recognition in unconstrained scenarios. **IEEE Transactions on Intelligent Transportation Systems**, IEEE, 2021.

TAKAHASHI, K. et al. Human pose as calibration pattern; 3d human pose estimation with multiple unsynchronized and uncalibrated cameras. In: **Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops**. [S.l.: s.n.], 2018. p. 1775–1782.

Wikipedia contributors. **Random sample consensus — Wikipedia, The Free Encyclopedia**. 2021. <https://en.wikipedia.org/w/index.php?title=Random_sample_consensus&oldid=1050189866>. [Online; accessed 31-October-2021].

XU, H.-k. et al. A new approach of the vehicle license plate location. In: IEEE. **Sixth International Conference on Parallel and Distributed Computing Applications and Technologies (PDCAT'05)**. [S.l.], 2005. p. 1055–1057.

YABO, A. et al. Vehicle classification and speed estimation using computer vision techniques. In: **XXV Congreso Argentino de Control Automático (AADECA 2016)(Buenos Aires, 2016)**. [S.l.: s.n.], 2016.

YAMAZAKI, F.; LIU, W.; VU, T. T. Vehicle extraction and speed detection from digital aerial images. In: IEEE. **IGARSS 2008-2008 IEEE International Geoscience and Remote Sensing Symposium**. [S.l.], 2008. v. 3, p. III–1334.

YOHIMORI, S. et al. License plate detection system by using threshold function and improved template matching method. In: IEEE. **IEEE Annual Meeting of the Fuzzy Information, 2004. Processing NAFIPS'04.** [S.l.], 2004. v. 1, p. 357–362.

ZHANG, Z. A flexible new technique for camera calibration. **IEEE Transactions on pattern analysis and machine intelligence**, IEEE, v. 22, n. 11, p. 1330–1334, 2000.

ZHANG, Z. et al. Practical camera calibration from moving objects for traffic scene surveillance. **IEEE transactions on circuits and systems for video technology**, IEEE, v. 23, n. 3, p. 518–533, 2012.

ZOU, Z. et al. Object detection in 20 years: A survey. **arXiv preprint arXiv:1905.05055**, 2019.