Lucas Caetano Possatti

Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars

Vitória, ES

2019

Lucas Caetano Possatti

Traffic Light Recognition Using Deep Learning and Prior Maps for Autonomous Cars

Dissertação de Mestrado apresentada ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo, como requisito parcial para obtenção do Grau de Mestre em Informática.

Universidade Federal do Espírito Santo – UFES

Centro Tecnológico

Programa de Pós-Graduação em Informática

Supervisor: Prof. Dr. Thiago Oliveira dos Santos

Vitória, ES

2019

Ficha catalográfica disponibilizada pelo Sistema Integrado de Bibliotecas - SIBI/UFES e elaborada pelo autor

---

Possatti, Lucas Caetano, 1993-
P856t Traffic light recognition using deep learning and prior maps for autonomous cars / Lucas Caetano Possatti. - 2019.
56 f. : il.

Orientador: Thiago Oliveira dos Santos.
Dissertação (Mestrado em Informática) - Universidade Federal do Espírito Santo, Centro Tecnológico.

1. Inteligência artificial. 2. Agentes inteligentes (Software). I. Oliveira dos Santos, Thiago. II. Universidade Federal do Espírito Santo. Centro Tecnológico. III. Título.

CDU: 004

TRAFFIC LIGHT RECOGNITION USING DEEP LEARNING AND PRIOR MAPS FOR AUTONOMOUS CARS

Lucas Caetano Possatti

Dissertação submetida ao Programa de Pós-Graduação em Informática da Universidade Federal do Espírito Santo como requisito parcial para a obtenção do grau de Mestre em Informática.

Aprovada em 21 de outubro de 2019:

Prof. Dr. Thiago Oliveira dos Santos
Orientador(a)

Prof. Dr. Alberto Ferreira De Souza
Membro Interno

Profª. Drª. Karin Satie Komati
Membro Externo

UNIVERSIDADE FEDERAL DO ESPÍRITO SANTO
Vitória-ES, 21 de outubro de 2019.

To my parents, Isaías and Gabrielle, for all their support throughout my life.

Acknowledgements

First off, I'm grateful to God for the opportunities open to me and his aid during the difficult moments. I want to thank my advisor, Prof. Dr. Thiago Oliveira dos Santos, for his invaluable guidance, without which I wouldn't have got this far. Also, Rodrigo Ferreira Berriel and Thiago Meireles Paixão for many fruitful discussions, which were essential for the completion of this work. I've learned a lot from them, and from all of my other friends at LCAD.

"I see now that the circumstances of one's birth are irrelevant.
It is what you do with the gift of life that determines who you are."
(Mewtwo)

Abstract

At complex intersections, human drivers can easily identify which traffic lights are relevant for the route they intend to follow, and what are their states (red, yellow, or green). However, this remains a challenging task for autonomous vehicles. In the literature, an effective solution to this problem is to merge traffic light recognition with prior maps of traffic lights. Deep learning techniques have showed great performance and power of generalization including for traffic related problems. Motivated by the advances in deep learning, some recent works leveraged some state-of-the-art deep detectors to locate traffic lights and classify their state from 2D camera images. However, none of them combine the power of deep learning-based detectors with prior maps to identify the state of the relevant traffic lights. Based on that, this work proposes to integrate the power of deep learning-based detection with prior maps of traffic light into our car platform, IARA (acronym for Intelligent Autonomous Robotic Automobile), to recognize the relevant traffic lights of predefined routes. The process is divided in two phases: an offline phase for map construction and traffic lights annotation; and an online phase for traffic light recognition and identification of the relevant ones. Two different types of model for detection and classification of traffic lights are approached. One is a single model, deep learning detector, that detects and classify the state of traffic lights in a single step. The other uses a deep learning detector for locating traffic lights, and a separate model for classifying their states. The proposed system was evaluated on five test cases (routes) in the city of Vitória, each case being composed of a video sequence and a prior map of relevant traffic lights for the route. Results showed that the proposed technique is able to correctly identify the relevant traffic light along the trajectory.

Keywords: traffic light. traffic light recognition. deep learning. prior map. IARA. YOLO.

Resumo

Em cruzamentos complexos, motoristas humanos conseguem facilmente identificar quais semáforos são relevantes para a rota que eles intendem seguir, e quais são os seus estados (vermelho, amarelo, ou verde). Porém, isso permanece uma tarefa desafiadora para veículos autônomos. Na literatura, uma solução efetiva para esse problema consiste em unir reconhecimento de semáforos com mapeamento prévio de semáforos. Técnicas de *Deep Learning* têm mostrado grandes resultados e poder de generalização, incluindo para problemas relacionados ao trânsito. Motivados pelos avanços em *Deep Learning*, alguns trabalhos recentes utilizam detectores estado da arte formados por técnicas de *Deep Learning* para localizar semáforos e classificar seus estados em imagens 2D. Porém, nenhum deles combina o poder desses detectores com mapas de semáforos para identificar o estado dos semáforos relevantes. Baseado nisso, este trabalho propõe integrar o poder de detectores baseados em *Deep Learning* com mapas de semáforos no nosso carro autônomo, IARA (acrônimo para *Intelligent Autonomous Robotic Automobile*), para reconhecer os semáforos relevantes de rotas pré-definidas. O processo é dividido em duas fases: uma offline para construção dos mapas necessários; e uma online para reconhecimento dos semáforos relevantes. Dois tipos de modelos para detecção e classificação de semáforos são abordados. Um é composto por um único modelo de *Deep Learning* que detecta e classifica o estado de semáforos em uma única etapa. O outro usa um detector *Deep Learning* para localizar semáforos, e um modelo à parte para classificar seus estados. O sistema proposto foi avaliado em cinco casos de teste (rotas) na cidade de Vitória, cada caso é composto por uma sequência de vídeo e mapas de semáforos relevantes para a rota. Os resultados mostram que o sistema proposto é capaz de corretamente identificar os semáforos relevantes ao longo das trajetórias.

Palavras-chaves: semáforo. reconhecimento de semáforo. deep learning. prior map. IARA. YOLO.

List of Figures

List of Tables

List of abbreviations and acronyms

ANN	Artificial Neural Network
CNN	Convolutional Neural Network
DNN	Deep Neural Network
BB	Bounding Box
TL	Traffic Light
TLR	Traffic Light Recognition
RF	Random Forest
SM	Single Model
CM	Compound Model
LiDAR	Light Detection And Ranging
IARA	Intelligent Autonomous Robotic Automobile
LCAD	Laboratório de Computação de Alto Desempenho (High Performance Computing Laboratory)
SVM	Support Vector Machines

Contents

1 Introduction

Autonomous driving is an essential topic of research in the development of intelligent transportation systems. Briefly, the great ambition with autonomous vehicles is fully replacing the human driver by a computer system without compromising and eventually improving safety and efficiency. To this end, the human ability of "seeing" the environment (e.g., the road, pedestrians, signs, and other vehicles) and behaving accordingly should be carefully reproduced by the computer system. In particular, autonomous terrestrial vehicles must be capable of perceiving traffic lights and identifying their current states (red, yellow, green). In the autonomous driving literature, this problem is generally known as Traffic Light Recognition (TLR).

Traffic intersections exist in different levels of complexity. Some of them include different traffic lights that control different lanes to different routes, and therefore can present discordant states at one given moment. Consequently, human drivers and autonomous cars must be able to discern which traffic lights are relevant to the route they intend to pursue. Most of the time, human drivers can easily identify the relevant traffic lights. Nevertheless, there are not always precise rules (e.g., an algorithm) that allow the differentiation of these traffic lights from the others in a scene. To deal with this issue, a common solution is to integrate traffic light recognition with prior maps which record position, direction, and other properties of the traffic lights [1, 2, 3]. With prior maps, an autonomous vehicle can be early aware of the presence of traffic lights on its vicinity, and can also fuse the map and real-time sensors' data (e.g., camera image) for robust detection and classification of the relevant traffic lights in a scene.

1.1 Motivation

Although TLR is widely addressed in the literature, there are still some challenges to be faced. Most methods [4, 5, 6, 7] focus on detecting and/or classifying all traffic lights in a scene and do not attribute any special meaning to the traffic lights that are relevant for the given context, i.e., the traffic lights in the vehicle's route and that the driver should obey. Other challenges include recognition in adverse conditions (e.g., rain, snow), early recognition (detecting traffic lights at greater distances), and recognition in different illumination settings (including night images). Despite their importance, these other challenges are out of the scope of this work.

Deep learning techniques have showed great performance and power of generalization in many areas and types of problems such as classification [8, 9] and detection [10]. General purpose object detectors have been well explored for traffic related problems (such as

detection of pedestrians, traffic signs, etc), and YOLO [11] is one of these state-of-the-art detectors. Motivated by the advances in deep learning, some recent works [12, 13, 14, 15] leveraged some state-of-the-art neural detectors to locate traffic lights and classify their state from 2D camera images. Other works, such as [16], combined prior maps with deep learning classification. However, to the best of our knowledge, none of them combine the power of the deep learning-based detectors with prior maps to identify the state of the relevant traffic lights.

Deep learning methods require many samples for training, but most traffic light datasets contain far fewer samples for yellow traffic lights when compared to the other classes. As a consequence, when training a deep learning model to detect and classify traffic lights, the performance for the yellow class usually suffers, as shown in [17]. There are two alternatives to this problem: encompassing red and yellow traffic lights into a single class (red-yellow); or decoupling the classifier from the detector, which is common for traditional methods.

1.2 Objectives

This work proposes to integrate the power of deep learning-based detection with prior maps of traffic light into our car platform, IARA (acronym for Intelligent Autonomous Robotic Automobile [18]), to recognize the relevant traffic lights of predefined routes.

To accomplish this task, the specific objectives are:

- Investigate the use of a deep learning model trained end-to-end for detecting traffic lights and classifying their states.
- Investigate the use of a model specialized at classifying traffic light state.
- Evaluate the entire TLR system's performance on real world data.

1.3 Proposal

In this work, the process of recognizing relevant traffic light states is divided in two phases: an offline phase for map construction and annotation of traffic lights into IARA's prior maps; and an online phase for traffic light recognition and identification of which of them are relevant. In the offline phase, after the prior map is constructed by driving IARA on routes of interest to collect camera, LiDAR, and other sensor data. Subsequently, a semi-automatic process is applied to these data in order to find 3D coordinates of traffic light candidates, which are further inspected on camera images to filter false positives and identify those that are relevant for the particular route. The relevant traffic lights' positions

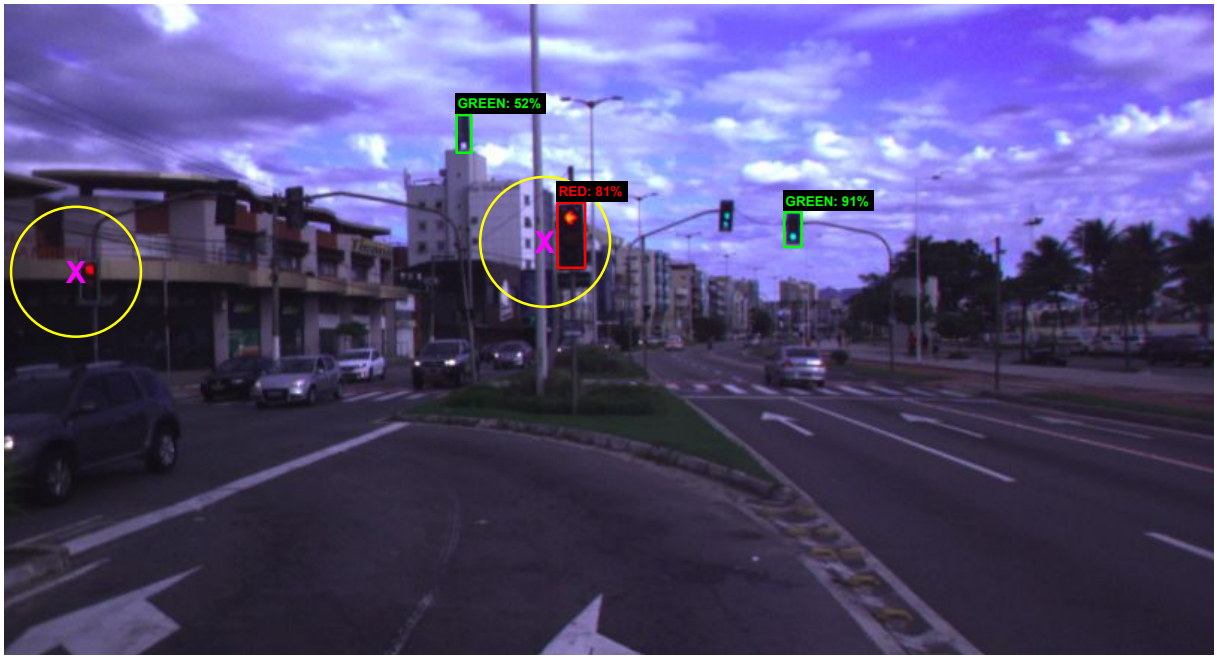


Figure 1 – Example of the proposed system running. The traffic lights delimited by bounding boxes (BBs) are located and recognized by a detector in a online phase. The text above BBs indicates the traffic lights' state followed by the confidence score of the detection. The pink crosses represent the projection of the mapped (during an offline phase) relevant traffic lights onto the 2D camera space, whereas the yellow circles define a threshold limit for these traffic lights. BBs whose center lies outside the yellow circles are filtered out. The BB closest to any of the red dots will dictate the traffic light (group) state.

(in 3D world coordinates) are then stored in the prior map. In the online phase, as the car approaches traffic lights, they are detected on camera frames, classified according to their states, and their annotated positions are projected to the image in order to filter the predictions (location and state). Finally, only the relevant traffic lights are taken into account in the decision-making process. The relevant state is dictated by the traffic light that is closest to the projected annotations. See Figure 1 for an illustration of the system behavior.

Two types of models for detection and classification of traffic lights are explored. One is a single deep learning model that can both detect traffic lights and classify their state in a single step. And the other is a two-stage model, composed of a deep learning detector that is responsible for detecting traffic lights, and a separate model (a Random Forest) for classifying their individual state. The first is trained to recognize red and yellow traffic lights as a single class due to the lack of yellow samples. The second one is designed to overcome this limitation of samples, and to be a model that is fully capable of classifying the three states (red, yellow, green) as separate classes.

For performance assessment, each individual model was tested with metrics and datasets specific for the tasks they were designed for. The detection models (single or

compound) were evaluated using Mean Average Precision (mAP) and precision-recall curves. The classification models were evaluated in terms of macro-averaged accuracy. And the system as a whole was evaluated on real-world data for five test routes in the city of Vitória, and its performance was measured in two ways: (i) how accurate the states predicted by the full system were, i.e., the system's ability to correctly assign the state of the relevant traffic lights throughout the car's progress on the road; and (ii) how early it can recognize traffic lights correctly.

1.4 Contributions

The main contributions of this work are:

- A YOLOv3 model trained for detecting traffic lights and classifying their state as red-yellow or green.
- A Random Forest model that can robustly classify traffic lights' state as red, green or yellow.
- A new method for annotating the 3D poses of traffic lights.
- A peer reviewed paper [19] accepted on IJCNN (International Joint Conference on Neural Networks).
- A TLR system that is integrated into IARA.

1.5 Structure

The remaining of this document is organized as follows: the theoretical background is discussed in Section 2; the proposed method is described in Section 3; the materials and methods are presented in Section 4; the experiments and their results are shown and discussed in Section 5; and, finally, Section 6 concludes and discusses future work.

2 Theoretical Background

This chapter approaches the subjects of deep neural networks used for object detection, Random Forests for classification, and related works that use deep learning or prior maps for traffic light recognition.

2.1 Deep Learning for Object Detection

Artificial Neural Networks (ANN) are a class of machine learning models inspired by the biological brain. Theses models are composed of artificial neurons (or nodes) with connections among them, and their interaction is modeled after biological neurons and synapses in the brain. ANNs are typically defined in terms of layers of neurons. When these layers are connected in a way that do not form a cycle, this type of ANN is a feedforward one [20]. During inference, a feedforward network will use the activation values of individual neurons in a layer as input to the next layer, and so on. In this setup, the first layer is commonly referred to as the input layer, the last one as the output layer, and the layers between the two are called hidden layers [21]. Figure 2 shows a representation of an ANN with four layers. When ANNs have a good number of hidden layers (although, how many is up for discussion [21]) they can be regarded as "Deep" Neural Networks (DNN).

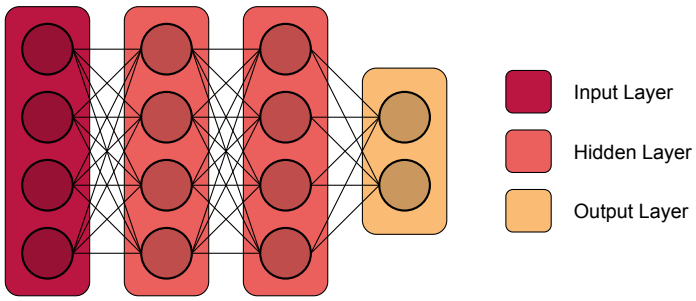


Figure 2 – An schematic of an ANN with one input layer, two hidden layers, and one output layer. All hidden and output layers in this example are fully connected layers, i.e., all of their neurons connect to all previous layer's neurons.

Recently, Deep Neural Networks have been widely used for the tasks of image classification, object detection and many others, specially since recent breakthroughs in many different areas of research [21]. Many of such breakthroughs were reached through the use of convolutional neural networks (CNN), which are a special type of ANN with layers that employ a mathematical operation known as convolution, and are thus called convolutional layers. A convolutional layer is composed of filters. When operating, a window, which is the same size as the filter, slides over the input features, and its values

are multiplied element-wise by the filter's weights and summed up to produce the window's output values. The output of each filter is commonly called a feature map. Figure 3 illustrates this process with a bi-dimensional input and a single filter. Also, a common pattern in convolutional layers is to pad the input features with zeros before doing the convolution, so that the output will have the same size as the input. Besides convolutional layers, CNNs often employ other types of layers, such as: pooling layers for reducing the spatial dimensions; shortcut connections for dealing with the vanishing gradient problem; and fully connected layers, usually at the end, for classification. There are many different architectures of CNNs for different tasks. For classification, some of the most popular architectures are ResNet [22], SqueezeNet [23], MobileNets [24] and VGG [25].

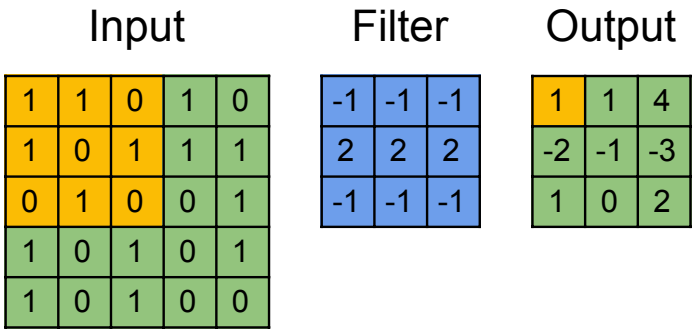


Figure 3 – General operation of a convolutional layer. The sliding window takes elements from the input features (yellow highlight in "input") and the filter's weights to produce values in output (yellow highlight in "output").

Object detection is a task of computer vision, in which an algorithm needs to find if an object class is present in an image, and, if so, localize each instance of it [26]. The localization part is typically done using bounding boxes. Deep learning models used for object detection (deep detectors) take a colored image as input for inference. The image is forwarded through the convolutional layers of a standard convolutional neural network (the detector's backbone, e.g., ResNet, VGG or Darknet) for feature extraction. Further layers process the output of this feature extraction process and generate a list of bounding boxes (BBs) with the objects' class probabilities.

One of such models is YOLO, a state-of-the-art deep learning architecture for object detection first implemented on the Darknet framework. The third version of this architecture, YOLOv3 [27], achieves 57.9 mAP (AP_{50}) on Microsoft's challenging COCO dataset, using an input resolution of 608×608 pixels. YOLO stands out from other models for achieving good performances while keeping fast inference times. The Darknet-53 (a CNN model with 53 convolutional layers) is used as its backbone, and it incorporates some features that were missing in its previous version and that are now considered essential for state-of-the-art models, like residual blocks (inside Darknet-53) and feature pyramids. Particularly, the inclusion of feature pyramids are much beneficial to this work, since they

improve accuracy for small objects [28].

Figure 4 displays a diagram of YOLOv3's architecture. First the image is forwarded through the layers of the backbone (Darknet-53) for feature extraction. Then, these features are routed through three different detection heads. The first one is composed of more convolutional layers and, at the end, an YOLO layer, which is responsible for producing the bounding box predictions. The second and the third detection heads carry out a similar process. First, they receive features from the previous detection head, upsample (i.e. increase their resolution) and concatenate them with more fine-grained features from the backbone network. And only then, they apply more convolutions and finally the YOLO layer. This procedure of merging together features from lower (more detailed) and higher (semantically stronger) regions of the network is the feature pyramid. Because the first YOLO layer makes its predictions over lower resolution feature maps, it is better at detecting larger objects, while the next YOLO layers use increasingly higher resolutions and are, thus, better at detecting smaller objects.

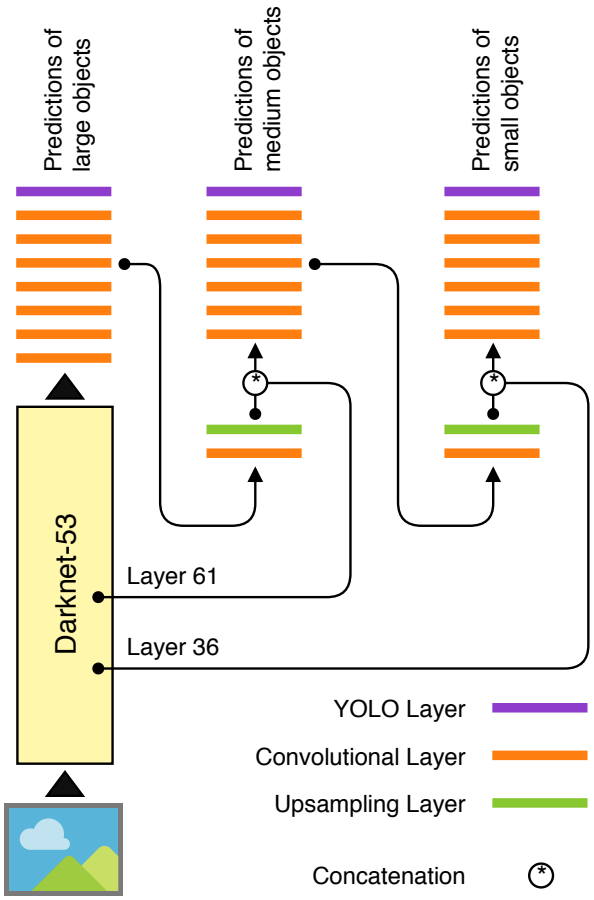


Figure 4 – Diagram of YOLOv3's architecture. The network receives the image at the bottom, and processes it upwards. After the feature extraction process, in Darknet-53, the features flow through three detection heads which are responsible for predicting the bounding boxes at different scales.

The bound boxes output by YOLOv3 each have an "objectness" score and con-

ditional class probabilities associated with them. The objectness is a prediction of how accurate the bounding box's coordinates are and how likely it contains an object of interest. And the conditional class probabilities are the raw class probabilities multiplied by the objectness.

2.2 Random Forest

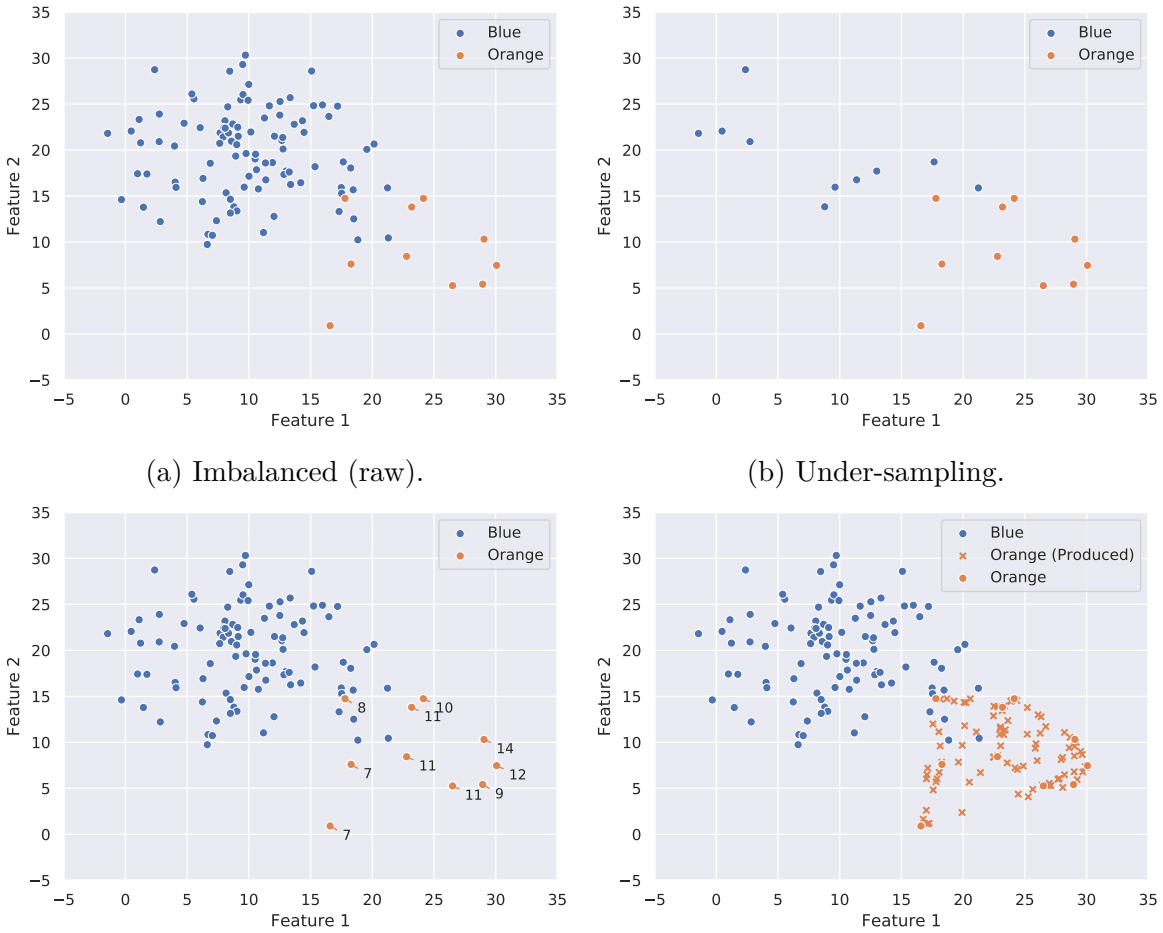
Random Forests are a machine learning method that constructs and combines the predictions of multiple decision trees for the tasks of classification or regression. Some of the most influential works regarding Random Forests are [29, 30, 31], specially [31], whose "Forest-RI" model is the reference model for many machine learning libraries, including OpenCV (which is used in this work). For this method, when training each individual tree, a new subset of the training data is drawn randomly with replacement (the same number of total samples is drawn) and the tree is trained using only that subset. This process is known as bootstrap. Additionally, when training the trees, at each decision node, only a determined number of features, randomly selected from all the input features, will be used for finding the best split. During inference, the final prediction is the most voted class among all the decision trees.

Random Forests have a few hyper-parameters that allow for tuning the model, so that it performs better for the task at hand. The most relevant of them are: the number of trees; maximum depth of trees; minimum number of samples for splitting a node; and maximum number of features used for finding the best splits. The number of trees is primordial for the model's performance. Generally speaking, the higher it is, the better the trained model is. Increasing the number of trees does not make the algorithm more prone to overfitting [31], however the greater the number of trees, the longer it takes to train the model and run inferences. So, a reasonable number will be high, but only if there is a significant gain from it when compared to lower values. Regarding the maximum depth, individual trees that are too deep have more room for overfitting, due to the the increased risk of learning the noise in the data. Nevertheless, Random Forests minimize this risk by combining the results of many trees together, yet this is a value that needs attention. There is also, the minimum number of samples required for splitting a node (i.e., turn it into a decision node). This number can vary from 2 to the maximum number of samples, however low values are recommended (OpenCV suggests 1% of the total samples), since higher values will hinder the creation of decision nodes, therefore preventing the trees from learning more. Finally, the maximum number of features sampled by a decision node when searching for the best split can range from 1 to the total number of input features (M). Two of the most commonly used values for this setting are $\sqrt{M}$ (OpenCV's default) and $\lfloor \log_2 M + 1 \rfloor$ (as in [31]). A common practice for choosing the hyper-parameters is to employ a grid search (or random search) over reasonable ranges of the hyper-parameter

space. In this process, multiple models are trained with different sets of hyper-parameters, and the best one is selected.

2.3 Techniques for Balancing Datasets

Raw datasets for classification are often imbalanced to some extent. This happens when at least one of the classes have many more (or many fewer) samples than the others. So, when training some machine learning models, it might happen that a few classes will have a bigger influence in the training algorithm than others. In some cases, even to the point of almost completely ignoring the classes with fewer samples. Balancing the data can be used to counter this issue. There are two main categories of balancing algorithms: under-sampling, which will reduce the number of samples of the majority class; and over-sampling, which will increase the number of samples for the minority class.



(a) Imbalanced (raw).

(b) Under-sampling.

(c) Naive over-sampling.

(d) SMOTE.

Figure 5 – In Subfigure a, a plot displaying the samples of an imbalanced synthetic dataset is observed. Subfigures b, c, and d are examples of the same dataset after transformed by some balancing technique. In all subfigures, both axis represent two independent features of the dataset, and the color of each data point represents the class it belongs to.

A synthetic dataset was built to better illustrate a few techniques used for balancing datasets. Each sample in the dataset contains values for two features only, "Feature 1" and "Feature 2", and each is labeled as one of two classes: Blue; or Orange. The Blue class comprise 100 samples, while the Orange class has only 10 samples. All these samples are plotted in Figure 5a, in there, each axis represents one of the two features available, and the color encodes which class the samples belong to. Figures 5b, 5c and 5d are laid out in a similar manner, but they represent the outcome after applying one of the balancing techniques that will be discussed here.

A common form of under-sampling is to simply eliminate random samples from the majority classes, until the desired ratio among the classes is achieved. Figure 5b displays the use of this technique for reaching a balance of 1:1 for the two classes. After the algorithm is applied there are 10 samples per class, and the dataset is balanced. In cases where the minority class has enough samples for training, this technique could be a good choice. However, this technique could impose a problem, if after applying it there were not enough samples to properly train the desired model.

Regarding over-sampling techniques, the most basic approach is to randomly duplicate samples from the minority class. Figure 5c displays the result after applying this technique. The numbers in the figure count how many samples are stacked in the same position. As it can be noticed, the new samples fall exactly over already existing samples in the plot, because they are indeed duplications of existing ones. By increasing the number of samples, some machine learn algorithms will more properly learn the minority class, instead of ignoring them, but since it only duplicates samples that are already present in the data, it does not add any new information.

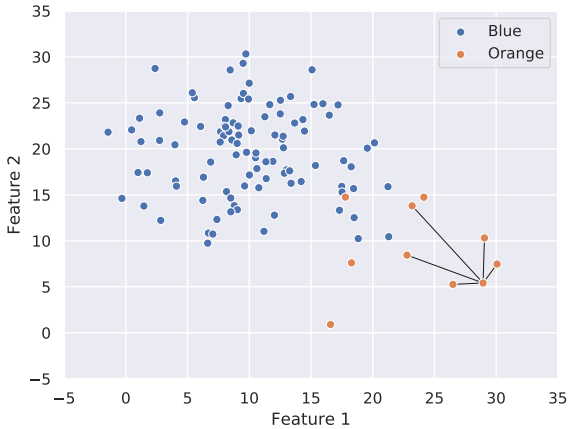


Figure 6 – Example of SMOTE's operation. The algorithm finds the 5 nearest neighbors of a sample, and creates new samples above the line segments (represented in black) that connects them.

A more elaborate method for over-sampling is an algorithm called SMOTE (acronym for Synthetic Minority Over-sampling Technique)[32]. For each sample belonging to the

minority class, the algorithm obtains the 5 nearest neighbors of each one. Then, according to how much over-sampling is needed, it will randomly select some pairs of samples and one of their neighbors, and for each pair it will lay down a new sample in-between the two through linear interpolation of their features. Figure 6 shows line segments connecting an arbitrarily chosen sample to its 5 nearest neighbors. The SMOTE algorithm will create synthetic samples that are placed over these line segments. This process will repeat until the desired balance between the classes is reached. Figure 5d displays the result after applying SMOTE on the synthetic dataset. Although SMOTE seems to have improved the state of the example dataset, one must be watchful if the synthetic samples resemble real world data. In some applications, the produced samples might be too artificial (i.e. their features' values would not happen naturally as a whole) and, therefore, could hinder the model from learning the underlying patterns of the class. As an example, Figure 7 displays two images (7a and 7b) of the same dog and the resulting linear interpolation between the two of them (7c).

(a) (b) (c)

Figure 7 – Two photographs of the same dog are displayed on the left and middle images. The one on the right is the result of a linear interpolation between the two others, which is how SMOTE creates new samples.

2.4 Related Work

This section covers the main works addressing the use of prior maps or deep learning applied to TLR. For a more comprehensive review that includes other approaches, the reader should refer to the surveys in [33, 3].

Prior maps containing annotations of traffic lights (e.g., position, direction) can be exploited to increase the robustness of TLR. In this context, the work of Lindner et al. [34] uses maps to recognize traffic lights and they propose a three-stage system: detection (based on handcrafted features), tracking, and state classification (using a feed-forward neural network). Map information and GPS data can be incorporated into the system to allow triggering the system only near intersections; therefore, possibly reducing the amount of false alarm detections. Despite of the use of prior maps, their work does not address explicitly the choice of the relevant traffic lights for the lane.

Fairfield and Urmson [1] present an automatic strategy to map traffic lights by fusing the precise location of the car (accuracy of less than 15 cm) with image information, and then estimating traffic lights' 3D positions using least squares triangulation. An heuristic is used to make initial guess of what routes each traffic light controls based on the position, orientation, and the average intersection width. Then, the outcome of this automatic process is verified and further refined by human annotators. Using their method, they were able to map over 4000 traffic lights across more than 1000 intersections in the area of San Jose, in San Francisco. Levinson et al. [35] propose a mapping procedure of traffic lights using tracking, back-projection, and triangulation. Traffic lights' states are computed in a probabilistic approach taking into consideration the previously constructed map. Their method is invariant to different lighting conditions, and it probabilistically combines predictions of redundant traffic lights to make a more robust prediction for the state of the intersection. In addition to prior maps, Frank et al. [36] locate relevant traffic lights by matching image-based features of intersections. Such features are extracted offline from manually labeled regions around the relevant traffic lights while a neural network is responsible for predicting their states. Then, each of them is tracked over time, to make predictions more reliable. With this setup, the vehicle drove autonomously for 6700 km during a test phase, with a final test taking place on Bertha Benz Memorial Route.

John et al. [16] combine prior maps and GPS to limit the region of interest (ROI; the search area) where traffic lights are expected to appear in the images. A custom feature space is built from multiple color spaces and it is binarized using two empirical thresholds to propose TL candidates. Candidates without a circular shape are discarded afterwards. Then, bounding boxes of multiple sizes are extracted from each candidate, and a new feature space is used for classification. Three CNNs (one for each class: green, red, and background) classify each bounding box. The CNN with highest output value determines the final class. Also, a saliency map is computed using daylight data in order to aid the system at night by further restricting the regions of interest. Jang et al. [37] propose a recognition system that explores the prior maps at every stage. First, a ROI is generated based on where the TLs are expected to appear on camera. Slopes on the road are taken into account when generating the ROIs. Then, their approach selects the appropriate detector and classifier according to the TL's face (three bulbs or four bulbs). An Adaboost classifier based on Haar-like features is used for detecting TLs. Each detection is then classified by two SVM classifiers, using HOG features and HSV histograms. The first one classifies whether the object is indeed a traffic light (which is useful for discarding false positives), and the second one classifies its state. Finally, they use the distance to the TL in order to aid the tracking algorithm.

Tiago Alves [38] developed a TLR solution for IARA that detects TLs and classifies their states. For detection, the Viola-Jones algorithm is used to search a fixed subregion in camera images where TLs are expected to appear. The detector reached 97.7% of precision

and 52.3% of recall in his experiments using a local dataset. Further, each detection is cropped from the image and rescaled to a resolution of 10×20 pixels; then, the red and green channels are combined into a single channel utilizing a pixel-wise operation, and the resulting features are normalized. An SVM classifier takes that as its input, and classifies the image crop as either red or green. The classifier reached an accuracy of 96.5% in his experiments. During system-wide tests, due to the low recall rate of the detector, only 52.2% of TLs were detected. From those correctly detected, 96.3% of them were correctly classified. He also developed a semiautomatic method for annotating traffic light poses (3D coordinates and orientation). In his method, the user visually selects a point from a 3D point cloud (generated from LiDAR) and its coordinates will be used as the coordinates for the annotation. These coordinates are projected onto camera images, so that it can be verified that they represent the TL of interest. Finally, the car's orientation is used as the orientation of the annotation. And, although he annotates TL poses, they are not used to aid the detection or classification algorithms at any moment. It is simply used to trigger the TLR system when there is a TL nearby, and to inform IARA of the distance to the TL.

The interest in deep learning methods for TLR has been observed in the recent years. Weber et al.[12] propose a neural network called DeepTLR for detection and classification of traffic lights. The network's architecture is partially inspired by AlexNet. It uses nine convolutional layers, and reaches 35 FPS when taking 640×480 RGB images. Behrendt et al. [13] modified YOLO [11] to detect traffic light candidates. In their solution, three crops from the upper part of the camera frames are independently fed into YOLO for detecting traffic lights. The YOLO model had its detection grid size changed from 7×7 to 11×11 to improve detection of more distant TLs. The classification part of the network was removed in order to improve detection's performance, and this task was left to a custom CNN, which classifies detected objects as either red, yellow, green, or "background" (i.e., the the object is not a TL). The predicted bounding boxes are expanded and rescaled before being fed to the CNN, so that the network can use this extra background as context for filtering out false positives. An odometry-based motion model, aided by another neural network, is used to track TL objects. The models were trained and tested on their own public traffic light dataset, *Bosch Small Traffic Lights Dataset*. A more detailed study on traffic light detection using YOLO can be found in [39]. Pon et al. [15] trained a modified version of the Faster R-CNN [40], with a hierarchical class structure, to simultaneously detect traffic lights and traffic signs. Most traffic light datasets do not label signs, and vice versa. This is a hindrance for training a model that can detect both. To counter the issue, the authors take advantage of the fact that traffic lights and signs are usually separated in physical space, so they discard RPN proposals that do not overlap with a ground truth bounding box.

Despite the fact that TLR can benefit from modern deep learning techniques,

additional information should be used to allow detection and classification of the relevant traffic lights. Therefore, this work proposes to combine a state-of-the-art deep learning model with precise location of the car (e.g., by exploiting LiDAR data) and prior maps for a real-world TLR application in autonomous vehicles.

3 Traffic Light Recognition

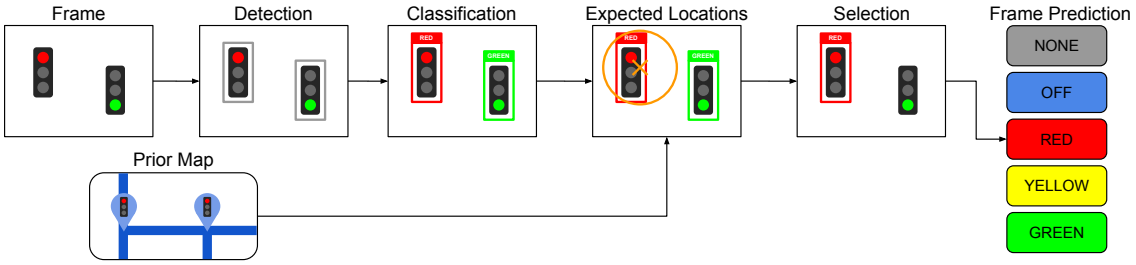


Figure 8 – Overall flow of the system. First, traffic lights are detected on the camera frame. Then, the traffic lights are classified according to their state. Subsequently, TL's world positions from the prior maps are projected into the image (orange cross) using the current localization pose of the vehicle. The orange circumference represents a threshold that accounts for imprecisions in localization. Finally, from BBs that have center within the threshold, the closest to any of the projected TL's positions is selected and used as final state prediction for that frame.

This work proposes a method for recognition of relevant traffic lights (TLs) and their state that combines deep learning with prior maps. The process is performed in two steps: offline map construction and annotation of the relevant traffic lights for the intended route, and an online detection and classification of the relevant traffic light state. Figure 8 displays an overall picture of the online phase of the method. On each camera frame, the traffic lights are detected, and further classified according to their state. Subsequently, information from prior maps of traffic lights and the localization of the vehicle are used to project the TL's world position to the current frame and then select only one BB to make the final state prediction for that frame. Additionally, a new method for annotating traffic lights that uses both the detection model and the LiDAR sensor is proposed. This unified approach to TLR is implemented in the autonomous car of our laboratory, IARA.

The next sections describe: the IARA platform (Section 3.1), the autonomous car on which the proposed approach was implemented; the traffic light detection and classification procedures (Section 3.2), which include the deep learning traffic light detector and the Random Forest state classifier; the methods for annotating the relevant traffic lights' world position for a given route (Section 3.3); and, finally, the general system operation (Section 3.4), that describes how these steps are connected.

3.1 Intelligent Autonomous Robotic Automobile (IARA)

IARA is the autonomous car built in our laboratory. It is an adapted Ford Escape Hybrid that features a variety of sensors such as: odometer, LiDAR, IMU, RTK-GPS,

and stereo cameras. The LiDAR is a Velodyne's HDL-32E and the stereo camera is a Point Grey's Bumblebee XB3 stereo camera. Both are mounted on top of the car, and the camera is mounted front-facing. The HDL-32E LiDAR uses 32 lasers spread at a 40° VFOV (+10° to −30°) to scan 360° horizontally. The Bumblebee XB3 camera has three 1.3MP CCD sensors with 66° of HFOV, and captures colored images of 1280 × 960 pixels at 16 FPS. Figure 9 points out where each sensor is positioned in the car.

Figure 9 – Photograph of IARA highlighting its main sensors.

IARA's software is based on the Carnegie Mellon Robot Navigation Toolkit (CARMEN), which is an open source collection of software for robot control [41]. Our laboratory, LCAD, maintains its own fork of CARMEN at <https://github.com/LCAD-UFES/carmen_lcad>. It includes all software that is currently used in the car. And its architecture is divided in two systems: perception and decision-making. Figure 10 depicts this division, their main subsystems and how they interact.

IARA's standard mode of operation, when driving autonomously, is to follow a predetermined route that was already driven by a human before. When the human drives along the route for the first time, all sensor data from this run is stored in a driving log. This log, in turn, is used offline to produce a few assets necessary for the autonomous operation: a Road Definition Data File (RDDF), and an offline map (occupancy grid map). The RDDF is a file with a sequence of states (position, orientation, velocity, etc.) from the car along the route. Many types of map could be used with IARA, but currently it uses an occupancy grid map (see Figure 11), which is generated by the Mapper. Many important

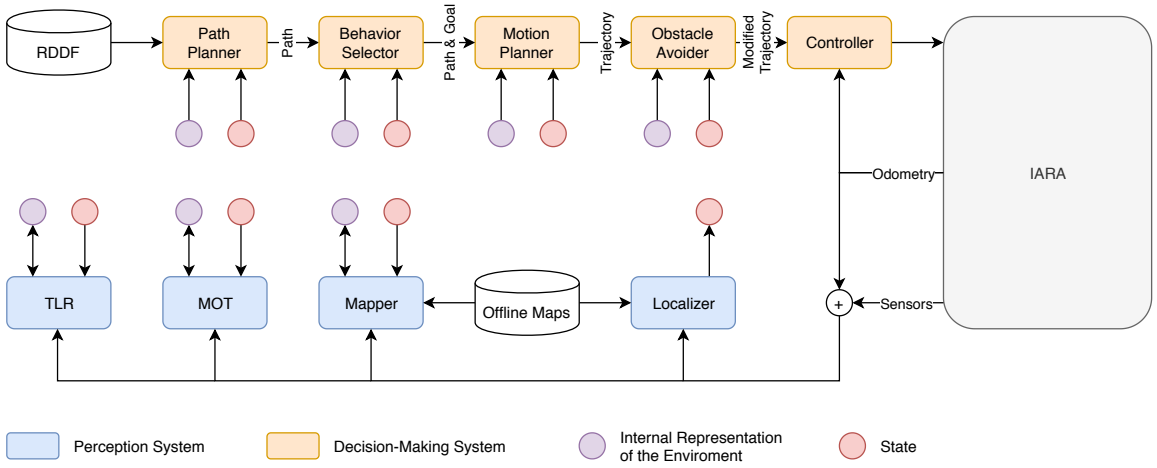


Figure 10 – Overview of IARA's software architecture. Most subsystems share an internal representation of the environment. And many of them receive the current state from the Localizer. The arrows denote the flow of messages among the subsystems.

features of the route are manually annotated into IARA's prior maps to further aid the car, such as traffic light positions, speed bumps, security barriers, crosswalks, speed limits, and so forth. The next paragraphs briefly explore the main subsystems of the car.

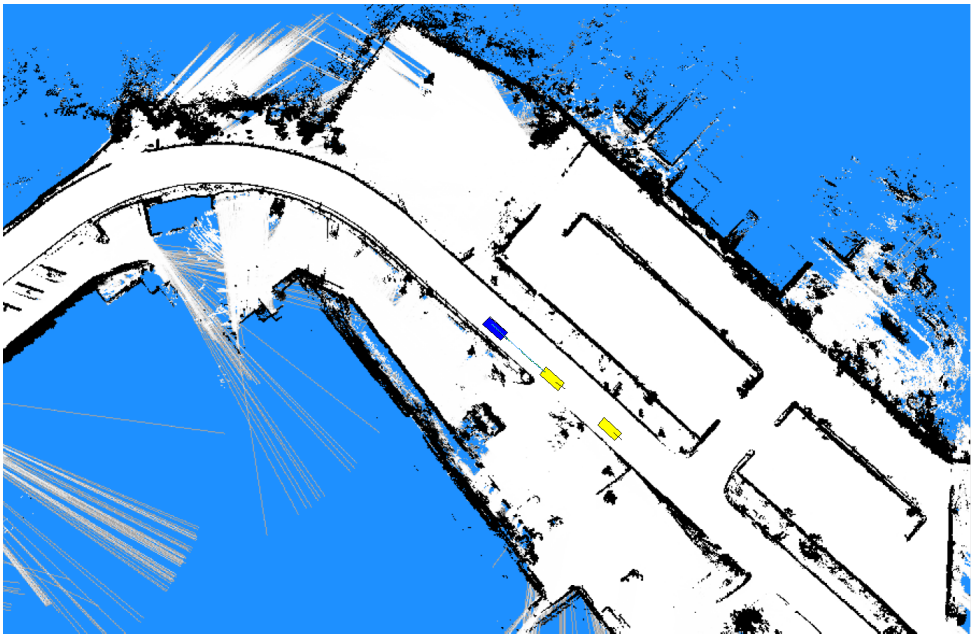


Figure 11 – Example of occupancy grid map. Each pixel is a cell in the grid map. The white pixels represent areas of free navigation; the black pixels are areas blocked by obstacles; and the blue ones are areas where the sensors did not reach and, consequently, unknown. The dark blue box in the map is the car pose predicted by the *Localizer* subsystem, which should be very accurate. The yellow boxes are goal states yielded by the *Behavior Selector*.

The *Traffic Signalization Detection (TSD)* subsystem is responsible for detecting traffic signalization, like traffic lights, traffic signs, markings on the ground, etc. Our

solution for traffic light recognition is part of this comprehensive subsystem, and the details on how traffic light detection, and classification are performed will be approached on the next sections, as well as the method for annotating traffic lights. The *Mapper* [42] uses 3D point clouds from LiDAR to construct 2D occupancy grid maps, which is a map where each cell in the grid represents a small area in the world and its value denotes the probability of being occupied by an obstacle or not. This subsystem is used offline for creating the offline maps, and online for updating them with new information. The *Moving Objects Tracking (MOT)* subsystem is responsible for detecting and tracking moving obstacles around the car, such as pedestrian, animals and other vehicles. The *Localizer* is responsible for predicting IARA's state relative to the world. To localize the car accurately, GPS coordinates are often not enough. For this reason, IARA's Localizer subsystem uses a localization method [43] based on Particle Filter localization. It is initialized with a pose from GPS and orientation from IMU, it then enters a cycle of two phases: prediction and correction. In the first phase, it predicts car poses using odometry data. In the second phase, the method corrects these poses by matching local occupancy grid maps with the offline one. Experiments showed that IARA's localization system operates within 0.28 meters of longitudinal error and 0.14 meters of lateral error [43].

The *Path Planner* extracts, from the RDDF, a sequence of equally spaced states that goes from IARA's current state to a goal state some meters ahead. The *Behavior Selector* chooses a behavior according to the path and the current scenario. It will, for example, choose to stop the car before red traffic lights or busy crosswalks; reduce the speed before a speed bump; or accelerate in an open highway. It will then establish a local goal state in the path, some seconds ahead of the current state, and tweak its pose and velocity according to its plan. The *Motion Planner* computes a trajectory (sequence of control commands) to take IARA from the current state to a local goal in the path. Each control command contains a linear velocity, a steering wheel angle, and an execution time. The *Obstacle Avoider* verifies the sequence of commands and modifies it, if necessary, in order to avoid collisions that were not foreseen. Finally, the *Controller* computes and sends the commands to the actuators of the steering wheel, throttle and brakes, in order to make the car execute the Modified Trajectory as best as possible.

3.2 Traffic Light Detection and Classification of state

The task of traffic light recognition can be broken into two steps: detection of traffic light objects, and classification of their state (red, yellow, green). In this work, two types of models are explored for this task: single model and compound model. The two were designed as standalone units, i.e., there is no switching between the two. The single model (SM) uses a single YOLOv3 model to perform both detection and classification at once. And the compound model (CM) uses a YOLOv3 model for detection only, and a separate

Random Forest model for classification of state. Due to the limitations with the number of yellow samples, the single model is trained with red and yellow traffic lights as a single class, while the compound model distinguishes between the two. Figure 12 illustrates the design of both SM and CM.

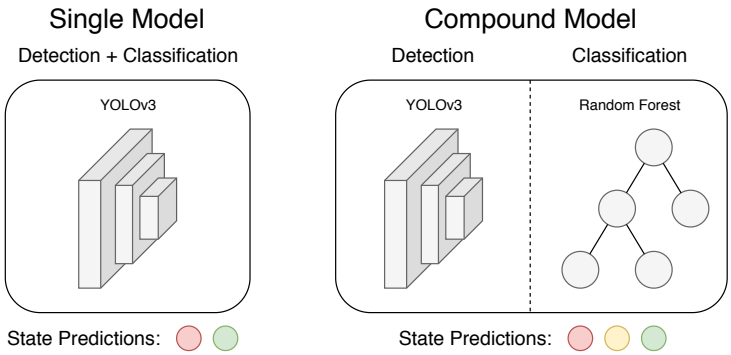


Figure 12 – Design of the Single Model and Compound model.

3.2.1 Single Model

The single model is composed solely of a YOLOv3 model, which is responsible for detecting traffic lights and also for classifying their state. This is achieved by training YOLO to treat traffic lights with different states as if they were different object classes. The model is trained end-to-end using colored images of traffic scenes. These images have to be annotated with axis-aligned bounding boxes surrounding the traffic lights, and each bounding box labeled with the appropriate class. Since deep learning models require many samples for training and most traffic light datasets have very few yellow samples, training a three-class model (red, yellow, green) is impractical. Therefore, the classes red and yellow are mixed together into one single class (red-yellow). For the purposes of our application, joining these classes is acceptable since it is better to have IARA stopping before a yellow traffic light than proceeding. Hereon, the YOLOv3 model resulting from this training will be referred to as YOLO-RG.

3.2.2 Compound Model

Even though there are not enough yellow traffic light samples to fully train a three-class deep learning detector, a Random Forest can be used specifically to classify the traffic lights' states, since it usually requires less samples for training. Hence, the compound model combines YOLO for locating traffic lights and a Random Forest to classify their states, and it aims at fully distinguishing the three states: red, yellow, and green. Random Forests have been chosen because they achieve good results, but are still very fast, both for inference and for training.

The compound model works in two stages. The first stage is a YOLOv3 model,

which behaves identically to the single model detector, i.e., an image is fed to the network, which in turn produces bounding box predictions. Any YOLO model that detects traffic lights could be used for this stage. In addition to using the YOLO-RG model for the first stage, we investigate the use of a generic YOLOv3 traffic light detector (hereinafter, known as YOLO-TL). It is generic in the sense that is does not identify the traffic light state. In the second stage, all information about traffic light state, if any, is dropped, and only the location of the the traffic lights is used, i.e., the bounding boxes' coordinates. All traffic light bounding boxes are used to crop the original image, with each bounding box yielding a distinct image crop, which is then resized to a resolution of 10×30 pixels. These crops are taken as input of a Random Forest classifier, which will predict the bounding box's final class (red, green, or yellow traffic light).

Both the YOLO model and the RF model are trained separately. The YOLO model is trained in a process similar to the Single Model. The only difference being that it might be trained to differentiate the state of the traffic lights (this information will be dropped during inference) or not. The RF model is trained using small colored images, each containing a single traffic light. The images are 10×30 pixels in size, and they are labeled according to the state of the traffic light they contain.

3.3 Annotation of Traffic Lights

Given that the driving log is already necessary for IARA's standard operation, it can also be used to create the prior map of relevant traffic lights for the respective RDDF. Making use of prior maps is necessary because until the present moment there is no clear algorithm or machine learning method that can robustly identify which traffic lights are relevant using image data only. The first step is to identify the traffic lights' positions in the world. For that, the log is played offline, while a traffic light detector locates the traffic lights in the image frames. Additionally, the LiDAR points are projected to camera coordinates, and those that "hit" inside any traffic light bounding box are accumulated in a buffer. The points are projected using the same process as presented in [10]. When eight frames have passed without a single detection, the accumulated point cloud is clustered using the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) algorithm [44]. The world position of the cluster centroids are used as traffic lights' positions. The accumulated point cloud is then reset, and the process repeats throughout the duration of the log. Finally, the centroids are manually filtered to discard false positives and traffic lights that are not relevant for the RDDF of interest. Moreover, traffic lights that share the same control semantics can be grouped together, thus their redundancy can come to use, i.e., when the state of one of them can not be determined, the state of the other is used instead. Traffic lights that are relevant for a particular route are grouped considering a maximum distance threshold (of 20 meters) from each other. One

important feature is that the annotated traffic lights can be transferred from one RDDF to the other, for example, when two RDDFs follow the same path (maybe in different lanes). Additionally, the traffic lights that were discarded for not being relevant for one RDDF could be reused for the prior map of another RDDF where they are relevant.

This clustering strategy has to cope with some difficulties: (i) TLs are small objects and LiDAR rays are sparse, thus, in some cases, only few rays will hit a specific TL or perhaps none at all; (ii) small imprecisions in the localization system may cause disturbances in the accumulated point cloud; (iii) if the BB is a little larger than the TL object, then some LiDAR rays may hit inside the BB, but they actually miss the real TL object, hitting something on the background instead. Thus, it is better to train the detector for the offline phase with tight BB annotations, so that BB proposals will also be tight. All of these difficulties, may cause the clustering algorithm to yield false positives and also some inaccurate TL positions. Therefore, they need to be manually filtered as mentioned. In case of any problem in the final accumulated points, there is always the possibility of choosing a single frame for extracting the traffic light world position.

3.4 General System Operation

When driving autonomously, IARA continuously checks for the relevant traffic lights associated with the RDDF. Whenever a group of relevant TLs comes within 100 meters, their 3D world positions are projected onto camera coordinates. Additionally, the model of choice (either single or compound model) is triggered for the current frame. Each annotated TL position is surrounded by a sphere of 1.5 meter of radius that is also projected to serve as a threshold for the localization error. The euclidean distance, in camera's coordinates, between each BB's center and each projected TL is calculated. Any BB that has its center outside of all projected spheres are discarded right away. From the remaining ones, that with the closest center to any of the projected TLs is selected, and its status serves as the final prediction for that frame. The final state output of the system is either one of the five: none, when there is no traffic light within a 100 meters; off, when there is a traffic light, but the state could not be determined; red, yellow or green, when the state could be determined. When using the single model , red-yellow is ultimately considered as red. In cases in which all BBs are discarded, the final state is set to "off". Before a red, yellow or off traffic light, IARA (according to *Behavior Selector*'s planning) will reduce its speed until a complete stop if necessary. Otherwise, before a green traffic light (or none at all), it will continue its trajectory.

4 Materials and Methods

In this chapter, the datasets for traffic light detection and classification of state are introduced, as well as, the driving logs for testing on real world data, and the metrics used in the experiments.

4.1 Traffic Light Datasets

The training and evaluation of the YOLOv3 detector leveraged two public available datasets: DriveU Traffic Light Dataset (DTLD) [45] and LISA Traffic Light Dataset (LISA-TLD) [3]. IARA Traffic Light Dataset (IARA-TLD), a dataset created using IARA, was also used. Additionally, datasets for traffic light classification were derived from LISA-TLD and IARA-TLD aiming at training and evaluating the classification model.

4.1.1 Datasets for Traffic Light Detection

The DriveU Traffic Light Dataset (DTLD) comprises traffic image sequences from 11 German cities. Each image (2048×1024 pixels) includes the following annotations about traffic lights: bounding box coordinates, state, relevance status (visually defined by annotators), horizontal or vertical orientation, occlusion, number of lights, type of traffic light (for pedestrians, cyclists, or cars), and other attributes. For our purposes, pedestrians and cyclists traffic lights were removed. From the remaining, only active (i.e., not "off") traffic lights which are facing the car and have at least three bulbs were selected. The images were cropped to a 1280×960 rectangle aligned to the top and centered on the horizontal axis. The bounding box annotations were changed accordingly to reflect the cropping. The dataset comes with a training-test split.

The LISA Traffic Light Dataset (LISA-TLD) was developed by the Laboratory for Intelligent and Safe Automobiles (LISA) at University of California. The dataset was created using a Point Grey's Bumblebee XB3 (the same model used in IARA) to record more than 44 minutes of traffic video sequences in San Diego, California, USA. Some video sequences were recorded during the day, and others during the night. The dataset is split into training and testing sets, and has seven TL classes: "go", "go forward", "go left", "warning", "warning left", "stop", and "stop left". The classes "go", "go forward", and "go left" were merged as "green"; "warning", "warning left" as "yellow"; and the other tow classes as "red".

For preliminary evaluation of the detection model, we used a local dataset named (in this work) IARA Traffic Light Dataset (IARA-TLD). It comprises images of traffic

scenes recorded in Vitória, Espírito Santo, Brazil with help of the IARA's camera. The dataset has a total of 5002 bounding box annotations distributed across three different classes: "green", "yellow", "red", and "off". Only "green", "yellow" and "red" annotations were used in this work.

Figure 13 – Image samples for DTLD, LISA-DTLD and IARA-TLD.

Microsoft's COCO[26] is a large dataset for object detection, segmentation and captioning. It is widely used as a benchmark for object detection, comprising samples for 80 categories of objects grouped by 11 super-categories, like food, accessories, animals, vehicles, sports, outdoor objects, etc. The dataset includes images and bounding box annotations for traffic lights, however there is no information on their statuses. A YOLOv3 model trained on COCO will be used in our experiments (YOLO-TL).

Table 1 – Distribution of the annotated traffic lights across the datasets' splits.

Class	DTLD (train)	DTLD (test)	LISA-TLD (train)	LISA-TLD (test)	IARA-TLD
Red	19096	8374	22084	9846	1813
Yellow	1840	773	1045	457	266
Green	33540	14403	14681	7717	2923

Table 1 shows the class distribution of bounding box across DTLD, LISA-TLD, and IARA-TLD, and some image samples can be seen in Figure 13. These three datasets were used for training and evaluating our YOLOv3 detector (YOLO-RG), but all images and annotations were scaled down to 640 × 480 pixels, and the classes red and yellow were merged into a single class: red-yellow.

4.1.2 Datasets for Traffic Light Classification

In order to train and validate the RF classifier, a dataset derived from LISA-TLD was created. The ground truth bounding box annotations from LISA-TLD were used to crop the original images, thus producing image crops that contain a single traffic light occupying most of the image. Each of these crops were labeled with the state of the traffic light they contain (red, green, or yellow), rescaled to a resolution of 10×30, and saved as separate images. The produced dataset will be referred to as LISA-GT-CROPS.

Two similar datasets were created from IARA-TLD, for testing purposes. For these, the bounding box predictions from YOLO-RG and YOLO-TL were used to make the crops, resulting respectively in the datasets named IARA-RG-CROPS and IARA-TL-CROPS. This was done with the intent of creating test datasets that will better resemble the input data the classification model will receive when in operation (i.e., when combined with the YOLO detector). When making the crops, only the predictions that have a confidence score higher than 0.01, and that yield IoU greater or equal to 0.4 with any ground truth box, were considered, and the class labels were drawn from their respective ground truth boxes (instead of from the predictions themselves). Table 2 indicates the number of images per class across the classification datasets, and Figure 14 displays some image samples.

Table 2 – Class distribution across the classification datasets.

Class	LISA-GT-CROPS (train)	LISA-GT-CROPS (test)	IARA-TL-CROPS	IARA-RG-CROPS
Red	22084	9846	1698	1807
Yellow	1045	457	238	187
Green	14681	7717	2385	2755

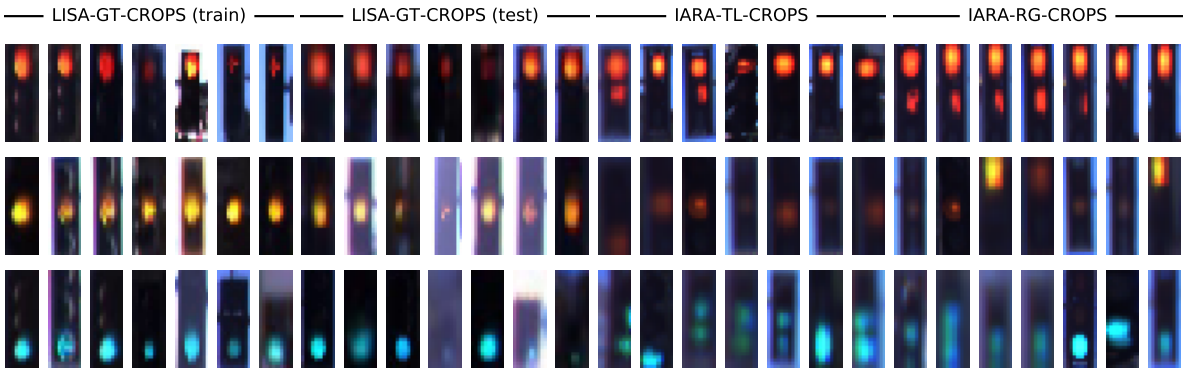
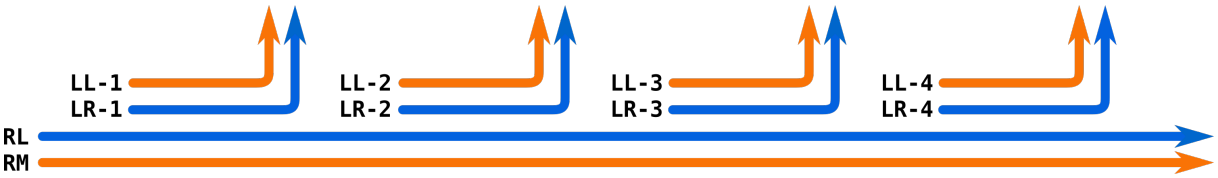


Figure 14 – Image samples for LISA-GT-CROPS, IARA-TL-CROPS and IARA-RG-CROPS.

4.2 Driving Logs

Driving logs are files containing detailed sensory information recorded during the car's trips, and serve to reproduce the IARA's sensors offline. These logs can be revisited (playback operation) as many times as needed, which allows testing offline modifications on IARA's software, and evaluating the car's new behavior. A total of 10 logs were recorded on Dante Michelini avenue (Vitória, Espírito Santo, Brazil), from which five were used to construct the prior maps, and the other five for full system evaluation. This avenue was chosen because it contains challenging situations, i.e., several traffic lights where some of them are potentially discordant. There are four bifurcations, each one with three lanes that go straight, and two other lanes (the two leftmost) that turn left. When turning left, the driver should take one of the two leftmost lanes and obey their corresponding traffic lights. Otherwise, the driver can keep on the three rightmost lanes ruled by another set of traffic lights. Regardless of the current lane, all traffic lights are visible to the camera most of the time. Figure 1 depicts this ambiguous situation.



(a) Simplified view of all logs. The figure does not contain real geographic information, i.e., the lengths and geometry of each path are not accurately depicted.

(b) Satellite photo of the first bifurcation. The two bottommost logs go through the entire avenue, while the other ones are local to that bifurcation. From top to bottom the logs depicted are: LL-1, LR-1, RL, RM.

Figure 15 – Representation of the logs' trajectories. A tube map of all logs is shown on (a), and (b) depicts the logs' trajectories that appear on the first bifurcation. Each path drawn represents a log. The orange paths represent logs that were used for creating prior maps, while the blue ones were used for testing the system.

Figure 15a displays, through a simplified view, the trajectory of every log recorded.

And Figure 15b shows a satellite photo depicting only the first bifurcation, and dashed arrows are used to represent the logs' trajectories. There are two logs that follow the avenue from beginning to end, these are identified as RL and RM. From the car's point of view, RL is the one on the left lane, and RM goes along the middle lane. At each of the four bifurcations there are two extra lanes for turning to the left: the leftmost lane is identified as LL, and the rightmost as LR. Each numbered according to their bifurcation.

4.3 Metrics

The detection performance was measured in terms of Mean Average Precision (mAP), Average Precision (AP) per class, and precision-recall curves. The calculation of these metrics follow the definition from the Pascal VOC 2012 competition [46, 47]. Essentially, mAP is the mean of all APs (one per class), which, in turn, are the values corresponding to the area under their respective precision-recall curves. An IoU (Intersection over Union) threshold of 0.5 is used to compute all these metrics.

To measure classification performance, macro-averaged accuracy and per class accuracy will be used. Samples belonging to the "yellow" class constitute only roughly 5% of the classification datasets used for testing. Due to this heavy class imbalance, the misclassification of "yellow" samples would contribute very little to a micro-averaged accuracy, and would thus hinder our ability of choosing a model that learned to equally classify each class correctly. The accuracy per class will aid in inspecting what the deficiencies of the classification models are.

For the full system evaluation, the state of the relevant traffic lights was predicted for all the frames of each test log. We reported the confusion matrix resulting from comparing the predictions with the ground truth annotations. Additional information was recorded (for each bifurcation/lane) in order to verify how early the system correctly perceived the traffic lights: the time the system took to produce the first correct prediction since the car entered the 100-meters range to the next set of traffic lights; and the car's distance to them when this occurred.

5 Experiments and Results

The experiments for this work comprise the training, evaluation and selection of the models that will be used, and also the final evaluation of the full TLR system on the driving logs.

It is important to notice that these experiments are limited in scope, and their results cannot be transferred to any other conditions for which they were not designed. For example, the data used in the experiments are composed only of images recorded during the day, with a clear sky. Therefore, it is unclear how the system is going to behave during the night, rain, snow, and other environments. Additionally, the experiments on real world data comprise only a few local tracks which are all very similar in many aspects, hence more extensive testing on different tracks that present a larger variety is necessary to ensure the safety and performance of the proposed system. Also, different neural networks could be studied with the aim of achieving yet higher performance.

5.1 Training the Single Model Detector

Currently there are easily available pretrained models that can detect traffic lights, such as those trained on the COCO dataset. But, usually, these do not identify the traffic light's state, so we trained our own model.

Our YOLOv3 detector (YOLO-RG) was trained for detecting two classes only: red-yellow or green traffic lights. DTLD's and LISA-TLD's train sets were used for training, and their test sets were used for validation. The model was trained for 15000 batches, with 64 images per batch, and a constant learning rate of 10^{-4}. The input resolution was 608×608 pixels as a compromise between inference time and accuracy. Many of the default parameters were also kept, such as: image augmentation with changes in hue, saturation, and exposure; batch normalization; and default anchors. Additionally, YOLO was allowed to change the input resolution from 608×608 to different resolutions following multiples of 32 starting from 320 (e.g., 320, 352, ..., 608) every 10 batches, as it is done in its original work. This, supposedly, makes the model more robust to different scales.

The YOLO-RG model was evaluated on IARA-TLD, the resulting precision-recall curves, APs, and mAP can be seen in Figure 16. The APs for the red-yellow and green classes are 80.08% and 68.14% respectively. The precision-recall curve is formed by considering bounding boxes with decreasing values of confidence scores. While this happens, it is usual for precision to decrease and recall to increase. From the precision-recall curves shown in Figure 16, one can see that both classes achieve good recall while maintaining descent

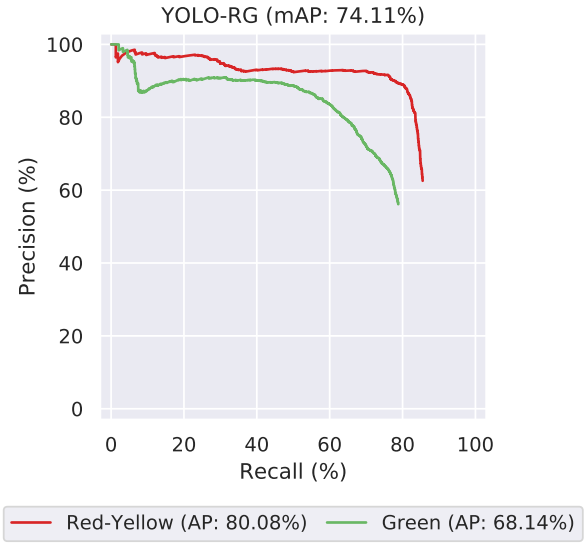


Figure 16 – Detection metrics for the model YOLO-RG on IARA-TLD (the detection test set).

precision rates, specially the red-yellow class.

Taking an image of 608×608 pixels as input, YOLOv3 takes about 47 milliseconds to make proposals on IARA's video card (Nvidia's Titan Xp). This is roughly 21 Hz, but this is capped by the low frequency of our camera: 16 Hz.

5.2 Comparison of Traffic Light Detectors

Any traffic light detector could be used for the first stage of the compound model. Besides YOLO-RG, a generic traffic light detector (YOLO-TL) is also considered: the YOLOv3 model pretrained on COCO[1]. This model detects traffic lights, but does not identify their states. And although it also detects many other classes (e.g., people, cars, dogs), due to the scope of this work, only the traffic light class will be considered.

In order to compare between YOLO-RG and YOLO-TL, solely regarding the detection of traffic lights (no state classification), the precision-recall curve and AP were used. The test was performed once again on IARA-TLD, and the results are shown in Figure 17. From the figure, it is noticeable that YOLO-TL's precision rapidly decreases, but YOLO-RG's is steady for the most part. This is probably due to the fact that YOLO-TL was trained on COCO, which includes traffic lights that are sideways, backwards, off, and even pedestrian and cyclist's traffic lights. Although IARA-TLD has instances of such traffic lights, they are not annotated, so whenever YOLO-TL predicts them, its precision decreases. Detecting off traffic lights could lead to complications, because YOLO-TL will operate in conjunction with a classifier that will assign either a red, yellow or green state

[1] <https://pjreddie.com/media/files/yolov3.weights>

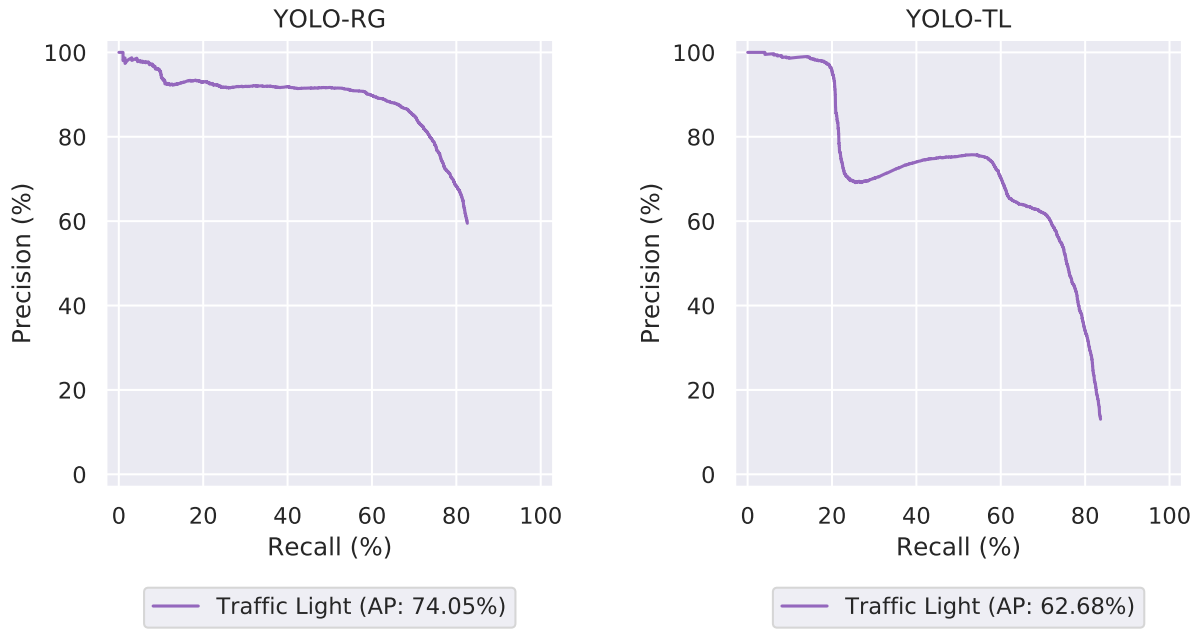


Figure 17 – Comparison of YOLO traffic light detectors, YOLO-RG and YOLO-TL, on IARA-TLD. Only raw traffic light detection is considered, not state classification.

randomly to it, and confusing an off traffic light for a green one could lead to serious accidents. Unfortunately, our experiments on logs do not include off traffic lights, so the systems' behavior in this situation can not be further investigated. Yet, the other false positives do not represent a problem for the proposed system, because almost all of them will be filtered out using information from prior maps.

5.3 Training the Classifier Model

A Random Forest model was trained for classifying traffic lights' states. LISA-GT-CROPS' train split was used for training, while its test split was used for validation, and IARA-RG-CROPS and IARA-TL-CROPS were used for testing. One can notice that the training set is heavily imbalanced, and since the model's performance for the yellow class is important for our goals, a balancing step is necessary. We opted for balancing the training set by under-sampling, in light that RFs do not require too many samples for training. When under-sampling, excessive samples from the more abundant classes are removed in a way that all classes will have equal number of samples. This process results in 1045 samples per class for the training set. The validation and test sets are left untouched.

In order to find the best combination of hyperparameters for training the model, a grid search was performed for the most important ones. The hyperparameters and their possible values are:

- *The number of trees (NTREES)*: any value in $\{50, 100, 150, 200\}$.
- *The maximum depth of the trees (MAXD)*: any value in $\{3, 5, 7, 10\}$.
- *The minimum number of samples necessary for a split to occur (MINSC)*: any value in $\{2, 5, 10\}$.
- *The number of features used to find the best split on decision nodes (ACTVC)*: $\sqrt{M}$ (OpenCV's default) and $\lfloor \log_2 M + 1 \rfloor$ (as in [31]) are used, where M is the number of input features. The images fed to the RF model have a resolution of 10×30 and 3 channels, so $M = 900$, therefore $\sqrt{M} = 30$, and $\lfloor \log_2 M + 1 \rfloor = 10$.

Table 3 – Hyperparameters for the best RF Model.

MAXD	MINSC	ACTVC	NTREES
10	2	30	150

Table 4 – RF results. Accuracies are in percentages (%).

Dataset	acc. (red)	acc. (green)	acc. (yellow)	acc. (macro)
LISA-GT-CROPS	92.59	89.94	100.00	94.18
IARA-TL-CROPS	92.81	96.85	81.09	90.25
IARA-RG-CROPS	84.61	98.37	49.73	77.57

The models from this grid search were ranked according to macro-averaged accuracy. The best performing model was selected to be used for the second stage of all compound models. Its hyperparameters are shown in Table 3, and its results on the validation set (LISA-GT-CROPS) and the test sets (IARA-TL-CROPS and IARA-RG-CROPS) are in Table 4.

The validation accuracy of the yellow class reached 100%, this is probably a coincidence due to the fact that yellow samples make up for less than 3% of the total in the validation set. The accuracies of the classes red and yellow are a lot lower for IARA-RG-CROPS when compared to IARA-TL-CROPS, especially for the class yellow. Consequently, the dataset's overall accuracy is lowered too. We suspect the bounding box predictions from YOLO-RG for the class red-yellow are coming more unstable, i.e. higher regression error on bounding box coordinates, specially when the traffic light's real status is yellow. This, in turn, might have yielded image crops that are more difficult for the RF to classify correctly.

An interesting analysis can be made. Each decision node in the model looks at a single feature of the image in order to make its decision, which is the value of a single

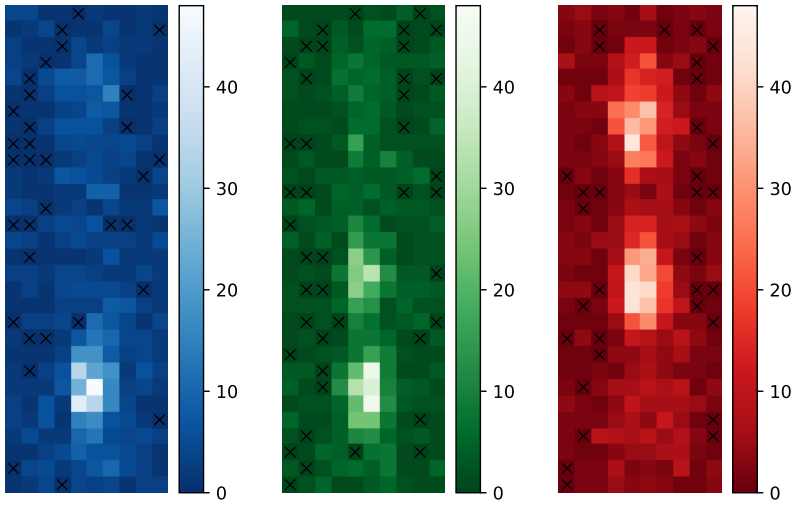


Figure 18 – A plot of how much attention the chosen RF model has for each pixel across the three color channels. The three images shown represent the three channels of any color image. Blue, green and red, respectively. The colors' brightness represent the number of decision nodes which are using that feature in order to make their decisions. Brighter values indicate more decision nodes are looking at that specific feature. The X's mark the features which receive no attention at all.

pixel in a single color channel. By counting how many decision nodes are looking at a feature, it's possible to see what the model is paying attention to. Figure 18 displays this visually. In the figure, there are three images representing the three color channels, and the brightness of each pixel represents how many decision nodes are using that specific feature to make their decisions. Pixels that are not used by any decision node are marked with X's to aid visualization. The brighter pixels in the figure are positioned where the three colored lights of a traffic light usually are, which is evidence that the model is paying attention to what it should. Their arrangement and distribution across the three channels also happens as expected. There is much attention on the red channel at where the red light is expected to be; on the red and green channels at where the yellow light is; and on the blue and green channels at where the green light is. Blue and green, because the green lights found in our classification datasets are usually not pure green, but *teal* instead (this can be confirmed by inspecting Figure 14 again). Also, almost all features are used by at least one decision node, even though there is more focus where the colored lights usually are. Only 107 features out of 900 (about 11.89%) receive no attention at all, and thus have no contribution for the model's final predictions. Still, the model's focus on where the lights usually are could present a problem. Since Random Forests are not invariant to translation, scale or rotation in the images, if a traffic light is positioned in the image differently from what is expected, the model's predictions could be unreliable.

Figure 19 displays some examples of images from IARA-TL-CROPS and IARA-

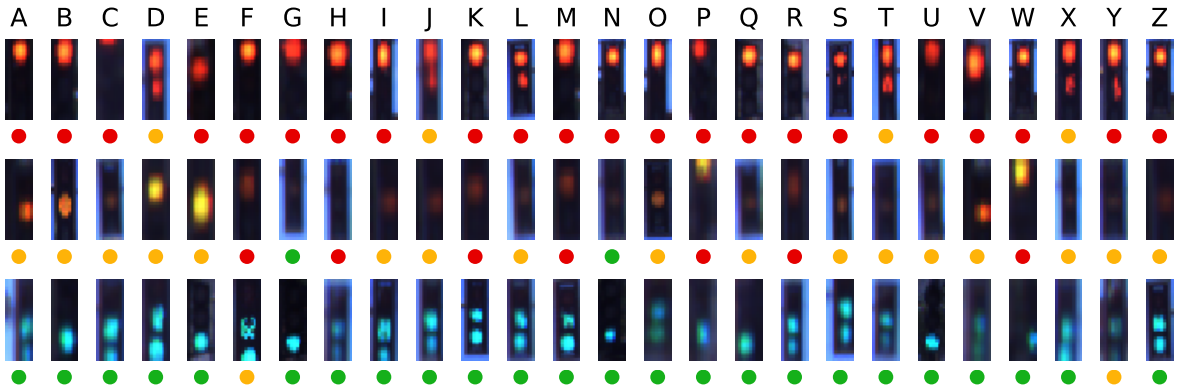


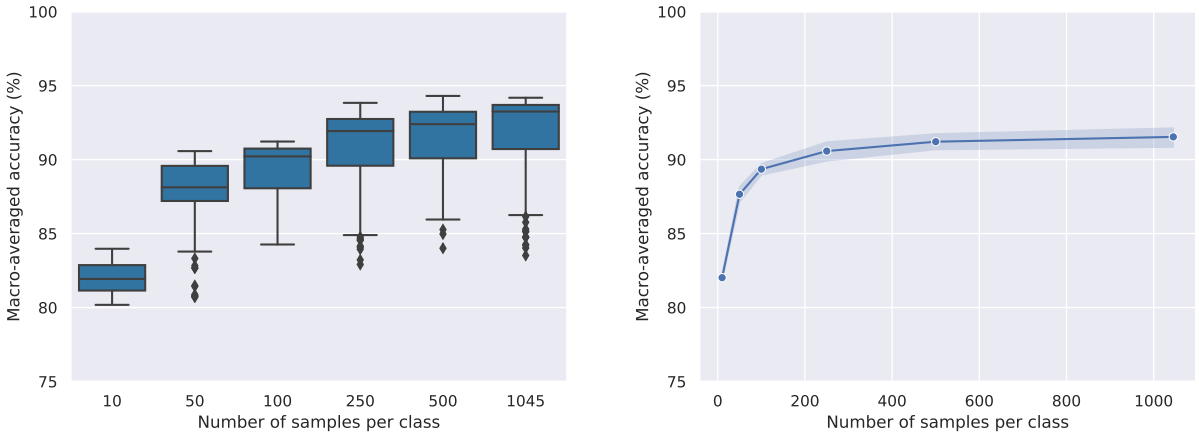
Figure 19 – RF predictions on randomly selected images from IARA-TL-CROPS and IARA-RG-CROPS. The examples are arranged in rows according to their GT class, from top to bottom: red, yellow, and green. The circle below each example has a color that represents their prediction. The letters on the top of the figure are used to reference the columns.

RG-CROPS, and what are the RF's predictions for each one of them (the colored circles below each image). In the figure's first row, there are examples of red traffic lights only. Most of them were classified correctly, but a few of them (columns "D", "J", "T" and "X") were misclassified as yellow. This could be due to the RF confusing their counters with yellow lights, even though there are other red traffic lights with counters that were classified correctly. In the second row (yellow traffic lights only), the TLs in "G", "H" and "N" were misclassified, probably because their lights are too dim. "F", "K", "M", "P", "R", and "W" were probably confused with red TLs because their yellow light is positioned more to the top of the image (easier to observe on "P" and "W"), instead of at the center, so the RF "thought" they were red lights. This awkward positioning of the yellow light can happen when the bounding box does not fit the traffic light very well, which causes a portion of the traffic light to be left out of the crop. In the third row (green traffic lights only), most of them were classified correctly, and it is difficult to identify why "F" and "Y" were classified incorrectly. For "Y", it could be because the light from the counter (at the image's center) is stronger than the green light at the bottom, which caused the RF to predict it as yellow.

5.4 Learning Curve Analysis of the Classifier Model

As it was mentioned in the previous section (5.3), after under-sampling the training set, we opted for using the maximum numbers of samples possible (1045 samples per class). This number was confirmed to be a good choice through a learning curve analysis, for which it is observed how the macro-averaged accuracy is impacted by increasing the number of samples per class used for training. The same grid search operation described in Section 5.3 was performed using different numbers of samples per class for training the

models (using values from the set $\{10, 50, 100, 250, 500, 1045\}$). Through this analysis, it is possible to understand how many samples it would be necessary for training the model appropriately.



(a) Box plot showing the performance of models (y-axis) across different numbers of samples per class used for training (x-axis).

(b) Learning curve of the classifier model. The graph shows the estimated mean macro-averaged accuracy (y-axis) for each number of samples per class used for training (x-axis). The bands are confidence intervals of the estimated means, using a confidence level of 95%.

Figure 20 – Performance of Random Forest models trained with increasing numbers of samples per class (x-axis on both graphs). The performance was measured using macro-averaged accuracy (y-axis on both graphs) on the validation set (the test split of LISA-GT-CROPS).

Figure 20a displays a box plot that summarizes the performance of the models (y-axis; using macro-averaged accuracy) that resulted from the grid searches using different values of samples per class (x-axis). The box plot used is a standard one, i.e., the blue boxes represent the interquartile range (IQR), and the whiskers extend to the minimum and maximum values within 1.5 times the IQR. The horizontal line within the blue box marks the median of the distribution, and the black diamond markers are outliers, i.e., they fall out of the $1.5 \times IQR$ range. Using the same underlying data, Figure 20b uses a line plot to display the estimated mean performance (y-axis) of the models that were trained with the same number of samples per class (x-axis). As it can be seen in both figures, using more samples seems to always bring gains in accuracy, but the gains are progressively diminished the more samples are used, mainly after 250 samples per class.

5.5 Evaluation of Compound Models

The compound model is composed of a YOLO detector for its first stage, and the RF classifier for the second one. Consequently, there are two compound models in this

work, one that uses YOLO-RG in the first stage, and another that uses YOLO-TL. For the second stage, the selected model from Section 5.3 was always used. The performance of both compound models was evaluated on IARA-TLD. The standard detection metrics apply: mAP, APs, and precision-recall curves. To compute these metrics, a confidence score per bounding box is necessary. As such, the objectness multiplied by the RF's winning class' probability was used for that purpose, which is similar to how YOLOv3 handles this internally. From both models, only that with highest mAP will be selected for the experiments on logs, since such experiments are much time-consuming. The performance metrics of each compound model when tested on IARA-TLD are displayed in Figure 21.

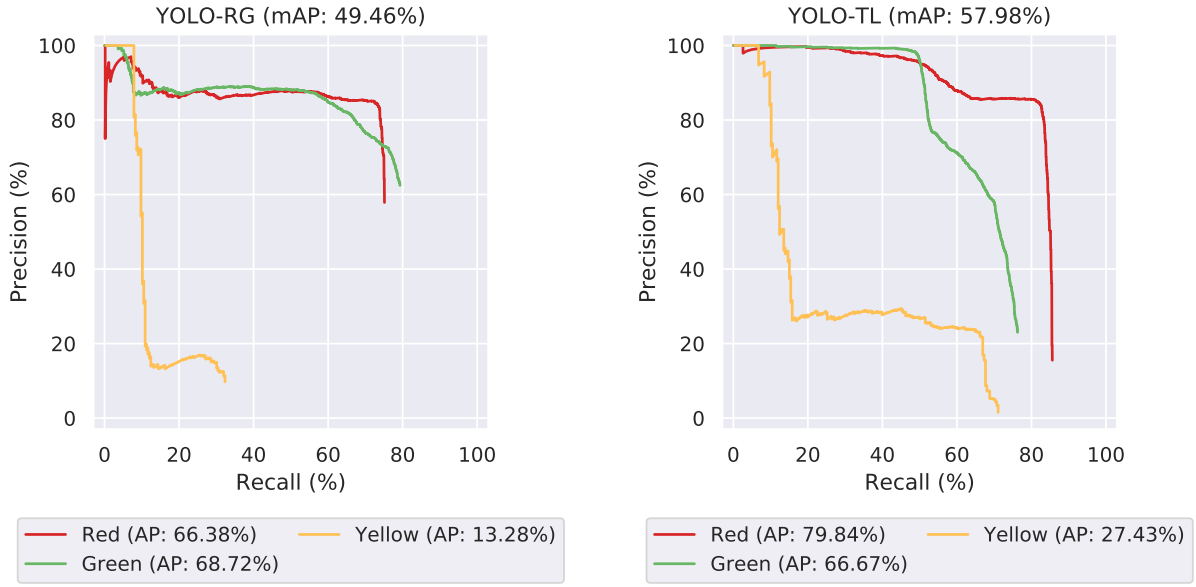


Figure 21 – Results for the compound models on IARA-TLD. Each sub-figure presents the result of one of the YOLO traffic light detectors when combined with the Random Forest state classifier.

For both models, it is noticeable how the detection of yellow traffic lights is worst when compared to the other classes. The unusual "L" shape of their precision-recall curves indicate that the compound models do not assign relevant confidence scores to predictions of this class specifically. Yet, YOLO-TL's precision-recall curve for the class yellow reveals some potential because it reaches descent recall values, even though, with little precision. Also, the class red's AP is higher when using YOLO-TL, instead of YOLO-RG. Considering these remarks, YOLO-TL is the most promising choice of first stage detector for a compound models in almost every aspect. Therefore, only YOLO-TL will be used for the compound model in the next section (Section 5.6).

5.6 Experiments on Logs

The test logs on avenue Dante Michelini were used to evaluate the overall system performance. RM was used to build the global occupancy grid-map of the region (Section 3.1). RM and LL-{1-4} were utilized for creating prior traffic light maps for the test logs, which were RL and LR-{1-4}. For the system evaluation, both single model (SM) and compound model (CM) are considered for recognizing traffic lights. A confidence threshold of $\tau = 0.2$ was used during YOLO's inference for both SM and CM; all bounding boxes with confidence scores below that value are discarded. The threshold value was chosen empirically, using the logs which are not utilized for testing. In the CM, the bounding boxes that pass through the threshold will be reclassified by the RF model, as normal. For each test log, the final predictions of our system are compared with the ground truth at every frame. Both the single model (SM) and the compound model (CM) are compared in this evaluation. Figure 22 shows this comparison in the format of a timeline. It displays, over time, the car's velocity, the ground truth (GT) state of the closest traffic light, and the predictions from the single model (SM) and compound model (CM). As time progresses, the car approaches different groups of relevant traffic lights. For all test logs, there are some frames without traffic lights ("none") within a hundred meters. As it can be seen, most of the predictions are consistent with the ground truth, and the inconsistencies usually occur when the car is first approaching a traffic light. The first correct predictions are represented on the figure as amber downward triangle markers. And, although they usually are representative of when the system begins to, consistently, make correct predictions for a TL group, that is not always the case (e.g., in LR-2 for the CM, it makes its first correct prediction quite early, but consistently gets it wrong until roughly 31 s). Table 5 also reports the first correct predictions, but for the shorter logs only (i.e., no RL). On this table, it is possible to see the time the system took to first predict the traffic light state correctly, and the distance remaining until the traffic light is reached. On LR-4, two different groups of TLs appear at different moments, so the table has two entries for LR-4. When using the SM, the first correct predictions occur in average with 0.65 seconds of delay and with around 93 meters of distance, whereas when using the CM, it occurs in average with 1.53 seconds of delay and with around 84 meters of distance.

Throughout the logs, the car's velocity ranged from 0 to 52 km/h (the speed limit on the avenue is 60 km/h). When using the CM, red traffic lights at longer distances (and occasionally at mid-distances) are often confused with yellow ones (e.g., LR-{1,2,3}). Also, in RL at 87 s, a red TL was still being predicted as yellow even after the car stopped. Sometimes, green is also confused with yellow (e.g., LR-4 at around 7 seconds). There are only three small sequences of yellow traffic lights (in LR-4 around 67 s, RL 40 s and 85 s), and they were often predicted correctly. The SM does not make a distinction between red and yellow traffic lights, so yellow traffic lights are considered red. Both SM and CM

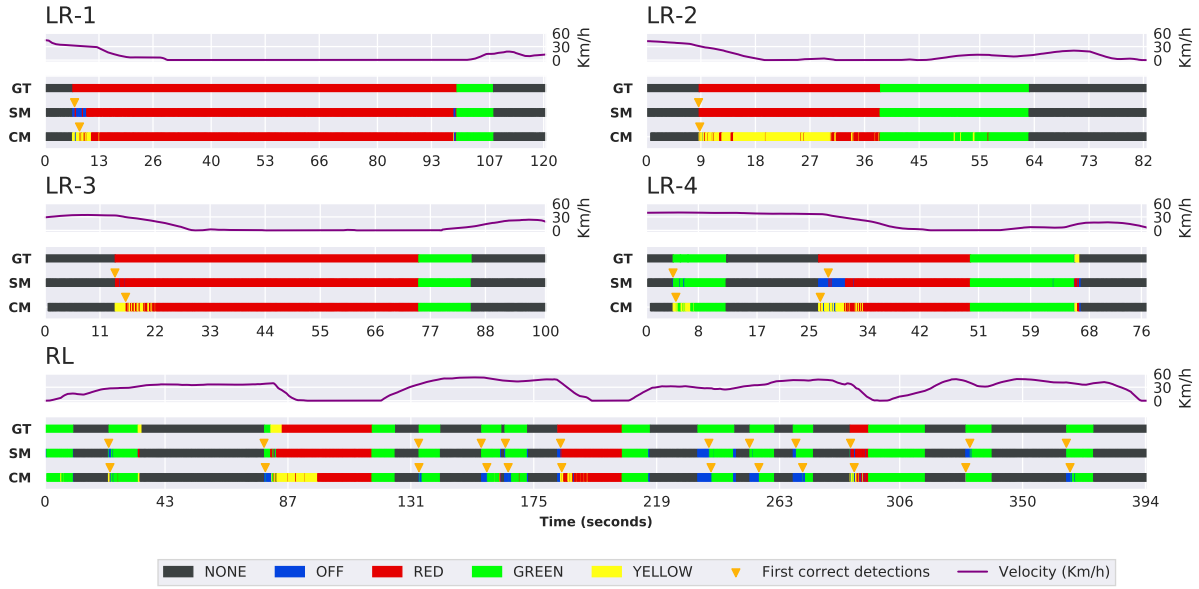


Figure 22 – Summary of results on each test log. The figure displays, for each log: a graph with a purple line that expresses the car's velocity (y-axis, in Km/h) over time (x-axis, in seconds); and a graph that displays, from top to bottom, the ground truth (GT) and the predictions from the single model (SM) and compound model (CM) over time. The earliest correct predictions for SM's and CM's timelines are highlighted with amber triangles.

Table 5 – First Correct Predictions on Logs.

	Log	Delay (seconds)	Distance (meters)
SM	LR-1	0.56	94.47
	LR-2	0.00	99.70
	LR-3	0.06	99.63
	LR-4 (I)	0.07	98.42
	LR-4 (II)	1.57	84.12
CM	LR-1	1.74	83.96
	LR-2	0.19	98.17
	LR-3	2.19	81.04
	LR-4 (I)	0.51	93.56
	LR-4 (II)	0.34	95.69

had difficulty with green traffic lights at long distances throughout RL, predicting "off" instead of "green", but SM performed significantly better in these cases (the only notable exception being the penultimate group of traffic lights in RL, starting at around 325 s). It is worth noting that a truck passed in front of the traffic light in LR-1 (right before the "green" state) resulting in a occluded traffic light that was not annotated, but can be seen in the sequence. Table 6 and 7 report the confusion matrices for the system using SM and CM, respectively, so that the reader can have a better idea of the overall prediction

performance of the system.

Table 6 – Confusion Matrices for Each Testing Log (SM).

	LR-1					LR-2					LR-3					LR-4					RL				
	N	O	R	G	Y	N	O	R	G	Y	N	O	R	G	Y	N	O	R	G	Y	N	O	R	G	Y
NONE (N)	273	0	0	2	0	432	0	0	0	0	433	0	0	0	0	420	3	0	0	0	2988	29	0	144	0
OFF (O)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
RED (R)	0	42	1294	0	0	0	0	467	1	0	1	2	914	0	0	0	61	287	0	0	0	33	912	2	0
GREEN (G)	0	0	0	128	0	1	0	0	388	0	3	0	0	156	0	1	5	0	351	0	10	159	1	1753	0
YELLOW (Y)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	9	0	0	10	4	54	10	0

Table 7 – Confusion Matrices for Each Testing Log (CM).

	LR-1					LR-2					LR-3					LR-4					RL				
	N	O	R	G	Y	N	O	R	G	Y	N	O	R	G	Y	N	O	R	G	Y	N	O	R	G	Y
NONE (N)	280	0	0	2	1	423	0	0	0	0	426	0	0	0	0	414	3	0	0	0	2989	20	0	136	8
OFF (O)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
RED (R)	0	9	1284	1	59	0	1	135	1	331	0	0	842	0	75	0	7	272	0	69	0	35	641	4	267
GREEN (G)	0	0	0	128	0	1	0	2	377	9	3	0	0	156	0	1	0	0	325	31	9	373	10	1517	14
YELLOW (Y)	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	6	10	6	6	9	47

For the SM (Table 6), it made correct predictions for almost the entirety of LR-2 and LR-3, except for a few mistakes. In LR-1 and LR-4, there are a reasonable amount of red TLs being predicted as off, and they occur at the very beginning of the TL groups, as can be seen in the timeline figure (Figure 22). This indicates that, at times, the YOLO model trained for SM is having trouble to detect traffic lights at longer distances. In RL, there a lot of incorrect predictions accumulated, since it is a very long log. Those that draw more attention are: many green (also red) TLs predicted as off; some yellow TLs predicted as green (the model was trained to predict them as red); none predicted as off or green; and, green and yellow predicted as none. The two last ones occur due to small differences from the localization system during the different playbacks (SM, CM, or GT). And they are very difficult to spot on the timeline, because they are very small (it is easiest to identify this in the second TL group of RL, although, still difficult). For the CM (Table 7), in LR-{1,2,3,4}, the most common type of error was confusing red TLs for yellow ones (also, green TLs for yellow ones in LR-4). This indicates that the state classifier (the RF) has a lot of room to improve in this aspect. The same thing happens in RL, which also accumulates a lot of incorrect predictions over time, most of them similar to the SM.

When the car approaches a group of relevant traffic lights, there are two major types of error that could occur. The first is predicting the traffic lights as red, yellow or off when they are actually green, which represents an inconvenience since it will cause the car to slow down to a stop when it could have proceeded. The second type of error is predicting green traffic lights while facing red ones (or off). This is by far the worst since it could lead the car into the middle of a busy intersection and cause accidents. In our experiments, the second type of error was very rarely observed (nothing special was

done during training, or any other stage, for that to happen), they are not visible on the timeline but can be seen on the confusion matrices.

For qualitative analysis, a video for the system running with the SM can be seen at <https://youtu.be/VhdLpuErJ8E>. The code, trained models, usage instructions and all the logs used in this work can be found at <https://github.com/LCAD-UFES/carmen_lcad/blob/master/src/traffic_light_yolo/README.md>.

6 Conclusions

We proposed a system for autonomous vehicles that combines deep learning and prior maps for traffic light recognition. First, traffic light detection and classification of state is performed. For this first step, a single YOLOv3 model or a compound model (YOLOv3 detector combined with Random Forest state classifier) were considered. Subsequently, prior maps are used to select only relevant traffic lights from the proposed detections. Additionally, a new method for creating prior maps is proposed, in which LiDAR points are projected to camera; points that hit inside detected bounding boxes are accumulated; and finally this point cloud is clustered to propose traffic lights' position in world coordinates. Even though results are promising, there are many areas for improvement, for example:

- A preliminary investigation has demonstrated that simply increasing YOLO's input resolution during inference time has the potential to deliver better results. But we could not explore this further due to the limited time available for the completion of this work.
- Investigate the reduction of the search space for traffic lights by proposing ROIs in the input image based on prior maps annotations (which is common in literature). This approach has two main potential benefits: saving processing time, or, alternatively, enabling the use of yet higher resolution images.
- Employ tracking of bounding box predictions, which is much common in the literature of traffic light recognition. Tracking can help with bounding box coordinates, and also to make state predictions more stable.
- Some form of slope compensation is probably needed before this work can be applied to more mountainous regions. [37] proposes a solution with slope compensation to operate inside Korea.

Bibliography

1 FAIRFIELD, N.; URMSON, C. Traffic Light Mapping and Detection. In: *International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE, 2011. Cited 2 times on pages 13 and 24.

2 BARNES, D.; MADDERN, W.; POSNER, I. Exploiting 3D Semantic Scene Priors for Online Traffic Light Interpretation. In: *Intelligent Vehicles Symposium (IV)*. Seoul, South Korea: IEEE, 2015. p. 573–578. Cited on page 13.

3 JENSEN, M. B. et al. Vision for Looking at Traffic Lights: Issues, Survey, and Perspectives. *IEEE Transactions on Intelligent Transportation Systems*, IEEE, v. 17, n. 7, p. 1800–1815, 2016. Cited 3 times on pages 13, 23, and 34.

4 CHARETTE, R. D.; NASHASHIBI, F. Traffic Light Recognition using Image Processing Compared to Learning Processes. In: *International Conference on Intelligent Robots and Systems (IROS)*. St. Louis, MO, USA: IEEE, 2009. p. 333–338. Cited on page 13.

5 CHARETTE, R. D.; NASHASHIBI, F. Real Time Visual Traffic Lights Recognition Based on Spot Light Detection and Adaptive Traffic Lights Templates. In: *Intelligent Vehicles Symposium (IV)*. Xi'an, China: IEEE, 2009. p. 358–363. Cited on page 13.

6 SIOGKAS, G.; SKODRAS, E.; DERMATAS, E. Traffic Lights Detection in Adverse Conditions using Color, Symmetry and Spatiotemporal Information. In: *International Conference on Computer Vision Theory and Applications (VISAPP)*. [S.l.: s.n.], 2012. v. 1. Cited on page 13.

7 PHILIPSEN, M. P. et al. Traffic Light Detection: A Learning Algorithm and Evaluations on Challenging Dataset. In: *International Conference on Intelligent Transportation Systems (ITSC)*. Las Palmas, Spain: IEEE, 2015. p. 2341–2345. ISSN 2153-0009. Cited on page 13.

8 BERRIEL, R. F. et al. Deep Learning Based Large-Scale Automatic Satellite Crosswalk Classification. *IEEE Geoscience and Remote Sensing Letters*, v. 14, n. 9, p. 1513–1517, Sept 2017. ISSN 1545-598X. Cited on page 13.

9 BERRIEL, R. F. et al. Automatic Large-Scale Data Acquisition via Crowdsourcing for Crosswalk Classification: A Deep Learning Approach. *Computers & Graphics*, v. 68, p. 32–42, Nov 2017. ISSN 0097-8493. Cited on page 13.

10 GUIDOLINI, R. et al. Handling Pedestrians in Crosswalks Using Deep Neural Networks in the IARA Autonomous Car. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. Rio de Janeiro: IEEE, 2018. p. 1–8. ISSN 2161-4407. Cited 2 times on pages 13 and 32.

11 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*. Las Vegas, NV, USA: IEEE, 2016. p. 779–788. ISSN 1063-6919. Cited 2 times on pages 14 and 25.

12 WEBER, M.; WOLF, P.; ZÖLLNER, J. M. DeepTLR: A single Deep Convolutional Network for Detection and Classification of Traffic Lights. In: *Intelligent Vehicles Symposium (IV)*. Gothenburg, Sweden: IEEE, 2016. p. 342–348. Cited 2 times on pages 14 and 25.

13 BEHRENDT, K.; NOVAK, L.; BOTROS, R. A Deep Learning Approach to Traffic Lights: Detection, Tracking, and Classification. In: *International Conference on Robotics and Automation (ICRA)*. Singapore: IEEE, 2017. Cited 2 times on pages 14 and 25.

14 BACH, M.; STUMPER, D.; DIETMAYER, K. Deep Convolutional Traffic Light Recognition for Automated Driving. In: *International Conference on Intelligent Transportation Systems (ITSC)*. Maui, Hawaii, USA: IEEE, 2018. p. 851–858. Cited on page 14.

15 PON, A. D. et al. A hierarchical deep architecture and mini-batch selection method for joint traffic sign and light detection. *arXiv preprint arXiv:1806.07987*, 2018. Disponível em: <http://arxiv.org/abs/1806.07987>. Cited 2 times on pages 14 and 25.

16 JOHN, V. et al. Traffic Light Recognition in Varying Illumination using Deep Learning and Saliency Map. In: *International Conference on Intelligent Transportation Systems (ITSC)*. Qingdao, China: IEEE, 2014. p. 2286–2291. Cited 2 times on pages 14 and 24.

17 KIM, H.-K.; PARK, J. H.; JUNG, H.-Y. An efficient color space for deep-learning based traffic light recognition. *Journal of Advanced Transportation*, Hindawi, v. 2018, 2018. Cited on page 14.

18 BADUE, C. et al. Self-Driving Cars: A Survey. *arXiv preprint arXiv:1901.04407*, 2019. Disponível em: <https://arxiv.org/abs/1901.04407>. Cited on page 14.

19 POSSATTI, L. C. et al. Traffic light recognition using deep learning and prior maps for autonomous cars. *arXiv preprint arXiv:1906.11886*, 2019. Disponível em: <http://arxiv.org/abs/1906.11886>. Cited on page 16.

20 SCHMIDHUBER, J. Deep learning in neural networks: An overview. *arXiv preprint arXiv:1404.7828*, 2014. Disponível em: <http://arxiv.org/abs/1404.7828>. Cited on page 17.

21 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. [S.l.]: MIT Press, 2016. <http://www.deeplearningbook.org>. Cited on page 17.

22 HE, K. et al. Deep residual learning for image recognition. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. Las Vegas, NV, USA: IEEE, 2016. p. 770–778. Cited on page 18.

23 IANDOLA, F. N. et al. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and < 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016. Disponível em: <http://arxiv.org/abs/1602.07360>. Cited on page 18.

24 HOWARD, A. G. et al. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. Disponível em: <http://arxiv.org/abs/1704.04861>. Cited on page 18.

25 SIMONYAN, K.; ZISSERMAN, A. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. Disponível em: <http://arxiv.org/abs/1409.1556>. Cited on page 18.

26 LIN, T. et al. Microsoft COCO: common objects in context. *arXiv preprint arXiv:1405.0312*, 2014. Disponível em: <http://arxiv.org/abs/1405.0312>. Cited 2 times on pages 18 and 35.

27 REDMON, J.; FARHADI, A. YOLOv3: An Incremental Improvement. *arXiv preprint arXiv:1804.02767*, 2018. Disponível em: <https://arxiv.org/abs/1804.02767>. Cited on page 18.

28 LIN, T. et al. Feature pyramid networks for object detection. *arXiv preprint arXiv:1612.03144*, 2016. Disponível em: <http://arxiv.org/abs/1612.03144>. Cited on page 19.

29 HO, T. K. Random decision forests. In: IEEE. *Proceedings of 3rd international conference on document analysis and recognition*. Montreal, Quebec, Canada, 1995. v. 1, p. 278–282. Cited on page 20.

30 HO, T. K. The random subspace method for constructing decision forests. *IEEE transactions on pattern analysis and machine intelligence*, IEEE Computer Society, v. 20, n. 8, p. 832–844, 1998. Cited on page 20.

31 BREIMAN, L. Random forests. *Machine learning*, Springer, v. 45, n. 1, p. 5–32, 2001. Cited 2 times on pages 20 and 42.

32 BOWYER, K. W. et al. SMOTE: synthetic minority over-sampling technique. *arXiv preprint arXiv:1106.1813*, 2011. Disponível em: <http://arxiv.org/abs/1106.1813>. Cited on page 22.

33 DIAZ-CABRERA, M.; CERRI, P.; MEDICI, P. Robust real-time traffic light detection and distance estimation using a single camera. *Expert Systems with Applications*, Elsevier, v. 42, n. 8, p. 3911–3923, 2015. Cited on page 23.

34 LINDNER, F.; KREßEL, U.; KAELBERER, S. Robust recognition of traffic signals. In: . Parma, Italy: IEEE, 2004. p. 49–53. ISBN 0-7803-8310-9. Cited on page 23.

35 LEVINSON, J. et al. Traffic Light Mapping, Localization, and State Detection for Autonomous Vehicles. In: *International Conference on Robotics and Automation (ICRA)*. Shanghai, China: IEEE, 2011. p. 5784–5791. Cited on page 24.

36 FRANKE, U. et al. Making Bertha See. In: *International Conference on Computer Vision (ICCV) Workshops*. Sydney, NSW, Australia: IEEE, 2013. p. 214–221. Cited on page 24.

37 JANG, C. et al. Traffic light recognition exploiting map and localization at every stage. *Expert Systems With Applications*, v. 88, p. 290–304, 2017. Cited 2 times on pages 24 and 51.

38 OLIVEIRA, T. A. de. *Mapeamento, Detecção e Reconhecimento de Semáforos*. Dissertação (Mestrado) — UFES (Campus de Goiabeiras), Vitória, ES, Brazil, 2014. Cited on page 24.

39 JENSEN, M. B.; NASROLLAHI, K.; MOESLUND, T. B. Evaluating state-of-the-art object detector on challenging traffic light data. In: *Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. Honolulu, HI, USA: IEEE, 2017. p. 882–888. Cited on page 25.

40 REN, S. et al. Faster r-cnn: Towards real-time object detection with region proposal networks. In: CORTES, C. et al. (Ed.). *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015. p. 91–99. Disponível em: <http://papers.nips.cc/paper/5638-faster-r-cnn-towards-real-time-object-detection-with-region-proposal-networks.pdf>. Cited on page 25.

41 MONTEMERLO, M.; ROY, N.; THRUN, S. Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit. In: *International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE, 2003. v. 3, p. 2436–2441. Cited on page 28.

42 MUTZ, F. et al. Large-scale mapping in complex field scenarios using an autonomous car. *Expert Syst. Appl.*, Pergamon Press, Inc., Tarrytown, NY, USA, v. 46, n. C, p. 439–462, mar. 2016. ISSN 0957-4174. Disponível em: <https://doi.org/10.1016/j.eswa.2015.10.045>. Cited on page 30.

43 VERONESE, L. de P. et al. A Light-Weight Yet Accurate Localization System for Autonomous Cars in Large-Scale and Complex Environments. In: *International Conference on Intelligent Transportation Systems (ITSC)*. Rio de Janeiro, Brazil: IEEE, 2016. p. 520–525. Cited on page 30.

44 ESTER, M. et al. A density-based algorithm for discovering clusters a density-based algorithm for discovering clusters in large spatial databases with noise. In: *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*. [S.l.]: AAAI Press, 1996. (KDD'96), p. 226–231. Cited on page 32.

45 FREGIN, A. et al. The DriveU traffic light dataset: Introduction and comparison with existing datasets. In: *International Conference on Robotics and Automation (ICRA)*. Brisbane, QLD, Australia: IEEE, 2018. p. 3376–3383. ISSN 2577-087X. Cited on page 34.

46 EVERINGHAM, M. et al. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, v. 88, n. 2, p. 303–338, 2010. Cited on page 38.

47 EVERINGHAM, M. et al. The pascal visual object classes challenge: A retrospective. *International journal of computer vision*, Springer, v. 111, n. 1, p. 98–136, 2015. Cited on page 38.