UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Tetiana Shtym

Traffic light detection by fusing object detection and map info

Master's Thesis (30 ECTS)

Supervisor: Tambet Matiisen, MSc
Supervisor: Meelis Kull, PhD

Tartu 2021

Traffic light detection by fusing object detection and map info

Abstract:
To share streets with human drivers, self-driving cars must locate traffic lights and recognize their states. While for human drivers recognizing a relevant traffic light does not require much effort, it is a challenging task for self-driving cars. Although, for state-of-the-art object detection methods detecting traffic lights is simple, identifying to which lane they apply is non-trivial.

The most common approach relies on precise locations of traffic lights on the high-definition map, localization of the car, and camera position with respect to the car. When a vehicle approaches a traffic light, traffic lights from HD-map are projected to the camera images. Then, regions that include traffic lights, or regions of interest (ROIs) for traffic lights are extracted and fed to the classifier. To mitigate localization errors, ROIs need to be enlarged. However, this can lead to imprecise classification as the bounding box might not capture traffic light adequately.

In this thesis, the problem is addressed by introducing traffic light recognition by fusing object detection and HD-map information. The process is divided into three phases: get 2D traffic lights' ROIs by projecting 3D bounding boxes from the map to the camera image; perform traffic light detection on the image to get 2D bounding boxes and traffic light states; associate traffic lights with lanes by matching detected bounding boxes with ROIs using Intersection-over-Union metric.

The proposed method was integrated into Autoware.AI and tested on prerecorded routes in Tallinn and Tartu. The approach achieved an accuracy of 93% and outperformed the approach currently used by Autoware.AI.

Keywords:
datasets, neural networks, object detection, traffic light recognition, autonomous driving, HD-map, YOLOv3

CERCS: P170 Computer science, numerical analysis, systems, control

Valgusfoori oleku määratlemine objektituvastuse ja kaardiandmete põhjal

Lühikokkuvõte:

Tavasõidukitega koos liiklemiseks peavad isejuhtivad autod leidma tänaval olevate valgusfooride asukoha ja tuvastama nende oleku. Kuigi inimeselt ei nõua suurt pingutust asjakohaste valgusfooride märkamine, on see isejuhtivatele autodele keerukas ülesanne. Kuigi valgusfooride tuvastus on lihtsasti lahendatav olemasolevate objektituvastus meetoditega, on fooride sidumine nendele vasatvate sõiduradadega keeruline.

Kõige tavalisem viis ülesande lahendamiseks hõlmab valgusfoori täpset asukohta täppiskaardil (HD map), auto asukohta ja kaamera positsiooni auto suhtes. Kui sõiduk läheneb valgusfoorile, projitseeritakse täppiskaardil olevad foorid kaamera pildile. Pildi osad, milles leidub valgusfoor ehk huvi pakkuvad alad (ROI), eraldatakse pildilt ja antakse sisendiks klassifikaatorile. Lokaliseerimisvigade vähendamiseks tuleb huvi pakkuvat ala (ROI) suurendada. See võib muuta valgusfoori oleku ennustused ebatäpseks, sest uus ala ei pruugi valgusfoori õigesti jäädvustada.

Lõputöös on valgusfooride paremaks tuvastamiseks ühendatud objektide tuvastamine ja täppiskaardi informatsioon. Protsess on jagatud kolme ossa: esiteks projitseeritakse kaardilt kolmemõõtmelised valgusfoori ümbritsevad kastid *(bounding box)* kaamera pildile kahemõõtmelisteks; teiseks tuvastatakse kaamera pildilt kahemõõtmelised valgusfoori ümbritsevad kastid ja fooride olek; kolmandaks sobitatakse tuvastatud valgusfoorid kaardilt projitseeritutega, mõõtes kahe kasti ülekattuvust (Intersection-over-Union).

Loodud meetod integreeriti Autoware.AI tarkvarasse ja seda testiti Tartus ja Tallinnas salvestatud marsruutidel. Meetod saavutas 93-protsendilise õigsuse ja edestas senist Autoware.AI poolt kasutatud meetodit.

Võtmesõnad:

andmekogumid, tehisnärvivõrgud, objektituvastus, valgusfoori tuvastus, isejuhtivad autod, kõrglahutusega kaart, YOLOv3

CERCS: P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine

Contents

1 Introduction

Autonomous driving has been a topic of active research interest in recent years. Self-driving cars have many advantages over human drivers. In particular, they do not get tired, distracted, or drunk. According to a study published by the U.S. Department of Transportation National Highway Traffic Safety Administration (USDOT NHTSA) [47], 94% of all car accidents that occurred between 2005 and 2007 were the result of driver error.

While safety is the major advantage of autonomous vehicles, they are also society- and environment-friendly. There are many people who cannot drive due to their age or disability. Self-driving cars offer mobility for disabled individuals, thereby making their lives more convenient and fulfilling. Moreover, human drivers usually misuse gas or petrol, which impacts the environment negatively. Autonomous vehicles can mitigate this problem by tuning their acceleration and deceleration profiles [50].

However, the ability to perceive the world in a way humans do remains a significant challenge in self-driving cars [39]. To obey road rules, they need to detect cars, pedestrians, road signs, and traffic lights. While for human drivers recognizing a relevant traffic light does not require much effort, for autonomous vehicles, it is not an easy task. The problem of detecting traffic lights and recognizing their states is known as *Traffic Light Recognition*. Despite the countless solutions proposed by different scientists, the problem still remains relatively unexplored and challenging. Furthermore, the task of identifying relevant traffic lights has not been addressed widely in the literature.

The most common traffic light recognition approach [17] relies on precise locations of traffic lights on the 3D prior map, or *high-definition map*, localization of the car, and camera position with respect to the car. When approaching traffic lights, a self-driving car already knows about them from the map and can combine this information and data from sensors to identify the relevant traffic light. Recently, deep learning methods proved to be efficient in processing sensor data such as camera images or LiDAR point clouds. However, detecting all traffic lights on the camera images still does not provide information on whether the particular traffic light applies to the lane a car is driving.

In this thesis, we analyze an approach used by Autoware.AI[1], and introduce an approach based on fusing object detection and high-definition map information to locate relevant traffic lights. The main idea is to obtain regions that contain traffic lights, or 2D regions of interest (ROIs) for traffic lights, by projecting them from the map to the camera image. At the same time, traffic light detection is performed on the camera image. As a result, we would have more reliable traffic light detection, and we would know if the traffic light is relevant or not based on matching the ROIs with bounding boxes detected from a camera image. The proposed method was evaluated on the routes recorded in Tallinn and Tartu. The proposed approach proved to be efficient and identified relevant

[1] https://www.autoware.ai/

traffic lights correctly.

The rest of this thesis is structured as follows: background is discussed in Section 2; the proposed method is described in Section 3; the experiments and their results are covered in Section 4; Section 5 interprets the results; and, finally, Section 6 concludes and covers future work.

2 Background

This section covers relevant background information on object detection algorithms, Traffic Light Recognition and high-definition map.

2.1 Object Detection

Object detection is a computer vision technique used to locate objects within the image and classify them. It is often confused with image classification task. While the latter only assigns a label to an image, object detection draws a box around the object (i.e. locates the object) and labels the box. Thus, object detection model predicts where each object is and what this object is [54].

Object detection is breaking into numerous fields [54]. It is used for crowd counting, tracking objects, face detection and text detection. Object detection is critical in autonomous driving as it provides a vehicle with the ability to perceive the world: recognize other cars, pedestrians, distinguish traffic lights and signs [25].

Object detection methods can be divided into two groups: classical (non-neural) machine learning-based approaches [46] and deep learning-based approaches [29, 52]. For non-neural approaches, features need to be defined separately using different computer vision techniques such as histogram of oriented gradients [9] or edge detection [48]. Later, these features are fed into a regression model (i.e. SVM [7] , AdaBoost [15], Random Forest [18]) that predicts object's location and label. In turn, convolutional neural networks, or CNNs, underlie deep learning-based approaches. They eliminate the need to define features separately as they behave as feature extractors. For example, for an image with a car, these features could be brake lights, wheels or door handles and are stored in what is known as a feature map.

Object detectors expect an image or video as input, while output is a list of all detected objects, their bounding boxes and labels. A bounding box describes the location of an object. Usually, the bounding box is represented by a rectangle with x-axis and y-axis coordinates in the top-left corner and the x and y-axis coordinates in the bottom-right corner [54].

There are two different types of object detection algorithms: one-stage detectors and two-stage detectors [54, 8]. Although the latter achieve high accuracy, they are also very slow. In the first stage, features from the input image are extracted, then Region Proposal Networks are used to obtain possible locations of the objects within the image. In the second stage, the region candidates and feature maps are used to compute final bounding boxes and class probabilities. An example of a two-stage detector is Faster-RCNN [43].

On the other hand, one-stage detectors are suitable for real-time applications due to the high inference speed. Unlike two-stage detectors, they are trained end-to-end, meaning they do not use Region Proposal Networks resulting in lower accuracy rates. YOLO [40, 41, 42], SSD [31] and RetinaNet [34, 27] are one-stage object detectors.

2.1.1 Traffic Light Recognition

Traffic Light Recognition (TLR) is another application of object detection. TLR is an essential yet challenging component of self-driving which has been actively addressed by researchers. Intelligent vehicles use real-time sensors' data (e.g. camera images) to perceive the environment. To obey road rules, self-driving cars must identify relevant traffic lights and their states [25]. Hence, the task of TLR is to locate and classify traffic lights in camera images.

Traffic lights are made to be visible. They consist of bright-coloured signal lamps and their housing. The most common traffic light sequence is red-yellow-green light [12, 22]. Red light indicates that a driver should stop, yellow or amber warns that light is about to turn red, green light means that a drive can start driving or keep driving. Colour and shape are major characteristics of traffic light and widely used by scientists. For example, in 2009, Omachi et al. introduced the approach of detecting a traffic light based on the Hough transform [33]. They convert RGB input images into normalized RGB images and use them to extract candidate regions for possible traffic lights. Then, a Sobel filter is used on the candidate regions to detect edges. Finally, they use Hough transform to detect a circle that represents a traffic light. However, colour- and shape-based methods are not reliable enough and not suitable for real-time detection as they are slow. Moreover, they might fail when traffic lights are partially occluded or light and weather conditions are poor. Another issue is a large number of false positives from brake lights, pedestrian crossing lights or billboards.

With the recent advancements in deep learning methods, researchers have intensively explored them for TLR.

In 2018, Bach et al. introduced a deep convolutional neural network for traffic light recognition [2]. It is based on the Faster-CNN and aims to detect traffic lights and classify them by type and state. The method was proved to be efficient for medium-sized traffic lights, but it failed to detect small traffic lights. Another issue is caused by an inability of the network to distinguish pedestrian traffic lights and drivers traffic lights, leading to many false positives.

Another example of deep learning approaches would be vision-based real-time traffic light classifier and detector, DeepTLR, proposed by Weber [51]. It is based on convolutional networks using RGB images of whole traffic scenes and includes class probability estimation and bounding box estimation. They analyzed different modifications of AlexNet and managed to achieve high recall and precision for different datasets on high-resolution images.

Although deep learning models have achieved impressive results, they still cannot identify relevant traffic lights. The most common way to solve the problem is to record the route beforehand and manually annotate relevant traffic lights in HD-map [17, 21, 36]. This will provide an autonomous vehicle with prior knowledge of the traffic lights locations. When a car approaches a traffic light, it already knows about it from the

HD-map. Positions of traffic lights in HD-map are then projected to camera images. This allows to identify state of the traffic light by using object detector or classifier on the entire image or only on regions that include traffic lights (regions of interest (ROIs) for traffic lights).

In 2018, Manato Hirabayashi et al. proposed an approach based on extraction of only part of the image related to traffic light [17]. To achieve that, they project 3D positions of traffic lights from HD-map to the image plane using cameras' intrinsics and extrinsics. After this, ROIs are fed to a classifier to get traffic light state. For classification, they used a Single Shot MultiBox Detector or SSD - an object detection algorithm that outputs both classes and bounding boxes (i.e. location of the object). As the authors' goal was to classify traffic lights, they ignored all bounding boxes and adopt the class with the highest probability as a traffic light state. While this approach has proved to be quite efficient, it fails when the traffic light only partially happens to be in the ROI.

Similarly, Jang et al. [21] extract ROIs, feed them into detection and classification models prepared for different traffic lights types (three or four bulbs TLs). As a detector, they use AdaBoost based on Haar-like features, while for classification, a cascade classifier is applied. It works in two stages: firstly, it distinguishes TLs from the background, then it identifies the exact state of the traffic light.

In 2019, Possatti et al. introduced an approach that uses deep learning and HD-maps [36]. When a car approaches the traffic light, it is projected from HD-map to the camera image. A sphere of a 1.5-meter radius surrounds each projected traffic light. Simultaneously, the camera frame goes through the object detector (i.e. YOLOv3) to get the state of the traffic light and its bounding box. Finally, they calculate the Euclidean distance between centres of detected bounding boxes and projected ones. Based on this, they select the traffic light that is the closest to the projected one. Authors claimed that detecting on a single image of the size of 608×608 takes about 47 milliseconds using YOLOv3. They also mentioned that the model confused red traffic light with green one and highlighted the danger of this type of errors.

While researchers have achieved impressive results in TLR, it still remains a major challenge in autonomous driving. The main challenges include [54, 22]:

- Recognition in different illumination conditions (including night or sunset images).
- Recognition in different weather conditions (e.g. rain, snow).
- Motion blur.
- Identifying relevant traffic lights, i.e. traffic lights in the vehicle's route.

2.1.2 Evaluation Metrics

Intersection-over-Union metric Since the classification model outputs only class probabilities, measuring its performance is a straightforward task as it is easy to identify

whether a prediction is correct or not [1, 54]. However, the object detection model predicts the bounding box and confidence score associated with it. To measure how many objects are detected correctly and how many are misdetected, Intersection-over-Union (IoU) [44, 16] is used.

The concept of IoU is fundamental for object detection [28, 13]. IoU indicates how well the predicted bounding box matches the ground-truth bounding box (i.e. targets in the test dataset). IoU is calculated as the overlap between the ground truth bounding box and predicted bounding box divided by their union (Figure 1). IoU ranges from 0 to 1, where 1 indicates that the boxes match perfectly.

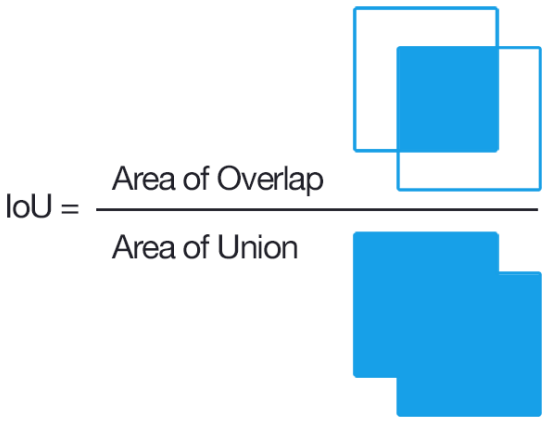


Figure 1. Intersection-over-Union metric [44]

By computing the IoU score for each detection against the ground truth box and setting a fixed value as a threshold θ, it is possible determine whether detection is a correct or wrong prediction. More precisely, each detection can be classified as follows [19, 1]:

1. True Positive (TP) is an outcome where IoU $\geq \theta$ and the predicted class matches the class of a ground truth, meaning that there is an object in the image and the model detects it correctly.

2. False Positive (FP) is an outcome where IoU $< \theta$ or the predicted class does not match the class of a ground truth, meaning the object is not there, and model detects it.

3. False Negative (FN) is an outcome where the object is in the image, but the model does not detect it.

It is worth mentioning that detection can be classified as True Negative (TN) when the model predicts that there is no object correctly. However, TNs do not provide any useful information and are not essential for the evaluation of object detection models [1].

Precision, Recall and F1-score The most common and most straightforward object detection performance metrics are precision and recall [1, 54, 37].

Precision measures how many detected objects are relevant or accurate, i.e. the proportion of correctly predicted objects among all predictions [35].

$$precision = \frac{TP}{TP + FP} \tag{1}$$

Precision ranges from 0 to 1. High precision means that most predicted bounding boxes match ground truth boxes. For example, precision $= 0.6$ when 60% of the time the model predicts positives correctly.

In turn, recall shows how many relevant objects are detected [35]. In other words, it measures the proportion of correctly predicted ground-truth objects.

$$recall = \frac{TP}{TP + FN} \tag{2}$$

Similarly, recall ranges from 0 to 1. For instance, recall $= 0.7$ when the model predicts 70% of objects correctly among all ground-truth objects.

Unfortunately, both these metrics are often in tension, meaning that higher recall leads to lower precision and vice versa [10]. High precision and low recall mean that the model detects all objects correctly and misses most ground truth objects (i.e. returns few results). A model with high recall and low precision detects most of the ground truth objects incorrectly.

There is also F1-score, which is defined as a harmonic mean of the model's precision and recall [45]. It takes both false positives and false negatives into account and therefore conveys a balance between precision and recall.

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{3}$$

Mean Average Precision It is improbable for a model to achieve high recall and high precision because of a trade-off between these two metrics, which depends on the IoU threshold [1, 10]. All objects for which the IoU score is less than the predefined IoU threshold θ are discarded. Thus, setting a low IoU threshold leads to more positive predictions and less false predictions and, as a result, high recall and low precision.

Precision-recall curve demonstrates a trade-off between precision and recall for different threshold values [5, 10, 1]. It is used to select an optimal threshold value. However, due to the trade-off, a curve can be wiggly, which complicates choosing the optimal threshold value. To eliminate the problem, interpolated precision is calculated by taking maximum precision measured for each recall level. Example of precision-recall curve is represented in Figure 2.

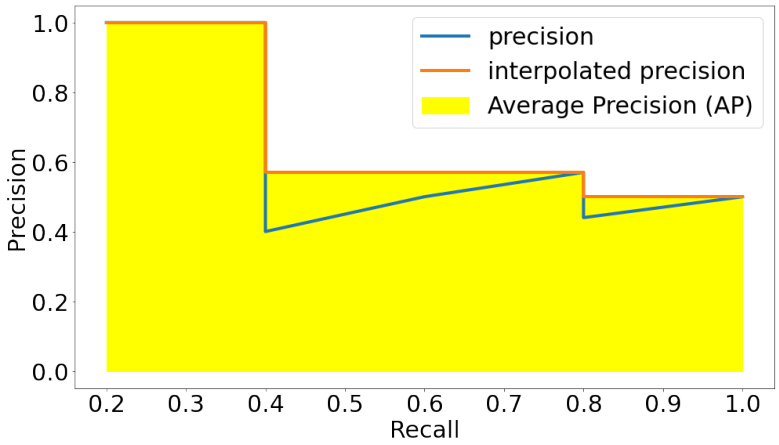


Figure 2. Precision-Recall curve. Yellow area denotes area under Precision-Recall curve, or average precision (AP). The blue line is wiggly as it before interpolation.

Mean Average Precision (mAP) is the most used performance metric for object detection models. Average Precision (AP) is the area under the Precision-Recall curve [10]. In turn, mAP averages AP over all classes N:

$$mAP = \frac{1}{N} \cdot \sum_{i=1}^{N} AP_i \tag{4}$$

Most state-of-the-art object detectors are evaluated on COCO 2017 dataset [28, 19]. In COCO 2017 challenge mAP is averaged over all classes and over 10 IoU thresholds. These thresholds range between 0.5 and 0.95 with the step of 0.05. Thus, $\mathrm{mAP}^{\mathrm{IoU}=0.5}$ indicates average precision at the IoU threshold of 0.5. Note that in COCO 2017 Challenge, there is no difference between AP and mAP. Comparatively, in PascalVOC 2007 challenge [13], mAP averages AP over all classes IoU threshold of 0.5.

2.1.3 YOLOv3

For safe self-driving, high accuracy and fast inference speed are crucial to prevent fatal accidents. Due to its inference speed, YOLOv3 is one of the most obvious candidates for object detection systems to use in autonomous driving vehicles [42, 20, 26]. Moreover, it is efficient in multi-scale detection, meaning it can detect objects of different sizes.

YOLOv3 works in the following way: an image comes into the network and goes to a feature extractor that produces feature maps at three different scales. This allows detecting small and large objects. Finally, these feature maps are passed to a detector that outputs bounding boxes and class probabilities.

YOLOv3 uses Darknet-53 as a feature extractor. It is composed of residual blocks consisting of consecutive 3×3 and 1×1 convolutional layers with skip connections. The last three residual blocks output feature maps (or grids) of different sizes at three different scales using strides of 32, 16 and 8. The size of the feature map is denoted as $S \times S$. Detection is done by applying 1×1 detection kernels on feature maps. The architecture of YOLOv3 is represented in Figure 3.

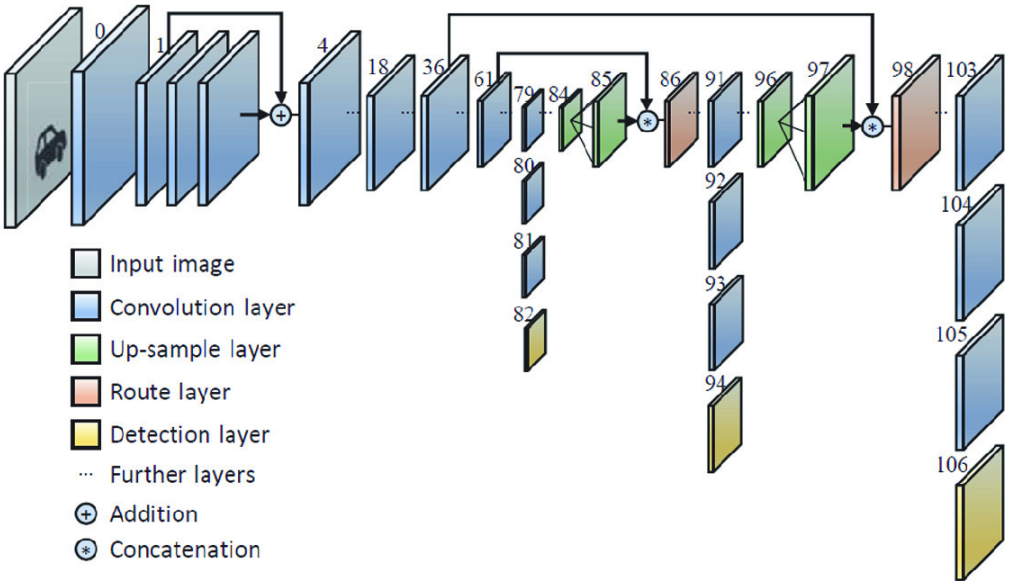


Figure 3. Architecture of YOLOv3 [6]. YOLOv3 consists of 106 layers. If a 416×416 image is fed, YOLOv3 yields three feature maps, or grids, of the size of 13×13, 26×26, 52×52 at layers 82, 94 and 106 respectively. Detection is done by applying 1×1 detection kernels on feature maps.

An input image is downsampled by the first 81 layers such that the 82st layer has a stride of 32. Thus, if a 416×416 image is fed, YOLOv3 yields a feature map of the size of 13×13. After this, the feature map from layer 79 goes through convolutional layers $80 - 84$ and then is upsampled by a stride of 2. The resultant feature map is concatenated with the feature map from layer 61. The second detection is made at layer 94 with stride 16 yielding the feature map of the size of 26×26. A similar procedure repeated again: the feature map from layer 91 is upsampled by a factor of 2 and concatenated with the feature map from layer 36. Finally, the third detection is made at the layer 106 of stride 8, which results in the feature map of the size of 52×52.

Detections at different layers address the problem of detecting objects of different sizes. Thus, a 13×13 layer detects large objects, while a 52×52 layer is responsible for detecting small objects. Upsampling procedure and concatenation with the previous layers preserves the fine-grained features used for detecting small objects.

Each grid cell is responsible for predicting B bounding boxes. For each bounding box YOLOv3 outputs the following elements:

- Center coordinates $(\hat{b}_x, \hat{b}_y)$.
- Width $\hat{b}_w$ and height $\hat{b}_h$.
- Objectness score $\hat{p}_{obj}$, which indicates how confident the model is that there is an object in the box. $\hat{p}_{obj}$ ranges between 0 and 1.

The main goal of object detection is to locate an object in the image by predicting its bounding box. However, predicting bounding box coordinates directly can lead to unstable gradients while training and the inability of the network to converge due to the large variance of scale and aspect ratio of boxes [26].

The problem has been addressed by Ren and al. [43], who introduced the concept of anchors. Anchors are predefined boxes, usually represented as width p_w and height p_h of the most common objects from the dataset. Anchors are generated by using the K-means clustering algorithm [32] on the entire training dataset. This is done before training. Anchors are the same for each cell.

In YOLOv3, each grid cell has B anchors ($B = 3$) and instead of predicting the bounding box's width and height, YOLOv3 predicts four offsets $\hat{t}_x, \hat{t}_y, \hat{t}_w, \hat{t}_h$ to anchors that are later transformed into coordinates of the bounding box, $\hat{b}_x, \hat{b}_y, \hat{b}_w, \hat{b}_h$. This is done in the following way:

$$\begin{aligned} \hat{b}_x &= \sigma(\hat{t}_x) + c_x \\ \hat{b}_y &= \sigma(\hat{t}_y) + c_y \\ \hat{b}_w &= p_w \cdot e^{\hat{t}_w} \\ \hat{b}_h &= p_h \cdot e^{\hat{t}_h} \end{aligned} \tag{5}$$

where:

$\sigma(x) = \frac{1}{1+e^{-x}}$

(c_x, c_y) - offset (normalized by image width and height) from top-left coordinate of the grid to the cell.

Finally, the bounding box responsible for detecting the object is that has the highest IoU with the ground truth box and above predetermined threshold that equals 0.5. In addition to offsets, YOLOv3 also outputs $\hat{t}_o$, which is passed through a sigmoid to obtain objectness score $\hat{p}_{obj}$ for each bounding box.

YOLOv3 outputs one set of C class probabilities for each grid cell regardless of the number of anchors B. Similarly, sigmoid is used to convert class scores to probabilities.

Authors stopped using softmax for classes as this suggests that classes are mutually exclusive, meaning that an object can belong only to one class.

To train the model, authors use a multi-part loss function consisting of three different losses [40, 20]:

1. Localization loss l_{box} measures sum of squared errors between predicted offsets $\hat{t}_x, \hat{t}_y, \hat{t}_w, \hat{t}_h$ and ground truth values t_x, t_y, t_w, t_h which can be easily computed by inverting the equations above. This loss is calculated only for the bounding box responsible for detecting the object.

$$\begin{aligned} l_{box} = \lambda_{coord} \sum_{i=1}^{S} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} [(\hat{t}_{x_i} - t_{x_i})^2 + (\hat{t}_{y_i} - t_{y_i})^2] \\ + \lambda_{coord} \sum_{i=1}^{S} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} [(\hat{t}_{w_i} - t_{w_i})^2 + (\hat{t}_{h_i} - t_{h_i})^2], \end{aligned} \tag{6}$$

   where:

   $\mathbb{1}_{ij}^{obj} = 1$ if j-th bounding box in the i-th grid cell is responsible for predicting the object (has the highest IoU with the ground truth box and IoU ≥ 0.5), otherwise 0.

   λ_{coord} increases the importance of localization loss.

2. Objectness loss l_{obj} represents the loss related to the the confidence of bounding box. The bounding box that has the highest IoU with the ground truth box is assigned 1 as a ground truth objectness score. If the bounding box does not have the highest IoU but overlaps a ground truth object by more than the predefined threshold that equals 0.5, it is ignored. In turn, if the IoU score between the bounding box and the ground truth box is less than threshold, the bounding box is assigned 0 as a target value. Objectness loss is calculated as cross-entropy loss:

$$l_{obj} = \sum_{i=1}^{S} \sum_{j=1}^{B} \mathbb{1}_{ij}^{obj} [-\log(\hat{p}_{obj_{ij}})] + \lambda_{noobj} \sum_{i=1}^{S} \sum_{j=1}^{B} \mathbb{1}_{ij}^{noobj} [-\log(1 - \hat{p}_{obj_{ij}})], \tag{7}$$

   where:

   $\mathbb{1}_{ij}^{obj} = 1$ if j-th bounding box in the i-th grid cell is responsible for predicting the object (has the highest IoUwith the ground truth box and IoU ≥ 0.5), otherwise 0.

   $\mathbb{1}_{ij}^{obj} = 1$ if the i-th grid cell does not contain the object or j-th bounding box in the i-th grid cell is not responsible for predicting the object (IoU ≥ 0.5 but not the highest), otherwise 0.

$\hat{p}_{obj_{ij}}$ is the objectness score of the j-th box in the i-th cell.

λ_{noobj} scales the loss of detecting background.

3. Classification loss l_{cls} uses binary cross-entropy loss to measure how well predicted class probabilities $\hat{\boldsymbol{p}}$ match one-hot-encoded ground-truth labels $\boldsymbol{p}$.

$$l_{cls} = \sum_{i=1}^{S} \sum_{c \in C} \mathbb{1}_i^{obj} [p_i(c) \log(\hat{p}_i(c)) + (1 - p_i(c)) \log(1 - \hat{p}_i(c))], \tag{8}$$

where $\mathbb{1}_i^{obj} = 1$ if object is detected in the i-th cell, otherwise 0.

The final loss is calculated as follows:

$$l = l_{box} + l_{obj} + l_{cls} \tag{9}$$

During inference, YOLOv3 tends to produce multiple bounding boxes for the same object. To remove duplicates, non-maximum suppression (NMS) is used. Firstly, it discards all predictions with objectness score less than predefined threshold. After this, it sorts all detections by confidence scores. Then, starting from prediction with the highest score, it uses IoU to compare the current bounding box with all other boxes and removes those with high overlap and are of the same class. The algorithm is applied iteratively until there are no more overlapping boxes. It is worth mentioning that NMS improves mAP by 2-3%.

YOLO is not only speedy but also outperforms SSD variants and is almost on par with RetinaNet in terms of mAP_{50}. Authors mentioned that with increasing of IoU threshold, YOLOv3's performance drops which is caused by an inability to precisely align bounding boxes with objects.

2.1.4 Tiny-YOLOv3

Tiny-YOLOv3 [42, 53] is the simplified and faster version of YOLOv3. Higher inference speed is achieved by reducing the number of convolutional layers in the backbone. While there are 53 convolutional layers in YOLOv3, Tiny-YOLOv3 has only 7. Features are extracted by 3×3 and 1×1 convolutional layers. Detection layers 16 and 23 has sizes of 13×13 and 26×26 respectively. In turn, the whole model consists of 13 convolutional layers and 6 max-pooling layers. Figure 4 demonstrates the architecture of Tiny-YOLOv3.

Tiny-YOLOv3 differs from YOLOv3 only in the architecture. Tiny-YOLOv3 similarly outputs centre coordinates, width and height of the bounding box, objectness score associated with it and class probabilities. For training, a cross-entropy loss is used for the class predictions and objectness. For bounding boxes, Tiny-YOLOv3 uses the sum of the squared error loss.

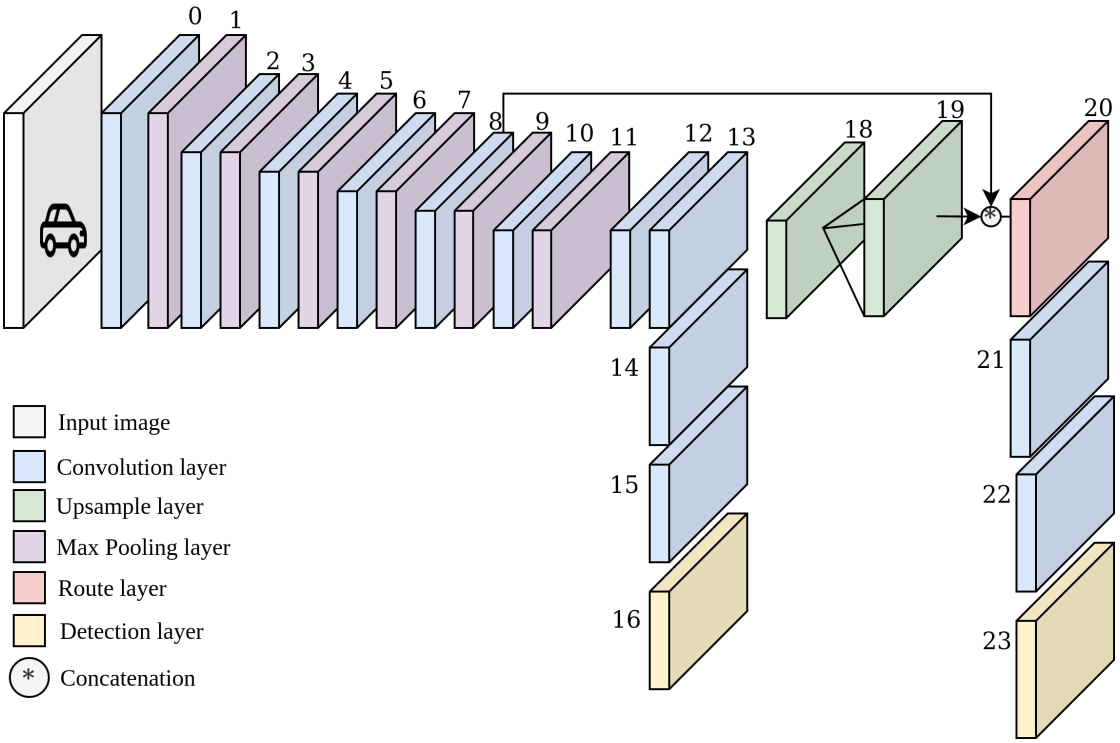


Figure 4. Architecture of Tiny-YOLOv3. Tiny-YOLOv3 consists of 23 layers.

The shallow and simpler architecture of Tiny-YOLOV3 demonstrates high inference speed. However, it also leads to a significant drop in accuracy.

2.2 Autonomous Driving

Autonomous driving is a rapidly advancing field. To achieve full autonomous driving (i.e. operating without any human involvement), self-driving cars must perceive the world as humans drivers do. Autonomous vehicles rely on different sensors to see and interpret objects in their route [14]. These sensors include:

- **Camera sensors**. They act as the eyes of the car. Cameras are mainly used for object classification, object detection and lane tracking. However, camera sensors do not perform well in poor weather and lighting conditions.
- **LiDAR sensors**. In self-driving, LiDARs are used to create a 360-degree 3D map by sending laser beams. This allows measuring distance to the surrounding objects and their shape. LiDAR sensors are mainly independent of environmental conditions and provide reliable 3D information.

- **Radar sensors.** Radar transmits radio waves to other objects, which allow to measure their speed and distance to them.
- **Global Navigation Satellite System (GNSS)**. It is used to estimate the vehicle's location and orientation.
- **Inertial Measurement Unit (IMU)**. It monitors the changes in the movements of a car. It tracks a car's velocity, position and attitude.

Self-driving cars fuse information from different sensors to enable safe autonomous driving.

2.2.1 High-definition map

High-definition map (HD-map), or vector map, is a highly precise 3D map mainly used in autonomous driving that contains a large amount of driving assistance information not present in traditional maps [30]. Autonomous vehicles cannot perceive the world and navigate at the level of humans. HD-maps overcome this limitation and enable autonomous vehicles to achieve centimetre-level precision.

- **Real-time layer** contains information from other vehicles and infrastructures. For example, if there is a traffic jam or an accident.
- **Experiential layer** represents information on previous experience: traffic flow, area of high accident risk.
- **Road Network layer** contains information on how lanes are connected to compose the road network.
- **Semantic layer** attaches meaning to the objects that contain rich metadata, for example, speed limits.
- **Geometric layer** usually contains geometric information (i.e. world position) of the physical objects.

There are several HD-map formats such as Navigation Data Standard, Lanelet2, OpenDrive and Autoware vector maps.

HD-maps provide prior knowledge of the environment, which makes autonomous driving easier. They allow autonomous vehicles to see further and do not get affected by weather and light conditions. However, the main disadvantage of HD-maps is high maintenance as it is very costly to keep them up-to-date.

3 Methods

Currently, Autonomous Driving Lab (ADL) uses Autoware.AI [23], the world's first "All-in-One" open-source solution for autonomous driving.

Autoware.AI provides a region-based method for Traffic Light Recognition. The method is based on the projection of HD-map locations of traffic lights (TLs) to the image coordinates using camera calibration parameters, localization of the autonomous vehicle and the map. Then, regions of interest (ROIs) for traffic lights are extracted and fed to the classifier. This approach is currently used by ADL and will be referred to as the baseline approach in this thesis.

However, this method is highly dependant on the accuracy of localization. Inaccurate localization can lead to projection errors which result in not precise ROIs for traffic lights. To accommodate localization errors, ROIs have to be enlarged. The possible errors associated with this (see examples in Figure 5) are as follows:

1. ROI does not contain a traffic light at all.
2. ROI contains only part of the traffic light, but its state is not visible.
3. ROI contains multiple traffic lights.

These errors can lead to the wrong classification.

Figure 5. Possible errors caused by inaccurate localization

To eliminate these problems, in this thesis, a new approach based on a fusion of object detection and HD-map is introduced. When a car approaches the traffic light, the traffic light's ROIs are extracted in the same way as the baseline approach. Simultaneously, we feed camera images to object detector to obtain possible traffic lights. Finally, the Intersection-over-Union metric is used on the ROIs and detected bounding boxes to identify relevant traffic lights and their states. We implemented the proposed approach on top of Autoware.AI. The proposed approach will be referred to as the fusion approach in this work.

In this section, we give a comprehensive overview of the proposed method. Firstly, we describe datasets used in this thesis. Then, we explain the baseline approach for a better understanding: we discuss the structure of HD-maps used in this thesis, ROI extraction procedure, and classifier architecture. After this, we present the proposed approach: we discuss training procedures of Tiny-YOLOv3, the fusion of ROIs and detected traffic lights. Finally, we provide the evaluation approach for our method and describe how the system was integrated into Autoware.AI.

3.1 Description of the data

In this work, we used three datasets: LISA Traffic Light Dataset, LaRA Traffic Light Dataset and Bosch Small Traffic Lights Dataset for training the object detector. For evaluation of our approach, we annotated the data collected by Autonomous Driving Lab.

3.1.1 LISA Traffic Light Dataset

LISA Traffic Light Dataset [22] is collected in San Diego, California, USA. The training clips consist of 13 daytime clips and 5 nighttime clips, while for testing, four daytime and two nighttime sequences of driving in Pacific Beach and La Jolla, San Diego are provided.

The database consists of continuous test and training video sequences, totalling 43007 images with a resolution of 1280×960 and 113,888 annotated traffic lights. There are **seven different labels** in the dataset: "go", "stop", "stop Left", "go Left", "go Forward", "warning", "warning Left". In this thesis, classes "stop" and "stop Left" were merged as **"red"**, "go", "go Left" and "go Forward" - as **"green"**, "warning" and "warning Left" - as **"yellow"**.

3.1.2 LaRA Traffic Light Dataset

LaRA Traffic Light Dataset [11] is collected in Paris, France. The dataset consists of 11179 daytime images with a resolution of 640×480 and 9168 hand-labelled traffic lights. There are 4 different labels - **warning** (treated as yellow), **go** (treated as green), **stop** (treated as red) and **ambiguous** (treated as unknown).

3.1.3 Bosch Small Traffic Lights Dataset

Bosch Small Traffic Lights Dataset (BOSCH Dataset) [3] is collected in Palo Alto, California, USA. It consists of 13427 camera images at 1280×720 pixels and contains about 24000 annotated traffic lights. The dataset includes different road scenes such as road works, light rain, dense stop-and-go traffic, substantial changes in illumination/exposure.

Annotations include bounding boxes of traffic lights (coordinates of top-left and bottom-right corners), the current state of each traffic light, and whether the traffic light is occluded. The training set consists of 5093 images and 24000 annotated traffic lights. It has ten different labels - **four states** ("Off", "Red", "Green", "Yellow") and **six specific versions of these states** ("Red Left", "Red Right", "Red Straight", "Green Left", "Green Right", "Green Straight"). In turn, the test set consists of 8334 images and 13486 annotated traffic lights and has four different labels - red, green, yellow, and off.

In this thesis, specific traffic light states were grouped into **"green", "red" and "yellow"**. All "Off" traffic lights are treated as **"unknown"**.

3.1.4 Autonomous Driving Lab Dataset

Autonomous Driving Lab Dataset was collected in Tartu and Tallinn, Estonia. It consists of 448 camera images with a resolution of 2064×1544 and 1333 annotated traffic lights. The dataset includes different road scenes such as light rain and substantial illumination changes.

For labelling, a graphical image annotation tool LabelImg [49] was used. Traffic lights were labeled as soon as they are 11 pixels wide or more. Red-yellow traffic lights were labeled as "red". Switched off traffic lights were labeled as "unknown". Hence, there are four different classes in the dataset: **"green"**, **"yellow"**, **"red"**, and **"unknown"**.

3.2 Baseline approach

The baseline approach is a region-based method for traffic light recognition currently used by ADL. It can be divided into two parts:

1. Extraction of regions that include traffic lights (ROIs) in the camera image.
2. Colour state recognition by feeding ROIs into a classifier.

Baseline approach is represented in Figure 6. Section 3.2.1 presents HD-map used in this thesis. Section 3.2.2 explains ROI extraction. Finally, Section 3.2.3 discusses the classifier used by baseline approach.

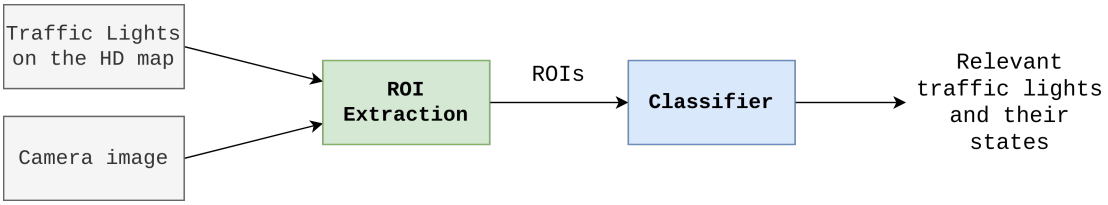


Figure 6. Overall flow of the baseline approach.

3.2.1 Autoware vector maps

Autoware.AI supports the simplified version of vector maps developed by Aisan Technology.

Autoware vector maps are represented as CSV files where each file introduces one specific category of the map elements, for example, point, line, lane or signal (traffic light). As this thesis focuses on the traffic lights, we cover only aspects of the HD-map related to them. Autoware vector map provides the following information for an individual traffic light:

- ID of each bulb, or lamp, belonging to the traffic light.
- World or 3D coordinates (x, y, z), of each bulb.
- Lamp Type (red, yellow, green).
- ID of the lane to which traffic light applies.
- Pole ID to which the traffic light bulb belongs.

3.2.2 ROI Extraction

ROI extraction is achieved by using locations of traffic lights on the HD-map, precise localization of the car and knowledge of camera position and angle with respect to the car.

The first step to obtain ROIs is to project traffic lights from the HD-map to camera images. For this, an autonomous vehicle must locate itself on the map. ADL uses Global Navigation Satellite System (GNSS)-Inertial Measurement Unit (IMU) approach for localization (GNSS and IMU are described in Section 2.2). GNSS provides the absolute pose (position and orientation) of the car with an accuracy of up to 3 meters which is not enough for autonomous driving. Higher accuracy (up to 3 cm) is achieved by using real-time kinematic (RTK) corrections [4]. To mitigate possible errors, measurements from different sensors are fused using Kalman filters. To project traffic lights from the map to image coordinates, the camera position is needed. To obtain this, first, the GNSS receiver (located in the trunk) position is acquired, applying an offset (between antennas and GNSS receiver) that was calibrated beforehand. Then, the LiDAR position is estimated based on the GNSS receiver position using the predetermined positional relationship. Finally, the camera position is obtained using LiDAR-camera transform. The entire process is demonstrated in Figure 8.

Each camera has its parameters (i.e. focal length, optical centre and skew coefficient) represented in the intrinsic matrix. However, the real camera does not know anything about the positions of traffic lights in the HD-map. Camera acts as a pinhole camera (see Figure 7), and its intrinsics are used to project traffic lights from the HD-map to

the camera image. This also can be viewed as the virtual camera, which has the same intrinsics. As a result, centre pixel coordinates (u, v) and projected radius r for each bulb of the traffic light are obtained in addition to the information from the HD-map (pole ID, lamp type, ID of the lane TL applies to).

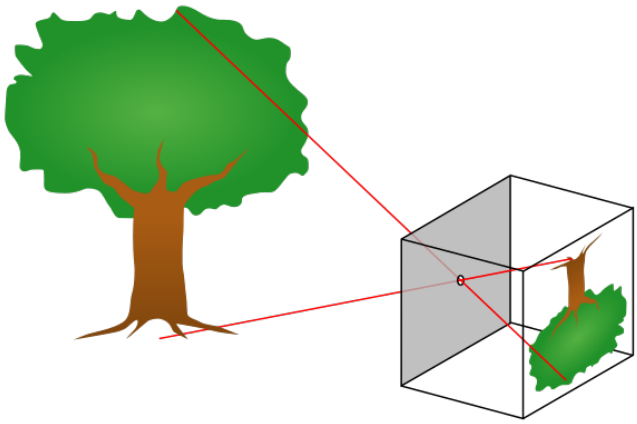


Figure 7. A diagram of a pinhole camera [24]. A pinhole camera is usually without any lens but with a small aperture (a tiny hole in one side). All light rays from the object go through the aperture, and as a result, the image plane shows inverse images of objects. A pinhole camera model describes the relationship between a point in 3D space and its projection on the 2D image plane.

Obtained coordinates are used to extract ROIs for traffic lights. Figure 9 shows ROI extraction procedure. Firstly, two most extreme bulbs of the traffic light are found. Then, their centre coordinates, (u_1, v_1) and (u_2, v_2), and projected radius, r_1 and r_2, are used to compute top-left (x_1, y_1) and bottom-right (x_2, y_2) corners of ROI. To mitigate possible localization errors, λr_1 and λr_2 margins are applied to both corners as follows:

$$
\begin{aligned}
x_1 &= u_1 - r_1 - \lambda r_1 \\
y_1 &= v_1 - r_1 - \lambda r_1 \\
x_2 &= u_2 + r_2 + \lambda r_2 \\
y_2 &= v_2 + r_2 + \lambda r_2, \\
&\text{where } \lambda \in R.
\end{aligned}
$$

Finally, extracted ROIs are specified as as top-left $(x_{\mathrm{ROI}_1}, y_{\mathrm{ROI}_1})$ and bottom-right $(x_{\mathrm{ROI}_2}, y_{\mathrm{ROI}_2})$ corners. Each ROI is associated with pole ID.

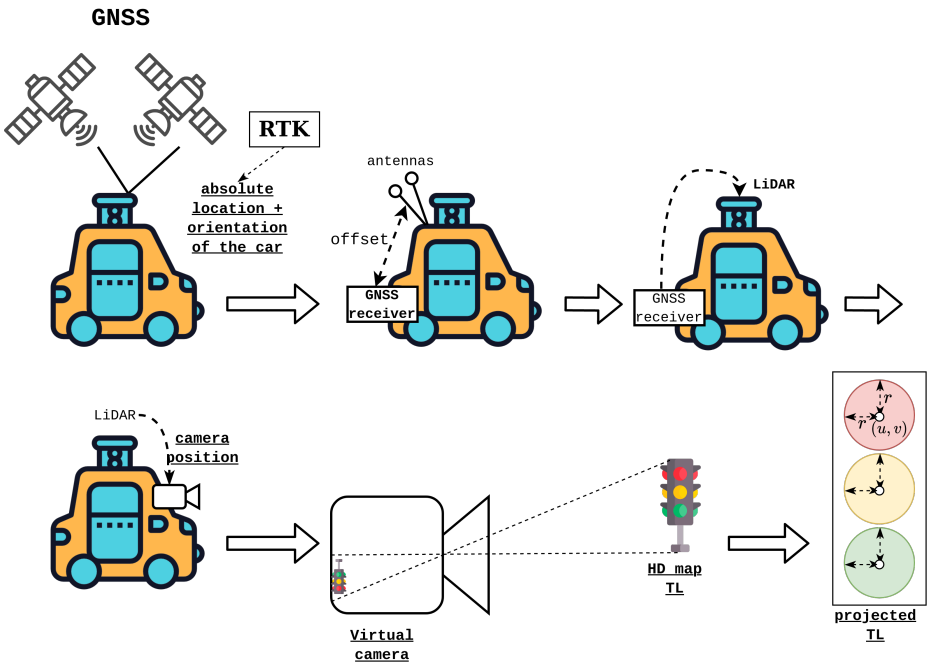


Figure 8. Projection of TLs from HD-map to the camera images.

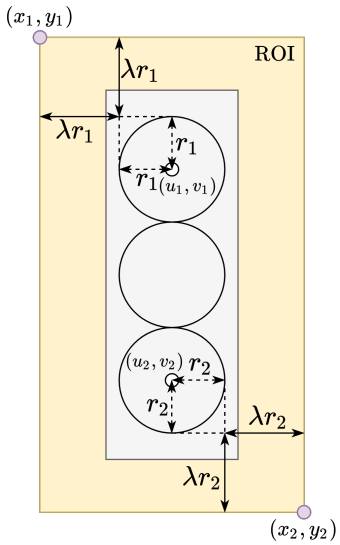


Figure 9. ROI extraction method. Yellow area represents ROI. Gray area denotes true traffic light.

3.2.3 Classifier

Colour state recognition in the baseline approach is done by feeding ROIs to the classifier. The classifier is a simple CNN that consists of three convolutional layers and three max-pooling layers. The model was provided by AutonomouStuff[2]. Architecture is represented in Table 1. The model has 831780 parameters in total.

Table 1. Architecture of the classifier.

Layer	**Output Size**	**№of filters**	**Filter size**	**Stride**	**Padding**
Input image	$128 \times 128 \times 3$	–	–	–	–
Convolution (ReLU)	$128 \times 128 \times 32$	32	3×3	1	1
Max Pooling	$64 \times 64 \times 32$	32	2×2	2	0
Convolution (ReLU)	$62 \times 62 \times 32$	32	3×3	1	0
Max Pooling	$31 \times 31 \times 32$	32	2×2	2	0
Convolution (ReLU)	$29 \times 29 \times 64$	64	3×3	1	0
Max Pooling	$14 \times 14 \times 64$	64	2×2	2	0
Flatten	12544×1	–	–	–	–
Dense (ReLU)	64×1	–	–	–	–
Dropout (0.5)	64×1	–	–	–	–
Dense (Softmax)	4×1	–	–	–	–

ADL retrained the model using the crops of traffic lights from LISA (without night-time data) and LaRA datasets. The categorical cross-entropy loss function and Stochastic Gradient Descent optimizer with a learning rate of 0.01 have been used for the training.

The model was implemented with the help of Keras[3], an open-source software library that provides a Python interface for artificial neural networks.

3.3 Fusion approach

Fusion approach uses Autoware vector maps. It can be divided into three parts (Figure 10):

1. Extraction of regions that include traffic lights (ROIs) in the camera image in the same way as in the baseline approach.

2. Object detection by feeding camera images into Tiny-YOLOv3.

3. Fusion of ROIs and detected bounding box using Intersection-over-Union (IoU).

[2]https://autonomoustuff.com/
[3]https://keras.io/

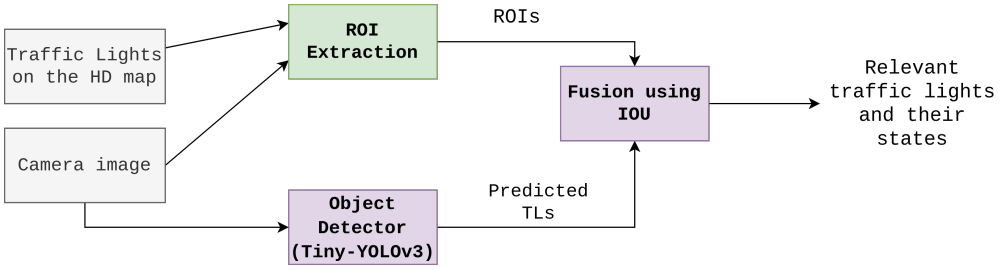


Figure 10. Overall flow of the fusion approach.

Section 3.3.1 presents architecture and training procedures of Tiny-YOLOv3. Section 3.3.1 presents architecture of Tiny-YOLOv3. Section 3.3.2 discusses fusion of camera and HD-map information. Finally, Section 3.3.3 introduces evaluation methods of the fusion approach.

3.3.1 Tiny-YOLOv3

Tiny-YOLOv3 is an one-stage detector. The choice of Tiny-YOLOv3 as an object detector for the fusion approach is stemmed from its high inference speed needed for real-time applications. Tiny-YOLOv3 was used for recognition of traffic lights and their TL states - red, green, yellow, unknown. The architecture of Tiny-YOLOv3 is represented in Table 2. By default, Tiny-YOLOv3 configuration is defined for 80 classes. Thus, the number of filters for the convolution layer preceding YOLO layers is $(5 + 80) \times 3 = 255$. In this thesis, 4 classes (red, green, yellow and unknown) is used. Therefore, the number of filters was changed to $(5 + 4) \times 3 = 27$.

3.3.2 Fusion

To identify states of relevant traffic lights, a fusion of ROIs and detected bounding boxes produced by Tiny-YOLOv3 is performed. The fusion process is as follows:

Step (1) Obtain the ROIs of traffic lights.

Step (2) Obtain detected traffic lights (bounding box and state) using Tiny-YOLOv3.

Step (3) Compute IoU scores between each ROI and detected bounding box.

Step (4) Match ROI with detected bounding box based on maximum IoU.

Step (5) Discard all matches where IoU is below the IoU threshold θ. This IoU threshold will be referred to as fusion threshold θ in this thesis.

The whole fusion process is represented in Algorithm 1.

Table 2. Architecture of Tiny-YOLOv3. YOLO layers represent layers 16 and 23 from Figure 4.

Layer	**Output Size**	**№of filters**	**Filter size**	**Stride**	**Padding**
Input image	$608 \times 608 \times 3$	–	–	–	–
Convolution	$608 \times 608 \times 16$	16	3×3	1	1
Max Pooling	$304 \times 304 \times 16$	32	2×2	2	0
Convolution	$304 \times 304 \times 32$	32	3×3	1	1
Max Pooling	$152 \times 152 \times 32$	32	2×2	2	0
Convolution	$152 \times 152 \times 64$	64	3×3	1	1
Max Pooling	$76 \times 76 \times 64$	64	2×2	2	0
Convolution	$76 \times 76 \times 128$	128	3×3	1	1
Max Pooling	$38 \times 38 \times 128$	128	2×2	2	0
Convolution	$38 \times 38 \times 256$	256	3×3	1	1
Max Pooling	$19 \times 19 \times 256$	256	2×2	2	0
Convolution	$19 \times 19 \times 512$	512	3×3	1	1
Max Pooling	$19 \times 19 \times 512$	512	2×2	1	0
Convolution	$19 \times 19 \times 1024$	1024	3×3	1	1
Convolution	$19 \times 19 \times 256$	256	1×1	1	1
Convolution	$19 \times 19 \times 512$	512	3×3	1	1
Convolution	$19 \times 19 \times 27$	27	1×1	1	1
YOLO	–	–	–	–	–
Route 13	$19 \times 19 \times 256$	–	–	–	–
Convolution	$19 \times 19 \times 128$	128	1×1	1	1
Upsample	$38 \times 38 \times 128$	–	–	–	–
Route 19 8	$38 \times 38 \times 384$	–	–	–	–
Convolution	$38 \times 38 \times 265$	256	3×3	1	1
Convolution	$38 \times 38 \times 27$	27	1×1	1	1
YOLO	–	–	–	–	–

3.3.3 Evaluation

Evaluation of the fusion approach consists of two parts: evaluation of the object detector and evaluation of the entire system. The main two criteria for evaluation are the following:

1. How well object detector can detect traffic lights.

2. How well the system can identify relevant traffic lights and their states.

Performance of Tiny-YOLOv3 was measured in terms of recall, precision, F1-score and mean Average Precision (mAP) using the ADL dataset. These metrics were calculated using confidence thresholds α of 0.05, 0.1 and 0.3 and the IoU threshold of 0.5 for

Algorithm 1: Fusion

Input: $R = \{R_1, ..., R_N\}$, $B = \{B_1, ..., B_N\}$, θ
R is the list of ROIs and traffic lights' IDs associated with them,
B is the list of detected bounding boxes and states associated with them,
θ is the fusion threshold.
Result: Relevant pole IDs T and their states S

```
for R_i in R do
    θ_max ← 0;
    for B_i in B do
        IoU ← calculate IoU score between R_i and B_i;
        if IoU ≥ θ_max then
            θ_max ← IoU;
            s ← B_{i.state};
    if θ_max ≥ θ then
        insert R_{i.ID} into T
        insert s into S
return T, S
```

the mAP calculation. It is worth noting that the mAP calculation provided by PascalVOC 2007 challenge was used.

The entire system was evaluated on two prerecorded routes in Tartu and Tallinn. Firstly, ROIs for traffic lights from the routes were obtained. Then, ROIs were manually labelled as "green", "red", "yellow", or "unknown". From both routes, there are 328 green ROIs, 415 red ROIs, 21 yellows ROIs and 26 unknown ROIs.

After this, fusion is performed and as a result, ROIs and associated with them states are acquired. Finally, predictions were compared with ground truths annotations and confusion matrices were provided. The performance of the entire system was compared to the baseline approach.

Moreover, the prerecorded routes were used to measure how early the system was able to detect traffic lights: the time system took to output the first prediction.

3.3.4 System integration

The fusion approach was implemented on the top of the Autoware.AI currently used by ADL. Autoware.AI consists of the following modules: Localization, Detection, Control, Prediction, and Planning. This work is particularly concerned with the Detection module that uses cameras and LiDARs with sensor fusion algorithms and deep neural networks.

Autoware.AI is based on Robot Operation System (ROS) [38]. Robot Operating

System or ROS[4] is an open-source framework for writing robot software. It contains tools and libraries that provide services for hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. The main concepts in ROS are nodes, packages, topics and messages. Nodes are processes that perform computation. They communicate with one another by publishing messages to topics. Usually, nodes are grouped into packages.

This thesis focuses only on Traffic Light Recognition package provided by Autoware.AI. The package contains different nodes responsible for TLR, for example, a node that uses the baseline approach to predict the state of the traffic light.

This package includes the node *"feat_proj"* responsible for the projection of 3D coordinates of a traffic light to the 2D image coordinates shown in Figure 8. The output of this node is ROI signals that describe a bulb in a traffic signal, as shown in Table 3.

Table 3. Attributes of ROI signals published by *"feat_proj"*.

Attribute	**Type**	**Explanation**
signalId	int	Traffic Signal Bulb ID
u,v	float	2D centre coordinates of the bulb
radius	float	Projected radius of the bulb
x,y,z	float	3D centre coordinates of the bulb
hang	float	Horizontal angle, clockwise from North
type	int	State of the bulb (red, yellow, green)
linkId	int	Closest lane for the traffic light
plId	int	Pole ID to which the traffic light bulb belongs to

A new ROS node *"tlr_yolo"* was added in Traffic Light Recognition package. The node listens to ROI signals published by *"feat_proj"*. It also subscribes to the topic with images from the camera. When ROI signals are coming to the node (i.e. car enters a 60-meters range from the next set of traffic lights), camera image is preprocessed and passed to Tiny-YOLOv3 to get detections. At the same time, ROIs are extracted as described in Section 3.2.2. After this, fusion on ROIs and detected bounding boxes is performed. As a result, we have relevant traffic lights (their pole IDs) and predicted states. The final step is updating ROI signals' states.

It is worth mentioning that the model was converted from Darknet to TensorRT[5] to achieve faster inference.

[4]https://www.ros.org/

[5]https://developer.nvidia.com/TensorRT

4 Results

This section covers the description of experiments and their results. First, we discuss the performance of the object detector, then the performance of the entire system, including the selection of optimal fusion threshold. After this, the effect of λ on the ROI extraction is investigated. Finally, the performance of the baseline and the fusion approach is compared on the prerecorded routes is discussed.

4.1 Detection Performance

Tiny-YOLOv3 was trained for locating only traffic lights and their states. In this thesis, we investigated the effect of different datasets presented in Section 3.1 on the performance of the object detector. Hence, we trained five models training settings of which are represented in Table 4.

We trained **Tiny-YOLOv3-BOSCH** on the BOSCH dataset, which contains traffic lights of different sizes collected in diverse weather conditions. In turn, LISA and LaRA datasets were collected in sunny weather and, therefore, more similar. Based on this, we trained **Tiny-YOLOv3-LISA-LARA** on both LISA and LaRA datasets. Nighttime images from the LISA dataset were not included for training as the proposed approach was planed to be used chiefly in the daytime.

Tiny-YOLOv3-LLB was trained on BOSCH, LISA and LaRA. Nighttime images from LISA dataset were excluded to improve the performance of classifying traffic lights in the daytime, which is the main practical use case of the fusion approach.

Table 4. List of models used in this thesis.

Model	**Datasets**	**Additional information**
Tiny-YOLOv3-BOSCH	BOSCH Dataset	-
Tiny-YOLOv3-LISA-LARA	LISA Dataset LaRA Dataset	Model was trained without nighttime data from LISA dataset.
Tiny-YOLOv3-LLB	LISA Dataset LaRA Dataset BOSCH Dataset	Model was trained without nighttime data from LISA dataset.
Tiny-YOLOv3-LLB-night	LISA Dataset LaRA Dataset BOSCH Dataset	Model was trained with nighttime data from LISA dataset.
Tiny-YOLOv3-LLB-aug	LISA Dataset LaRA Dataset BOSCH Dataset	Model was trained without nighttime data from LISA dataset. Image augmentation was modified: saturation $= 1.1$, exposure $= 1.75$, hue $= 0.05$.

Tiny-YOLOv3-LLB-night was trained on BOSCH, LISA and LaRA. In comparison to **Tiny-YOLOv3-LLB**, nighttime images from LISA dataset were kept to investigate their effect on the performance.

Tiny-YOLOv3-LLB-aug was trained on BOSCH, LISA and LaRA. Nighttime images from LISA dataset were excluded by the same reasoning as with **Tiny-YOLOv3-LLB**. However, image augmentation was added to make images darker. This was done due to a large number of overly bright images in the datasets.

LaRA dataset was apportioned into the training set and test set, with an 80-20 split. Models were trained using the training sets of BOSCH, LISA and LaRA. Distribution of classes is represented in Table 5. All labels in the training sets were converted into YOLO format - bounding boxes' coordinates must be in normalized $xywh$ format (in the range of $0 - 1$), where (x, y) - centre coordinates, w, h - width and height of an object respectively.

Tiny-YOLOv3 implemented on the Darknet[6] framework was used in this work. It is worth noting that all models were trained with default parameters (anchors, batch normalization, image augmentation: saturation $= 1.5$, exposure $= 1.5$, hue $= 0.1$) unless otherwise stated.

Table 5. Distribution of the classes in the training sets and evaluation set.

Class	**LISA**	**LaRA**	**BOSCH**	**ADL**
Red	26089	3471	4157	666
Green	24166	2538	5399	589
Yellow	1555	77	444	61
Unknown	0	142	726	17

Each model was trained for 100000 batches with 64 images per batch and a learning rate of 0.001 using Nvidia Tesla V100 GPU. Camera images are of the size 2064×1544. They were resized to the dimensions of 608×608 (as a compromise between time and accuracy) and only then fed to Tiny-YOLOv3. Bounding boxes produced by Tiny-YOLOv3 are represented in the original image coordinates.

Object detector was evaluated on the ADL dataset using mAP, precision, recall and F1-score. Confidence thresholds $\alpha \in \{0.05, 0.1, 0.3\}$ and IoU threshold θ of 0.5 for mAP calculation were used. Results are demonstrated in Table 6.

Model trained on LaRA, LISA and BOSCH datasets including nighttime data achieves mAP_{50} of 50.01% at $\alpha = 0.1$ on the ADL dataset. It also can be observed that reducing α leads to an increase in recall, but it also results in a drop in precision. In this work, a higher recall is more important since HD-maps are further used for filtering out bounding boxes. **Tiny-YOLOv3-LLB-night** outperforms any other model in terms of recall as

[6]https://github.com/AlexeyAB/darknet

Table 6. Detection performance on ADL Dataset (%).

Confidence	Model	$\mathbf{mAP_{50}}$	Precision	Recall	F1
$\alpha = 0.05$	Tiny-YOLOv3-BOSCH	36.15	36.09	64.74	46.35
	Tiny-YOLOv3-LISA-LARA	23.35	38.67	54.16	45.13
	Tiny-YOLOv3-LLB	44.30	57.78	74.94	65.25
	Tiny-YOLOv3-LLB-night	**50.04**	**62.75**	**79.75**	**70.23**
	Tiny-YOLOv3-LLB-aug	45.98	59.19	73.22	65.45
$\alpha = 0.1$	Tiny-YOLOv3-BOSCH	32.94	40.36	63.61	49.39
	Tiny-YOLOv3-LISA-LARA	23.01	43.02	52.73	47.39
	Tiny-YOLOv3-LLB	44.10	62.23	74.42	67.78
	Tiny-YOLOv3-LLB-night	**50.01**	**67.31**	**78.61**	**72.52**
	Tiny-YOLOv3-LLB-aug	45.83	63.15	72.77	67.62
$\alpha = 0.3$	Tiny-YOLOv3-BOSCH	27.86	48.92	59.79	53.81
	Tiny-YOLOv3-LISA-LARA	21.34	50.16	46.66	48.34
	Tiny-YOLOv3-LLB	42.97	70.66	72.47	71.55
	Tiny-YOLOv3-LLB-night	**49.05**	**75.22**	**76.07**	**75.64**
	Tiny-YOLOv3-LLB-aug	45.12	68.74	70.59	69.65

well. Interestingly, setting α to 0.05 did not improve recall drastically, however, it reduced precision significantly. Based on this, $\alpha = 0.1$ was chosen for the evaluation of the entire system.

Performance of **Tiny-YOLOv3-LLB-night** on the different classes (red, green, yellow and unknown) at $\alpha = 0.1$ is represented in Table 7. It can be seen that model was able to identify about 75% of red traffic lights and approximately 88% of green traffic lights. It also classified half of the yellow traffic lights correctly. However, it failed to detect all traffic lights that are switched off. It can be explained by the small amount of "unknown" traffic lights in the training set (868 instances out of 67896).

Table 7. Detection performance for each class (%).

Class	$\mathbf{AP_{50}}$	Recall	Precision
Red	65.02	74.62	64.21
Green	82.12	87.78	74.49
Yellow	52.91	55.74	53.97
Unknown	0	0	0

In addition to Table 7 confusion matrix for **Tiny-YOLOv3-LLB-night** is provided in Table 8. It can be observed that the model tends to confuse yellow and red traffic lights and vice versa. This confusion does not have real-life implications as both of them are treated as "red". The number of background FP indicates that the model detected objects

that are not present in the ground truth, such as car taillights. Moreover, the model was able to detect far traffic lights, which affected the number of FP. As a reminder, in the ADL dataset, traffic lights were labelled as soon as they are 11 pixels wide or more.

Table 8. Confusion matrix for Tiny-YOLOv3-LLB-night.

		Predicted				
		Red	Green	Yellow	Unknown	Background FN
Ground Truth	Red	497	0	14	0	169
	Green	1	517	0	0	72
	Yellow	24	0	34	0	27
	Unknown	0	2	0	0	17
	Background FP	263	176	5	24	–

All further experiments were conducted only on the best model - **Tiny-YOLOv3-LLB-night**.

4.2 The full system performance

The full system was evaluated on two prerecorded routes in Tartu and Tallinn. The fusion approach heavily relies on the fusion threshold θ. The average area of ROIs is approximately 73131 pixels in the coordinates of the original camera image (for $\lambda = 1.5$). The average area of the detected bounding boxes is 4714 pixels which is almost 16 times smaller than ROI. This implies that the fusion threshold must not be high. Fusion thresholds $\theta \in \{0.025, 0.05, 0.075, 0.1\}$ were investigated in this work.

Table 9 shows the results of the experiment and compares the fusion approach with the baseline.

Table 9. Comparison of the fusion approach and the baseline approach(%). As this is a multiclass classification problem and the evaluation set is imbalanced, weighted averaging is used. Firstly, for each class metrics are calculated. Then their average weighted by the number of true instances of each class is found.

Approach	**Recall**	**Precision**	**Accuracy**	**F1**
Fusion approach ($\theta = 0.025$)	93.03	88.93	93.03	90.83
Fusion approach ($\theta = 0.05$)	92.52	88.44	92.52	90.33
Fusion approach ($\theta = 0.075$)	90.62	86.67	90.62	88.52
Fusion approach ($\theta = 0.1$)	82.64	79.46	82.64	80.94
Baseline approach	89.68	84.54	89.68	86.64

It can be seen that the fusion threshold θ of 0.025 leads to higher performance and therefore was used for the rest of the experiments. The fusion approach outperformed the

baseline approach by achieving accuracy of 93%. It also achieved F1-score of 90.83% on Tallinn and Tartu routes which is higher than the baseline by approximately 3%.

Confusion matrices for the fusion approach and the baseline approach are represented in Table 10 and Table 11 respectively. It can be seen that both approaches were not able to detect unknown traffic lights. The fusion approach proved to be more efficient in detecting green traffic lights. Both the baseline approach and the fusion approach classified almost all yellow traffic lights as red.

The baseline approach misclassified 9 red traffic lights as green. Similarly, the fusion approach detected 5 red traffic lights as green. This behaviour can cause accidents.

Table 10. Confusion matrix for the fusion approach.

		Predicted			
		Red	Green	Yellow	Unknown
Ground Truth	Red	404	5	6	0
	Green	0	328	0	0
	Yellow	19	0	2	0
	Unknown	7	19	0	0

Table 11. Confusion matrix for the baseline approach.

		Predicted			
		Red	Green	Yellow	Unknown
Ground Truth	Red	406	9	0	0
	Green	27	301	0	0
	Yellow	21	0	0	0
	Unknown	4	22	0	0

It should be mentioned that currently, Autoware.AI treats all yellow traffic lights as red ones. Therefore, in this case, misclassifying red traffic lights as yellow and vice versa will not cause any problems.

Time needed for baseline and fusion approach to predict traffic light state was compared. From Figure 11 can be seen that the fusion approach performs slower than the baseline approach. Interestingly, it took about 21 milliseconds (in average) to predict on the single image using Tiny-YOLOv3 while for the baseline approach the inference time was 23.9 milliseconds. However, post-processing (non-maximum suppression) took 11.42 milliseconds which resulted in the increasing of the performance time of the fusion approach 1.4 times.It is worth mentioning that the fusion takes about 0.073 milliseconds which is the reason why it was not included into the plot.

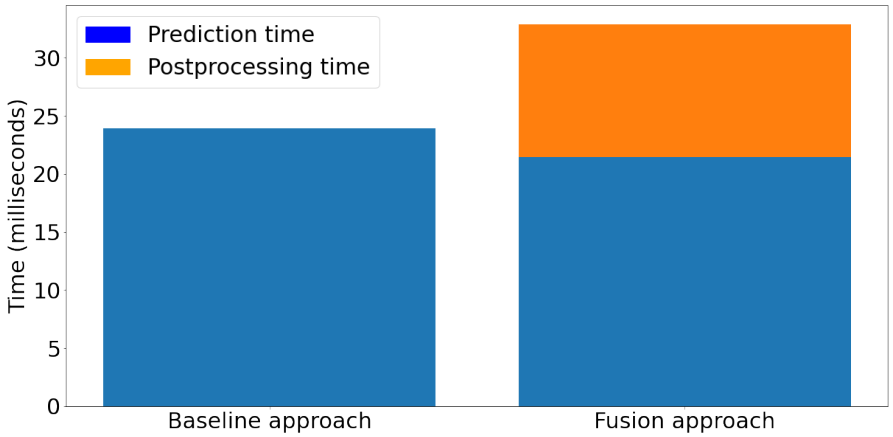


Figure 11. Time needed for both approaches to detect traffic lights on the single image.

4.3 Effect of ROI extraction on the performance

ROI extraction procedure described in Section 3.2.2 uses λ to mitigate possible localization errors. The default value of λ used by the baseline approach is 1.5. By reducing λ, the ROI area will decrease. Since the fusion approach uses Tiny-YOLOv3 that outputs exact bounding boxes of traffic lights, a lower value for λ could be beneficial: the lower λ, the smaller area of ROIs and, as a consequence, the IoU score between ROI and the detected bounding box is higher. Therefore, it is more likely that the detected box will be associated with ROI.

Hence, eight values for λ, $\lambda \in \{0.05, 0.1, 0.25, 0.5, 0.75, 1.0, 1.25, 1.5\}$ were investigated. Effect of λ on the performance of baseline approach was investigated as well.

Figure 12 represents how much ROI area reduces with decreasing of λ. It can be observed that decreasing λ from 1.5 to 0.5 results in a drop of ROI area almost two times.

Figure 13a shows the results of the experiment. It can be seen that changing λ did not improve the performance of the fusion approach. It can be explained by the fact that the proposed approach is heavily dependant on the fusion threshold.

However, from Figure 13b can be seen that reducing λ from 1.5 to 0.75 improves the accuracy of the baseline approach by approximately 3%. Figure 14a shows the ROI that was extracted using $\lambda = 1.5$. ROI includes two traffic lights which leads to the confusion of a classifier and, as a consequence, the wrong prediction. Setting λ to 0.75 made ROI smaller and almost discharged one traffic light (Figure 14b). This resulted in a correct prediction.

It is worth noting while decreasing λ reduces ROI area, the risk arises that localization errors will be penalized less. Therefore, it is more likely that ROI will not include traffic light at all.

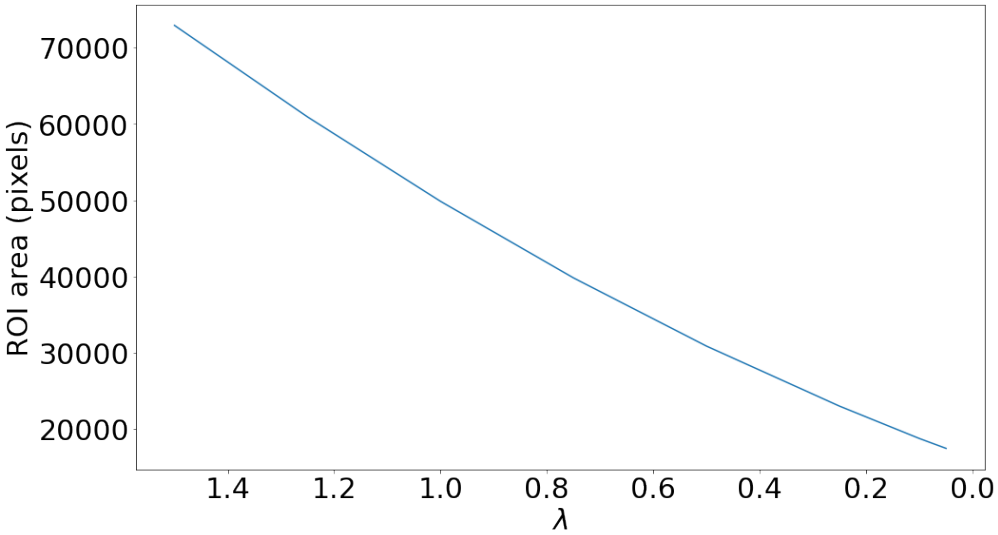
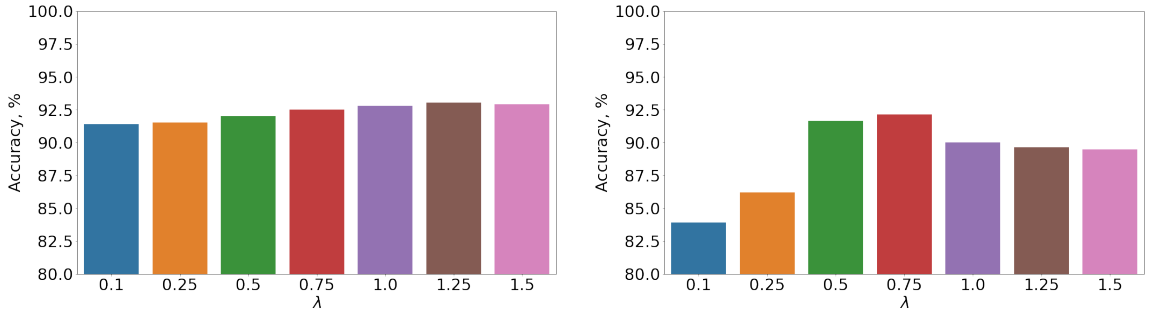


Figure 12. Effect of λ on ROI area.



(a) The fusion approach

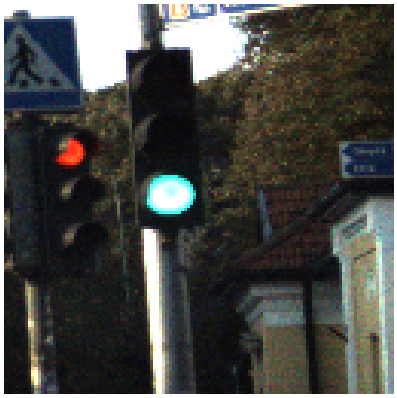
(b) The baseline approach

Figure 13. Effect of λ on the performance.

4.4 Results on Prerecorded Routes

We measured how early the system was able to detect traffic lights: the time the system took to output the first prediction. We provide two plots for Tallinn and Tartu routes. Each plot shows ground truth and performances of both baseline and fusion approaches over time. On the plot, each colour corresponds to the traffic light state. The black colour indicates that there were no traffic lights on the car's route.

The measurements are represented in Figure 15 for Tallinn route. It can be seen that the fusion approach produced the first correct prediction faster. In turn, baseline outputted "unknown", which can be explained by localization errors and, as a consequence ROIs that did not contain traffic lights. It is worth explaining that baseline still outputs red



(a) $\lambda = 1.5$. Predicted by the baseline approach TL state - red.

(b) $\lambda = 0.75$. Predicted by the baseline approach TL state - green.

Figure 14. Example of effect of λ.

color (for example, the first red TL) while ground truth is already green because the state changes only if the prediction was the same for two frames.

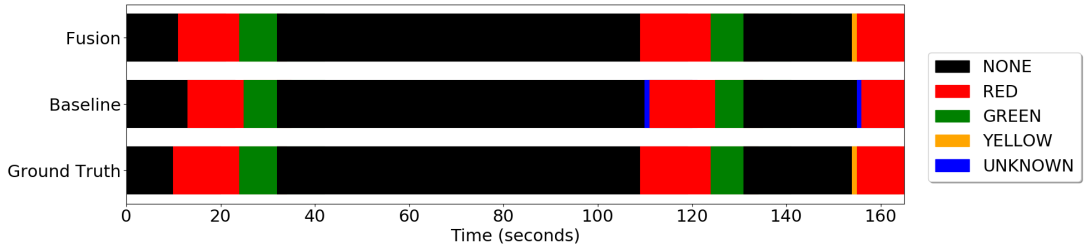


Figure 15. Tallinn route.

Measurements for Tartu route are represented in Figure 16. There are two "unknown" traffic lights in the ground truth:

1. Since the traffic light is not facing the camera, its state cannot be observed in the beginning.

2. None of the traffic light's bulbs are active.

The fusion approach again produced the first correct prediction faster than the baseline. Localization errors affected the performance of the baseline, which resulted in "unknown" output.

It can be observed that Tiny-YOLOv3 detected the first red traffic light much later than the car approached it. However, the baseline classified the same traffic light as green.

It happened as ROI included both relevant red traffic light and green pedestrian traffic light.

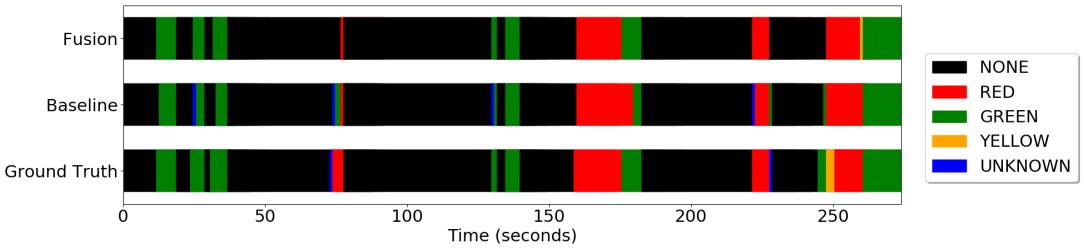


Figure 16. Tartu route.

5 Discussion

This section discusses and interprets obtained results.

Tiny-YOLOv3 was trained on the different datasets: LISA, LaRA and BOSCH. It was discovered discovered that the model trained only on BOSCH achieved higher recall, precision and mean Average Precision (mAP) on the ADL dataset than the model trained on both LISA (without nighttime clips) and LaRA. The BOSCH dataset consists of traffic lights of different sizes. It was collected in various weather conditions. In turn, most images from LISA and LaRA datasets were taken in sunny weather. Thus, the model trained on BOSCH managed to handle different weather conditions and sizes of traffic lights better.

One of the models was trained on all mentioned datasets only on daytime data as it is the primary practical use case of the model. It outperformed the model trained only on the BOSCH dataset. From this, we can draw a conclusion that the number of training instances affects the performance significantly. Another model was trained on all datasets with additional data augmentation (to make images darker). However, it did not improve performance.

The best model was trained on the BOSCH, LISA and LaRA datasets, including nighttime clips. Our results demonstrated that the model could correctly classify 75% red and 88% green traffic lights from the ADL dataset. We can conclude that this model was able to generalize on different lighting conditions better. However, the model failed to detect any unknown traffic lights, resulting from the small number of "unknown" instances in the training set. Considering that currently, Autoware treats yellow traffic lights as red, we can conclude that the model could detect almost 80% red traffic lights from the ADL dataset correctly.

Section 4.2 describes the results of the proposed method, which showed that it was able to identify relevant traffic lights and their states. It achieved F1-score of 90.83%, on the prerecorded routes which is higher than the baseline by almost 3%. Moreover, it also outperformed the baseline in terms of accuracy by yielding 93%. It can be explained by the fact that the proposed method mitigates localization errors by applying Intersection-over-Union. For example, if extracted ROI does not contain the traffic light, the baseline will fail to classify the traffic light. In turn, the fusion approach associates each ROI with one of the traffic lights detected by Tiny-YOLOv3. This suggests that the performance of the fusion approach heavily relies on the performance of Tiny-YOLOv3 and the fusion threshold.

In this thesis, experiments were conducted with different λ for ROI extraction. It was discovered that lowering λ did not improve the performance of the fusion approach and explained that by the importance of the fusion threshold. However, changing λ improved the performance of the baseline. The possible reason for that could be the training set baseline was trained on.

Finally, time needed for both approaches to produce the first detection was measured.

The fusion approach produced the first correct prediction faster for both routes. It can be observed that the baseline is easily affected by localization errors which led to classifying traffic lights as "unknown". Moreover, for brief moments baseline detected red as green. This type of errors is the worst as it can lead to fatal accidents.

The proposed approach suffers from the limitations associated with the HD-map. If a new traffic light was installed, it must be added to the HD-map. However, it is very costly to keep HD-maps up-to-date. Furthermore, the fusion approach relies on the fusion threshold which has to be selected carefully to guarantee that none of the traffic lights are missed.

6 Conclusion

This thesis introduces the fusion of HD-map information and object detection for traffic light recognition problems. First, an object detector performs traffic light recognition on the camera image. Then relevant traffic lights are selected from the HD-map and associated with one of the detected traffic lights state using Intersection-over-Union. The proposed system was implemented on top of Autoware.AI.

Tiny-YOLOv3 was trained on different datasets and it was discovered that training on LISA, LaRA and BOSCH datasets, including both daytime and nighttime clips outperformed any other model. Object detector was evaluated on the ADL dataset annotated beforehand. Tiny-YOLOv3 was able to achieve a recall of almost 80% and mAP of 50%. However, Tiny-YOLOv3 failed to detect "unknown" traffic lights and could not identify the state of the traffic lights that did not fully face the camera.

The proposed method was compared to the baseline approach on two prerecorded routes. The system proved to be efficient and identified relevant traffic lights correctly. The fusion approach yielded an accuracy of almost 93%. In comparison, the baseline approach achieved only 89%. Moreover, the fusion approach correctly identified the first traffic light on the route faster than baseline. It proved to be more robust to the localization errors as well.

Although results seem promising, some changes can be made that the proposed method can benefit from. In particular, it is important to improve the performance of the object detector. Moreover, dependency on the fusion threshold could also be mitigated. As future work, it would be interesting to combine both fusion and baseline approaches to understand if it can help compensate for each method's drawbacks. Additionally, deciding directly from the camera images if traffic light applies to the car could resolve limitations associated with HD maps.

References

[1] Manal El Aidouni. Evaluating Object Detection Models: Guide to Performance Metrics. https://manalelaidouni.github.io/Evaluating-Object-Detection-Models-Guide-to-Performance-Metrics.html#key-notions.

[2] Martin Bach, Daniel Stumper, and Klaus Dietmayer. Deep convolutional traffic light recognition for automated driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 851–858. IEEE, 2018.

[3] Karsten Behrendt, Libor Novak, and Rami Botros. A deep learning approach to traffic lights: Detection, tracking, and classification. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1370–1377, 2017.

[4] Guillaume Bresson, Zayed Alsayed, Li Yu, and Sébastien Glaser. Simultaneous localization and mapping: A survey of current trends in autonomous driving. *IEEE Transactions on Intelligent Vehicles*, 2(3):194–220, 2017.

[5] Michael Buckland and Fredric Gey. The relationship between recall and precision. *Journal of the American society for information science*, 45(1):12–19, 1994.

[6] Jiwoong Choi, Dayoung Chun, Hyun Kim, and Hyuk-Jae Lee. Gaussian yolov3: An accurate and fast object detector using localization uncertainty for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 502–511, 2019.

[7] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[8] Abhinav Dadhich. *Practical Computer Vision: Extract Insightful Information from Images Using TensorFlow, Keras, and OpenCV*. Packt Publishing, 2018.

[9] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, volume 1, pages 886–893. Ieee, 2005.

[10] Jesse Davis and Mark Goadrich. The relationship between precision-recall and roc curves. In *Proceedings of the 23rd international conference on Machine learning*, pages 233–240, 2006.

[11] Raoul De Charette and Fawzi Nashashibi. Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates. In *2009 IEEE Intelligent Vehicles Symposium*, pages 358–363. IEEE, 2009.

[12] Moises Diaz, Pietro Cerri, Giuseppe Pirlo, Miguel A Ferrer, and Donato Impedovo. A survey on traffic light detection. In *International Conference on Image Analysis and Processing*, pages 201–208. Springer, 2015.

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[14] Rui Fan, Jianhao Jiao, Haoyang Ye, Yang Yu, Ioannis Pitas, and Ming Liu. Key ingredients of self-driving cars. *arXiv preprint arXiv:1906.02939*, 2019.

[15] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

[16] Lieve Hamers et al. Similarity measures in scientometric research: The jaccard index versus salton's cosine formula. *Information Processing and Management*, 25(3):315–18, 1989.

[17] Manato Hirabayashi, Adi Sujiwo, Abraham Monrroy, Shinpei Kato, and Masato Edahiro. Traffic light recognition using high-definition map features. *Robotics and Autonomous Systems*, 111:62–72, 2019.

[18] Tin Kam Ho. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition*, volume 1, pages 278–282. IEEE, 1995.

[19] Jonathan Hui. mAP (mean Average Precision) for Object Detection. `https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173`.

[20] Jonathan Hui. Real-time Object Detection with YOLO, YOLOv2 and now YOLOv3. `https://jonathan-hui.medium.com/real-time-object-detection-with-yolo-yolov2-28b1b93e2088`.

[21] Chulhoon Jang, Sungjin Cho, Sangwoo Jeong, Jae Kyu Suhr, Ho Gi Jung, and Myoungho Sunwoo. Traffic light recognition exploiting map and localization at every stage. *Expert Systems with Applications*, 88:290–304, 2017.

[22] Morten Bornø Jensen, Mark Philip Philipsen, Andreas Møgelmose, Thomas Baltzer Moeslund, and Mohan Manubhai Trivedi. Vision for looking at traffic lights: Issues, survey, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, 17(7):1800–1815, 2016.

[23] S. Kato, E. Takeuchi, Y. Ishiguro, Y. Ninomiya, K. Takeda, and T. Hamada. An open approach to autonomous vehicles. *IEEE Micro*, 35(6):60–68, 2015.

[24] Larry Kirkpatrick and Gregory E Francis. *Physics: A conceptual world view*. Cengage Learning, 2009.

[25] Jesse Levinson, Jake Askeland, Jan Becker, Jennifer Dolson, David Held, Soeren Kammel, J Zico Kolter, Dirk Langer, Oliver Pink, Vaughan Pratt, et al. Towards fully autonomous driving: Systems and algorithms. In *2011 IEEE Intelligent Vehicles Symposium (IV)*, pages 163–168. IEEE, 2011.

[26] Ethan Yanjia Li. Dive Really Deep into YOLO v3: A Beginner's Guide. `https://towardsdatascience.com/dive-really-deep-into-yolo-v3-a-beginners-guide-9e3d2666280e`.

[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[29] Li Liu, Wanli Ouyang, Xiaogang Wang, Paul Fieguth, Jie Chen, Xinwang Liu, and Matti Pietikäinen. Deep learning for generic object detection: A survey. *International journal of computer vision*, 128(2):261–318, 2020.

[30] Rong Liu, Jinling Wang, and Bingqi Zhang. High definition map for automated driving: Overview and analysis. *The Journal of Navigation*, 73(2):324–341, 2020.

[31] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *European conference on computer vision*, pages 21–37. Springer, 2016.

[32] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA, 1967.

[33] Masako Omachi and Shinichiro Omachi. Traffic light detection with color and edge information. In *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pages 284–287. IEEE, 2009.

[34] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 821–830, 2019.

[35] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[36] Lucas C Possatti, Rânik Guidolini, Vinicius B Cardoso, Rodrigo F Berriel, Thiago M Paixão, Claudine Badue, Alberto F De Souza, and Thiago Oliveira-Santos. Traffic light recognition using deep learning and prior maps for autonomous cars. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2019.

[37] David MW Powers. Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation. *arXiv preprint arXiv:2010.16061*, 2020.

[38] Morgan Quigley, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, Andrew Y Ng, et al. Ros: an open-source robot operating system. In *ICRA workshop on open source software*, volume 3, page 5. Kobe, Japan, 2009.

[39] M. S. Ramanagopal, C. Anderson, R. Vasudevan, and M. Johnson-Roberson. Failing to learn: Autonomously identifying perception failures for self-driving cars. *IEEE Robotics and Automation Letters*, 3(4):3860–3867, 2018.

[40] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[41] Joseph Redmon and Ali Farhadi. Yolo9000: better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[42] Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[43] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[44] Adrian Rosebrock. Intersection over Union (IoU) for object detection. `https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/`.

[45] Yutaka Sasaki. The truth of the f-measure. *Teach Tutor Mater*, 01 2007.

[46] Sanjivani Shantaiya, Keshri Verma, and Kamal Mehta. A survey on approaches of object detection. *International Journal of Computer Applications*, 65(18), 2013.

[47] Santokh Singh. Critical reasons for crashes investigated in the national motor vehicle crash causation survey. Technical report, 2015.

[48] Vincent Torre and Tomaso A Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, (2):147–163, 1986.

[49] Tzutalin. LabelImg. `https://github.com/tzutalin/labelImg`.

[50] C. Urmson and W. ". Whittaker. Self-driving cars and the urban challenge. *IEEE Intelligent Systems*, 23(2):66–68, 2008.

[51] Michael Weber, Peter Wolf, and J Marius Zöllner. Deeptlr: A single deep convolutional network for detection and classification of traffic lights. In *2016 IEEE intelligent vehicles symposium (IV)*, pages 342–348. IEEE, 2016.

[52] Xiongwei Wu, Doyen Sahoo, and Steven CH Hoi. Recent advances in deep learning for object detection. *Neurocomputing*, 396:39–64, 2020.

[53] Zhang Yi, Shen Yongliang, and Zhang Jun. An improved tiny-yolov3 pedestrian detection algorithm. *Optik*, 183:17–23, 2019.

[54] Zhengxia Zou, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *arXiv preprint arXiv:1905.05055*, 2019.

Appendix

I. Glossary

Acronyms

ADL Autonomous Driving Lab. 20, 22, 26, 29

AP Average Precision. 13

GNSS Global Navigation Satellite System. 19, 23

IMU Inertial Measurement Unit. 19, 23

IoU Intersection-over-Union. 11–13, 15–17, 26, 28, 40

mAP mean Average Precision. 13, 17, 28, 29, 32, 40

NMS non-maximum suppression. 17, 35

ROI region of interest. 2, 6, 10, 20–22, 26, 29, 30, 36, 37, 39

TL traffic light. 20, 24, 27, 38

TLR Traffic Light Recognition. 8–10, 20, 30

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Tetiana Shtym**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Traffic light detection by fusing object detection and map info,
   *(*title of thesis)

   supervised by Tambet Matiisen and Meelis Kull.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Tetiana Shtym
13/05/2021