# Extending Traffic Light Recognition: Efficient Classification of Phase and Pictogram

Matthias Michael and Marc Schlipsing
Institut für Neuroinformatik
Ruhr-Universität Bochum
Email: *firstname.lastname*@ini.rub.de

***Abstract*—While much work in the domain of traffic lights recognition is invested in the detection of traffic lights, classification of their exact state (including color phase and possible arrow pictogram) is often neglected. In this paper, we propose a robust approach for efficient video-based classification of said state with particular attention to the displayed pictogram and an additional ability to reject false detections. The currently active lights are identified and used to classify the phase. The lights are extracted and transformed into a HOG feature representation that is used to classify the pictogram with the help of machine learning classifiers. In order to gain optimal results, we compared the performance of different algorithms, namely LDA, kNN, and SVM. We provide an evaluation of our method on individual images and demonstrate that the classification rate of the phase lies at 96.7 % and at 92.8 % for the pictogram, with the use of SVMs providing best results. This leads to an overall classification quality of 89.9 %. With a runtime of less than 1 ms per image section our algorithm can easily be integrated in every traffic light recognition pipeline.**

## I. Introduction

For applications in the field of assisted and autonomous driving it is of huge importance to build and maintain a representation of the environment of the car in order to execute appropriate actions or – in the case of assisted driving – to display the currently most relevant information to the driver. Many parts of this environment are regulated by traffic law and have a standardized appearance (e. g., traffic signs and street markings) which greatly simplifies automatic recognition.

A considerable amount of effort has been invested in the detection of traffic lights – however, many methods concentrate on the detection alone and ignore the actual state of the traffic light. At the utmost, a coarse classification in red, green, and possibly yellow traffic lights is made – either explicitly after, or implicitly during the detection process. By doing this, a large amount of usable information is disregarded since traffic lights may also provide the information to which lane they apply.

This information includes the permission to drive for green lights and the prompt to stop at red lights, which are the most simple cases. A yellow light's informative content depends on the position of the car relative to the traffic light. If the car is far away, the breaking process should be initiated. However, if the car is rather close to the traffic light this is not necessary. In some countries traffic lights provide additional information in a fourth phase where the red and the yellow light are switched on simultaneously. This phase occurs between the red and the green phase and cues the driver to start driving.

Additionally, the lights sometimes contain information on the direction the driver can follow. In particular at intersections with multiple lanes, the traffic light indicates the possible turning maneuvers for that lane. This is depicted by arrows on the traffic light which are displayed in black on top of the red and yellow lights while the green light itself consists of a colored arrow on top of a black background. The most common directions are *left*, *right*, and *top*. Figure 1 gives an overview of the different types of traffic lights we aim to distinguish in this work. To our knowledge, no previous work has considered extracting this information from video sequences.

Additionally, the classification provides the opportunity to evaluate the correctness of the detected traffic lights. Usually only a small number of detections should occur in every frame. Therefore, it is possible to apply rather costly evaluation methods which would require too much time in an earlier step of the recognition pipeline.

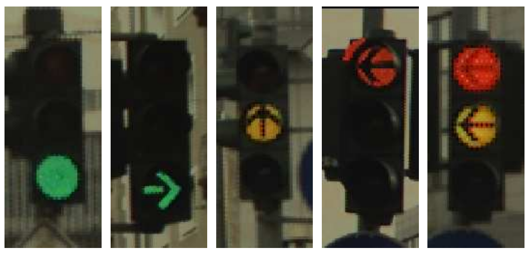
Fig. 1. Examples of the different types of traffic lights we aim to distinguish. All four different phases and possible pictograms are shown.

As shown in Tab. I, there are four different color phases a traffic light can enter and four different pictograms it can display which leads to a total of 16 classes. We chose to simplify the classification process by dividing it into two parts. During the first part, the color phase is determined by finding ROIs (regions of interest) around the active lights. During the second part we determine the pictogram on the active light; therefor we evaluate the performance of the Linear Discriminant Analysis, the k-Nearest Neighbor classifier and the Support Vector Machine on this problem.

We provide an overview of current techniques which can be used to detect and (partially) classify traffic lights in video sequences in Sec. II. The proposed method is explained in detail in Sec. III. Afterwards, in Sec. IV and V, we present the conducted experiments and the results that were achieved. Finally, in Sec. VI, a conclusion is given.

TABLE I. AN OVERVIEW OF THE DIFFERENT ATTRIBUTES A TRAFFIC LIGHT CAN DISPLAY FOR ITS PHASE AND ITS PICTOGRAM. THE ACTUAL CLASSES ARE GIVEN BY THE CROSS PRODUCT BETWEEN **PHASE** AND **PICTOGRAM** LEADING TO A TOTAL OF 16 CLASSES.

| **Phase** | **Pictogram** |
|---|---|
| green | none |
| yellow | left |
| red | top |
| red-green | right |

## II. Related Work

As stated in the previous section, a major part of research is concerned with the detection of traffic lights while the further distinction of their phase or other aspects is often neglected. Examples for this work are Omachi and Omachi [1], Wang et. al [2], and Gong et. al [3]. In the following we will briefly describe current methods that can be used to detect traffic lights while also providing a minimal amount of classification.

Charette et. al [4] applied a spot light detection approach based on the Top-hat morphological operator in order to identify light sources in an image which allows them to detect suspended as well as supported lights. The operator extracts hypotheses for possible locations of traffic lights – here a large number of false positives is allowed. These hypotheses are filtered and an adaptive template matching is applied to the remaining locations. A basic classification has to be made here since different templates have to be used for green, yellow, and red traffic lights.

John et. al [5] propose the use of deep learning methods for traffic light recognition. They use GPS information together with a traffic light location database in order to limit the regions in the camera images which may contain the lights. Inside this region, candidate ROIs are identified using color, intensity, and shape information. These ROIs are then fed into a convolutional neural network which provides final detection as well as classification by deciding for each ROI, if it contains a red, a green, or no traffic light. In environments with low illumination additional robustness is provided by the use of a saliency map. While achieving a good detection result, the presented system is not real-time capable and only distinguishes between red and green lights.

An entire recognition pipeline consisting of detection, tracking, and classification is proposed by Lindner et. al [6]. Different detectors are used depending on the availability of color or gray scale cameras. For the classification a feed forward neural network is applied which distinguishes between red, yellow, and green traffic lights and – similar to our work – can also reject wrong detections.

Cai et. al [8] are among the few authors that propose an holistic method for the detection and classification of traffic lights including color and arrow state. They apply shape and color constraints for detection and use a kNN-Classifier on features obtained by a 2D Gabor wavelet transform and 2DICA for classification. While achieving a recognition rate of 91 %, their work still leaves room for improvement. Classification of a single image requires 56 ms processing time which leads to a total framerate of 6.57 fps, making the method not real-time capable. All arrow lights in their dataset consist of an illuminated shape on black background which eliminates the difficulties of recognizing a black shape on an illuminated background. Additionally, the authors do not provide information on their choice of parameter values and the distribution of their data into training and test set for the kNN-Classifier.

## III. Method

It is noteworthy that the term "traffic light" is used to describe the entirety of the three circular, colored lights on a rectangular plate, while "light" refers to an individual lamp. As stated in Section I, the proposed method is divided into two distinct steps – in the first step it is decided which colored light (or which combination of lights) is displayed by the traffic light (also denoted as the "phase" of the traffic light); in the second step it is determined, in which direction the arrows are pointing if they are present (or which "pictogram" is displayed by the traffic light).

Images showing only the traffic light (as well as a small surrounding region) – identical to those shown in Fig. 1 – serve as input to our system. They can be produced by applying any of the methods described in Sec. II. For further processing a few assumptions are made about the input images and the traffic lights, which, however, should be easily satisfiable:

- The entire traffic light is included in the image and it is roughly centered.
- Only one traffic light is included in a single image.
- The active light has a diameter of at least 4 pixels.
- From top to bottom the colors red, yellow, and green are present.
- An arrow – if present – is either black on a red or yellow background, or green on a black background and points in one of the directions specified in Tab. I.

### A. Phase Classification

The goal of this first step is to find precise rectangular ROIs $R_g$, $R_y$, and $R_r$ around each of active lights. The classification decision can be inferred from the presence (and absence) of a ROI for a certain color. If only a single ROI can be found, the current phase of the traffic light can be classified as "green" for the presence of $R_g$, "yellow" for the presence of $R_y$, and "red" for the presence of $R_r$.

For multiple ROIs only the simultaneous appearance of $R_y$ and $R_r$ is considered valid. In any other case – no found ROI or a different combination of ROIs –, the current input image is regarded as invalid and not processed any further. As mentioned before, this provides the possibility to validate the results of the previous traffic light detection. The process of obtaining the ROIs is described in the following.

Since the colors of the individual lights are clearly defined, color segmentation is a reasonable approach to identify candidate pixels which may be part of a light. The threshold values for the segmentation are defined in the HSV space. In this color space the intervals describing the possible colors of a light are coherent, which makes it possible to perform the color segmentation with only six comparisons for each pixel and color. The hue primarily defines the color while value and saturation can be chosen in such a way that only bright pixels which may be part of a light emitting source are segmented.

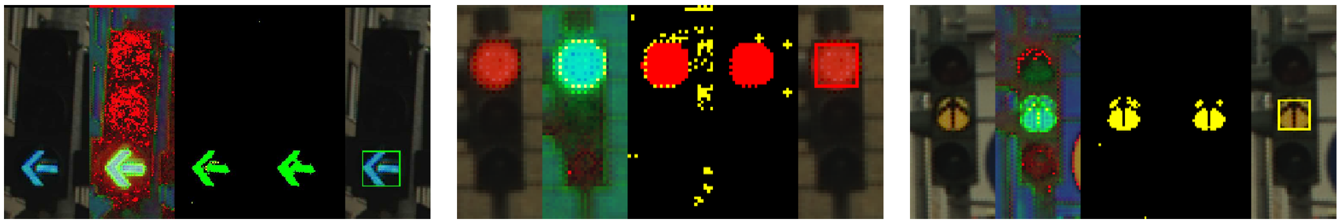

Fig. 2. Easily classifiable examples of the intermediate steps during the phase classification. In every image the left part shows the original input image. The left-to-center part shows the HSV color representation of the image with the channels mapped to the RGB color space; the lights display a bright greenish-blue color since they have high saturation and value, which is mapped to the green and blue color channels, respectively. The center part shows the initial color segmentation with the parameters given in Tab. IV. The right-to-center part shows the color segmentation after the morphological operators. Finally, the rightmost part shows the final ROI hypothesis for the light.

During this initial segmentation we aim to include as many pixels of a light as possible while accepting erroneously segmented pixels which will be removed in subsequent processing steps. From the segmentation we receive three binary images $I_g^s$, $I_y^s$, and $I_r^s$ for the green, yellow, and red color definitions respectively, where candidate pixels are marked as 1 while the remaining pixels are set to 0.

In order to remove noise and small errors we perform an opening on $I_y^s$ and $I_r^s$. Because the pictogram on a green light is inverted, this procedure is not suitable for the segmented green pixels. The illuminated ones are smaller and not entirely connected since there is a small gap between the two parts of the arrow. Applying an opening, one would risk to delete correctly segmented pixels which carry valuable shape information for the second classification step. Thus, we perform a closing on $I_g^s$ in order to close said gap if an arrow should be present. As structural element, a cross with the size of $3 \times 3$ pixels is used.

Based on these denoised binary images we want to determine the position of the actual light. In order to do this it is necessary to identify the number of connected segmented regions. If more than one region for a certain color is present, it has to be decided which one – if any – is most likely to describe a light. We approach this problem by performing a clustering on the segmented pixels which is based on the 8-connected neighborhood of each pixel and can return several clusters $C_g^i$, $C_y^i$, and $C_r^i$, where $i$ is the running index for all clusters of a single color.

This clustering is refined by joining clusters that are particular close to each other. Two clusters of a color are merged together if the size of the bounding box around both regions exceeds the summed size of both individual bounding boxes by less than a certain amount $t_{growth}$; i. e., if

$$s(C_x^i \cup C_x^j) < \left(s(C_x^i) + s(C_x^j)\right) \cdot t_{growth}, \tag{1}$$

where $x$ denotes one of the colors and $s(\cdot)$ returns the size of the bounding box around a cluster. This allows to gather all pixels of a green light in a single cluster even if the closing operation was not able to close the gaps. This is especially important for red and yellow lights with pictograms where the opening may have split the light in separate regions.

After the clustering, each clustered region is validated with respect to their size, shape, and location in the image. We require an active light to have a diameter of at least 4 pixels, which can for convenience be expressed in the configurable threshold $t_{diameter}$. Smaller clusters cannot represent a valid light and can, thus, be ignored.

A single light is roughly circular; therefore the aspect ratio of a clustered region should be close to one. In some cases we observed a failure of the color segmentation to accurately segment the top half of lights with an arrow to the top. Therefore, we chose a rather loose threshold $t_{aspect} = 2$ for the aspect ratio – if the length of one side of the bounding box amounts to more than double the length of the other side, the current cluster is discarded.

Due to the strict layout of traffic lights – as stated in the assumptions at the beginning of this section – we require that red segmented pixels need to lie in the upper, yellow segmented pixels in the middle, and green segmented pixels in the bottom third of the input image in order to belong to the respective light. We enforce this requirement by checking the area of the cluster's bounding box that overlaps with the respective third of the image an applying a threshold $t_{intersect}$. Let $I_t$ be the top third of the image; a cluster is discarded if

$$s\left(I_t \cap C_r^i\right) < s(C_r^i) \cdot t_{intersect}. \tag{2}$$

A rather small value of $t_{intersect}$ allows for less accurate – and therefore faster – detectors since the traffic lights can be slightly off-center.

In many cases this filtering process already removes all clusters except for the correct one around which we can construct the final ROI $R_i$. If there should still be multiple hypotheses for the location of the light, we simply choose the largest one. After that we end up with at most one ROI for each color. We observed that in some cases ROIs are found around objects with similar colors in the background of the traffic light; e. g., a priority sign that appears at the same height as the yellow light. These errors – as well as other wrong detections – share the aspect, that the ROIs are comparatively small. However, they can not be eliminated solely on their size since they are still big enough

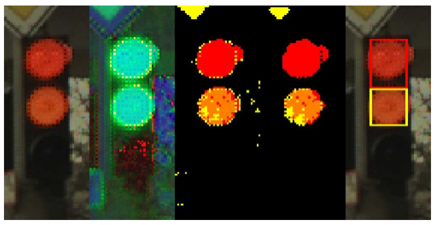

Fig. 3. In this example the color definitions for red and yellow overlap (as detailed in Sec. V). This is indicated by orange color in the segmentation images. The shapes of $R_r$ and $R_y$ are adapted so that they do not intersect.

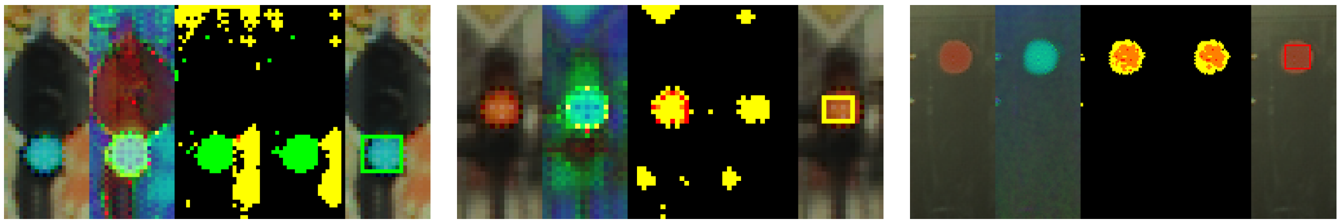

Fig. 4. Examples that are more difficult to classify. The structure of each individual image is identical to those in Fig. 2. *Left:* Here a large part of the background was segmented as yellow; due to the check for the maximal aspect ratio our algorithm was able to correctly detect only the green light. *Middle:* Multiple hypotheses for a yellow light exist; all erroneous ones could be eliminated due to their size and position. *Right:* The foggy environment led to a simultaneous segmentation of the light as red and yellow; due to its position, the light could be identified as red.

to resemble a dim light of a traffic light that was not cut out accurately. If ROIs of different colors were found, we have the possibility to validate their sizes with respect to each other. Again we apply a threshold $t_{size}$ on the maximal size differences between two ROIs $R_i$ and $R_j$. $R_i$ is removed if $s(R_i) \cdot t_{size} < s(R_j)$.

The appearance of hypotheses $R_r$ and $R_y$ for the red and the yellow light at the same time gives us additional opportunities for improvement. Due to the layout of the lights, $R_r$ and $R_y$ should be vertically centered and should not intersect. We adapt the horizontal coordinates of the ROIs to enforce the centering and remove intersections by shrinking the larger ROI. An example where this is necessary is given in Fig. 3.

After this step we have obtained the final ROIs which we can infer the phase classification result from. Figure 2 gives an overview of the intermediate steps described so far. Table II summarizes the adjustable threshold parameters that have been applied.

### B. Pictogram classification

Besides wanting to obtain the best possible classification result there is another reason to invest large effort in finding ROIs around active lights. These ROIs serve as input to the next step – the classification of the pictograms on the lights. The process that is described in the following is performed on the contents of each ROI individually – for each ROI a classification result is calculated. In the case of multiple detected ROIs these results are compared afterwards; if they contradict each other, no final decision regarding the arrow direction can be made.

TABLE II. THE DIFFERENT PARAMETERS THAT TAKE EFFECT DURING THE PHASE CLASSIFICATION STEP, THEIR MEANING, AND THE VALUES WE CHOSE FOR THEM.

| Variable | Meaning | Value |
|---|---|---|
| $t_{diameter}$ | Minimal diameter of a traffic light in pixels | 4 |
| $t_{aspeact}$ | Maximal aspect ratio of segmented area | 2 |
| $t_{growth}$ | Maximal factor the joined ROI might be bigger than both individual ROIs during the clustering | 1.5 |
| $t_{intersect}$ | Minimal amount of a ROI that needs to lie in its respective third of the image | 0.3 |
| $t_{size}$ | Maximal factor a ROI is allowed to be smaller than another ROI | 2 |

We applied state-of-the-art machine learning algorithms for the classification. While classification on the raw image data might be possible we searched for descriptive image features which expose the most relevant information for classification. The arrows on the lights are mainly defined by their differently oriented edges, which led us to the decision to use *Histogram of Oriented Gradients* (HOG) features [9]. HOG features are invariant against translation and scaling and have been proven efficient for several other detection and classification tasks [10], [11].

The HOG features are calculated on cut out regions of the original input image defined by the ROIs. Following the recommendation of the original authors, the gradients themselves are calculated on each RGB color channel individually and the maximum is chosen. To ensure consistent results and to simplify the classification procedure by using feature vectors of identical length, we scale all ROI images to $24 \times 24$ pixels before applying the HOG descriptor. This includes downsampling larger ROIs.

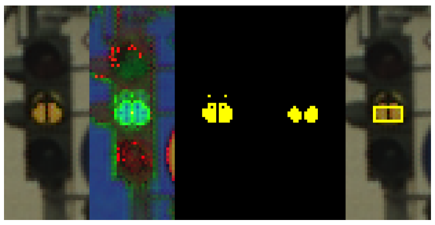

Fig. 5. In some cases the upper part of a yellow arrow to the top is cut off. A ROI can still be found, however it only includes the bottom part of the arrow containing the straight line. By using a separate classifier for the yellow lights we account for this source of errors.

Since the green pictograms are inverted, they should be fed into a separate classifier. The red and yellow lights however could be processed by a single classifier. Nevertheless we decided to use one distinct classifier for each color. This decision was made because the classifiers are trained with the ROI images that are created as described in the previous section. Depending on the parameter settings, the ROIs of the red and the yellow lights might have a different structure; e. g., the head of yellow arrows pointing to the top is often cut off (see Fig. 5) while this problem did not occur with red lights. By using separate classifiers for each color we hope that the classifiers have a better chance to adapt to the characteristics of the different colors.

In order to solve the problem of classifying large, real-valued vectors, many efficient algorithms in the field of machine learning have been established. Since we have to distinguish between more than two different arrow directions, only multi-class classifiers are suitable. We tested classification with three different approaches: *Linear Discriminant Analysis* (LDA) [12], *K-Nearest Neighbor Classifier* (kNN) [13], and *Support Vector Machines* (SVM) [14], which are briefly presented in the following.

*a) Linear Discriminant Analysis:* The LDA attempts to find a linear separation between the classes while minimizing the variance of the samples within each and maximizing the variance in-between different classes. One of the main advantages of the LDA is its universal applicability since it does not depend on parameters that have to be tuned beforehand. Additionally, classification of a single sample can be done very fast and efficient which makes it practicable in environments that require real-time capability.

*b) K-Nearest Neighbor Classifier:* During the training phase, the feature representation of all training examples and their labels are simply stored – for a shorter processing time in an efficient tree structure. The kNN classifier then assigns a label to a new example based on the labels of the $k$ nearest neighbors of that example in the feature space. The performance of this classifier strongly depends on the parameter $k$ which determines the number of neighbors that should be taken into account. Classification of a single example is more time consuming compared to the other methods since it is necessary to find relevant adjacent examples in the feature space. However, real-time capability can usually be ensured.

*c) Support Vector Machines:* A few training examples are selected as the eponymous *Support Vectors* that define the class separation. They are chosen such that the margin to the separating hyperplane is maximized. SVMs have the possibility to allow wrong classifications during the training phase to avoid overfitting. The parameter $C$ determines how strong such a misclassification is penalized. For small $C$ it is more important to maximize the margin than the adaption to the examples of the training dataset. We use non-linear SVMs with a Gaussian RBF kernel. They introduce an additional parameter $\gamma$, which determines the flexibility of the separating hyperplane. A small $\gamma$ leads to a greater flexibility but also increases the risk of overfitting during training. Originally, SVMs are designed for binary classifications; by using a certain combination of binary SVMs and sophisticated training methods however, they can also be applied to multi-class classification. In the available Shark implementation, classification with an SVM is only marginally slower than with an LDA.

An evaluation of the quality and efficiency of these three methods for the problem at hand is given in Sect. V.

The final classification result is produced as a combination of the labels for the phase and the pictogram. In case of contradictions in either classification step we decided in our implementation to return a label belonging to an additional error class. Depending on the desired field of use it might also be suitable to return the most plausible label.

## IV. Experimental Setup

For our experiments we acquired a dataset consisting of 1802 single cut out image sections of german traffic lights containing 946 green traffic, lights 230 yellow traffic lights, 509 red traffic lights, and 117 red and yellow traffic lights. The pictograms are distributed such that 1180 display no arrow, display an arrow to the left, display an arrow to the top, and 61 display an arrow to the right. Furthermore we added 125 images that do not contain a traffic light but were nevertheless selected by a Viola-Jones detector [15]. The distribution of available images is detailed in Tab. III. It can be seen, that some classes are underrepresented; e. g., we had only six images of a yellow traffic light with an arrow to the left. Therefore, some results may not be representative and would benefit from additional data.

TABLE III. The total amount of data available for each class. This data is split into training (70 %) and testing (30 %) datasets.

| | **None** | **Top** | **Left** | **Right** |
|---|---|---|---|---|
| **Green** | 668 | 167 | 81 | 30 |
| **Yellow** | 161 | 48 | 6 | 15 |
| **Red** | 254 | 49 | 190 | 16 |
| **Yellow-Red** | 97 | 8 | 12 | 0 |
| **Empty** | 125 | | | |

We also considered using a publicly available benchmark – the only relevant being the LARA traffic lights recognition benchmark[1]. However, the lights in this benchmark do not display pictograms and the overall variety is rather low since a majority of the images were taken waiting in front of red traffic lights. Therefore it was unsuitable for our demands and we decided against using it.

One possible approach to this problem might follow the idea of Salmen et. al [16], who used Google Street View to gather large amounts of training and testing data for traffic sign recognition applications in very little time. They accessed the image via the Street View API and applied a Viola-Jones detector trained on traffic signs to extract hypotheses. For our needs, the same approach could be taken with a detector for traffic lights.

For further experiments we randomly divided all images into a training and a test set with the former containing 1355 (roughly 70 %) and the latter containing 572 (roughly 30 %) samples. We only used the images contained in the training dataset for developing the method for the phase classification described in Section III-A as well as training the classifiers described in Section III-B.

The overall training and testing procedure was divided into two parts according to the two parts of our method. No explicit training is necessary for the phase classification since all relevant parameters were already adjusted by hand during the development according to the training dataset.

TABLE IV. Threshold definitions in the HSV space that were used for each individual traffic light color. These might have to be adapted depending on the attributes of the camera.

| | ***Green*** | | ***Yellow*** | | ***Red*** | |
|---|---|---|---|---|---|---|
| | **Min** | **Max** | **Min** | **Max** | **Min** | **Max** |
| **Hue** | 85 | 149 | 10 | 45 | 250 | 12 |
| **Value** | 75 | 255 | 120 | 255 | 120 | 255 |
| **Sat.** | 70 | 255 | 65 | 255 | 120 | 255 |

[1]http://www.lara.prd.fr/benchmarks/trafficlightsrecognition

TABLE V. CONFUSION MATRIX FOR THE COLOR CLASSIFICATION. THE ROWS INDICATE THE GROUND-TRUTH LABEL WHILE THE COLUMNS INDICATE THE OUTPUT OF OUR CLASSIFIER. FOR BETTER READABILITY THE LABELS IN THE HEADER ARE ABBREVIATED.

| | G | Y | R | YR | E |
|---|---|---|---|---|---|
| **Green** | 0.98 | 0 | 0 | 0 | 0.02 |
| **Yellow** | 0 | 1 | 0 | 0 | 0 |
| **Red** | 0 | 0 | 0.99 | 0 | 0.01 |
| **Yellow-Red** | 0 | 0 | 0.18 | 0.79 | 0.03 |
| **Empty** | 0.03 | 0.13 | 0 | 0 | 0.84 |

The color thresholds we chose for our data are displayed in Tab. IV.

It is noteworthy that the hue is defined circularly which explains the minimum of 250 and the maximum of 12 for red lights. Additionally the definitions for red and yellow lights partly overlap for hues of 10–12. This is due to our training data in which – depending on the environmental conditions – rather dark yellow lights can fall into the regime of a bright red light as displayed in Fig. 6.

In an application scenario, the pictogram classifiers will receive the ROIs produced by this first step as their input. We also want this relation to exist during training; therefore we run the phase classification on the original training dataset and build a new training datatset for the pictogram classification from only those ROIs that were segmented correctly. Thus, the training dataset for the pictogram classification consists of images of single colored lights subdivided by their arrow direction. Since we use a different classifier for each color, we also divide the new training set by color, so that each classifier is trained only on examples of a single color.

The kNN classifier and the SVM provide tunable parameters that affect the overall classification quality. All of these parameters were adjusted by the means of a 3-fold cross-validation. The number of neighbors $k$ is an integer; therefore we simply compared the performance of the kNN classifier on the validation set for all $k \in [1, 10]$. Figure 7 gives an overview of this performance for different $k$. The parameters $C$ and $\gamma$ of the SVMs were adjusted using a grid search with the Jaakkola Heuristic [17]. For the LDA no parameter tuning is necessary.

For our implementation we used the *Shark Machine Learning Library*[2] [18], which provides a wide range of state-of-the-art algorithms for classification as well as other

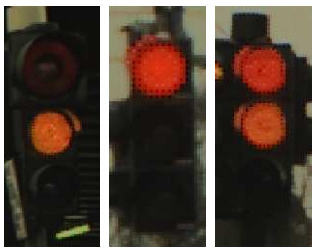

Fig. 6. The color of red and yellow lights might be nearly identical which explains the overlap in the thresholds for the hue in Tab. IV.

[2]http://image.diku.dk/shark

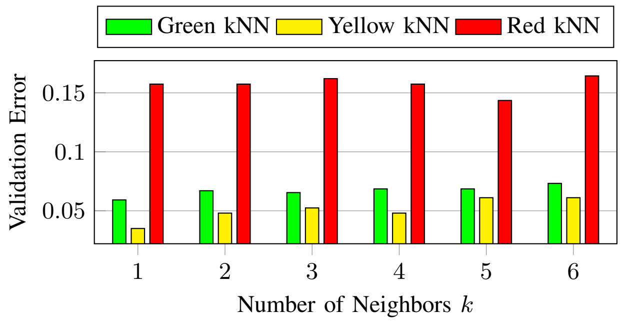

Fig. 7. The error of each kNN classifiers on the validation set for different values of $k \in [1, 6]$. Minimal values are found at $k = 1$ for green and yellow kNNs and at $k = 5$ for the red kNN. For better readability we omitted values between 7 and 10 – here the performance on the validation set is worse.

machine learning problems like supervised or unsupervised learning, optimization, or regression. All experiments were conducted on an Intel Core i7-2630QM CPU with 8 GB ram.

## V. RESULTS

Following the two-part training routine, the evaluation will also be divided into two parts. At first we present the quality of the phase classification in Subsec. V-A followed by a comparison of the different classifiers in Subsec. V-B. Conclusively we give an overview of the combined performance.

### A. *Phase Classification*

The phase classification achieves an overall classification quality of 96.7 %. Figure 8 shows precision and recall for all classes. In Tab. V the confusion matrix for the phase classification is displayed. It can be seen that there exist mainly three causes of error. Firstly, it can happen, that an existing light is not found – e. g., 5 green traffic lights have been labeled as “Empty”. Secondly, in some cases the yellow light of a traffic light that is simultaneously red and yellow could not be found. Furthermore, our algorithm is not able to identify all erroneous detections since it labels 6 images containing no traffic light as “green” or “yellow”.

For such errors, traffic light recognition pipelines usually contain a tracking module, which is able to smooth small inconsistencies in the detection and ensures a stable detection over several frames before it is considered for further processing.

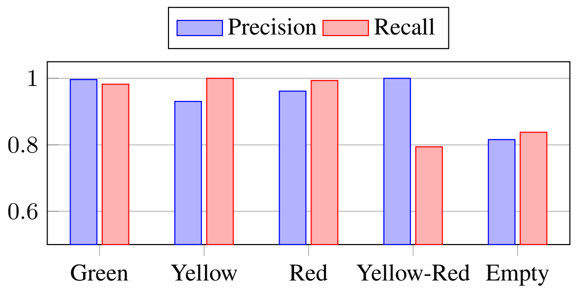

Fig. 8. Precision and recall for the phase classification step.

TABLE VI. CONFUSION MATRIX FOR THE PICTOGRAM CLASSIFICATION WITH THE SVM.

| | None | Top | Left | Right |
|---|---|---|---|---|
| **None** | 0.98 | 0.01 | 0.01 | 0 |
| **Top** | 0.14 | 0.85 | 0.01 | 0 |
| **Left** | 0.16 | 0.03 | 0.81 | 0 |
| **Right** | 0.25 | 0 | 0 | 0.75 |

## B. *Pictogram Classification*

For the pictogram classification step we evaluated 9 classifiers in total. An overview of their classification quality – on each color individually as well as the overall performance – is given in Fig. 9. It can be seen that the LDA systematically performs worst, although still showing 89.7 % correct classifications. The kNN classifier exhibits slightly better results with 90.3 % correct classifications. SVMs perform best with an overall quality of 92.8 %. These results suggest to consider only SVMs for our further studies.

In Tab. VI, the confusion matrix for the arrow classification is displayed. The matrix shows that most errors are made by classifying lights with an arrow as a plain light without a pictogram. We investigated the images which were classified erroneously by the SVM and found that predominantly small images that were captured further away from the traffic light were affected – e. g., those shown in Fig. 11.

For more accurate results we divided the training data for the arrow classifiers into three parts based on the diameter of the active light. This can also be understood as dividing them with respect to the distance between camera and traffic light. Lights with a diameter of 8 pixels or fewer were denoted as “Far”, lights with a diameter of more than 14 pixels were denoted as “Close”, and all lights between were labeled “Medium”. The classification errors with respect to this diameter are shown in Fig. 12.

This demonstrates that classification of nearby traffic lights is more robust and less error prone than the classification of distant traffic lights. While the phase of distant traffic lights can still be identified with rather high accuracy, the classification of the pictogram is more difficult. For the final application scenario of traffic lights recognition this implies, that the initial guess after first observing a traffic light from a distance might not be correct. However, when the traffic light is approached, the classification gets more stable – especially when applying one of the aforementioned trackers.

Another important observation that can be deduced from Fig. 7 and 9 is that pictograms on red lights are considerably harder to classify than their counterparts on yellow lights. A possible explanation is that red lights are darker than yellow ones in general, which leads to a less prominent gradient between light and arrow. Figure 13 shows some examples of those red traffic lights with a pictogram that is hard to recognize – even for the human eye. It is subject to further research to determine if this difficulty is an inherent property of the traffic lights recognition problem or if it can be countered by the use of different cameras.

## C. *Combined Performance*

When evaluating the phase and the pictogram classification steps at the same time, the overall quality is worse than the individual amounts of correct classifications (96.7 % and 92.7 %). Here we achieve an error of 10.1 % with 89.9 %

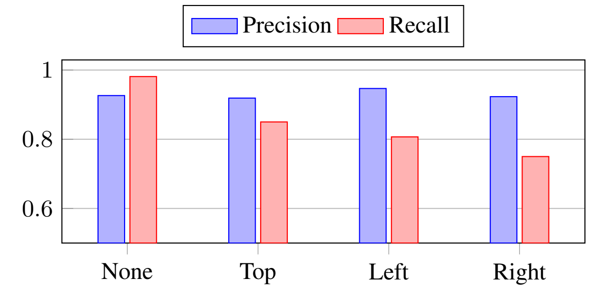

Fig. 10. Precision and recall for all classes in the pictogram classification step.

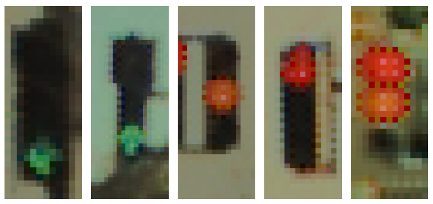

Fig. 11. Examples of traffic lights that were captured from further away. Their lights have only a very small diameter and it is therefore rather difficult to recognize a pictogram – even for the human eye.

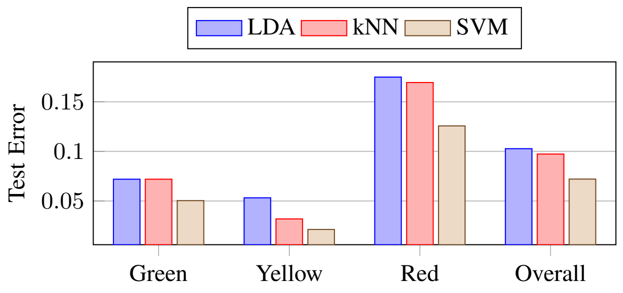

Fig. 9. The amount of error on the test dataset for all classifiers. Since individual classifiers are used for each color, one bar represents a single of these classifiers. The rightmost group shows the combined performance of all classifiers of a single type (LDA, kNN, or SVM).

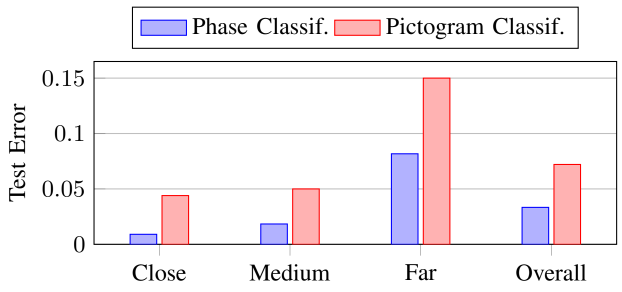

Fig. 12. The error on the test dataset with respect to the distance between camera and traffic light. This distance has direct implications on the size of the observed lights. “Close” represents lights with a diameter of more than 14 pixels. “Far” represents lights with a diameters of 8 pixels or fewer. Lights with a diameter between 8 and 14 fall into the category “Medium”.

of correctly classified images.

When integrating an algorithm in a larger pipeline, the runtime is always an important aspect. Even algorithms that are real-time capable when executed alone might not be fast enough when sharing resources with other applications. Depending on the used machine-learning classifier, the following amounts of time are needed by our implementation to classify a single image from the testing dataset:

- LDA: 0.24 ms
- kNN: 4.69 ms
- SVM: 0.51 ms,

where the speed of the kNN classifier is tied to the number of neighbors that need to be considered.

As stated above, the qualitative results suggest using the SVM for pictogram classification. Even though it requires the double amount of time compared to the LDA, with a runtime of less then one millisecond it should still be fast enough even for highest demands.

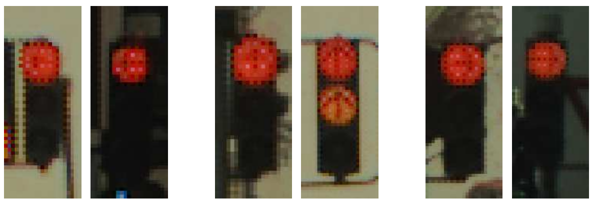
Fig. 13. Examples of pictograms on red lights. It can be seen, that even on lights of medium size the arrow is very blurry and hard to recognize. The two left images show arrows to the left, the two middle images to the top, and the two images on the right to the right.

## VI. Conclusions

In this paper we presented a method for efficiently classifying the color phase and the arrow pictogram of a traffic light for video-based driver assistance. A tailored segmentation pipeline, state-of-the-art image features, and supervised learning methods were applied successfully. With a correct classification rate of 96.7 % and 92.7 %, both sub-tasks exhibit a very good individual performance, as well as a considerable quality of 89.9 % when combined. All results presented here were achieved on individual images, while most errors occur on small images that were captured from a far distance to the traffic light.

Compared to the work of [8], our results seem worse. However, Cai et. al require a traffic light to occupy at least 300, and an individual light at least 50 pixels before they attempt classification. Our approach in contrast is able to process even smaller lights. Ignoring samples from the "Far" category, our method surpasses their classification rate of 91 % while using significantly less processing time.

By means of the developed method one is able to easily complete an existing real-time capable traffic light recognition module and, thus, offer detailed state information to the driver or autonomous or guided driving systems.

Nevertheless, some further improvements are targeted. A first step is the development of better means to reject wrong detections during the phase classification step. It also needs to be investigated if the error of the pictogram classifiers can be reduced by employing a larger dataset or if further measures are necessary.

Another promising approach is the introduction of temporal integration of the classified state and, therewith, the track-based evaluation on video sequences. Thus, the significance of single misclassifications will be reduced, which might boost the overall performance of our system.

## References

[1] M. Omachi and S. Omachi, "Traffic light detection with color and edge information," in *Proceedings of the IEEE International Conference on Computer Science and Information Technology*, 2009, pp. 284–287.

[2] C. Wang, T. Jin, M. Yang, and B. Wang, "Robust and real-time traffic lights recognition in complex urban environments," *International Journal of Computational Intelligence Systems*, vol. 4, no. 6, pp. 1383–1390, 2011.

[3] J. Gong, Y. Jiang, G. Xiong, C. Guan, G. Tao, and H. Chen, "The recognition and tracking of traffic lights based on color segmentation and camshift for intelligent vehicles," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2010, pp. 431–435.

[4] R. de Charette and F. Nashashibi, "Real time visual traffic lights recognition based on spot light detection and adaptive traffic lights templates," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2009, pp. 358–363.

[5] V. John, K. Yoneda, B. Qi, Z. Liu, and S. Mita, "Traffic light recognition in varying illumination using deep learning and saliency map," in *Proceedings of the IEEE International Conference on Intelligent Transportation Systems*, 2014, pp. 2286–2291.

[6] F. Lindner, U. Kressel, and S. Kaelberer, "Robust recognition of traffic signals," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2004, pp. 49–53.

[7] N. Fairfield and C. Urmson, "Traffic light mapping and detection," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2011, pp. 5421–5426.

[8] Z. Cai, M. Gu, and Y. Li, "Real-time arrow traffic light recognition system for intelligent vehicle," in *Proceedings of the IEEE International Conference on Image Processing, Computer Vision, & Pattern Recognition*, 2012, pp. 848–854.

[9] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2005, pp. 886–893.

[10] Q. Zhu, M.-C. Yeh, K.-T. Cheng, and S. Avidan, "Fast human detection using a cascade of histograms of oriented gradients," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2006, pp. 1491–1498.

[11] M. Dahmane and J. Meunier, "Emotion recognition using dynamic grid-based HoG features," in *Proceedings of the IEEE International Conference on Automatic Face Gesture Recognition and Workshops*, 2011, pp. 884–888.

[12] T. Li, S. Zhu, and M. Ogihara, "Using discriminant analysis for multi-class classification: an experimental investigation," *Knowledge and Information Systems*, vol. 10, no. 4, pp. 453–472, 2006.

[13] P. Cunningham and S. J. Delany, "K-nearest neighbour classifiers," *Multiple Classifier Systems*, pp. 1–17, 2007.

[14] K.-B. Duan and S. S. Keerthi, "Which is the best multiclass SVM method? an empirical study," in *Multiple Classifier Systems*. Springer, 2005, pp. 278–285.

[15] P. Viola and M. Jones, "Robust real-time object detection," *International Journal of Computer Vision*, vol. 57, no. 2, pp. 137–154, 2001.

[16] J. Salmen, S. Houben, and M. Schlipsing, "Google street view images support the development of vision-based driver assistance systems," in *Proceedings of the IEEE Intelligent Vehicles Symposium*, 2012, pp. 891–895.

[17] T. Jaakkola, M. Diekhans, and D. Haussler, "Using the Fisher kernel method to detect remote protein homologies." in *Proceedings of the International Conference on Intelligent Systems for Molecular Biology*, 1999, pp. 149–158.

[18] C. Igel, T. Glasmachers, and V. Heidrich-Meisner, "Shark," *Journal of Machine Learning Research*, vol. 9, pp. 993–996, 2008.