Freie Universität Berlin

Bachelor Thesis at the Department of Mathematics and Computer Science
Dahlem Center for Machine Learning and Robotics

Development of an image based traffic light detection for model cars

Jörn Andreesen
student ID: 5077181
joern.andreesen@fu-berlin.de

advisor: Prof. Dr. Daniel Göhring

first examiner: Prof. Dr. Daniel Göhring
second examiner: Prof. Dr. Dr.(h.c.) habil. Raúl Rojas

March 9, 2020

Selbstständigkeitserklärung

Ich erkläre gegenüber der Freien Universität Berlin, dass ich die vorliegende Bachelorarbeit selbstständig und ohne Benutzung anderer als der angegebenen Quellen und Hilfsmittel angefertigt habe.

Die vorliegende Arbeit ist frei von Plagiaten. Alle Ausführungen, die wörtlich oder inhaltlich aus anderen Schriften entnommen sind, habe ich als solche kenntlich gemacht.

Diese Arbeit wurde in gleicher oder ähnlicher Form noch bei keiner anderen Universität als Prüfungsleistung eingereicht.

Berlin, den _____ _____

Jörn Andreesen

Abstract

For autonomous vehicles it is important to be able to detect traffic lights. This work suggests a method for the Freie Universität Berlin's AutoMiny model cars to detect traffic lights. This method uses image thresholding to find candidate regions and uses the normalized rgb colors of the candidate region to check if a traffic light was found. The evaluation shows that this method is able to reliably detect traffic lights in short distance but struggles with traffic lights further away. Also a detection in real-time is not possible with the hardware of the car.

Contents

List of Figures

List of Tables

1 Introduction

In recent years there has been a lot of advances in the development of autonomous cars and driver assistance systems. Many researchers wrote papers and tech companies and car manufacturers started the development of their own autonomous cars. There is already a wide variety of assistance systems in modern cars, but since car2car and car2infrastructure communication is not available everywhere yet, most systems have to rely on sensors like camera, radar, etc. to learn about their surroundings and analyze the situation.

One important assistance system is the detection of traffic lights using camera images. There have papers suggesting the use of neural networks to detect traffic lights. Weber et al.[1] use a Convolutional Neural Networks based on AlexNet that won the Large Scale Visual Recognition Challenge 2012 to detect traffic lights and were able to achieve more than 90% precision and recall. Lee et al.[2] use a Deconvolutional Neural Network that extracts features with an encoder, decodes them into a feature map and then uses a detector network to detect the traffic lights. Behrendt et al.[3] uses the "You only look once" or YOLO[4] architecture to detect even very small traffic lights in images with high precision but also use a second network to classify the traffic lights and required powerful hardware to detect while driving.

While these neural networks can achieve good recall and precision in detecting traffic lights, they require a lot of time and labeled data for training. Furthermore neural networks need more processing power and are therefore slower than other approaches without neural networks. Charette et al.[5][6] search for spot lights in the image and use adaptive template matching to find the shapes of traffic lights around these spot lights. Alam et al.[7] also match templates for different states off traffic lights in Regions of Interest to detect the state and position of a traffic light in an image. Omachi et al.[8][9] detect edges in a normalized image and use Hough transformation[10] to find circles that represent traffic lights. Sooksatra et al.[11] detect red traffic lights using fast radial symmetry transform.

This work proposes a system to detect traffic lights with a camera on the model cars of the Freie Universität Berlin and evaluates the proposed system in different situations.

2 Used Technologies

2.1 The model car

The car used is the model car of the AutoMiny project of the Freie Universität Berlin. It uses an Intel Real Sense D435 infrared stereo camera mounted 20cm above ground on the car[12]. The AutoMiny software is composed of modules for the Robot Operating System (ROS) which is installed on the cars.

2.2 ROS

The Robot Operation System[13] is a framework for robot software. The software is written as a collection of modules called ROS nodes. These nodes control the various parts of the robot and implement the control logic. For example some nodes control the driving motor or the steering motor, other nodes read input from sensors like the camera and the Lidar and yet other nodes implement the control logic to react to the inputs from the sensors by creating commands for steering and acceleration. To communicate the nodes use topics. Every topic has a name that usually describes what kind of information can be found there. For example in the AutoMiny software the topic name for colored camera images is `/sensors/camera/color/image_rect_color`. On these topics messages are exchanged. Every node can subscribe to topics they are interested in and will then receive all messages exchanged on this topic. A node can also create messages and publish them to a topic to make their information available for other nodes.

A common example would be a node reading the output of a sensor like the camera and creating a message containing the sensor information and publishing this message to a topic for sensors input. Other nodes subscribed to this topic will then receive the message with the senor information. With this sensor information a node containing control logic could make a decision to steer the car. To do this it would create another message containing instructions for the node controlling the steering motor and publish it to a steering topic. The steering motor node is subscribed to this topic and receives the message containing instructions and executes them.

ROS also offers supportive tools for recording and visualizing the messages. One of the tools is rosbag which allows to record message exchanged on selected topics. With these recordings it is easy to test nodes even with-

out the robot or in this case model car itself. By replaying the recordings the node would receive the same messages as it would on the car and the behavior of the node can be observed without running the node on the car.

2.3 Robotics Lab Environment

Unlike real cars the model car does not drive outside on the road. It is only used inside the Freie Universität's Robotics Lab. This controlled environment creates a few favorable factors. There is no weather inside, the light conditions are always the same and there are almost no other light sources that could be confused with a traffic light.

Also there is currently only one traffic light in the Lab. The traffic light is 33 cm high and 7cm wide. The red, yellow and green lights are positioned at a height of 30 cm, 27 cm and 24 cm respectively. The traffic light has four states and changes from red to red and yellow to green to yellow and then back to red and repeats.

2.4 Precision, Recall and F-Measure

When testing the method on an images there are five possible outcomes:

1. The image contains a traffic light and the traffic light is detected correctly (true positive)
2. The image contains no traffic light but a traffic light is detected (false positive)
3. The image contains no traffic light and no traffic light is detected (true negative)
4. The image contains a traffic light but the traffic light is not detected (false negative)
5. The image contains a traffic light and is detected but the detected traffic light has the wrong state (wrong classification)

The quality of a detection is evaluated by its recall, precision and the F-measure. Recall is the proportion of traffic lights that are correctly identified and is defined as[14]:

$$\text{recall} = \frac{\text{true positives}}{\text{true positives+false negatives+wrong classification}} \tag{1}$$

Precision is the proportion of detected traffic lights that are actual traffic lights and is defined as[14]:

$$\text{precision} = \frac{\text{true positives}}{\text{true positives+false positives+wrong classification}} \tag{2}$$

The F-measure or F_1-measure is the harmonic mean of precision and recall and is defined as[15]:

$$\text{F} = \frac{2 * \text{precision} * \text{recall}}{\text{precision+recall}} \tag{3}$$

2.5 Confusion matrices

A confusion matrix visualizes the predictions of a classifier. The columns represent the actual classes while the rows represent the predicted classes. The values in the matrix are a counter for how many of each actual class were predicted as a different or their correct class.[14]

In the example in table 1 there are three classes. One class for red traffic lights, one for green traffic lights and one class "empty" for no traffic lights. The visualization as confusion matrix makes it easy to see the number of true positives and true negatives on the diagonal axis, false negatives in the first two columns of the last row and false positives in the first two rows of the last column.

		Truth		
detectetd as		red	green	empty
	red	52	5	2
	green	13	44	0
	empty	0	4	20

Table 1: Example confusion matrix

2.6 Hill Climbing

Hill climbing is an search algorithm that can also be used to find good parameters for classifier. Starting with an inital parameter the quality of the classifier is measured for example with the F-measure. Then the parameter is changed by previously defined step value and the classifier is evaluated

again. If the quality of the classifier improved the new parameter is accepted and the process is repeated. If there was no improvement the parameter is changed back to the previous value, the step value is changed and the process is repeated with a different step value. The way the step value is changed also has to be decided on beforehand. For example the step value could be halved or multiplied by -1. The algorithm ends when there is no step left to make that would improve the results.

This algorithm is easy to implement and can improve the quality of the results, but it does not find optimal values for the parameters. If the algorithm finds a local maximum it will stop because there is no improvement to make in step range.[16]

2.7 Image Thresholding

Thresholding is a simple method to find bright spots in a grayscale image. Thresholding compares the intensity of every pixel to a threshold value. In a grayscale image the value of a pixel is its intensity. If the pixel value is higher than the threshold the pixel is set to the maximum value. For an 8bit grayscale image the maximum value would be 255. Pixels with a value lower than the threshold are set to zero. If $f(x, y)$ is the intensity of the pixel at (x, y) then the intensity of the pixel in the thresholded image would be $g(x, y)$, defined as:[17]

$$g(x,y) = \begin{cases} 255 & \text{if } f(x,y) > T \\ 0 & \text{else} \end{cases} \tag{4}$$

The resulting image is a binary image with only either completly black or white pixels. An example can be seen in figure 2.

2.8 Normalized RGB Color Space

Images taken by the camera of the model car are in the RGB color space. RGB values are affected by the light conditions. Even though the light conditions in the Robotics Lab usually do not change, by converting to image to normalized RGB the colors in the image become independent of changes in the light conditions.[8] To convert the color space to normalized RGB each of the values red, green and blue is divided by the sum of the three values.[18]

$$r = \frac{R}{R+G+B} \tag{5}$$

$$g = \frac{G}{R+G+B} \tag{6}$$

$$b = \frac{B}{R+G+B} \tag{7}$$

Examples can be seen in figure 3.

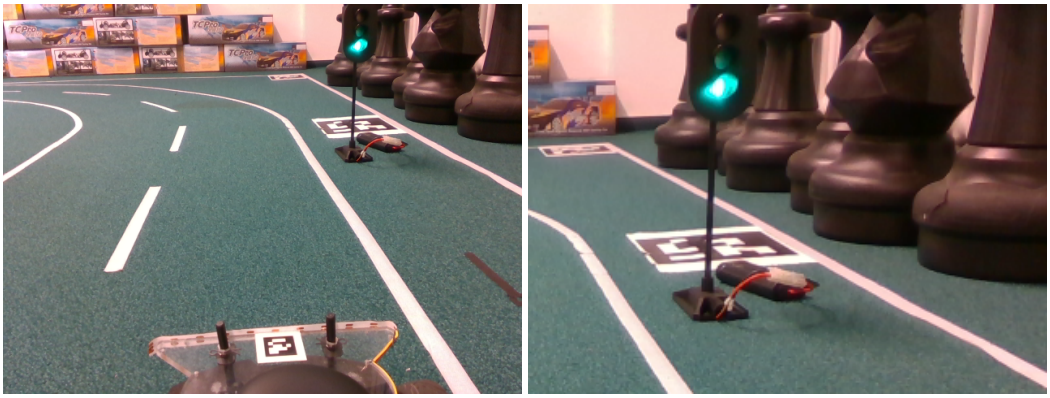


Figure 1: Cropping the image

3 Implementation

The detection algorithm is implemented as a ROS node that gets images from the `/sensors/camera/color/image_rect_color` topic where color images taken by the camera are published. Then the following method is used to detect traffic lights in image if present and inform other nodes of their presence. The method uses 5 steps to find traffic lights:

1. cropping the image
2. finding bright spots
3. filtering the bright spots by shape
4. determining the color
5. publishing the results

3.1 Cropping the image

Since the traffic lights in this environment are always expected to be on the right of the lane, the image will be cropped to the upper right quarter as seen in figure 1. This reduces the area in which to look for traffic lights and therefore reduces the time to process an image. Furthermore possibly distracting light sources in other parts of the image are eliminated.

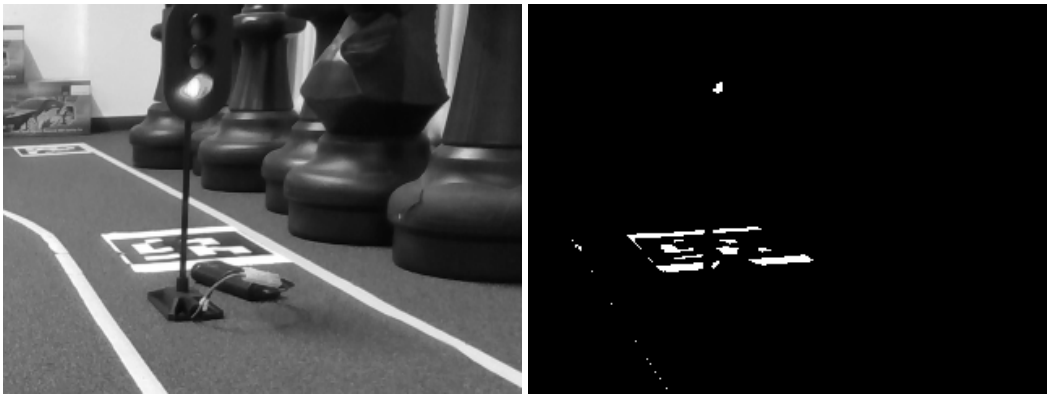


Figure 2: Grayscale and binary image

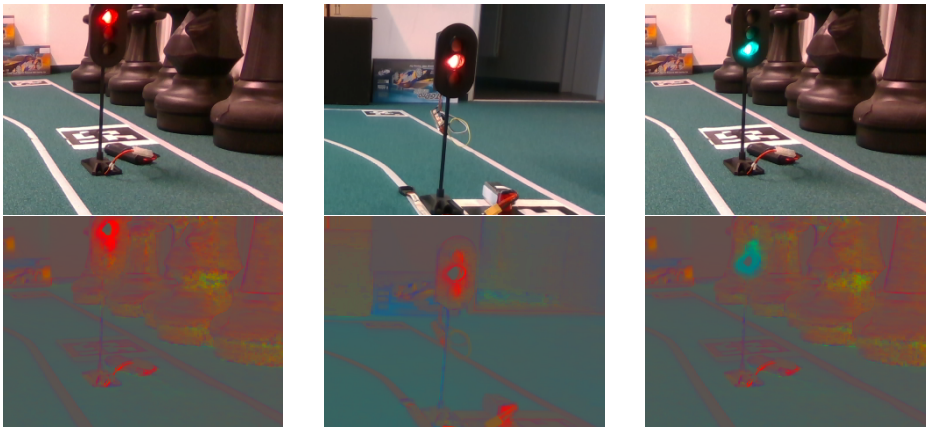
3.2 Finding bright spots

To find bright spots the image will be converted into a greyscale image. This makes it easier to see the intensity of a pixel, because in grayscale it is only encoded in one intensity value per pixel rather than a red, green and blue color value. Then a threshold is applied and all pixels below a specified intensity will have their intensity set to 0 while the remaining brighter pixels will have their intensity set to the maximum value. This results in a black and white image as seen in figure 2 where the bright spots are clearly visible as white pixels before a black background.

3.3 Filtering the bright spots by shape

This step requires to group white pixels that are connected with each other. All white pixels are grouped together to a shape. If two white pixels are adjacent to each other both pixels are grouped into the same shape. If a white pixel has no adjacent other white pixel, it becomes a shape on its own, containing only this one pixel. Two pixels at (x_1, y_1) and (x_2, y_2) are adjacent if $|x_1 - x_2| \leq 1$ and $|y_1 - y_2| \leq 1$.

Then all shapes are filtered. Shapes containing less than 10 pixels are filtered out because they are too small. The height and width of the shapes is measured and shapes are filtered out if they don't satisfy the criteria $\max(\text{width}, \text{height}) \leq 2 * \min(\text{width}, \text{height})$. The idea for this criteria is taken from Charette et al.[5].



(a) red (b) yellow (c) green

Figure 3: Traffic lights converted to normalized rgb color space

3.4 Determining the color

After the last step there are still shapes that have not been filtered out yet but are not part of a traffic light. Therefore it is necessary to determine the color of the possible light source. This filters out the remaining shapes and the color also defines the state of the traffic light. The greyscale image has been used to find circular bright spots in the image, but to determine the color of the light the original colored image is needed.

Because the led of the traffic light is too bright, the color cannot be determined at the center of the light source. In the normalized rgb color space the center of the light is due to its brightness normalized to a gray color as seen in figure 3 .The color is instead taken at 8 points on a circle of a specified radius around the center. The mean of the normalized red, green and blue values of the points is taken and compared to predefined values for a red, green or yellow traffic light as seen in listing 1. If the normalized rgb values are within these bounds the spot is classified as a traffic light of the the respective color.

This can be seen in figure 4, where the yellow circle shows shapes that were not filtered and where the color was tested. The blue point shows where the traffic light was found.

```
1 def is_red(b,g,r):
2     return r>= 0.75 and b < 0.15 and g < 0.15
3
4 def is_green(b,g,r):
5     return r < 0.1 and b >= 0.4 and g >= 0.4
6
7 def is_yellow(b,g,r):
8     return r>=0.5 and b < 0.25 and g < 0.25
```

Listing 1: Color test functions

3.5 Publishing the results

If and only if a traffic light is detected a message of the `traffic_light_info` message type is send to the new topic `/traffic_lights/info`. The new message type `traffic_light_info` contains information about where in the camera image the traffic light was found and the state of the traffic light. Also an image similar to figure 4 showing the detected traffic light is published to `/traffic_lights/image`.

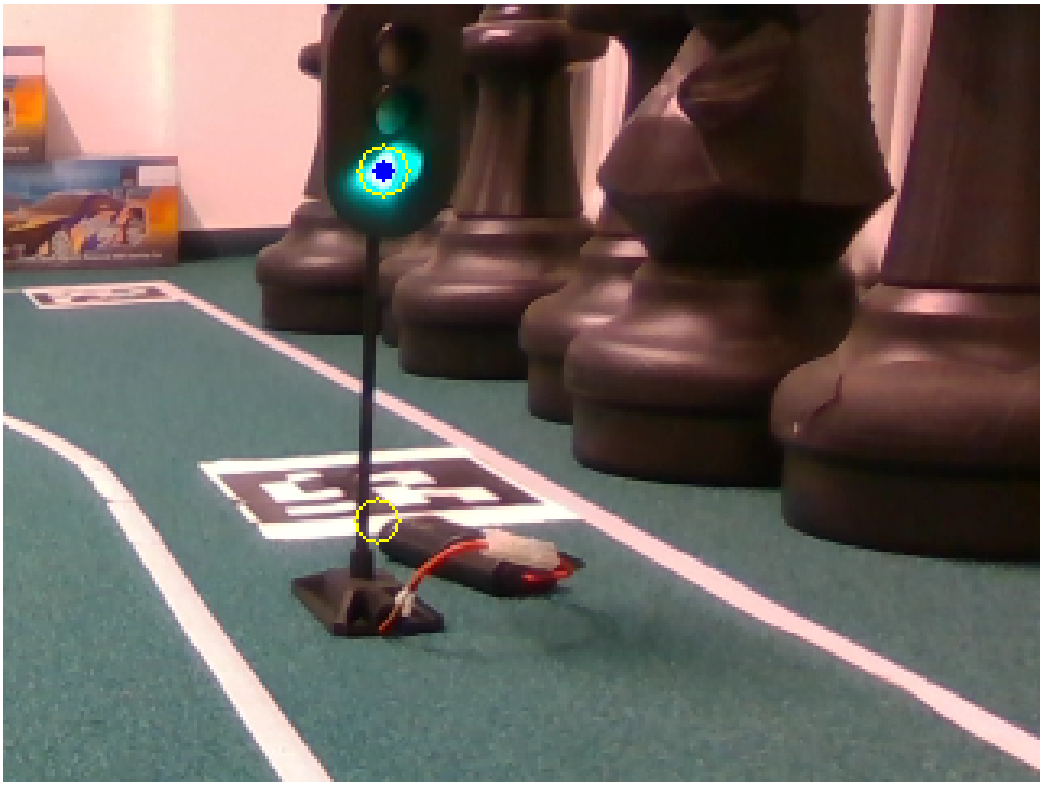


Figure 4: Points where color was tested marked in yellow and the detected traffic light marked in blue

4 Evaluation

This method was tested with labeled images. The images were taken from rosbags recorded on the model cars. The rosbags contain the record of the `/sensors/camera/color/image_rect_color` topic. For each image message in each rosbag the image was extracted from the message, assigned a label by hand and then saved as image file with the label in the filename. This resulted in 4 sets of labeled images with a total of 1904 images. One set with traffic lights in 1 meter distance, one with traffic lights in 2 meter distance and one with traffic lights in 3 meter distance. The last one contains images taken while driving at various speeds. As can be seen in Table 2 867 of these images show red traffic lights, 653 show a green traffic, 90 show yellow traffic lights and 110 show traffic lights with red and yellow lights on. The driving

Label	1 meter	2 meter	3 meter	driving	Total
red	431	250	65	121	867
green	133	272	141	107	653
yellow	27	26	24	13	90
red-yellow	59	33	6	12	110
empty	0	0	0	184	184
Sum	**650**	**581**	**236**	**437**	**1904**

Table 2: Composition of the test image sets

set also contains 184 images where there is no traffic light or the traffic light is concealed. These images are labeled as empty. Naturally the most images are those of red and green traffic lights because most of the time the traffic light is either in the red or green state.

At first the quality of detection at various distances is evaluated to see if it is possible to detect even distant traffic lights. Then detection while driving is tested and at last the time to process an image is measured

4.1 Traffic lights in the distance

There are multiple parameters that can be adjusted to improve the detection. Those parameters are the threshold value in the range $[0, 255]$ for the thresholding in section 3.2, the distance to the center of the points where the color is checked in section 3.4 and the minimum number of pixels a shape must contain to not be filtered out as too small. As initial values for these parameters a threshold of 250, a distance of 5.0 and a minimum of 10 pixels required for a shape were chosen by manually testing different values on single images from the set. A test run with these parameters on the 1m image set results in the confusion matrix seen in Table 3. The F-measure of this run is 94.8%.

When trying to detect traffic lights in 2-3 meter distance with the same parameters as in the first run only a few traffic lights are detected correctly as seen in table 4. The F-measure is only 2.3%. To make it easier to detect traffic lights in the distance the parameter for minimum required pixels for a shape was manually reduced to 5 pixels. To determine the best values for the remaining parameter a simple hill climbing algorithm was used to find values for the parameters that result in higher f1 scores than the scores of

		Truth				
		red	green	yellow	red-yellow	empty
detectetd as	red	428	0	0	0	0
	green	0	77	0	0	0
	yellow	0	0	26	1	0
	red-yellow	2	0	0	58	0
	empty	1	56	1	0	0

Table 3: Detection of traffic lights in the 1 meter image set with a threshold = 250, distance = 5.0 and minimum pixels = 10 as parameters

		Truth				
		red	green	yellow	red-yellow	empty
detectetd as	red	0	0	0	0	0
	green	0	0	0	0	0
	yellow	0	0	10	26	0
	red-yellow	0	0	0	0	0
	empty	322	413	40	6	0

Table 4: Detection of traffic lights in the 2 and 3 meter image set with a threshold = 250, distance = 5.0 and minimum pixels = 10 as parameters

		Truth				
		red	green	yellow	red-yellow	empty
detectetd as	red	122	0	0	5	0
	green	0	413	0	17	0
	yellow	195	0	50	10	0
	red-yellow	5	0	0	0	0
	empty	0	0	0	0	0

Table 5: Detection of traffic lights in the 2 and 3 meter image set with a threshold = 216, distance = 3.0 and minimum pixels = 5 as parameters

		Truth				
		red	green	yellow	red-yellow	empty
detectetd as	red	23	0	0	1	0
	green	0	51	0	0	0
	yellow	72	0	5	3	0
	red-yellow	0	0	0	3	0
	empty	26	56	8	5	184

Table 6: Detection of traffic lights in the driving image set with a threshold = 216, distance = 3.0 and minimum pixels = 5 as parameters

the previous runs. This led to the values threshold = 216 and distance = 3.0 as parameters. The results of a test run with this parameters can be seen in table 5. With this parameters the F-measure has improved to 71.6%.

4.2 Detecting traffic lights while driving

A further challenge is to detect traffic lights while driving. Driving especially at high speeds can produce blurry images and the originally circular shapes of the traffic lights get distorted to elongate shapes as can be seen in figure 5. Using the same parameters as in the test before results in the confusion matrix in table 6. The f1 score here is only 39.9%.

4.3 Runtime

Although the hill climbing does improve the quality of the traffic light detection, a lower threshold as parameter also increases the runtime. This is

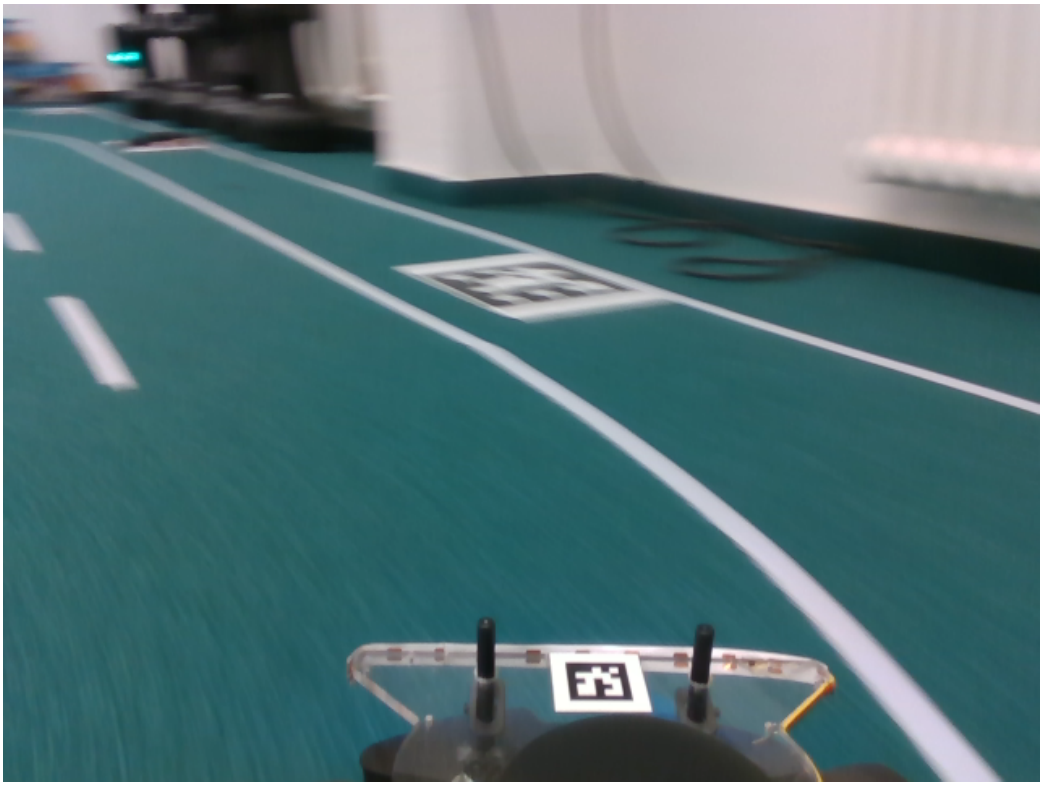


Figure 5: Traffic lights get blurry and distorted when driving fast

because with a lower threshold the binary image created in 3.2 will contain more white pixels and with more white pixels more and bigger shapes are created in 3.3. With more shapes there are more candidates for traffic lights that need to be checked thus increasing the overall runtime. On the laptop running the tests with an Intel® Core™i5-8250U, a test with the initial threshold of 250 took only 3 minute and 41 seconds for all 1904 images or 0.1161 seconds per image. A second run with a threshold reduced to 216 took 19 minutes and 1 second for the 1904 images or 0.5993 seconds per image. This means the time needed per image has increased more than 5 times.

It is also important to measure to runtime on the hardware of the model car, since this is where the software will be used. To measure the runtime on the model car whenever an image is received from the subscribed image topic the current time is taken. After the detection method has done its work the time is taken again and the difference between the time from the start and

threshold	**250**	**216**
minimum Runtime	0.5790 sec	5.0202 sec
mean Runtime	0.6375 sec	11.3422 sec
maximum Runtime	0.9049 sec	15.8051 sec

Table 7: Runtime on the model car

the time taken at the end is the runtime. This runtime is then published to a new topic `/traffic_lights/runtime`. In table 7 the min, max and mean time per image can be seen. Two tests were carried out with 250 and 216 as parameter for the threshold. For each test the car was standing for 1 minute in front of a traffic light and for another minute the traffic light was removed from the cars view. As can be seen the mean runtime per image on the car increased by more than 17 times.

Traffic light detection in real-time would therefore be impossible on the hardware of the car as traffic lights would only be detected with a delay of more than 10 seconds. Only with a higher threshold or external hardware is it possible to detect traffic lights within a reasonable time.

5 Results and Future Work

The presented method is capable of detecting the traffic lights in the Robotics Lab environment but is only reliable at short distances about 1 meter. Greater distances decreased the number of correctly identified traffic lights and detecting traffic lights while driving at high speeds also proved difficult for this method due to blurry and distorted images.

Running on external hardware the detection was fast enough but the runtime measurements on the model car itself showed that the method cannot be used for real-time detection on the hardware of the car.

At the time of testing there was only one traffic light in the Robotics Lab but with multiple traffic lights it would also be necessary to check which traffic light controls which lane. In the case of multiple traffic lights for one lane the results could be verified by comparing then with other detections.

The tracking of already detected traffic lights could be used to predict the position of a traffic light in the next image. Anticipating the position in an image allows to narrow down the search area even further and to double check if no traffic light is detected where one is to be expected. This could also be extended to not only tracking the traffic light frame by frame but also tracking its position globally. The model car would then know where to expect traffic lights when revisiting an area.

References

[1] M. Weber, P. Wolf, and J. M. Zöllner, "Deeptlr: A single deep convolutional network for detection and classification of traffic lights," in *2016 IEEE Intelligent Vehicles Symposium (IV)*, pp. 342–348, IEEE, 2016.

[2] E. Lee and D. Kim, "Accurate traffic light detection using deep neural network with focal regression loss," *Image and Vision Computing*, vol. 87, pp. 24–36, 2019.

[3] K. Behrendt, L. Novak, and R. Botros, "A deep learning approach to traffic lights: Detection, tracking, and classification," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1370–1377, IEEE, 2017.

[4] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 779–788, 2016.

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

[6] R. De Charette and F. Nashashibi, "Traffic light recognition using image processing compared to learning processes," in *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 333–338, IEEE, 2009.

[7] A. Alam and Z. A. Jaffery, "A vision-based system for traffic light detection," in *Applications of Artificial Intelligence Techniques in Engineering*, pp. 333–343, Springer, 2019.

[8] M. Omachi and S. Omachi, "Traffic light detection with color and edge information," in *2009 2nd IEEE International Conference on Computer Science and Information Technology*, pp. 284–287, IEEE, 2009.

[9] M. Omachi and S. Omachi, "Detection of traffic light using structural information," in *IEEE 10th International Conference on Signal Processing Proceedings*, pp. 809–812, IEEE, 2010.

[10] P. V. Hough, "Machine analysis of bubble chamber pictures," in *Conf. Proc.*, vol. 590914, pp. 554–558, 1959.

[11] S. Sooksatra and T. Kondo, "Red traffic light detection using fast radial symmetry transform," in *2014 11th International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON)*, pp. 1–6, IEEE, 2014.

[12] "Autominy core." `https://autominy.github.io/AutoMiny/docs/autominy-core/`. Accessed: 2020-02-20.

[13] "Ros.org." `https://www.ros.org/about-ros/`. Accessed: 2020-02-20.

[14] D. Powers, "Evaluation: from precision, recall and f-factor to roc, informedness, markedness andamp; correlation," *J Mach Learn Technol*, vol. 2, pp. 2229–3981, 2008.

[15] Y. Sasaki, "The truth of the f-measure. 2007." `https://www.toyota-ti.ac.jp/Lab/Denshi/COIN/people/yutaka.sasaki/F-measure-YS-26Oct07.pdf`, 2007. Accessed: 2020-03-05.

[16] S. Russell and P. Norvig, "Artificial intelligence: a modern approach (global 3rd edition)," *Essex: Pearson*, pp. 122–125, 2016.

[17] R. C. Gonzales and R. E. Woods, "Digital image processing, 4th edn.," 2017.

[18] T. Gevers and A. W. Smeulders, "Color-based object recognition," *Pattern recognition*, vol. 32, no. 3, pp. 453–464, 1999.

Appendices

A Confusion matrices

detectetd as \ Truth	r	g	y	r-y	e
r	428	0	0	0	0
g	0	77	0	0	0
y	0	0	26	1	0
r-y	2	0	0	58	0
e	1	56	1	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	0	0	0	0	0
g	0	0	0	0	0
y	0	0	10	26	0
r-y	0	0	0	0	0
e	322	413	40	6	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	20	0	0	0	0
g	0	6	0	0	0
y	6	0	2	0	0
r-y	0	0	0	0	0
e	95	101	11	12	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	448	0	0	0	0
g	0	83	0	0	0
y	6	0	38	27	0
r-y	2	0	0	58	0
e	428	570	52	8	184

(d) all images

Table 8: threshold = 250, distance = 5.0, minimum Pixels=10

detectetd as \ Truth	r	g	y	r-y	e
r	427	0	0	1	0
g	0	77	0	0	0
y	1	0	26	1	0
r-y	2	0	0	57	0
e	1	56	1	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	0	0	0	0	0
g	0	0	0	0	0
y	0	0	10	26	0
r-y	0	0	0	0	0
e	322	413	40	6	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	11	0	0	0	0
g	0	6	0	0	0
y	16	0	2	0	0
r-y	0	0	0	0	0
e	94	101	11	12	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	438	0	0	0	0
g	0	83	0	0	0
y	17	0	38	27	0
r-y	2	0	0	57	0
e	427	570	52	9	184

(d) all images

Table 9: threshold = 250, distance = 3.0, minimum Pixels=10

detectetd as \ Truth	r	g	y	r-y	e
r	167	0	0	13	0
g	0	0	0	0	0
y	261	0	26	23	0
r-y	2	0	0	20	0
e	1	133	1	3	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	0	0	0	0	0
g	0	0	0	0	0
y	0	0	0	4	0
r-y	0	0	0	0	0
e	322	413	50	28	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	1	0	0	0	0
g	0	1	0	0	0
y	22	0	1	0	0
r-y	0	0	0	0	0
e	98	106	12	12	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	168	0	0	4	0
g	0	1	0	1	0
y	283	0	27	8	0
r-y	2	0	0	19	0
e	431	652	63	61	184

(d) all images

Table 10: threshold = 250, distance = 2.0, minimum Pixels=10

detectetd as \ Truth	r	g	y	r-y	e
r	203	0	0	11	0
g	0	133	0	2	0
y	226	0	27	13	0
r-y	2	0	0	33	0
e	0	0	0	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	0	0	0	0	0
g	0	0	0	0	0
y	252	0	33	12	0
r-y	5	0	0	0	0
e	65	413	17	20	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	16	0	0	0	0
g	0	21	0	0	0
y	26	0	2	12	0
r-y	0	0	0	0	0
e	79	86	11	0	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	219	0	0	2	0
g	0	154	0	5	0
y	513	0	62	23	0
r-y	7	0	0	32	0
e	145	499	28	31	184

(d) all images

Table 11: threshold = 216, distance = 5.0, minimum Pixels=10

detectetd as \ Truth	r	g	y	r-y	e
r	429	0	0	2	0
g	0	133	0	1	0
y	0	0	27	1	0
r-y	2	0	0	57	0
e	0	0	0	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	122	0	0	5	0
g	0	155	0	5	0
y	195	0	50	10	0
r-y	5	0	0	0	0
e	0	258	0	12	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	23	0	0	3	0
g	0	22	0	0	0
y	29	0	5	8	0
r-y	0	0	0	0	0
e	69	85	8	1	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	577	0	0	7	0
g	1	310	0	6	0
y	230	0	82	9	0
r-y	7	0	0	58	0
e	69	343	8	15	184

(d) all images

Table 12: threshold = 216, distance = 3.0, minimum Pixels=10

detectetd as \ Truth	r	g	y	r-y	e
r	429	0	0	0	0
g	0	133	0	0	0
y	0	0	27	0	0
r-y	2	0	0	59	0
e	0	0	0	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	93	0	0	4	0
g	0	155	0	5	0
y	222	0	50	8	0
r-y	7	0	0	4	0
e	0	258	0	11	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	21	0	0	7	0
g	0	22	0	0	0
y	31	0	10	3	0
r-y	0	0	0	1	0
e	69	85	3	1	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	550	0	0	3	0
g	0	310	0	4	0
y	255	0	87	9	0
r-y	10	0	0	64	0
e	69	343	3	13	184

(d) all images

Table 13: threshold = 216, distance = 2.0, minimum Pixels=10

detectetd as \ Truth	r	g	y	r-y	e
r	429	0	0	0	0
g	0	77	0	0	0
y	0	0	26	1	0
r-y	2	0	0	58	0
e	0	56	1	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	4	0	0	0	0
g	0	0	0	0	0
y	121	0	26	2	0
r-y	0	0	0	0	0
e	197	413	24	30	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	23	0	0	0	0
g	0	6	0	0	0
y	11	0	5	0	0
r-y	0	0	0	0	0
e	87	101	8	12	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	456	0	0	2	0
g	0	83	0	1	0
y	132	0	57	5	0
r-y	2	0	0	59	0
e	294	570	33	26	184

(d) all images

Table 14: threshold = 250, distance = 5.0, minimum Pixels=5

detectetd as \ Truth	r	g	y	r-y	e
r	427	0	0	1	0
g	0	77	0	0	0
y	2	0	26	1	0
r-y	2	0	0	57	0
e	0	56	1	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	0	0	0	0	0
g	0	0	0	0	0
y	125	0	26	2	0
r-y	0	0	0	0	0
e	197	413	24	30	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	14	0	0	0	0
g	0	6	0	0	0
y	21	0	5	0	0
r-y	0	0	0	0	0
e	86	101	8	12	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	441	0	0	2	0
g	0	83	0	1	0
y	148	0	57	5	0
r-y	2	0	0	58	0
e	293	570	33	27	184

(d) all images

Table 15: threshold = 250, distance = 3.0, minimum Pixels=5

detectetd as \ Truth	r	g	y	r-y	e
r	167	0	0	13	0
g	0	0	0	0	0
y	262	0	26	23	0
r-y	2	0	0	20	0
e	0	133	1	3	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	0	0	0	0	0
g	0	0	0	0	0
y	0	0	1	4	0
r-y	0	0	0	0	0
e	322	413	49	28	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	1	0	0	0	0
g	0	1	0	0	0
y	23	0	1	0	0
r-y	0	0	0	0	0
e	97	106	12	12	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	168	0	0	4	0
g	0	1	0	1	0
y	285	0	28	8	0
r-y	2	0	0	19	0
e	429	652	62	61	184

(d) all images

Table 16: threshold = 250, distance = 2.0, minimum Pixels=5

detectetd as \ Truth	r	g	y	r-y	e
r	203	0	0	11	0
g	0	133	0	2	0
y	226	0	27	13	0
r-y	2	0	0	33	0
e	0	0	0	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	0	0	0	0	0
g	0	117	0	2	0
y	252	0	33	12	0
r-y	5	0	0	0	0
e	65	296	17	18	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	16	0	0	0	0
g	0	27	0	0	0
y	46	0	2	7	0
r-y	0	0	0	0	0
e	59	80	11	5	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	221	0	0	2	0
g	0	277	0	7	0
y	529	0	62	24	0
r-y	7	0	0	32	0
e	127	376	28	28	184

(d) all images

Table 17: threshold = 216, distance = 5.0, minimum Pixels=5

detectetd as \ Truth	r	g	y	r-y	e
r	429	0	0	2	0
g	0	133	0	1	0
y	0	0	27	1	0
r-y	2	0	0	57	0
e	0	0	0	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	122	0	0	5	0
g	0	413	0	17	0
y	195	0	50	10	0
r-y	5	0	0	0	0
e	0	0	0	0	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	23	0	0	1	0
g	0	51	0	0	0
y	72	0	5	3	0
r-y	0	0	0	3	0
e	26	56	8	5	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	575	0	0	8	0
g	2	597	0	15	0
y	270	0	82	9	0
r-y	10	0	0	58	0
e	27	56	8	5	184

(d) all images

Table 18: threshold = 216, distance = 3.0, minimum Pixels=5

detectetd as \ Truth	r	g	y	r-y	e
r	429	0	0	0	0
g	0	133	0	0	0
y	0	0	27	0	0
r-y	2	0	0	59	0
e	0	0	0	0	0

(a) 1 meter

detectetd as \ Truth	r	g	y	r-y	e
r	93	0	0	4	0
g	0	272	0	7	0
y	222	0	50	8	0
r-y	7	0	0	4	0
e	0	141	0	9	0

(b) 2-3 meter

detectetd as \ Truth	r	g	y	r-y	e
r	21	0	0	1	0
g	0	63	0	0	0
y	74	0	10	1	0
r-y	0	0	0	8	0
e	26	44	3	2	184

(c) driving

detectetd as \ Truth	r	g	y	r-y	e
r	543	0	0	4	0
g	0	468	0	7	0
y	297	0	87	8	0
r-y	17	0	0	64	0
e	27	185	3	10	184

(d) all images

Table 19: threshold = 216, distance = 2.0, minimum Pixels=5

B Plots