Danmarks Tekniske Universitet

Anker Engelunds Vej 1

Bygning 101A

2800 Kgs. Lyngby

01.08.2018

# Synthesis Course Report

Vision System for the Autonomous Shell Eco-Marathon car

s171811 Mikołaj Patalan

# Table of contents

## 1 Introduction

The objective of this Synthesis Course was to build a vision system that could extend perception of surrounding environment of DTU Ecocar for the Autonomous UrbanConcept Competition(AUC) 2018 competition, which is part of the Shell Eco-Marathon(SEM). The SEM main competition is focused on achieving highest fuel efficiency by students' self-build vehicles. At 2018 edition of SEM, new category of challenges was added to the competition. The new AUC category revolved around self-driving cars solving diverse tasks, including navigation through complex track, obstacle avoidance and parking a vehicle in a marked rectangle. This project addressed the latter problem, which related to 2 of 5 AUC challenges. Parking spot detection was included in:

- Maneuverability Challenge – a vehicle had to find and maneuver into a parking spot marked on the track with black and yellow tape, which formed a rectangle. The rectangle was 4x3m and a blue block 3x0.5m and was placed behind the rectangle.
- Last-Minute Challenge – a vehicle had to park within one of 4 parking spots. Parking spots had same dimensions as in Maneuverability Challenge, however there was no block behind them. Right before the challenge, a number from 1 to 4 was provided by AUC organizers, and the car had to maneuver into that parking spot.

The vision system had to find a real world coordinates of a parking spot center in car's global coordinate system at least 10 meters before the parking spot with frequency of at least 2Hz. This part of the project dealt with detection of a parking spot on an image and statistical analysis of detected parking spots, whereas Tomasz Firynowicz focused on hardware, camera calibration, ROS integration and transformation from image coordinates to real world coordinates.

## 2 Overview of the vision system

During the system development two approaches to finding a parking spot were considered. First was to detect black and yellow striped lines on the ground that marked a parking rectangle. This method was used as a primary tool to detect parking spot. It was tested on many occasions and proven to be working correctly. However, in certain situations where shadows were casted on the parking spot, the program could fail to detect a parking spot. Therefore, a second solution was developed, where an algorithm focused on detecting a blue block that stood behind a parking spot. As can be seen on Fig.2.1 the left side of the flowchart corresponds to the lines detection and the right side to the block detection. The system would start to process new camera frame with the first method, and in the case where the vehicle couldn't find a parking spot for $d_{block}$ meters, then the system would switch to the blue block detection. Both methods will be described in the next chapters. If either parking spot lines or a blue block coordinates were found, then the program would send it to the statistical analysis part of the system, where all the previous detections were kept and segregated. The program was divided into 3 source files:

- *parking_spot_detection.cpp* – handled ROS setup, receiving new camera frames, publishing detected parking spots, perspective transformation and statistical analysis
- *parking_spot_processing.cpp* – detected parking lines

- *parking_spot_processing_block.cpp* – detected a blue block

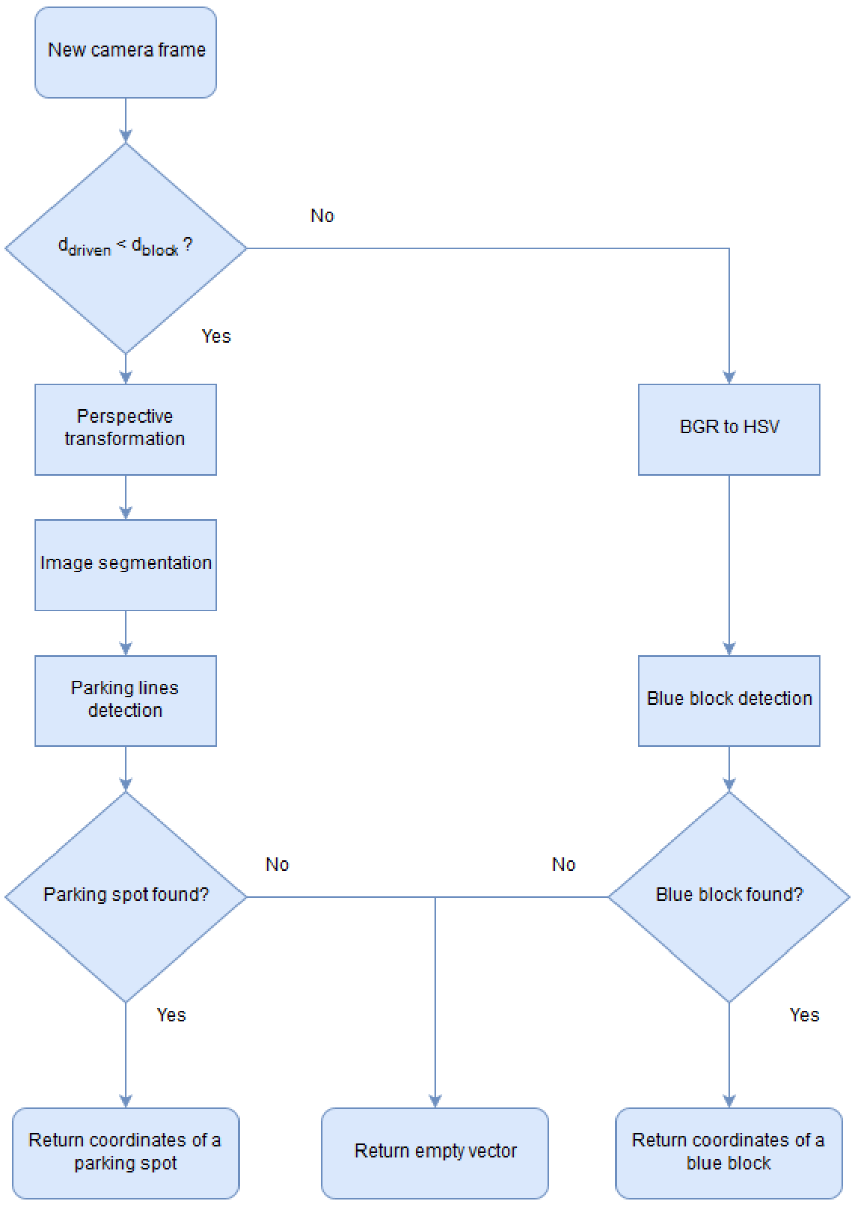


*Fig.2.1 Flowchart representing overview of the parking spot detection system.*

## 3 Parking lines detection

Primary algorithm for the parking spot detection is based on finding black and yellow striped lines on the ground. In general, program uses changes of color between lines and ground to detect shape of the whole parking spot, rather than particular color or striped patterns. The main advantage of this approach is that it is more robust to lighting conditions in comparison to color detection techniques. Many methods for pure color detection were tested prior to finding the final solution. Color segmentation techniques, such as adaptive thresholding, Otsu's method and k-means, failed to segment lines because parking lines does not create a dominant color in an image taken in the unknown, natural environment. Moreover, simple thresholding techniques, such as testing if pixel's color is in a preset range of all HSV channels would fail in various lighting conditions. The other tested method was template matching for black and yellow stiped patterns of lines. This solution would have high certainty and low false positive rate, since such patterns rarely exit in natural scenes. Unfortunately, strips couldn't be detected from far enough to be useful as black parts blended with yellow in an image. In the end, rectangular shape of parking spot was used to detect center and orientation of a parking spot. In next the paragraphs, all the parts of the used algorithm will be explained as consecutive steps of an image processing pipeline. Fig.3.1 shows steps of the image processing pipeline.

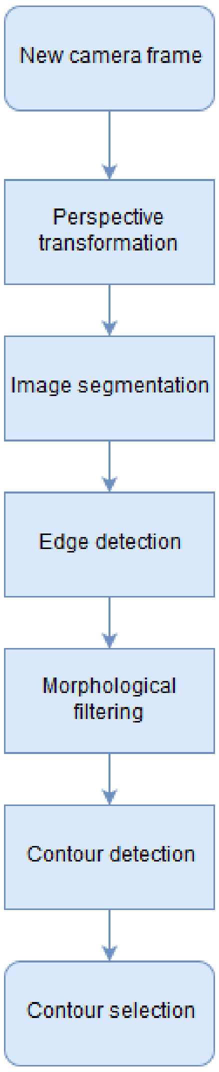


*Fig.3.1 Image processing pipeline of the parking spot lines detection.*

## 3.1 New camera frame and perspective transformation

This part of the project was done by Tomasz Firynowicz and is described in details in his report for the Ecocar project. Therefore, it will be covered very briefly. New camera frames are acquisitioned with 15fps. Since processing time of the whole algorithm is about 3-5Hz, depending on chosen image resolution, the program cannot analyze every new frame and a buffer always keeps a latest frame. Afterwards, perspective transformation is applied to an image in order to transform an image from camera coordinates to bird's eye view. In this part of the project it is used to remove perspective distortions related to the ground, so that parking spot lines ideally create rectangular shape. Example of source image and perspective transformation can be seen in Fig.3.2 and Fig.3.3.

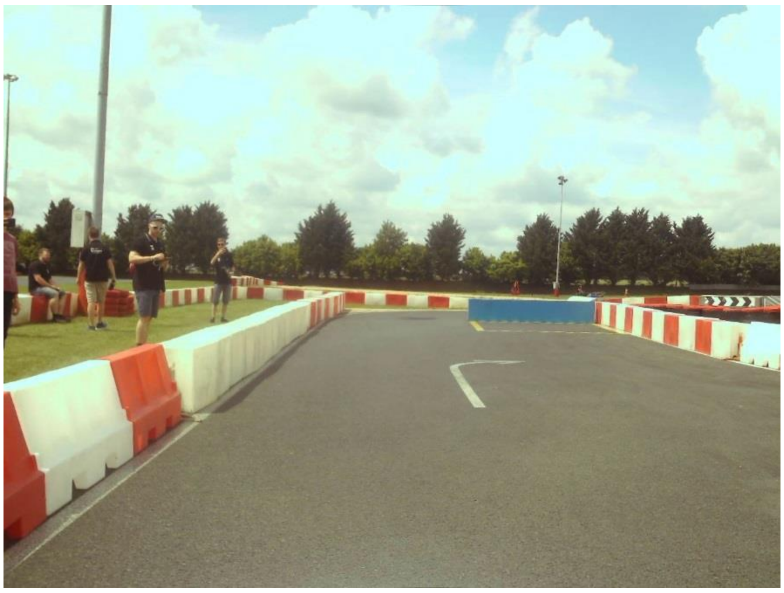


*Fig.3.2 Source image.*

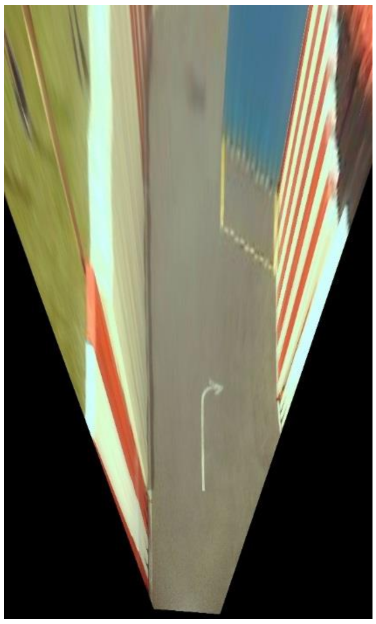


*Fig.3.3 Image after perspective transformation.*

## 3.2 Image segmentation

Image segmentation is done to separate a road from environment. GrabCut algorithm from OpenCV library was used for segmentation purposes. Before using the actual function image is downsampled by the factor of 4, in order to improve processing time. First, user has to define two bounding boxes, one containing pixels that are within region of interest(ROI) and second containing likely non-ROI pixels. In road segmentation problem, the fact that ROI is always in front of the car can be leveraged to optimize GrabCut algorithm. All pixels are selected as non-ROI, except for the 36x140 box, located in the bottom center of an image, which marks ROI pixels. GrabCut is graph based algorithm, that treats all the pixels as Random Markov Field and tries to find an optimal cut in a connected graph. It returns binary mask of segmented road, which has to be upscaled to the original resolution. Pixels that have value of 1 in the mask are kept, while all others are set to the average color of the segmented road. Fig.3.4 shows example of an image with segmented road. Image segmentation is done to make the program less prone to falsely detecting parking spots that are outside of ROI. It is not a necessary step in the image processing, but it is still recommended. For speed up of program execution it might be disabled. Lastly, as a further improvement it could be combined with lidar measurements to detect barriers.

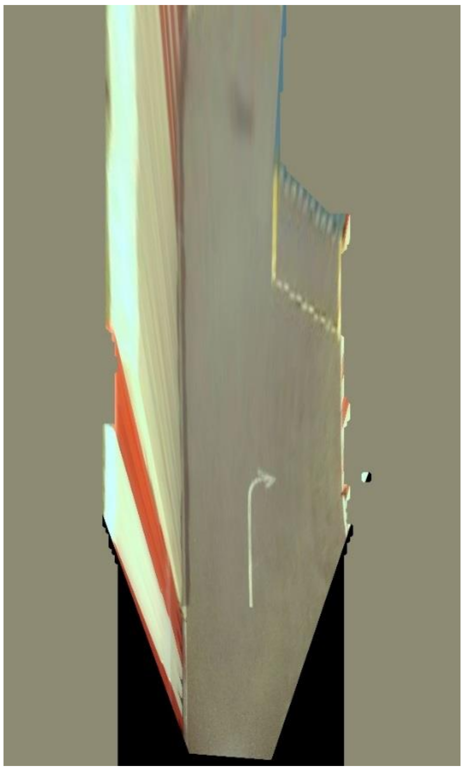
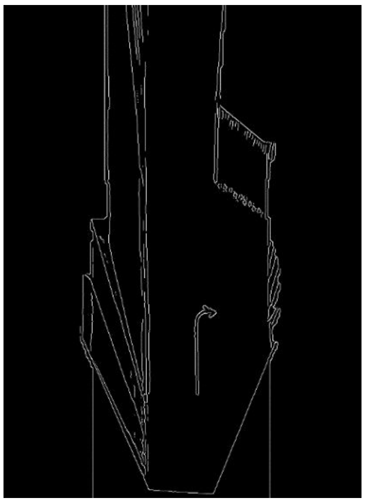
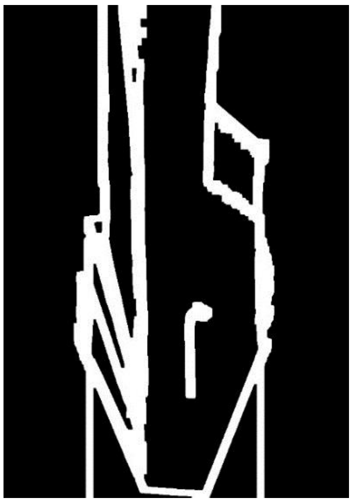


*Fig.3.4 Road segmentation.*

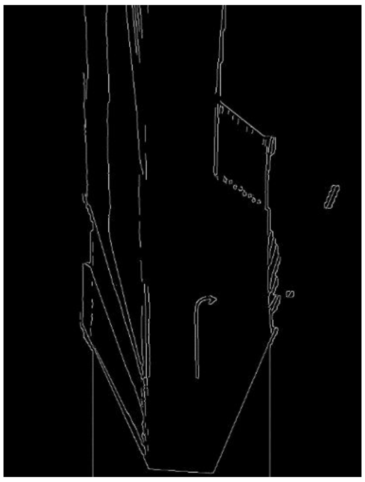
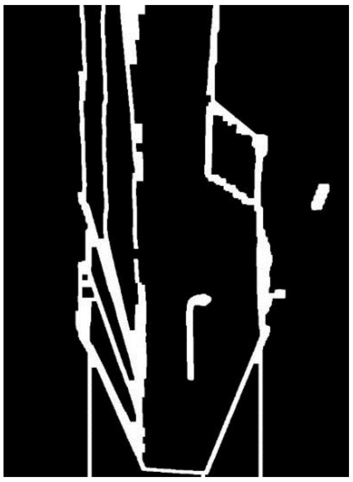
## 3.3 Edge detection and morphological filtering

Firstly, Gaussian blur was used to remove high frequency noise in an image. Then, Canny edge detection implemented in OpenCV library was applied to segmented BGR image. Its purpose is to detect changes in color between lines and ground. One thresholding parameters for Canny edge detection couldn't be specified for all the lighting conditions, therefore program runs three sets of parameters, that should work in various situations, ranging from cloudy to sunny weather. Next, morphological filters are applied to obtain connected regions in a binary image. Fig.3.5 shows an example image after Canny edge detection and morphological filters processed with 3 sets of parameters. The major disadvantage of the whole method for parking spot detection lies in the aforementioned usage of canny edge detection. For this project, assumption was made that there are no considerable color changes inside the parking spot. Sometimes during testing this assumption couldn't hold, since cracks in the road, different asphalt materials and shadows would create edges that obstructed proper functioning of the rest of the image analysis pipeline. Partially, this problem was solved by increasing Gaussian blur to obtain more uniform color inside parking spot. However, it also decreased sharpness of parking spot lines.

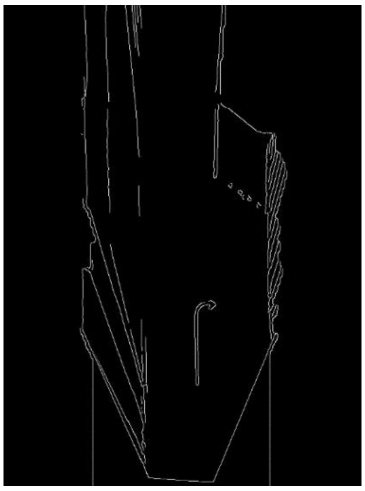
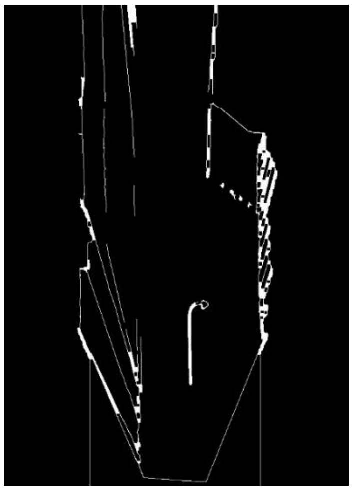
1. Low Canny = 800, High Canny = 850

Dilatile1 = 5, Dilatile2 = 7
Erode1 = 4,Erode2 = 2

2.Low Canny = 900, High Canny = 1100

Dilatile1 = 5, Dilatile2 = 5
Erode1 = 3,Erode2 = 4

3.Low Canny = 1100, High Canny = 1400

Dilatile1 = 2, Dilatile2 = 3
Erode1 = 2,Erode2 = 2

*Fig.3.5 Canny Edge detection on the left and corresponding morphological filtering on the right. Low and high Canny relates to thresholding values. Dilatile and erode describes number of iterations.*

## 3.4 Contour detection and selection

The task was to find four corners in an image that corresponds to parking spot corners. In the beginning, Hough transform for lines was used to find probable lines that would form a parking spot. This approach was difficult to work with, because selection of right lines was hindered by detection of multiple lines that were outside parking spot. In the final approach, all possible contours were analyzed to find the one making up a parking spot. Firstly, contours were found by the function *findContours* from the OpenCV library. Then, they were approximated by Douglas-Peucker algorithm(OpenCV) to obtain similar curves to originals, but with fewer points. Afterwards series of requirements for rectangularity were applied to select only contours that could possibly be a parking spot. Requirements for parking spot test:

- 4 corners with angles close to 90 degrees
- area of contour within specified range
- convex contour
- ratio of longest to shortest line within specified range
- average color inside contour should be similar to average road color

All contours that passed these checks were kept in a vector. Then, if contours were very similar location-wise, they were counted as duplicates and were averaged out. This could happen when contours of inside and outside of a parking were found or when different sets of parameters during edge detection found the same parking spot. Vector of parking spots is returned if any spots are found, otherwise an empty vector is returned.

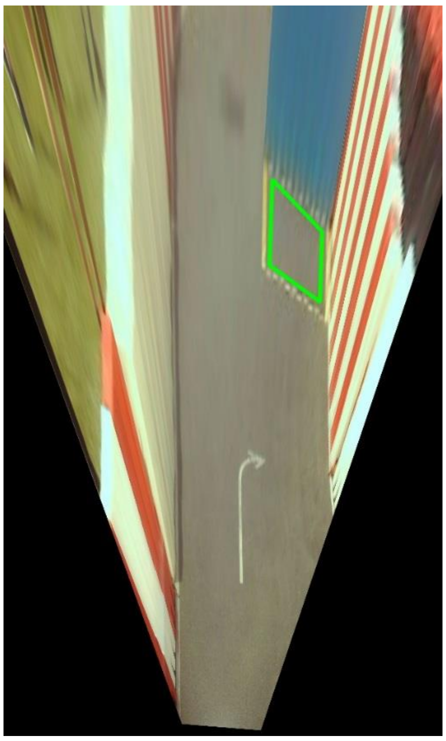


*Fig.3.4 Image after perspective transformation with detected parking spot lines drawn as the green quadrangle.*

## 4 Blue block detection

As mentioned in the introduction, if the algorithm for parking spot lines detection couldn't find a parking spot for $d_{block}$ meters, then the algorithm for the blue block detection was used. In comparison to lines detection, the blue block has large area and uniform color, therefore it was possible to apply simple thresholding techniques to detect it. First, an image was converted from BGR to HSV colorspace in order to separate hue and saturation channel from value channel. In theory, this procedure should ensure robustness to light changes. In practice, it only partially made method invariant to lighting conditions. Next, all the pixels in hue and saturation channels are checked if they were within set range of values, that relates to blue color. Two binary masks from hue and saturation channels were then combined by logical AND operator to create a final mask. Afterwards, contour detection and selection, as described in the previous chapter, were used to find a rectangular shape of a blue block. Fig.3.4 shows source image with found blue block drawn as a green quadrangle.

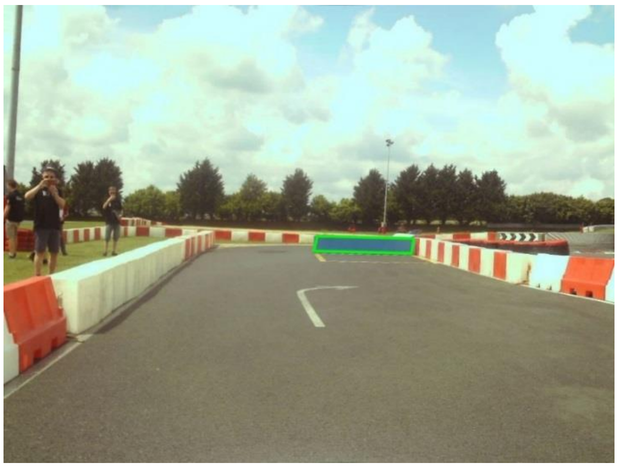


*Fig.3.4 Source image with detected blue block drawn as the green quadrangle.*

## 5 Statistical analysis

Detection history of all found parking spots was kept over time to perform statistical analysis. In order to compare them, they had to be transformed to global coordinate system. Firstly, they had to be transformed from image coordinates into local real world distances. Then by knowing car's odometry parking spots coordinates could be transformed into global coordinate system. Grouping of new parking spots was based on location of their center. If a parking spot center was close to average of parking spot group centers, then it was added to that group. Closeness was determined by the condition that an Euclidean distance from a new parking spot center to an average of a group had to be smaller than the grouping parameter. If a new parking spot couldn't be added to any group, then a new group would be created. Afterwards, a group of parking spots was validated if it consisted of at least 3 detections. This validation size of 3 detections ensured that single false detections weren't send as an actual parking spots. Then program behaved differently for two parking spot challenges:

- single parking spot: from all validated groups choose a group with most detections and publish an average of the group to navigation system
- multiple parking spots: sort all groups by their distance to a global coordinate origin and publish an nth average of the group, were n is specified by a user

# 6 Conclusion

## 6.1 Project objectives

The aim of this project was to create a vision system that could find a parking spot for two challenges at Shell Eco-Marathon 2018 competition for self-driving vehicles. Many solutions had been considered and tested before deriving the final solution. The created system for parking spot detection was able to successfully find parking spots for both challenges during qualification event in Paris 2018 as well as during Shell Eco-Marathon London 2018. DTU Roadrunners was the only team to solve both parking challenges, because other teams used lidar perception instead of camera vision, which couldn't detect parking spots in the fifth challenge. The vision system worked with 3-5Hz frequency, could detect parking spots up to 20m and almost never returned false positives, hence the vision system met the set requirements.

## 6.2 Unsuccessful attempts

On two occasions program failed to work properly. First was during the fifth challenge, where the car had to distinguish correct parking spot among multiple parking spots. The program was able to detect parking spots, however wrongly adjusted parameters in statistical analysis part resulted in not enough detections for the program to publish a parking spot to the navigation part of the project. Too small grouping parameter caused the program to classify the same parking spot into different groups, resulting in more detected parking spots than in reality. Also, validation size of parking spots was set too high, therefore only once in few attempts the program found the same parking spot multiple times and could mark it as validated spot to publish. Second situation when the vision system failed to work was during the Eco-car presentation for Prince Joachim. On this occasion, calibration was done on completely flat terrain, whereas the presentation took place on a road sloped to the sides. This resulted in a skewed image transformation, where parking spot contour had angles outside set tolerances. Afterwards, constrains for 90 degrees angles where relaxed to prevent similar situations.

## 6.3 Possible future improvements

The major flaw of this system is that if there is any significant change of color inside the parking spot, then the algorithm fails to detect a parking spot. For example, this might be caused by a shadow during sunny day or by partially wet asphalt after rain. Shadow removal from an image is a well-known problem in image processing without a standard automatic solution. Methods for increasing color constancy in different illumination conditions, such as histogram equalization, resulted in no improvement. Researching solutions that target hardware properties of a camera might be a solution.