ISSN 2464-4617(print)
ISSN 2464-4625(CD)
Computer Science Research Notes
CSRN 2801
Full Papers Proceedings
http://www.WSCG.eu

Real-Time Visual Off-Road Path Detection

Marc Steven Krämer, Lars Kuhnert and Klaus-Dieter Kuhnert
Department of Electrical Engineering & Computer Science
Institute of Real-Time Learning Systems
University of Siegen, Germany
marc.kraemer@uni-siegen.de, lars.kuhnert@uni-siegen.de, kuhnert@fb12.uni-siegen.de

ABSTRACT

In this paper, we propose a fast and real-time capable system for visual off-road path detection. We equipped our robot AMOR with a single monocular camera and explored unstructured environments like woods. In these areas, it is almost harder to identify and classify drivable and non-drivable parts in an image. In urban regions, roads can be detected by lane markers or delimitations whereas the boundaries of a forest path blend into the environment almost seamlessly. In our work, we developed a software system that is based on mostly simple and low computationally intensive algorithms. We developed and tested the functions with a large dataset of camera images and also generated a manually Ground Truth for the evaluation.

Keywords

Road Detection, Off-road Autonomous Navigation, Image Segmentation, Terrain Classification, Mobile Robots

1 INTRODUCTION

In recent years, interest in autonomously operating robots has increased. The main task is the navigation from point A to point B as well as the necessary road detection. In general, a distinction can be made between structured roads and unstructured roads (off-road) [8]. Most of the problems in identifying roads in urban areas have widely been solved by the automotive industry[1]. The road boundaries and landmarks are a great help for the detection task in this environment [6]. Therefore, the current challenge is the path recognition in natural, off-road environments. Since it is not necessary to have roads available, the area is divided into passable and non-passable corridors.

The developed path recognition systems are based on various sensors. A laser scanner or stereo camera can detect areas close in front of the vehicle precisely in 3D. However, in order to reach higher speeds it is also necessary to cover other distances that radar or monocular cameras can detect [8]. Other previous projects mainly use stereo cameras .[8][7][4][9][6][5]. In the case of monocular camera-based methods, the main focus was on terrain classification by means of textures and obstacle detection. The classifier used for the path recognition is previously trained with different types of road images. This training is expensive because it has to be done for many types of roads [2] [5] [1]. Another approach splits the camera image into separate grid cells. The cells are now compared by their similarity and subdivided into drivable and non-drivable areas. This is based on a cell directly in front of the vehicle, which is assumed as being a part of the road [3].

The majority of the previously presented procedures have been developed for the use during the DARPA Challenge in America. This runs through the Mojave Desert. In this Paper, we also present a monocular camera based system. The algorithms used in this work have been created especially for our local areas and environments. These include forest areas with more or less fortified paths and different seasons.

This paper is organized as follows: In the next section, we describe the hardware setup and the generated test dataset. The general software architecture and some sub functions are explained in section 3. In section 4, the analyses algorithms for detecting roads are described and will be evaluated in the following section before the paper is finally concluded.

2 HARDWARE SETUP

In this section, we describe our experimental setup. First, we introduce our off-road robot AMOR followed by a description of the test areas and the test image data set.

2.1 Robot AMOR

For our tests, we used the Robot AMOR (Autonomous Mobile Outdoor Robot) which is shown in Figure 1. The outdoor suitable robot was built on the base of

ISBN 978-80-86943-40-4
https://doi.org/10.24132/CSRN.2018.2801.9

a quad from Yamaha. The basic concept was the fusion of a large number of redundant sensors to achieve a robust function even in complex and variable environments. The robot is equipped with various different cameras, starting from simple USB webcams to industrial highly-sensitive monocular or stereo camera systems.

Figure 1: Autonomous Mobile Outdoor Robot.

2.2 Test Dataset

To achieve a robust functioning system we needed to collect a huge set of test data. Therefore, we selected five different test paths with a total length of 7.5km and changing road surfaces. The individual routes were approached at different seasons and weather conditions. Thus, we generated a dataset with 23 sequences and 43961 images.

Figure 2: Manual Ground Truth with binary mask.

To evaluate our results, we manually created a Ground Truth and labeled a part of our dataset. This resulted in 1884 images pairs of an original camera image and a corresponding binary mask, where we marked the drivable path (see Figure 2).

2.3 Auxiliary Rectangles

The recordings of the dataset were made with different cameras on various positions on the robot. In addition to the resolution and the aspect ratio even the opening angle of the lens and the position differs. In addition, there are other objects in the visual field, like a lidar sensor or parts of the robots frame. To deal with these problems, we used auxiliary rectangles. As a simple and good option, it has been found to use three rectangles. One rectangle, which points to the road directly in front of the vehicle and two more to the right and left environment. Figure 3 shows this.

Figure 3: Auxiliary rectangles.

These rectangles are not fixed, so the position and size are variables. In a future system, it is planned to use a three-dimensional sensor, like a lidar or stereo-camera, to precisely detect the near surroundings of the robot and classify them into drivable and non-drivable areas. Based on these informations we can adjust the road and environment rectangles to detect the distant surroundings in a fast image processing.

3 SOFTWARE ARCHITECTURE

This section describes the basic structure of the software. The development was based on the idea of a modular and extendible system. In order to understand this structure, we briefly go through the processing chain and describe the individual functions. Figure 4 shows the interaction of the software components in a flow chart.

At the beginning of the path recognition, a pre-test on the recorded image is done which will detect unusable and useless images. In this, for example, care is taken that the image is not extremely over- or underexposed, so that further processings would be impossible. A detailed description is given later in this chapter. If the pre-test fails, the image analysis is canceled and the software waits for the next image. Otherwise, the path type is determined. For this purpose, the current image is classified into a specific path type in a fast process. For example, differences are made between a forest path and a tarred road.

Based on the type of path thus determined, the analysis algorithms used for further analysis are selected. Each analysis algorithm tries to find the path by another method. For example, based on the saturation in the image, the natural environment can be distinguished from the road or, in another method, the edges of the street can be tracked. These chosen analysis algorithms are then executed in parallel.

Each of these functions provides a binary black-and-white mask as output, which specifies where is a path in the image and where not. This mask is evaluated with a plausible test. The results of all analysis algorithms are then merged. This results in a matrix in which parts of the path have a high and the environment a low value.

ISBN 978-80-86943-40-4

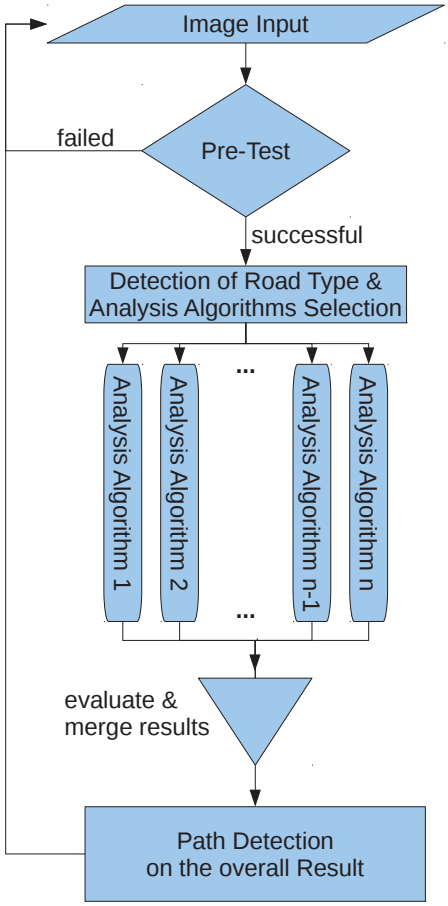

Figure 4: Software flow chart.

In other words, when expressed as an image, the result of the addition is a gray-scale image in which the recognized path has a color value greater than zero. The higher this value is in one pixel, the safer it is a drivable part.

3.1 Image Pre-Test

To detect unusable images we use a pre-test. These can occur, for example, while driving in places where individual shots are overexposed. One of the many other possibilities here is the position directly in front of a house wall, where there is no path to detect. To save computation time we recognize these images and prevent the execution of the analysis algorithms. The method developed here therefore tries to cautiously detect inappropriate images. We use the image areas within the rectangles to make a statement with as little effort as possible whether an image is usable or not. For this, we calculate the mean values of the saturation and blue channel in the segments and compare the ratio between road and environment with a threshold value. In the following equation, this is done by the example of the left environment rectangle and the blue channel.

$$\frac{max(mean_{blue}^{left}, mean_{blue}^{road})}{min(mean_{blue}^{left}, mean_{blue}^{road})} < \text{minDiff} \qquad (1)$$

Analogue for the right rectangle and the saturation of both ones. We found 1.4 for the best working minDiff value. If two or more of the four tests fail, the image is rejected.

During the image pre-test, additionally a binary mask is created which indicates bad points due to high exposure. These occur especially during trips in dark forest areas. In order to detect this, the brightness values (V) in the HSV color space are checked against a threshold value (250) up to the half of the image height, like Figure 5 shows. All image points that are contained in the mask are set to zero.

Figure 5: Highlight test with binary mask on the right.

3.2 Road Type Classification

After a successful pre-test, we detect the road type. Based on this type, the appropriate analysis algorithms will be selected and executed. The road type is distinguished between the drivable path and the surroundings. It is classified into normal road (for example tarred road), dirt road and dirt road with tracks. Figure 6 shows these different path types.

Figure 6: Path types: normal road(1), dirt road saturated environment(2), dirt road (3), dirt road with tracks(4).

We differentiate the environment only between saturated and unsaturated, where a saturated environment is characterized by a (natural) green vegetation. Figure 7 shows how to distinguish a road and a dirt road. On the bottom of the images a histogram of the road rectangle (see Figure 3) is drawn. The histogram of a dirt

ISBN 978-80-86943-40-4

road is usually relatively wide, whereas the histogram of normal roads are quite narrow. To make efficient use of this, we calculate the standard deviation of all RGB-channels. The mean of these values is used to assess the condition of the path in dirt road or normal road. A value greater than 12 means dirt road, a lower one normal road. In case of a dirt road, we also try to detect tracks for the sub type dirt road with tracks. This algorithm is explained in 4.2.

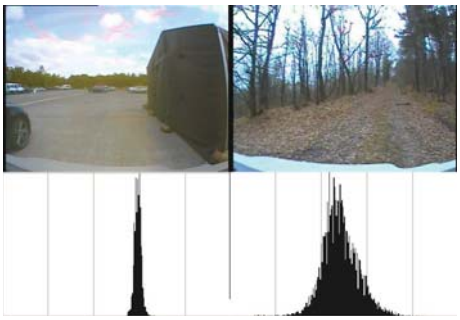

Figure 7: Road Types and Histograms.

In a final step, we have look at the environment. For this, we consider the standard deviation of the saturation channel. If this exceeds the threshold value of 20, the environment is saturated.

4 ANALYSE ALGORITHMS

The following section describes the individual analysis algorithms. Since the total number of algorithms is thirteen, they are only briefly discussed and not considered in detail. However, due to relatively simple procedures, these are easy to understand. Every algorithm gets as an input image a RGB, HSV or grayscale image and returns a binary mask. This mask separates the drivable from the non-drivable parts of the image. All following functions are named by their general idea.

4.1 Normal Road and Dirt Road

In the first part, we describe the analyses algorithms, which we use for the detection of normal and dirt roads.

Blur and Contours Detection

The algorithm BlurAndContours is a method to detect roads within a natural environment. After the image has been blurred, a contour recognition is performed. The result is then converted into a polygon. Figure 8 shows the processing steps. The blurring of the image eliminates fine structures that are common in forest roads, caused for example by fallen leaves. As a result, larger structures are detected during the following contour detection.

The contour detection is performed using the integrated OpenCV function. The necessary threshold value is determined iteratively. For this purpose, the contour

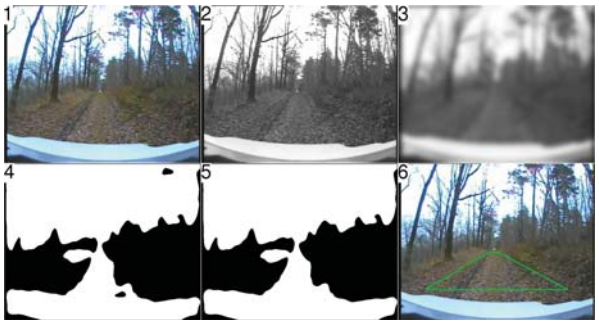

Figure 8: Algorithm Blur and Contours: Input (1), greyscale image (2), greyscale image after bluring (3), mask (4), mask without holes (5), fitted Polygon(6)

recognition is carried out until the suspected path at the top has a width of more than 75 pixels, the last threshold is accepted. The result is usually a large contour, where the narrowest point is the upper edge of the path. After this, we produce a polygon, which starts on two pre-defined origin points in front of the vehicle and ends in this point. The inner of this polygon represents the returned mask.

Equalize Saturation

The Equalize Saturation analysis method is based on changing the saturation values in the HSV color space and comparing the before and after RGB images. At first a histogram equalization in the saturation channel is done. In the histogram equalization process, the color values of a single channel (here saturation) are stretched so that they fill the entire range (0..255), as Figure 9 shows.

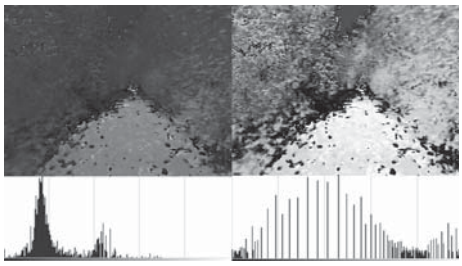

Figure 9: Saturation Histogram Equalization

Subsequently, a comparison of the blue and saturation channels is performed before and after the saturation equalization. Figure 10 shows the differences. In order to decide whether the difference images can be used for a road detection, we calculate the mean difference values in the road and both environment rectangles for the blue color channel. If the changes on the road are above a threshold and the ratio between road and environment changes is very low, the image is rejected, because the differences are not high enough. After the blue channel, the same process is repeated for the saturation channel and both results are added. Figure 11 gives a complete overview of the algorithm.

ISBN 978-80-86943-40-4

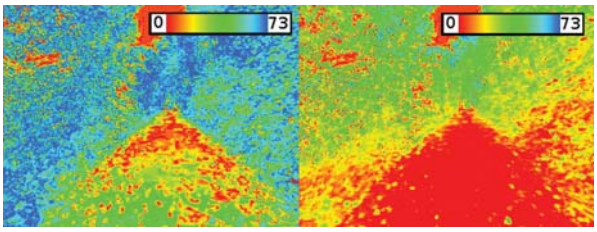

Figure 10: Differences before and after saturation equalization in blue (left) and saturation(right)-channel

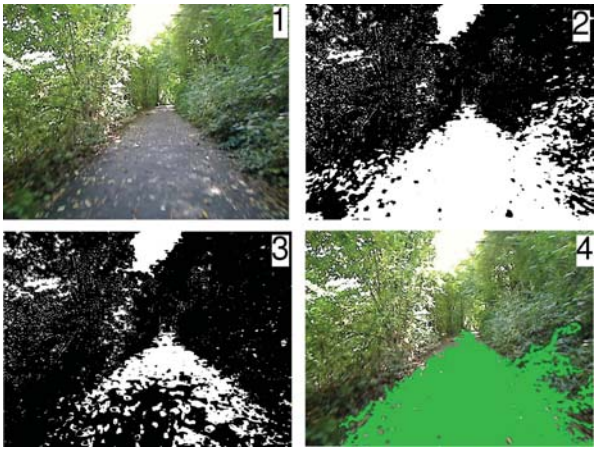

Figure 11: Equalize Saturation: Input(1), blue channel result(2), saturation channel result (3), final result(4).

Polygon Fitting

The polygon fitting method (see Figure 12) offers a very simple and fast, but often well-working way to recognize the road. Here the mask is represented by a polygon whose lower vertices are defined by the street rectangle and upper one by a simple procedure. The result of the algorithm is a mask into which a trapezoid is drawn. The trapezoid is determined by the two bottom and two top points. The two lower points are calculated using an auxiliary point in the center of the street rectangle, where a constant value is added to both sides. For the top points, another upper auxiliary point is required whose X coordinate determines the image column with the highest sum of color values and Y coordinate the image line with the smallest sum of color values as shown in Equation 2 and 3. In order to define two upper points for the trapezoid, here as well as at the lower points, a constant value is added to the left and right point.

$$P_x = x, \text{ with } max(\sum_{y=1}^{height}(I_{xy}^{blue})), x \in \{1,..,width\} \quad (2)$$

$$P_y = y, \text{ with } min(\sum_{x=1}^{width}(I_{xy}^{blue})), y \in \{1,..,height\} \quad (3)$$

Histogram Back Projection

The Histogram Back Projection is a very simple method, which is working on a single channel. The color values inside the road rectangle are stored. All image pixels are compared with this list. If the current value is listed, the pixel is marked as road, otherwise as environment. Before the execution, a pre-test is made on the road rectangle. The distance between the maximum and minimum pixel value has to be lower than a given threshold of 100. Figure 13 shows an example. An alternative method of this function uses the values in the range between the minimum and maximum value in the road values and not only the exact values.

Figure 12: Polygon Fitting: Input(left), Resulting Polygon with auxiliary points(right).

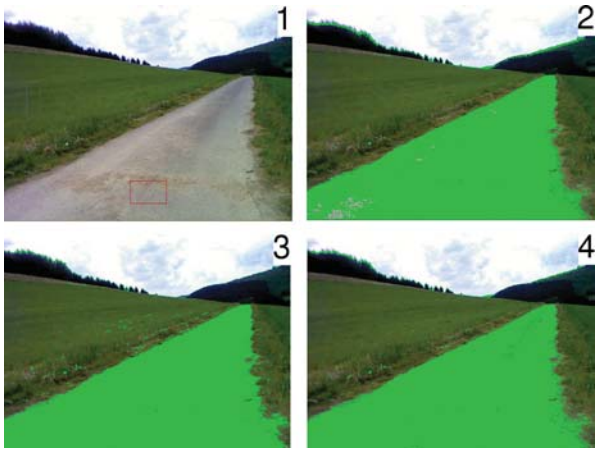

Figure 13: Histogram Back Projection: Input (with road rectangle)(1), Blue channel result (2), Green channel result (3), Val channel result (4).

Template Matching

The template matching method uses the standard OpenCV algorithm. As template area, we use the road rectangle. The following correlation function is used for the template matching process.

$$R_{ccorr}(x,y) = \sum([T(x',y') * I(x+x',y+y')]^2) \quad (4)$$

As result we get a 2-dimensional matrix, with the values of the match, like the third image in Figure 14 shows. To separate the drivable and non-drivable part, a threshold is needed. We calculate this one from the matrix values in the road rectangle as follows

$$threshold = minVal - \frac{meanVal - minVal}{2} \quad (5)$$

ISBN 978-80-86943-40-4

After a binarization of the matrix based on this threshold the resulting image is generated.

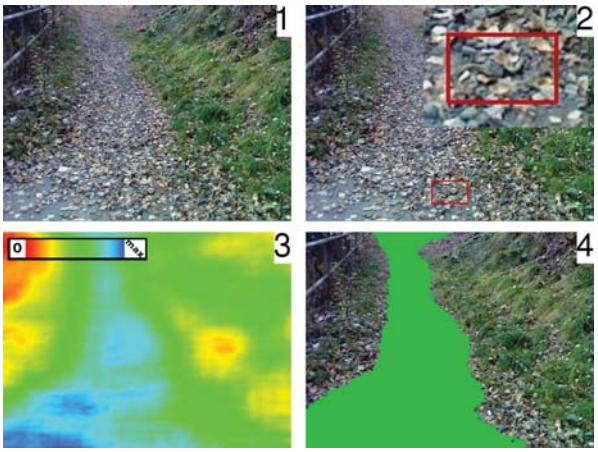

Figure 14: Template Matching:Input(1), Template(2), Template Matching Matrix(3), Result(4).

Region Growing

In a region growing procedure all neighboring points (starting from an initial point) whose colors are within a certain tolerance are marked. In our case, the initial point is in the middle of the road rectangle. As described above, the method sounds quite simple. The challenge is to determine the tolerance values. The easiest way to allow the whole color range within the street rectangle mostly misses. This is because even small disturbances, such as leaves, extend the color space too much. To solve this problem we calculated a histogram of the street rectangle and used only the values in the second and third quartile, as shown green highlighted in Figure 15.

Figure 15: Region Growing: Input with drawn road rectangle(left), histogram and quartile (right).

Before the region growing process, the interquartile range is checked against a threshold. If it is higher than 30, the image is rejected. Figure 16 shows the complete algorithm, which is very well working on normal roads. After the region growing, we eliminate holes in the resulting mask by walking through the lines and selecting the first and last pixel greater than zero.

Increase Saturation

Here we use, similar to the Equalize Saturation method, the difference in the RGB color space before and after a saturation increase. In the RGB color space, differences of the environment are higher than on the street. In contrast to the Equalize Histogram version, a factor for the saturation enhancement is determined in this method. This is done iterative by increasing the factor and comparing the average blue values in the road and environment rectangles until they differ clearly. We found 50 as a good working threshold.

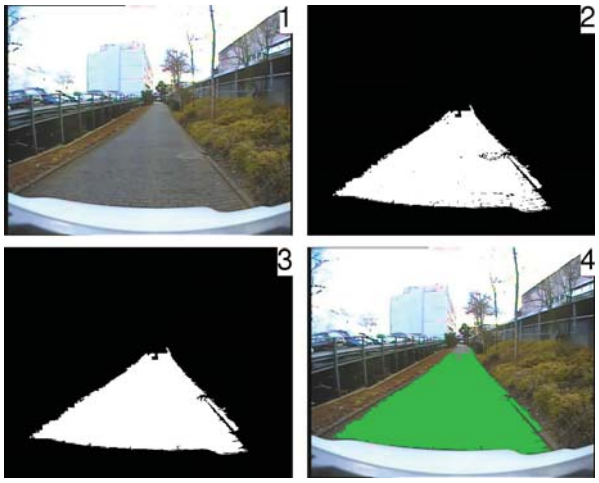

Figure 16: Region Growing: Input(1), Mask after Region Growing(2), Hole Elimination (3), Final Result (4).

In order to determine the bounds for the comparison, the average change in the road rectangle and the maximum change downwards and upwards between the original and higher saturated image are calculated. *road* means the road rectangle:

$$diffLow = min(I^{orig}_{x,y} - I^{highSat}_{x,y}), (x,y) \in road \quad (6)$$

$$diffHigh = max(I^{orig}_{x,y} - I^{highSat}_{x,y}), (x,y) \in road \quad (7)$$

$$meanDiff = \frac{1}{n}\sum(I^{orig}_{x,y} - I^{highSat}_{x,y}), (x,y) \in road \quad (8)$$

We found the following dynamic threshold values as a good choice for the bounds.

$$low = |(|meanDiff| - |diffLow|)| * 0.5 \quad (9)$$

$$high = |(|meanDiff| - |diffHigh|)| * 0.5 \quad (10)$$

Based on the lower and upper limits, the original image is compared with the higher saturated one. If the color values are within the limits, the point is marked as road, otherwise as environment. An Example is shown in Figure 17. To improve the result we only use parts connected to the road rectangle.

Simple Method Function

This method is a very simple and fast but good method for road detection. It is based on a comparison of all

ISBN 978-80-86943-40-4

Figure 17: Increase Saturation: Input(1), increased saturation(2), difference between 1 and 2 (3), Final Result (4).

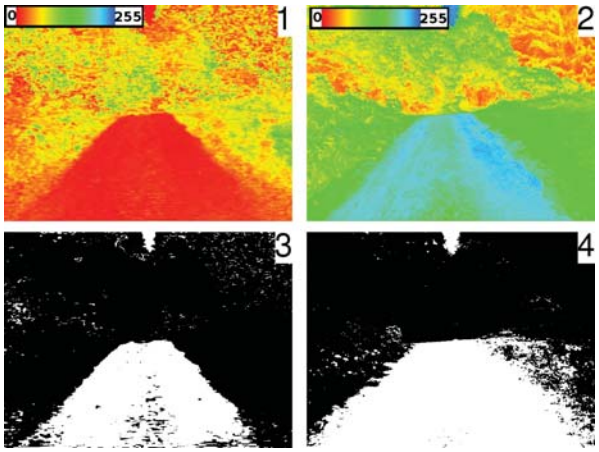

Figure 18: Simple Method Functions: Input saturation(1), Input Value(2), Result saturation (3), Result Value (4).

pixel values with a threshold. The first version is working in RGB color space with the blue channel. We calculate the minimum, maximum and mean blue values in the road rectangle of the image. Based on these values a threshold is determined as follows

$$thr_{blue} = \frac{max_{blue} + 3 * min_{blue} + mean_{blue}}{5} \quad (11)$$

Pixel values greater than this threshold will be selected as road, lower or equal as environment. An analogue function can be applied in HSV color space for the saturation and value channel. Here we use the average values of both environment rectangles for the threshold generation.

$$threshold_{sat} = \frac{max_{sat} + 2 * mean_{sat}}{3} \quad (12)$$

$$threshold_{val} = min_{val} \quad (13)$$

An example in the HSV-Color space is shown in Figure 18. Furthermore, all three simple functions perform a pre-test based on the comparison between road and environment rectangle to reject images.

Roadside Detector

The Roadside detector method searches the two roadsides to differentiate the road from the environment. Figure 19 shows the procedure. Beginning from a start point (center of the road rectangle), we follow scanlines and try to find the border. In order to detect it, we are going through each scanline and compare the mean color value in a rectangle with the color value at the starting point. This is shown for a single and for all scanlines in parts two and three of the Figure. The angles of the scanlines are calculated using the following angular step function. From the roadside points determined this way, a polygon is then assembled which indicates the road.

$$\alpha_{i+1} = \alpha_i + (10^\circ - |sin(2 * \alpha_i| * \alpha_{step}/2) \quad (14)$$

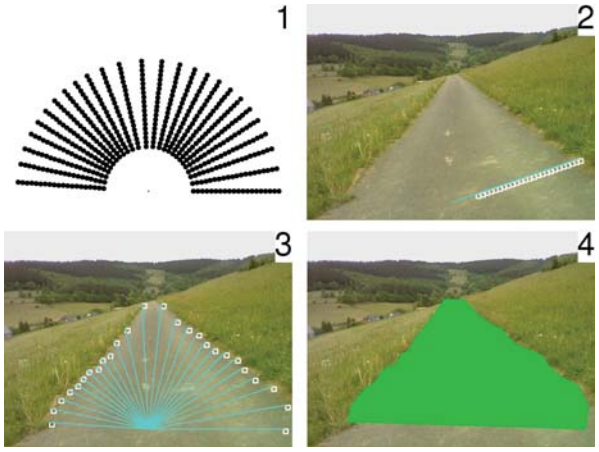

Figure 19: Roadside Detector: Scanline scheme(1), Single Scanline(2), Detected Roadside (3), Final Result (4).

4.2 Dirt Road with Tracks

Next to the previously mentioned roads, we developed special functions for the detection of dirt roads with tracks. At first, we developed a method to detect if tracks in a dirt road are present. The tracks are searched based on a horizontal baseline defined by the position of the road rectangle. This can be seen on the left in Figure 20. Rectangles are now formed on this line, within which the average color values (saturation and blue) are calculated and stored. Subsequently, starting from the middle point, the right and left rectangles are gradually compared with each other. If the saturation subsides, i.e. the green area in the middle has been exceeded and the color values of the rectangles match, these two determine the path tracks. In addition to the statement as to whether tracks are present, the function supplies one point each within the right and left track.

For the detection of dirt roads with tracks, we mostly adapted the previously used functions to this special

ISBN 978-80-86943-40-4

Figure 20: Find Tracks: Horizontal Scanline (1), Detected Tracks (2).

case. Thus, the basic idea is not new and we will only describe these new algorithms brief.

Follow Tracks

The core of the algorithm provides the functionality to follow a track from a given starting point. For this, we compare the color values inside rectangles (cells) as shown in Figure 21. The color values used are the mean values in the saturation, blue, green and red channel.

$$mean_{ch} = \frac{1}{n}\sum(I_{ch}(x,y)),\ (x,y) \in cell \qquad (15)$$

We try to find the best fitting cell in the next horizontal line. It starts with a cell drawn around the starting point and continues recursively until the color values differ too much (difference in two channels is more than 20). For the comparison, we calculate the following metric for each cell

$$diff^i = 2*mean^i_{sat} + 2*mean^i_{blue} + mean^i_{green} + mean^i_{red} \qquad (16)$$

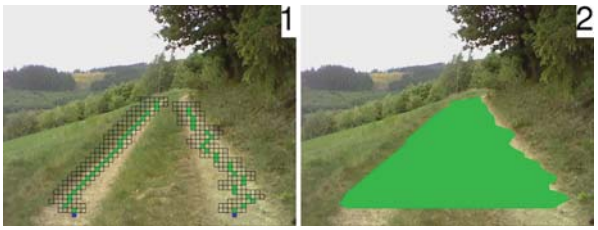

Figure 21: Follow Tracks: Detected Tracks(1), Final Result(2).

Other Track Detection Functions

All other track detection methods are adapted versions of already existing functions and will only be listed and explained within a few sentences. We extended the roadside detection for tracks by running the origin analysis method twice, once in the right and once in the left track. For the result, a polygon with respective outer points is drawn. In the case of the histogram back projection, the reference values can be calculated from rectangles in each of the two tracks instead of the road rectangle, as Figure 22 shows. With the help of the highest points in the individual tracks, a mask can now be generated which describes the road.

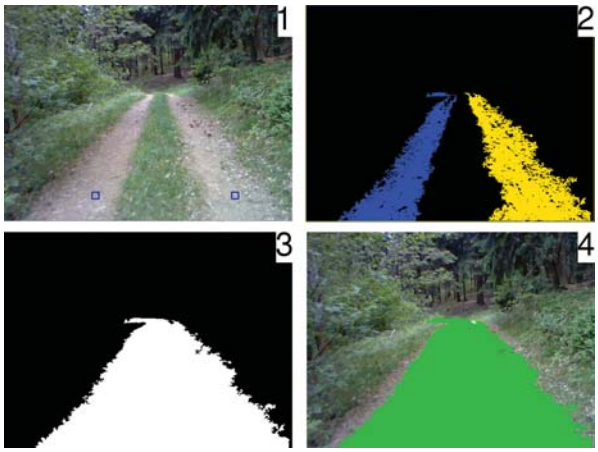

Figure 22: Histogram Back Projection Tracks: Detected Track Startpoints(1), Left and Right Tracks(2), connected mask(3), Final Result(4).

Analog to the Histogram Backprojection we adapted the template Matching and both Simple method functions with two seeding rectangles inside the tracks.

4.3 Evaluation Function

As already described in the last sections, some of the analysis functions detect in an additional pre-test whether the input image is suitable for the implemented method. Generally, we make a plausible test with the resulting mask for all algorithms. For this purpose, several conditions are checked and a scoring with a value between 0 and 100 is done. At first, we check the coverage of the rectangle in the mask. The road rectangle should be covered at least with 25 percent and the environment rectangles with a maximum of 20 percent. If one of these conditions fails, we devalue the score for this mask. The same if the whole mask (more than 95 percent) is covered.

4.4 Algorithm Selection

We made many tests with our dataset to generate a matrix, in which we assign the algorithms to the road and environment types. Some algorithms can be used with RGB or HSV input images, others only with single channels. Thus, the matrix in the following Figure 23 lists also the possible input data and provides as a result a lookup table for which method can be used in which conditions.

As can be seen in the matrix, a larger amount of analysis algorithms is available for the different road types. As already shown in the flow chart in Figure 4, the individual analysis algorithms are executed in parallel and the result masks are combined into a total result. A resulting image built with six algorithms is shown in Figure 24 with a simple implemented point structure for

ISBN 978-80-86943-40-4

	normal road	normal road, sat. Env	dirt road	dirt road sat. Env	dirt road sat. tracks
Simple Method	RGB	RGB,HSV	RGB,HSV	RGB,HSV	
Region Growing	RGB,B	B,S	RGB,B	B,S	
Road Border Detection	B	B,S	B	B,S	
Polygon Fitting	RGB	RGB	RGB	RGB	RGB
Hist. Back Projection	B	B	B	B,S	
Blur and Contours		RGB	RGB	RGB	RGB
Equalize Saturation		RGB		RGB	
Increase Saturation		RGB		RGB	
Track Algorithms					x

Figure 23: Analysis algorithm matrix.

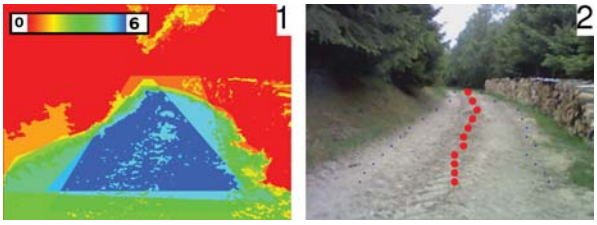

Figure 24: Total Result: pseudo-colored(1), path point structure(2).

the path description. The evaluation function returns a quality value after each analysis method. If the number of algorithms is restricted, only the best n methods of the previous run will be used. Since the rating is continuous, the used algorithms can change with each new input image. This ensures that only those methods are used that have proven themselves in past images. With the help of this option and the number of threads, it is now possible to adapt the complete software system to the processing power of the hardware.

In our software we describe the resulting road with a path-point structure, as shown in Figure 24 on the right. This structure is simply created on the total result mask of all analysis algorithms. To build the point structure, the algorithms walks, starting in the middle of the road rectangle, vertically and determines the width of the road in the total mask. Thus a path is generated where the points lie in the middle of the road.

5 EXPERIMENTAL RESULTS

In this section, we will verify the result of the analysis algorithms with the test dataset. In a first step, it makes sense to compare the resulting mask with the manually created ground truth mask. This comparison evaluates how many points are covered in the ground truth mask and how many additional ones are outside. The two values are given here in percent. So they reflect what percentage of the road was detected and what percentage of the environment was falsely classified as road.

The sole comparison of the mask values as described above would not be enough, because some algorithms (for example Simple Method Functions), usually generate road-pixels outside of the ground-truth mask. Since they occur only sporadically, these are not significant in the generation of the road point structure. Therefore, the main idea of how well the analysis worked can be best determined by the positions of the road points. So we compare the road structure generated on the ground truth mask with the one on the algorithm result mask and calculate the mean and maximum deviation (in pixels).

Our observation showed, that with an average difference of less than 85 pixels the result is useful and describes the path well. If the difference is higher, the result will be worse or unusable. If more road points are detected in one structure than in the other, we devaluate the result. In our evaluation we combined every analysis algorithm with every possible input data (RGB, HSV, red, blue, ...) and interpreted the results. Based on these informations we created the matrix in Figure 23.

In Figure 25 we show one plot as an example. Here it is the algorithm Histogram Back Projection with the blue channel as input data in a short 275 images sequence. At the beginning, the robot was heading towards a wall so that there was nothing to detect. The top plot shows, that during the drive in the later image sequence, the algorithm detects mostly more than 90 percent of the road (green graph). Nevertheless, there are also wrong detections outside the ground truth mask, but they are mostly low (blue graph). On the bottom of the Figure we compared the road point structure created from the algorithms result mask against the one from the ground truth mask. The mean deviation is drawn in red and the maximum (per point) in orange. Additionally there is a line at 85px. This represents the above mentioned border for useful path detections. Between image 200 and the end of the sequence the algorithm gets problems to detect the road. This can be seen by the graphs on the top. In most of these cases the mask evaluation function detected that the images are not suitable. Thus, we rated the deviation in the road point structure as -10, because our software was able to detect, that one function is not working and rejected the image instead of delivering wrong results.

All in all, using a variety of algorithms can cover a wide range of possible mobile robot operating areas. Roads are reliably recognized. Nevertheless, in some parts of the test data the detection unfortunately does not work satisfactorily with any of the algorithms. It is noteworthy, however, that the software detects this and thus does not provide any false results.

5.1 Runtime

An important goal in the implementation was the real-time capability of the software. We determined the runtime for each analysis algorithm on a single core with

ISBN 978-80-86943-40-4

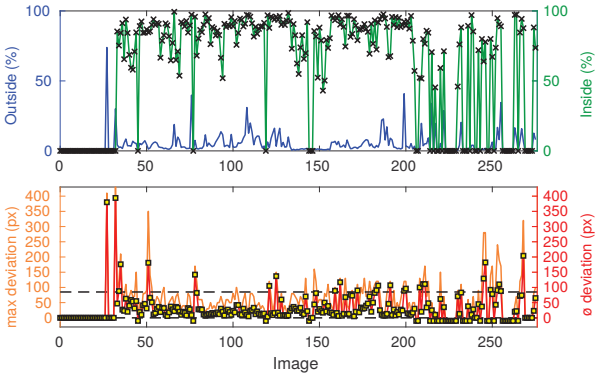

Figure 25: Evaluation of a single analysis algorithm.

the complete image set. The measurement was done on an AMD 635 X4 CPU and a 32-bit Linux operating system. Current high-end CPUs have about ten times the power. The runtimes of the individual algorithms are shown in Figure 26. The short runtime of the template matching is caused on the OpenCV multi-core implementation.

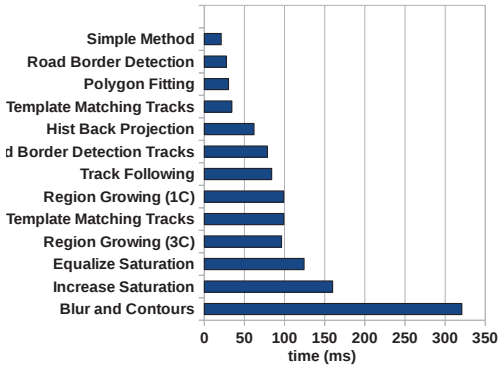

Figure 26: Runtimes of the analysis algorithms.

6 CONCLUSION

In this paper, we presented a software that can perform real-time path detection in different terrains based on monocular camera images using simple algorithms. The results have been obtained and validated with the help of a large test and ground truth dataset. Our software system can be adapted to the available computing resources by an intelligent scheduler, so that it can also be used on low power machines like ARM and no high-end machine is needed. Because of the required real-time capability, more complex approaches to path recognition were not pursued. In a further step, the auxiliary rectangles should be determined dynamically using 3D sensors. For example, a lidar scanner detects the drivable area in front of the vehicle with the spatial information. This road area is now used as a reference for the camera-based path recognition. With his extension, we plan to test the system for autonomous driving with our robot. In addition, the algorithms are to be tested with image data sets of other authors and compared with their results.

7 REFERENCES

[1] Y. Alon, A. Ferencz, and A. Shashua. Off-road path following using region classification and geometric projection constraints. *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 1:689–696, 2006.

[2] A. Angelova, L. Matthies, D. Helmick, and P. Perona. Fast terrain classification using variable-length representation for autonomous navigation. In *Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on*, pages 1 –8, june 2007.

[3] A. Broggi, C. Caraffi, S. Cattani, and R. Fedriga. A decision network based frame-work for visual off-road path detection problem. In *Intelligent Transportation Systems Conference, 2006. ITSC '06. IEEE*, pages 951 –956, 2006.

[4] A. Broggi, C. Caraffi, R. Fedriga, and P. Grisleri. Obstacle detection with stereo vision for off-road vehicle navigation. In *Computer Vision and Pattern Recognition - Workshops, 2005. CVPR Workshops. IEEE Computer Society Conference on*, page 65, june 2005.

[5] S. Cattani, P. Medici, and G. Vezzoni. Path detection system for autonomous off-road navigation.

[6] H. Kong, J.-Y. Audibert, and J. Ponce. General road detection from a single image. *Image Processing, IEEE Transactions on*, 19(8):2211 –2220, aug. 2010.

[7] R. Manduchi, A. Castano, A. Talukder, and L. Matthies. Obstacle detection and terrain classification for autonomous off-road navigation. *Auton. Robots*, 18(1):81–102, Jan. 2005.

[8] A. Nefian and G. Bradski. Detection of drivable corridors for off-road autonomous navigation. In *Image Processing, 2006 IEEE International Conference on*, pages 3025 –3028, oct. 2006.

[9] P. Vernaza, B. Taskar, and D. Lee. Online, self-supervised terrain classification via discriminatively trained submodular markov random fields. In *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*, pages 2750 –2757, may 2008.

ISBN 978-80-86943-40-4