UNIVERSITY OF GOTHENBURG

Indoor Navigation with Model Vehicles

Bachelor of Science Thesis in the Programme Software Engineering and Management Programme

ANNA ORAZOVA

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
Göteborg, Sweden, June 2013

Indoor Navigation with Model Vehicles

Anna Orazova

Supervisor: Christian Berger

University of Gothenburg
Chalmers University of Technology
Department of Computer Science and Engineering
SE-412 96 Göteborg
Sweden
Telephone + 46 (0)31-772 1000

Department of Computer Science and Engineering
Göteborg, Sweden, June 2013

Abstract

This paper describes a vision-based approach to miniature robot vehicle position estimation and navigation, that integrates data from on-board camera, tachometers and steering servo. The goal is to navigate a miniature vehicle on an indoor track using visual data from lane markings and dead reckoning from odometry. Then the estimated position of the miniature vehicle will be visualized in real-time and overlayed on a photo of the actual track. This will be used to evaluate how well this estimation performs and what its limitations are.

Acknowledgements

I would like to thank Benjamin Vedder for provided help and inspiration. Also for constructing the 1:24 model vehicle used in this experiment and providing dead reckoning and positioning functionalities for it. As well for provided applications for remote viewing of debug images.

I would also like to thank Christian Berger for supervising the project.

Anna Orazova - Monday 17th June, 2013

Glossary

Grayscale is an image in which the value of each pixel is a single sample, that is, it carries only intensity information. 8

RCM Ranging and Communications Module. 2, 3

RGB color model is an additive color model in which red, green, and blue light are added together in various ways to reproduce a broad array of colors. 8

Contents

1

Introduction

Positioning and navigation are essential problems for mobile robots. These problems get even more severe within the topic of autonomous vehicles where proper positioning and navigation are critical for normal functioning and safety.

In most cases expensive hardware is used to perform this task. This research is carried out to try to achieve comparable results by combining data from an onboard camera, tachometers and a steering servo. Since odometry data is provided by the vehicle's hardware and an onboard camera is relatively cheap and compact, this method is expected to decrease the costs, the power consumption and save more space onboard of a vehicle. By using this method the vehicle should be able to estimate its own position and navigate to next estimated position by detecting markings on the floor. The estimated position of the miniature vehicle will be visualized in real-time and overlayed on a photo of the actual track. This will be used to evaluate how well this estimation performs and what its limitations are.

In this research we are trying to implement a robust method for lane following and combine it with already implemented in Gulliver miniature vehicles[1] dead reckoning of the position of the vehicle.

The experiment will be set up on a miniature model vehicle, because a platform based on such vehicles can be used as a step between pure simulation of traffic scenarios and evaluation on full-scale vehicles [1]. Experiments on a miniature platform are likely to give more accurate results than a pure simulation, and require significantly less resources and costs than a full-scale vehicular platform.

[1]URL `www.chalmers.se/hosted/gulliver-en/`

1.1 Research questions

How robust will be a lane following algorithm based on techniques of cross-correlation and linear regression (Q1)? Is it possible to implement indoor localization of an autonomous miniature vehicle by combining functionalities of lane following and dead reckoning provided by only inexpensive onboard camera, tachometers and the steering servo (Q2)?

1.2 Goal of this Work

The goal of this project and intended contribution is to propose and implement a method for position estimating and navigating for an autonomous miniature vehicle:

- Implement robust method for line recognition.
- Implement perspective transformation of input image data to associate it with the vehicle's actual position.
- Integrate line recognition with image coordinates with already existing ”drive to position” functionality of the Gulliver miniature vehicles[2] designed by Benjamin Vedder[3] to perform lane following.
- Estimate position of the vehicle by combining implemeted in this study lane following algorithm with dead reckoning functionality of the Gulliver miniature vehicles.

1.3 Delimitations

Since to find the vehicle's absolute position data we would need to implement a more advanced system, it is almost impossible to examine the difference between the estimated data and the real vehicle's position. This can be also seen as a proposal for future research.

[2]URL `www.chalmers.se/hosted/gulliver-en/`
[3]URL `http://vedder.se/`

2

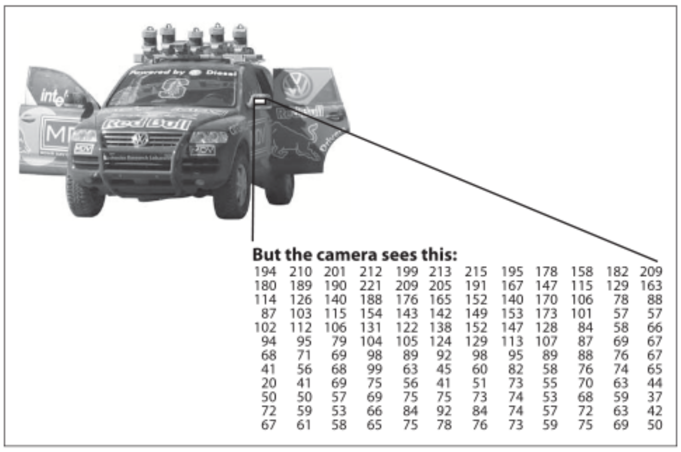
Theory and Related Studies

THIS CHAPTER explains the theories used in this paper.

2.1 Computer Vision

Computer vision is a field of methods for transformation of data from a photo or video source into numerical information to produce decisions or a new representation [4]. Different transformations are performed depending on the desired outcome. Examples of the input data may be video from a security camera or a photo of ocean surface. The decision might be ”there were 5 people passing the camera” or ”there is an abnormal big wave”. In our case the input data are photos from a camera located on top of a model vehicle and decision is the middle point of two lines in front of the vehicle. A new representation might mean removing camera motion from an image sequence [4] or cutting out part of the image. For this experiment we turn color image into grayscale image (3.3.1) and perform perspective transformation (3.3.1).

Because our brain is so advanced, it is easy for us to focus on different features of the images we see and make decisions using our previous experience. It is not as easy for a computer to perform on the same level because everything, that a computer ”sees” is just a grid of numbers (figure 2.1). Manipulating those numbers to achieve desired outcome is the basic idea of computer vision.


But the camera sees this:

194	210	201	212	199	213	215	195	178	158	182	209
180	189	190	221	209	205	191	167	147	115	129	163
114	126	140	188	176	165	152	140	170	106	78	88
87	103	115	154	143	142	149	153	173	101	57	57
102	112	106	131	122	138	152	147	128	84	58	66
94	95	79	104	105	124	129	113	107	87	69	67
68	71	69	98	89	92	98	95	89	88	76	67
41	56	68	99	63	45	60	82	58	76	74	65
20	41	69	75	56	41	51	73	55	70	63	44
50	50	57	69	75	75	73	74	53	68	59	37
72	59	53	66	84	92	84	74	57	72	63	42
67	61	58	65	75	78	76	73	59	75	69	50

Figure 2.1: Computer representation of a car's side mirror[4]

2.2 Model vehicles and Gulliver project

In this project we use model vehicle which is a modified radio controlled car. The modifications include removing radio control component, adding camera, onboard computer and motor controller. We use model vehicle as it is cheaper and easier to modify. Model vehicles are also closer to reality than pure simulation [1] and have the same level of inconsistencies (for example light noise, small errors in steering and driving systems) as real vehicles. The model vehicle used in this project is part of Gulliver project[1]. Gulliver is a platform for studying and testing vehicular systems, which proposes usage of miniature vehicles as a step between pure simulation and full size vehicle [1]. Numerous amount of miniature vehicles was built and tested within this project [2] and they show good results in usage and provide odometry data for dead reckoning.

2.2.1 Main hardware modifications

We chose ODROID-U2[2] because it is twice smaller in size and with its clock running at 1.7 Ghz it is more powerful, then most of other platforms of the same price group (for example PandaBoard[3] with only 1.2 Ghz). The size is especially important since it opens possibility of employment of model vehicles of smaller size.

To perform this experiment we chose PLAYSTATION Eye camera[4] because it runs 120 hertz at 320x240 pixels resolution. During experiments it was observed, that on more

[1]URL `http://www.chalmers.se/hosted/gulliver-en/`

[2]URL `http://www.hardkernel.com/renewal_2011/products/prdt_info.php?g_code=G135341370451&tab_idx=1`

[3]URL `http://en.wikipedia.org/wiki/PandaBoard`

[4]URL `http://en.wikipedia.org/wiki/PlayStation_Eye`

powerful machines or without computations the camera can run up to 166 hertz and about 66 hertz on the vehicle with all other algorithms running. It is valuable, because more frequent update rate gives more robust results for the lane following algorithm.

2.3 Navigation

Navigation is a process of controlling and monitoring the movement from one position to another in relation to a specific reference. As mentioned before navigation is one of the basic problems of mobile autonomous robots (1) as they should perform this task on their own. Number of different approaches has been presented in the past both based on various sensors [3, 5, 6, 7] and computer vision [8, 9]. One of the related projects is the localization project [3] for Gulliver cars, which was carried out to improve the vehicles' position estimation and make it work with large number of vehicles simultaneously. The method used in that project combined data from Ranging and Communications Modules (RCMs) and odometry. While providing sufficient results RCMs are quite expensive and take a lot of space onboard a vehicle, in fact RCMs are so big, they would not fit onboard of the vehicle used in this project.

2.3.1 Setting

To have navigation references we should start navigation in proper setting. The setting for this project is an indoor location with a track (path) for the vehicle to navigate through. The size of the track should be proportional to the size of the vehicle. The track should be mapped by two continuous lines on some contrast background.

2.4 Cross-Correlation

Cross-correlation is a standard approach for detecting features on an image by measuring similarity of two waveforms [10]. In this experiment it is performed by taking each row of the pixels' values of a picture matrix and comparing them to the filter vector. By adjusting the filter's shape and length in accordance to the width and the properties of the road's marking lines, we can then improve efficiency and performance of the line detection.

There are different ways of matching with correlation. For example, to measure the sum of squares of differences between the filter vector and the pixels' values or use normalized correlation [10]. In this paper we will experiment with various methods to distinguish the one which performs better in the given setting.

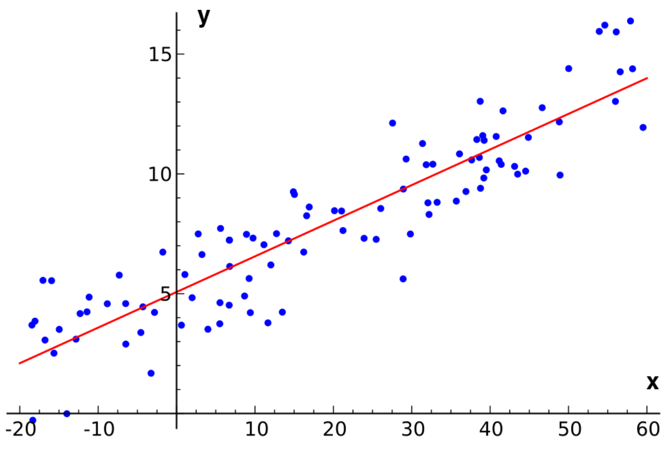


Figure 2.2: Linear regression[5]

2.5 Linear regression

Linear regression is an idea, that through each given set of points goes one straight line with optimal distance to each point (calculated on a perpendicular through the point to the line). Figure 2.2 illustrates linear regression.

That means, we can find such linear function, which describes that line and we can also find the points' correlation coefficient [11]. Correlation coefficient is a value from -1 to 1, where -1 and 1 mean, that the given points are perfectly aligned and values closer to 0 mean, that the points a poorly aligned. We can use correlation coefficient as a criterion for noise reduction (the reduction of the points, which are not on the line markings).

$$y = a \times x + b \tag{2.1}$$

If equation 2.1 describes the function of the best fitting line, then a determines the slope or the gradient of the line. By analyzing the slope of the line we can adjust the method to only work with lines, which have specific angle in relation to the image sensor's point of view. In our case we will use mostly vertical lines.

[5]Sewaqu, Linear regression.svg. URL `http://en.wikipedia.org/wiki/File:Linear_regression.svg`

3

Research Method

To answer our research questions we can not reflect only on already acquired knowledge. Since we want to try to implement a new technology in an existing field, we need to implement this technology and analyze its performance. Thus, we chose design research as research method for this study as it brings the design activity into focus at an intellectual level [12].

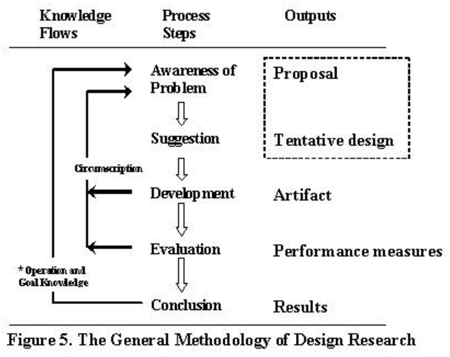


Figure 3.1: The general methodology of design research[1]

According to the figure 3.1 design research in this project follows the above mentioned steps:

[1]URL `http://desrist.org/design-research-in-information-systems/`

3.1 Awareness of problem.

This is the stage of compiling an idea of future research. The awareness of the problem came from participating in Carolo Cup competition[2], where we struggled with implementing proper lane following algorithm. The idea about combining lane following and position estimation algorithms comes from reflecting on the position estimation solutions of Gulliver project (2.3). The topic itself is important and interesting for mobile robots and autonomous vehicles (1). In this stage it is important to create criteria for future analysis and state preliminary research questions. In this study they are: which techniques can improve robustness of a lane following algorithm and is it possible to perform indoor navigation and position estimation of a model vehicle with reduced costs by combining lane following and dead reckoning functionalities?

3.2 Suggestion.

This is the step of suggesting new methods for old problems. After studying relevant literature and reflecting on already conducted studies (2), we propose a suggestion of a method, which is not yet implemented and which would combine in itself different promising techniques presented in the theory chapter (2). This way we come up with the final research questions (1.1), which are derived from preliminary research questions by acquiring more knowledge in the field. The problem domain resulting in the first research question indicates that we need to implement a lane following algorithm. The algorithm should only use data from onboard camera. It should be able to detect lines by employing cross-correlation (2.4) and filter false values using linear regression (2.5). The method should be able to supply the motor controller with proper position data information so that the vehicle can follow provided path. To answer second research question we need to combine already implemented in the motor controller dead reckoning functionality[3] with the lane following functionality resulting from the first research question. To perform with better results we should also find the most suitable hardware.

3.3 Development.

During this stage all suggested ideas are implemented. Precisely an application, which uses line detection with cross-correlation and linear regression (2) for detecting lines. And integrates lane following algorithm with dead reckoning data to localize the vehicle.

As dead reckoning functionality is already implemented in the motor controller software to perform the experiment it is only needed to find most suitable hardware, implement

[2]URL `http://www.chalmers.se/hosted/gulliver-en/carolo-cup-2012`
[3]URL `http://vedder.se/`

lane following functionality and integrate the next desired position data from lane following with the dead reckoning functionality.

3.3.1 Line recognition

This section will describe the procedures performed to archive line recognition.

Grayscale

To find lines on the track we will only be interested in their thickness and color intensity. That means, that we do not need to save the color information. So we convert each received RGB color model image into Grayscale image.

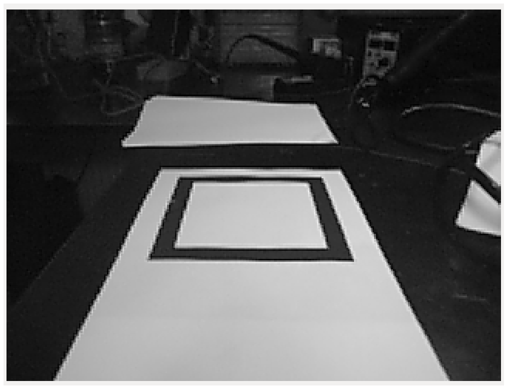


Figure 3.2: Sample image without perspective transform

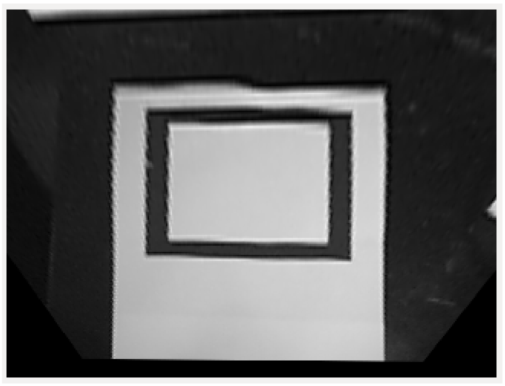


Figure 3.3: Sample image with perspective transform

Perspective transformation

Using cross-correlation (2.4) method to detect the lines we assume, that at each point the lines are of the same thickness (3.3.1). But since the camera is situated on top of the model vehicle and is facing forward, everything it sees, including the marking lines is in perspective. To fix this, we apply perspective transformation on the image. To find what are the proper values for the transformation we draw a square on the floor with size of a side equal to width of the model vehicle and point the vehicle's camera to it. Then, by calculating difference in pixels on X coordinate (horizontal) between upper and lower corners' we try to achieve the square shape to be same in our image. Figures 3.2 and 3.3 show the input image from the camera which was converted to Grayscale and the same image with perspective transform applied. By applying this transformation, we cut out a bit of the field of view the camera can have, but it still sees far enough to predict the vehicle's next move (the camera can see up to 650mm ahead). We use linear interpolation in this transform to save some computational power.

Detecting probable lines with cross-correlation

After the image is converted to Grayscale and the perspective transform (3.3.1) is done, we start to search for probable lines. As from now, we assume, that all the lines we are interested in have the same thickness. To apply cross-correlation (2.4) on the row we should distinct what is our filter vector. The length of the filter is determined by width of a marking line in pixels. For this experiment we use black electric tape, which is 1,77mm wide. On the camera image its width is about 9 pixels, that is why our filter has 9 positive values. The shape of the filter differs depending on what results we want to achieve. In this experiment we want to detect lines of specific thickness and not just areas with high color intensity. Thus the negative values on the sides of the filter are used for identifying a line rather, then just some dark (light) space on the image. When the filter is cross-correlated to a darker part of the image, the negative values multiplied to higher numbers are going to reduce the final correlation value in this area, while at the places with lighter sides and darker middle parts the correlation value should be higher. Figure 3.4 displays used filter size and shape.

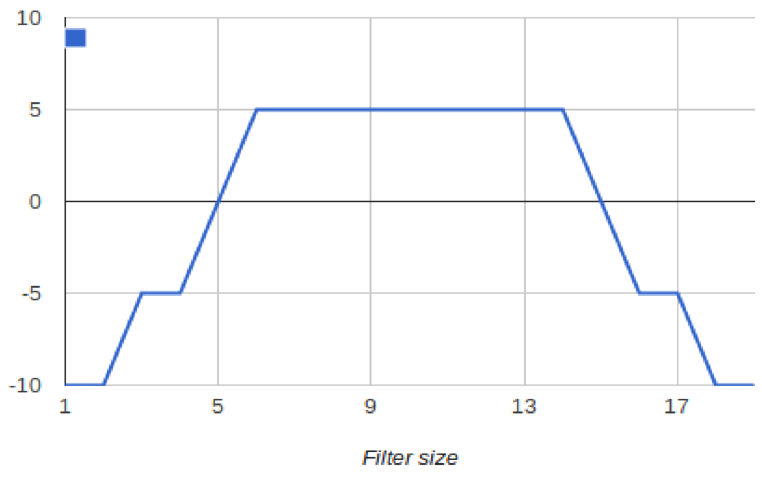


Figure 3.4: Plotted filter vector

There are different ways of matching with correlation. In this study we experimented with integral correlation and normalized correlation. Normalized correlation provided the best results.

The image is in Grayscale and the marking lines have contrast background, if we take each row of the image matrix and cross-correlate it to our filter vector, in the resulting graph, we should be looking for two peaks on the resulting graph if the background is darker, then the lines. If the lines are darker, than the background, we can subtract each pixel value from 255 (maximum pixel value as each pixel value is represented by unsigned char) to invert it. Once we have found two peaks, we can assume, that in the given row those are the marking lines. This way we go through the whole image matrix, analyzing each fifth row (experiment show, that this is enough for robust line detecting). Since each image we receive from the camera is 240 pixels in length and we analyze only each fifth row, in the end we get 48 pairs of probable line markings locations. Each pair is then saved to a structure with their x and y coordinates and the values at the peaks. We can also always track whether each value belongs to the right or to the left part of the image. Figure 3.5 shows an image with applied cross-correlation. Red and blue ovals mark the places, where the probable lines are found. Red ovals belong to the left line and blue ovals belong to the right line. The width of the ovals represents the size of the filter vector in pixels.

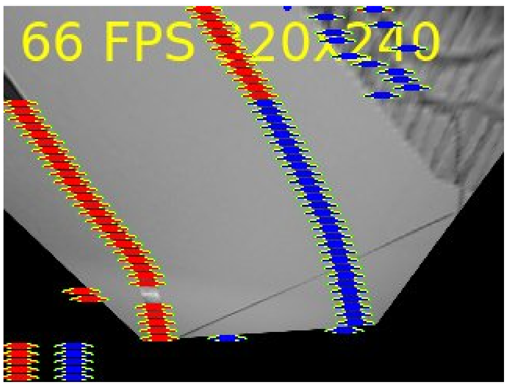


Figure 3.5: Sample image after finding probable lines with cross-correlation

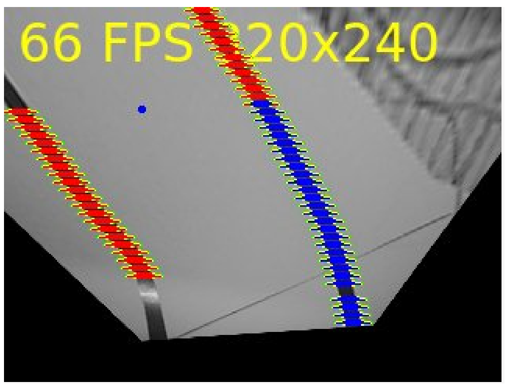


Figure 3.6: Sample image with reduced fault values with linear regression

Filtering noise with linear regression

Because after applying cross-correlation (3.3.1), we only look for two highest values, on each row we will always get two peak values whether there was a line or not. To determine which of those peaks are actually part of the lines, we are looking at them as right and left lines. All the left values are in the left line and all the right values are in the right line. Then each line is separated into smaller segments of 6 values on which we apply linear regression and calculate their correlation coefficient and the slope of the line (2.4). The lines we are looking for have correlation coefficients closer to 1 or -1 (from -1 to -0.8 and from 0.8 to 1), what means, that the points are aligned. But we also need to control the slope constant. If we want to find lines with the maximum slope of 45 degrees, then slope constant should be within -1.5 and 1.5.

We are analyzing each line segment by taking out point by point and comparing the correlation coefficients retrieved. This way we are trying to find a point, which is least aligned to the rest of the points of the given segment. Here we want to find the points, which are out of the line, to remove them. If total amount of points left in the segment is 4 and they are still not aligned, we can dismiss the whole segment. After applying this algorithm, the only points left in the segment are those, which can be aligned into lines. Figures 3.5 and 3.6 display the effect of applying noise reduction.

3.3.2 Navigating the model vehicle

Estimating next desired position

After we had found two lines, we can estimate, where is the vehicle's next desired position. To do this we find a pair of points closest to the top of the image. Our next desired position should be in the middle of the distance between those points. The blue dot between the lines on Fig. 3.6 is the next desired position. Now we calculate x and y coordinates for this position in the coordinate system of the image (its pixel address).

Conversion of the image coordinates into vehicle's coordinates

Since we use OpenCV[4] for computer vision libraries, the image coordinate system starts from left upper corner of the image and y coordinate is inversed. Each image is $320x240$ pixels, that means, that the maximum values for x and y are 320 and 240 relatively. The vehicle's positioning system on the other hand has different coordinate system (fig. 3.8), where center is in between vehicle's rear wheels and y coordinate is not inversed. To convert the pixel's coordinates to the vehicle's coordinates, we take pixel coordinates from the image's upper corners and mark them on the actual ground plane (where camera sees them). Then we can measure the actual distances between the vehicle's rear wheels and the image's corners to find ratio between those distances. The ratio is then used to convert pixel's coordinates (Fig. 3.7) to vehicle's position coordinates (Fig. 3.8).

Driving

Once the coordinates for vehicle's next position are found, we can send them via packet interface to the vehicle's motor controller. The motor controller computes steering values for the vehicle to arrive to the set position. The packet interface and motor controller functionality were provided by Benjamin Vedder[5].

[4]URL `http://opencv.org/`
[5]URL `http://vedder.se/`

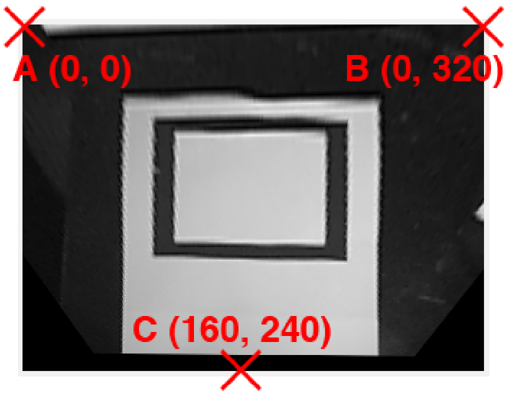


Figure 3.7: Reference points with image coordinates

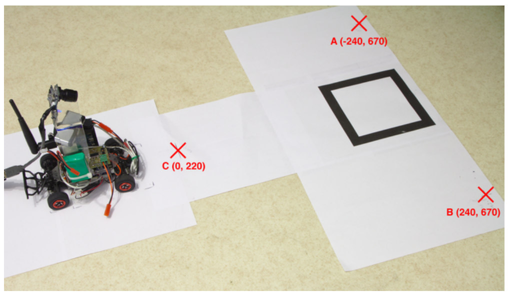


Figure 3.8: Reference points with model vehicle coordinates

3.4 Evaluation.

In this step we evaluate resulting application and its performance (4). Deviations from expectations are documented and explained. Also we produce hypothesis about behavior of the resulting artifact and discuss whether it behaves as we expected or not. Since it is design research, final results are rarely ”final” and can be used as basis for further research. In this phase we will produce quantitative analysis of how the validity threats affect the results. Following the research questions (1.1) we need to evaluate robustness of the implemented lane following algorithm and precision of position estimation. There are no internal threats to validity coming from the vehicle as we always use the same vehicle with the same hardware, software and fully charged battery. The internal validity threats are the experimenter bias, and it is present in validation of the position estimation results. We can not validate precision of the position estimation just by looking at the running vehicle and the picture of estimated positions. To compare real driven path with the estimated path we will attach a marker to the vehicle and run it. Then we can compare the drawn driven path to the driven path estimated by dead reckoning in the motor controller. Part of external validity threats come from the situation as different lightning and shape of the testing track can be perceived differently by the camera. That is why we need to test the algorithm with different light conditions to evaluate till what extend the algorithm can handle light change. We also need to test the algorithm with different track faults (such as missing lines and size change) to understand if the study results can be generalized for different settings.

3.5 Conclusion.

This is the final phase of the research. Here we are trying to categorize firm knowledge gained in the research, compare the results to previously conducted studies and also provide subjects for further research.

4

Analysis of Outcome

Since there are two research questions (1.1), validation of navigation (lane following algorithm 4.1) and position estimation (4.2) should be done separately. We will go through and analyze three threats of validity of lane following algorithm and one threat of validity of position estimation algorithm.

4.1 Evaluation of the Lane Following algorithm

To evaluate performance and robustness of lane following algorithm and provide statistical data for it, we are performing visual observations while the vehicle is following the track lines. The main criteria for the lane following algorithm is that the vehicle should always stay within the marking lines. Changing different external variables (such as amount of light, width of the lane and consistency of the marking lines), we can calculate how many times the vehicle lost the track. For each condition we will test the vehicle with 10 laps. Since the vehicle is small in size, its battery is also small. So increasing testing time (10 laps is about 2-3 minutes depending on how good the vehicle performs) increases charging time and danger of battery failure.

To perform the tests, each time the vehicle will be placed to the same start position. If the vehicle gets out of lane markings while running, it will be placed on the start position again and the lap will be evaluated as failed.

4.1.1 Light conditions

It is important to test the algorithm with various light conditions to be aware of its level of robustness and to which extend findings of this study can be generalized in different settings. The testing is done by running the algorithm with different kinds of lightning. Of course it is impossible for us to test the algorithm with every possible light condition and due to lack of equipment it is impossible to calculate the amount of light on the track, so we analyze the results in relation to amount of light the sources provide. As the vehicle is only used indoors, we compare its functioning in different kinds of indoor light. So we chose three main scenarios: cloudy daylight (about $1000lm$), usual indoor light ($600lm$), deemed indoor light ($400lm$). It is also important to test how the algorithm responds on mixed light situations: shadows and sudden bright spot lights. So we test each scenario with creating shadows and adding bright spot light (electric torch of $500lm$).

The system was tested with various light conditions and performed equally good with most of them (see table 4.1). Though it does not function with bright straight light (for example bright electric torch) pointing at a part of the track, which makes the marking lines overexposed (comparing to the rest of the track) and thus not recognizable for the camera. The system would also give poor results in dark rooms where amount of light is not sufficient. On the other hand it is valuable, that the algorithm handles shadows well as those are most common part of the light change in the real traffic scenario.

Results presented in table 4.1 show, that the implemented lane following algorithm performs well in different light conditions without any previous (light) adjustments and in 70% of the tested situations percentage of failure is below 50%. That is an enormous improvement comparing to Carolo Cup[1] vehicles, which were very sensitive to different light conditions and needed adjustments for each different light situation. We achieve these results by employing cross-correlation technique in line detection and applying proper filtering.

4.1.2 Missing line markings

It is important for lane following algorithm to be robust. To achieve robustness we should assume such situation as missing line markings (whether due to faults of the track or faults of the camera). We perform this test by covering part of the track with white paper (to change the shape of the track - the external variable) and analyzing the vehicle's behavior. For each test we use different sizes of the cover. It is important to test handling of missing lines on straight parts of the track and curves to detect system's weak points. Also it is valuable to see if there is any difference in handling left missing lines or right missing lines.

We perform variety of experiments with different lengths of missing line spaces. To decrease size of the result table, we group the results by three different lengths of missing marking spaces: less than $11cm$, between $11cm$ and $17cm$, more than $17cm$ (see table 4.2). During experiments it was also figured, that position of the missing line (left or

[1]URL `http://www.chalmers.se/hosted/gulliver-en/carolo-cup-2012`

Table 4.1: Relation of different light conditions to the performance of lane following algorithm

Type of light	Times out of track	Failure percentage
Day light (from windows) or bright day light lamps	0	0
Day light or bright day light lamps + bright spot light on parts of track	0	0
Day light or bright day light lamps + shadows on parts of the track	0	0
600lm lamp	0	0
600lm lamp + bright spot-light on parts of the track	4	40
600lm lamp + shadows on parts of the track	0	0
400lm lamp	3	30
400lm lamp + bright spot light on parts of the track	9	90
400lm lamp + shadows on parts of the track	5	50
No light in the room (only lights from onboard of the vehicle)	10	100

right line) does not play any distinct role (in more, than 80% of the cases the results were the same no matter left or right line was missing).

Analyzing results of table 4.2, it is visible, that the implemented lane following algorithm handles very well (less, than 50% failure) missing lines up to about 17cm. The reason why longer missing lines are not handled as good is the way we compensate for camera's narrow field of view. Every time the camera sees only one line, it sends the vehicle to opposite direction in order to find the second line. While shorter parts of missing lines get passed by vehicle quite quick and do not interfere with the decisions making, the longer missing lines are recognized by the vehicle as a reason to move in an opposite direction. For improving the vehicle performance in handling missing lines, some new algorithm for compensating should be found.

Table 4.2: Relation of size and placement of missing lines to the performance of lane following algorithm

Size and placement of missing line	Times out of track	Failure percentage
Less, than 11cm, straight	0	0
More, than 11cm, but less, than 17cm, straight	3	30
More, than 17cm, straight	8	80
Less, than 11cm, curve	0	0
More, than 11cm, but less, than 17cm, curve	4	40
More, than 17cm, curve	10	100

Table 4.3: Relation of the track width to the performance of lane following algorithm

Width of the lane	Times out of track	Failure percentage
From 16cm till 22cm	0	0
From 22cm till 27cm	6	60
From 27cm	10	100

4.1.3 Width of the track

During various testing it was figured, that vehicle performance is related to the width of the track. To evaluate if this statement is true and to which degree the width affects the performance, we test the vehicle on a track, where about $1/4th$ of it has different width. This way we are also able to evaluate how good the algorithm can handle change of the lane's width (and to provide information on which sizes of tracks this study can be generalized). To perform this experiment we run the vehicle on a track, where $1/4th$ of it was changed in size (the rest of the track will stay 20cm wide). The width of the vehicle is 10cm, so the minimum width for the track to start testing is about 16cm (since we evaluate the performance by visually observing and calculating how many times the vehicle is out of the track lines, we need to have some space in the lane to let the vehicle drive) and we conduct experiments increasing the width of the specified part of the track by 2cm at a time. The experiments show, that the algorithm can only handle width of the track of maximum around 27cm, so to decrease size of the table we separate the results in three groups: from 16cm and till 22cm, from 22cm till 27cm, from 27cm.

Table 4.3 shows relation of the width of the track to lane following algorithm's performance. It is visible, that as width grows, the performance gets worse. Furthermore, those only 4 times out of 10, when the vehicle was able to finish the lap, it was just a matter of luck from which angle it came to the wide track area and if the angle of camera view was enough to record both lines or not. That means, that width of the track criterion is only related to the angle of the camera and the wider the camera angle, the wider tracks the vehicle is able to follow.

4.2 Evaluation of Position Estimation algorithm

Since in this paper we are trying to implement a method for position estimation, it is important to evaluate its precision as with low precision we do not achieve desired goal. This evaluation is done by plotting the position data from the vehicle while it is running on the track. The position data is provided by dead reckoning in the motor controller and it is plotted on a photo of the actual track. The dead reckoning functionality is implemented by Benjamin Vedder[2]. By driving number of laps on the track and plotting the vehicle's estimated positions we can visually observe and evaluate the system's performance. Furthermore we can make the vehicle leave a trace on the actual track and compare it to the trace the vehicle estimated.

4.2.1 Perspective transformation of the photo of the track

Since the photo of the track was not made from a point above, it has perspective distortion. To withdraw this distortion and properly scale the photo to the plotting, we marked five points on the track as reference coordinates and measured distances between them. After perspective transformation of the photo, the distance between the points on it should be relative to the distance between those points in the reality. Fig. 4.1 shows the photo of the track after perspective transformation.

4.2.2 Precision of Position Estimation

It is not possible to evaluate precision of the position estimation just by looking on the running vehicle and its estimated path. To compare the reckoned trace to the vehicle's actual trace on the track, we mount a marker on the rear wheels of the vehicle and record its actual path on the track. Then we make a photo of it and apply perspective transformation, just like with the photo of the track. Then we can detect the actual trace the vehicle left and overlay it with the estimated path (fig. 4.2). After that by calculating difference in distance between those paths at each $5cm$ through the whole track we will get a graph showing relation between difference of the paths in cm and the distance the vehicle drove (fig. 4.3). Fig. 4.3 shows, that after one lap the position

[2]URL `http://vedder.se/`

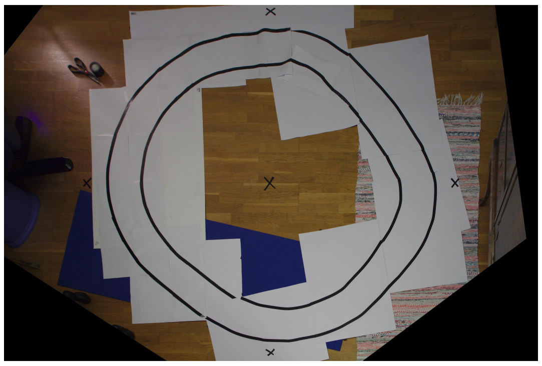


Figure 4.1: Photo of the track after perspective transformation

estimation (cayenne line) is really precise and differs from actual driven path (magenta line) only up to $4cm$. 72.38% of the measured difference values are under $2cm$, which means that more than half of the time precision was within $2cm$. This can be evaluated as a good result for positioning. But this is only one lap and to get the full picture we need to evaluate position precision through time (4.2.3). Reflecting on the result of performed experiment we can see, that two traces follow each other closely, but the trace from the actual vehicle is more wavy. Our suggestion is that the difference in those two traces appears due to the real vehicle's load (the hardware onboard of the vehicle) and the track material resistance to vehicle's wheels.

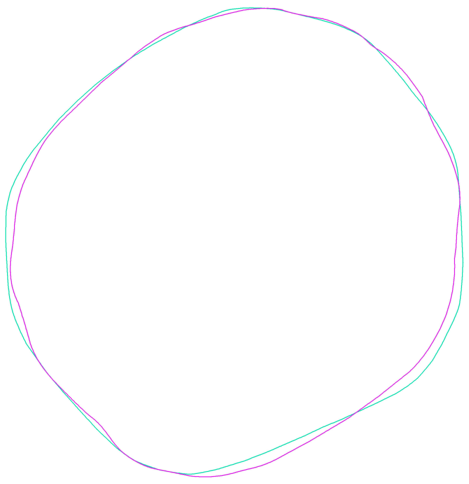


Figure 4.2: Overlay of the actual driven path (magenta) and the estimated driven path (cayenne)

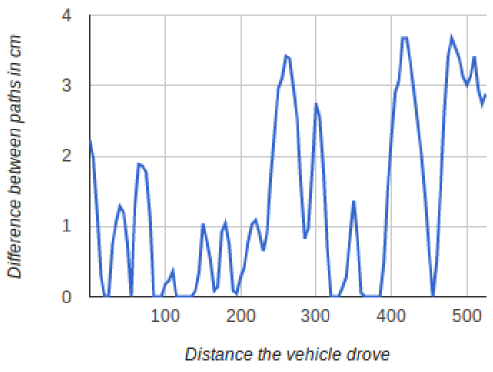


Figure 4.3: Difference between driven paths (estimated and real)

4.2.3 Position estimation drift with time

After analyzing the plotting of one finished lap (Fig. 4.4, sec. 4.2.2) we can see, that the vehicle follows the track closely. Also position estimation works with great precision. As the vehicle finishes three laps (Fig. 4.5) and six laps (Fig. 4.6), the amount error is growing. And though we can not quantify the amount of error, we can visually observe how the estimated position data drifts over time due to the fact that all the small errors such as servo non-linearities, wheel slip etc. add up, so the more the vehicle drives, the more off the position data is going to be. That is why one big improvement for the system would be increased amount of information sources and their synchronization. Furthermore the implemented position estimation algorithm while performing with good results on short periods of time (4.2.2) can not be used for precise position estimation on longer periods of time without significant improvements of the system.

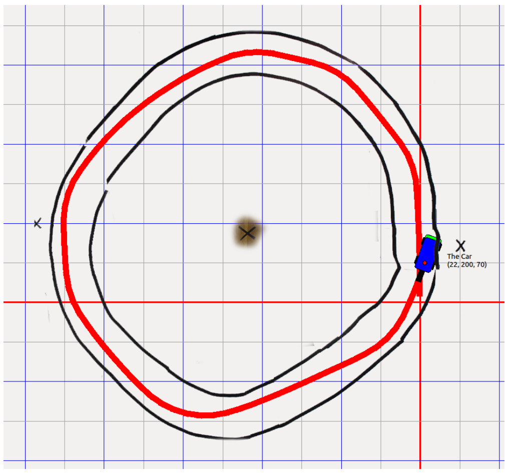


Figure 4.4: Trace of 1 driven lap overlaid on the track photo

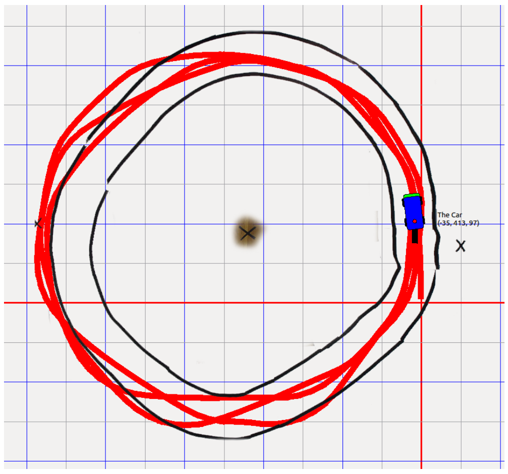


Figure 4.5: Trace of 3 driven laps overlaid on the track photo

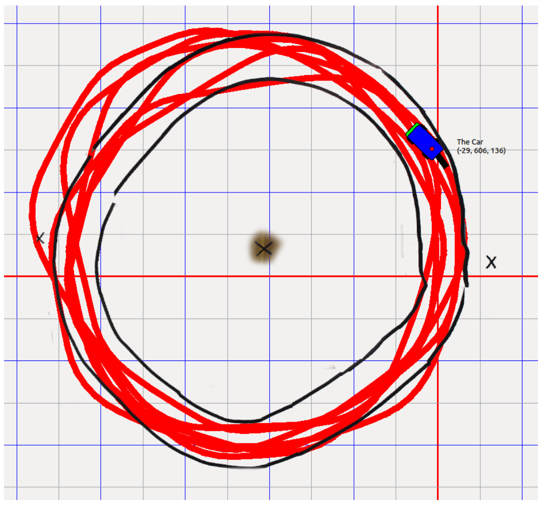


Figure 4.6: Trace of 6 driven laps overlaid on the track photo

5

Conclusions

In this paper we had presented a method for indoor navigation and position estimation of a mobile vehicle. The method uses computer vision, lane following and dead reckoning. The algorithm implemented for lane following and navigation shows high level of robustness and good results in various settings (4.1). Looking at the analysis of outcomes of localization algorithm (4.2) we can say, that the method can yet not be fully employed for intended purposes. But since the method shows good precision in estimating positions in short periods of time, we believe, that with more information sources and reference points this method can be applied for indoor position estimation of model vehicles in future. Thus, by improving and employing this method in model vehicle's we will not only get robust lane following functionality but also positioning system. This method is cheaper, than advanced positioning systems and has smaller size of hardware equipment.

Since there are no scientific papers describing lane following algorithm combining cross-correlation and linear regression, it is hard to compare perceived in this study knowledge with already existing studies. On the other side there is a lot of research about localization methods where computer vision is used together with odometry data [8, 13, 14]. Most of them use expensive omni-directional cameras and surroundings as reference points to reduce the odometry drift. This research extends known status quo by suggesting new method for robust lane following and implementing similar localization system as localization systems for mobile robots mentioned in the previous studies on a miniature model vehicle with inexpensive hardware. And though this research failed to provide precise positioning of the model vehicle on longer periods of time, it can be seen as the first step towards inexpensive and robust localization for indoor mobile vehicles.

5.1 Suggested improvements for the implemented system

- One of the most important properties for line recognition is the filter vector's size and shape. The filter vector can be generated significantly better by analyzing pixel values of a line sample from the debug image.
- Another improvement in line recognition would be further experimentations with different methods for matching filter vector with pixel values.
- The camera lens right now does not provide wide angle of view. This limits the distance we can have between the marking lines, because from time to time both of the lines do not get into the camera view. Right now maximum distance between marking lines for proper functioning is about $21cm$, but with wide angle lens, the vehicle will perform better on wider tracks.

5.2 Suggested future work

Ideas for further research based on the work in this experiment:

- Real-time visualization shows how the position data drifts over time (4.2.3). The smaller the drift, the closer we are to absolute positioning system. To reduce the drift the system needs more information sources and reference points. It would be valuable improvement to conceptualize new information sources and their synchronization with the system.
- To bring the system closer to real full size vehicle, it should be able to operate on usual roads. In this experiment the marking lines were represented as two continuous lines, while in reality the lines are not always continuous and camera's field of view can cover three and more lines.
- It would be interesting to validate the results with more advanced positioning system.

Bibliography

[1] M. Pahlavan, M. Papatriantafilou, E. M. Schiller, Gulliver: a test-bed for developing, demonstrating and prototyping vehicular systems, in: Proceedings of the 9th ACM international symposium on Mobility management and wireless access, MobiWac '11, ACM, New York, NY, USA, 2011, pp. 1–8.
URL `http://doi.acm.org/10.1145/2069131.2069133`

[2] B. Vedder, Gulliver: Design and implementation of a miniature vehicular system, Master's thesis, Chalmers University of Technology (2012).

[3] A. Altby, T. Boström, T. Sibgatullin, K. Stjärne, Robusta och noggranna positioneringssystem baserade på avståndsmätningar mellan mobila noder, Tech. rep., Chalmers University of Technology (2012).

[4] G. Bradski, A. Kaehler, Learning OpenCV, O'Reilly Media Inc., 2008.
URL `http://oreilly.com/catalog/9780596516130`

[5] M. Kam, X. Zhu, P. Kalata, Sensor fusion for mobile robot navigation, in: proceedings of the IEEE, Vol. 85, 1997.

[6] P. Hiemstra, A. Nederveen, Monte carlo localization.

[7] J. H. Connell, Sss: A hybrid architecture applied to robot navigation, in: Proceedings of the 1992 IEEE Conference on Robotics and Automation (ICRA-92), pp. 2719-2724., pp. 2719–2724.

[8] H. B. Jurgen Wolf, Wolfram Burgard, Robust vision-based localization by combining an image retrieval system with monte carlo localization, in: IEEE Transactions on Robotics.

[9] H. H. Claudio Facchinetti, François Tièche, Self-positioning robot navigation using ceiling image sequences, in: Proceeding ACCV'95 (Asian Conference on Computer Vision) 3, pp. 814–818.

[10] D. Jacobs, Correlation and convolution, Class Notes for CMSC 426.

[11] S. Chatterjee, A. S. Hadi, Regression Analysis by Example, Fourth Edition, John Wiley and Sons, Inc, 2006.

[12] H. Simon, The Sciences of the Artificial, Third Edition, Cambridge, MA, MIT Press.

[13] C.-C. Wang, C. Thorpe, Simultaneous localization and mapping with detection and tracking of moving objects.

[14] F. Chenavier, J. L. Crowley, Position estimation for a mobile robot using vision and odometry, in: Proceedings of the 1992 IEEE International Conference on Robotics and Automation.

List of Figures

List of Tables