Autonomous driving solely dependent on camera input and classical computer vision methods

Arne Smit

***Abstract*—Autonomous driving is expected to be the next revolution in transportation. In recognition of this, the RDW each year hosts a self-driving challenge. Within the context of this contest, this thesis focuses on how autonomous driving can be achieved solely dependent on camera input and classical computer vision methods. The focus of the thesis is placed on lane detection, vanishing point based control, real time image stitching and optical flow odometry. Lane detection showed great results for multiple scenarios. Vanishing point based control worked really well as long as both road sidelines are visible. Camera stitching was unfortunately not achieved in real time due to poor environmental keypoints and unstable cameras, but only when manually calibrated on a video frame. Optical flow based odometry shows decent results but is likely out competed by IMU units and GPS.**

I. Introduction

A. Social relevance

A little over a hundred years ago horse carriages filled the streets of cities in the western world. Since then, cars have become the primary mode of human transportation. Cars have seen a lot of improvements over the years making them faster, safer, and more comfortable. The automotive industry is now working on the next step: autonomous driving. Autonomous vehicles have many advantages. because they are more efficient, safer, have positive economic benefits and give freedom to those who can't currently drive by themselves. They are more efficient due to shorter reaction times and communication between vehicles which reduces congestion. They are safer because human errors due to distractions or fatigue are eliminated. Both these previous factors have economic benefits due to less congestion, damage and delays from accidents. Moreover, because people don't have to drive themselves they are able to be more productive as well. Furthermore autonomous vehicles give independence to those who are currently considered unsuitable to drive, such as the disabled. [2]. For these reasons, fully autonomous vehicles are expected to be the future of human transportation [1] [3].

B. Scientific relevance

Although lane detection has been researched for many years already, there is still a lot that has to be explored. Even though model based detection approaches are very popular nowadays, feature based approaches still have their place. Model based lane detection algorithms are often black box models making it more difficult to understand what's going on. Furthermore, in many applications in robotic limited hardware is available or expensive options are not desirable due to the production scale of a product.

C. RDW selfdriving challenge

The RDW is an independent Dutch governmental organisation that is responsible for the national registration of motorised vehicles. Since they recognize the importance of developing autonomous vehicles, they each year host the RDW self driving challenge. This is a competition for educational institutions to complete a race track as quickly as possible using a self-driving kart. In the 2024 edition there are two categories: the closed and open category. The open category has the freedom to design and build their own car within the rules of the RDW. Closed category participants all share the same car provided by the RDW and cannot make any hardware adjustments. For this research only the closed category will be considered. This means that the hardware is pre-determined by the RDW and outside the scope of this paper.

D. Research Scope

There are several factors that narrow down the scope of the research. First of all, due to the challenge task division, any tasks involving object detection and lidar involvement are not considered here. This mainly involves the recognition of traffic signs, traffic lights and other objects like the pedestrian, the car that needs to be overtaken and the barrier around the parking spot. Furthermore, the closed category of the challenge does not allow for any adjustments to be made to the car. Therefore, the entire hardware of the car is outside the scope of this research as well. Lastly, the time and weather conditions are limited to dry daytime weather. A lot of electronics are placed uncovered on the outside of the car, it is consequently not possible to drive during rain or in whet conditions. Regarding the daytime limit, all testing and recording time slots were during the day, so no nighttime footage could be recorded.

E. Research Questions

Based on everything the following research question was decided: "How can autonomous driving be achieved solely dependent on camera input and classical computer vision methods?"

This can be further subdivided into several questions.

- How can optical flow be utilized to track the movement of an autonomous vehicle and construct a road map?
- How can lane detection be achieved only using cameras and classical computer vision methods?
- How can real-time camera stitching be realized with unstable cameras?
- How can road markings be used to find the vanishing point and control the vehicle?

F. Report structure

The report is structured as follow. Section 2 contains related work of all relevant topics to the research. Section 3 includes the methods chosen and experiments conducted. The results of the experiments are then displayed in section 4. Section 5 interprets and discusses the results found and finally section 6 is the conclusion, where the research questions will be answered.

II. Related Work

Reda et al (2024) divide the Autonomous Driving System into six stages: sensors, perception, localization, assessment, path planning, and control [14]. These are the six layers required to achieve autonomous driving. The sensor stage is just measuring the environment. In this case this is mainly done through the three cameras, but also by the other included sensors. perception is about interpreting this data. This includes lane detection. localization is about determining the cars position. Vehicle tracking does this relative to the staring point. assessment is about risk calculation. This step is not considered in this research. Path planning concerns generating a path from the current position of the car to the desired location. Finally, control ensures the car stays on this desired path.

A. Lane Detection

For a vehicle, to be able to drive autonomously, based on a camera input, it should be able to distinguish the road its driving on. This is called road lane detection or lane detection. Because lane detection is a fundamental step in autonomous driving, there are many papers published on lane detection. They can generally be subdivided into two categories: feature based lane detection and deep learning, or model based lane detection [6] [7]. As the name suggests deep learning based methods are based on deep learning and use neural networks to find the road lanes. An overview of deep learning based methods can be found at Zao et al (2020) [7]. Feature based or traditional lane detection relies on classical computer vision methods, which use mathematical techniques to detect road markings as geometric shapes. Traditional approaches typically consist of two steps. The first step takes an image and after a couple of prepossessing steps, implements some form of edge detection, typically canny edge detection is used. The second step takes the edges and applies a line fitting algorithm to find the lane lines [7] [8]. There are several ways of implementing line fitting. Muthalagu et al (2020) present a minimalistic approach that uses Hough transform and linear regression for line fitting [20]. Hu et al (2010) use guided RANSAC to find the road markings [21]. Jiang et al (2011) first change perspective to a birds eye view, before applying edge detection and line fitting in a top perspective [19]. Sometimes a tracking step is included to use information of a previous frame to find the lines in a consecutive frame. Bottazzi et al (2014) for example use Lucas Kanade tracking to track the lane lines [18].

Traditional approaches have several advantages in comparison to deep learning based lane detection. First of all they are very accurate in detecting geometric features in a clear environment, which is very beneficial when looking for road markings. Secondly, Traditional methods are computationally less expensive, making them run more efficient with limited hardware options. In addition to this no dataset and model training is required. Lastly feature based lane detection is easier to understand than deep learning based approaches. Deep learning often relies on a black box model where it is difficult to understand why decisions are made. As a result deep learning based models are harder to understand, explain or debug. Deep learning based approaches typically excel in their versatility, adaptability and ability to perform when lanes are less visible. Traditional approaches are typically tuned to specific circumstances and their accuracy declines when applied in a different environment or with less visibility.

B. Vanishing point

A vanishing point is a point where all lines that are parallel in the real world, intersect in an image. The vanishing point is always located on the horizon. On a road with no elevation, all road lane lines intersect in the vanishing point. For this reason the vanishing point can be very useful when trying to find road lanes. T. Youjin et al use the vanishing point to estimate the best road lines [12]. Moghadam et al uses road structures and markings in unstructured environments to locate the vanishing point [13].

C. Odometry

Odometry uses sensor data to estimate robotic movement. In case of a car, the this means tracking its traveled path. In many applications GPS is used to determine a location. Since gps is not super accurate on shorter distances and does not always work as well, like inside a tunnel, its is often combined with other sensors. Many applications in robotics use an inertial measurement unit (IMU). An IMU contains an accelerometer, a gyroscope and a magnetometer. These can predict the orientation of the sensor based on its movement. There are several other methods to realize odometry. Bohlmann et al (2012) implemented automated odometry self-calibration for car-like robots [26]. Zhang et al (2014) use Lidar based odometry for real time mapping [23]. Odometry based on camera inputs is called visual odometry. Bahnam et al (2021) use multiple cameras for stereo based odometry, while Zhang et al (2022) researched monocular visual odometry, which is only based on a single camera [25] [24].

D. Control

Once it is determined where the road lanes are, the car still needs to determine where to drive to stay on the road. This is where the path planning and control stages of the autonomous driving system come in [14]. Path planning entails creating a path to get from where the car is currently, to where it wants to go. Control then ensures that it actually stays on this path. Due to the stable nature of autonomous cars, they typically

do not require a very complicated controller. Often a simple PID controller suffices. Path planning can be subdivided into global and local path planning. Global path planning is based on a map with known information regarding road information. Local path planning is more concerned with sudden traffic situations such as overtaking another car. The most common path planning algorithms are tradition algorithms, graph search algorithms and group intelligent optimization algorithms [22].

E. Image stitching

Image stitching is the process of stitching multiple images together to create one bigger image instead of several smaller ones. This technique is used for creating google street view images and to create panorama images. Image stitching typically consists of three stages: registration, seam finding and blending [10]. The registration step finds the transformation matrix between two images and transforms them into the same plane. This can be done by deriving the extrinsic and intrinsic camera parameters manually. Alternatively they can be found by matching key points in the overlapping sections of the camera images. This is done with keypoint detection and a keypoint matching step. There are many keypoint detection algorithms such as ORB and SIFT [11]. Keypoint matching takes the keypoints in one image and looks for their match in another image. An example of keypoint matching is FLANN based matching [27]. Seam finding is used to ensure every pixel on the border between the two images has a source image assigned, ensuring a smooth transition. Lastly blending is used to fix minor inconsistencies between images such as color or brightness deviations. An example of using image stitching to create a panorama image can be found in Xiong and Pulli [9]. They used a fast stitching approach to get a high quality panorama image suitable for mobile phone implementations.

III. Method

As a participant in the RDW self driving challenge the goal is to complete the track as quickly as possible. To achieve autonomous driving several stages need to be considered [**?**]. The stages that will be researched in this pare are perception in the shape of lane detection, optical flow odometry for localization and path planning and control through a vanishing point based controller. First, however, it is crucial to understand the RDW self driving challenge.

A. The Car

The car provided by RDW has a go-kart as its base. It has a 3.5 kW electric direct-drive motor, powered by a 'fixed-speed' motor controller that automatically regulates output power, a hydraulic brake line powered by a linear actuator and a servo motor mounted to steering shaft to control the steering angle. It has three Logitech StreamCam cameras mounted to the front of the kart in a 3d printed camera holder. The middle camera is facing forward at a slight downward angle. The left and right camera are adjacent to the middle camera and a rotated down and away from the centre. The front of the car also contains a planar 360 degrees lidar with a range of 6 meters. Furthermore, the car contains a steering angle sensor and a speed sensor on the back axle that measures the wheel rotations with a 1 km/h accuracy. The Car is controller by an Intel NUC mini-computer. The NUC contains an Intel Core i5-1135G7 processor with 16 GB RAM and a 512 GB NVME SSD with Ubuntu Linux installed. Notably it does not contain a gpu. Any internal communication between the NUC and actuators is done through a can-bus system [5].

Fig. 1. The RDW car used for the closed category [4]

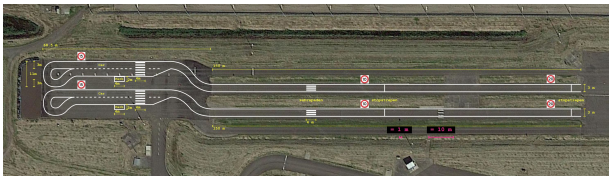

Fig. 2. The RDW racing track. Note that there are two parallel tracks of which the bottom one was used for the finale [4]

B. The Track

The track for the challenge is about 250 meters long with 3-meter-wide lanes. The entire track is made from asphalt with white continuous lines marking the track boundaries. The track starts with a straight section which includes several challenges. First of all, the car needs to adhere to the speed limit determined by the speed sign on the side of the road. Secondly, when encountering a red traffic light, the car needs to stop within a set distance of the stop line. Along this first section, there also is a zebra crossing, where a pedestrian may or may not cross. It is important to note that there is also a section included with the remnants of an old zebra crossing, which is no longer considered a pedestrian crossing. After a turn, the car enters a section with two lanes, where a striped line separates the two lines. After another pedestrian crossing, the car needs to switch lanes to overtake a parked car. Finally, there is a sharp turn (over 180 degrees) and a parallel parking objective to finish the challenge. The goal of the challenge is to complete the track and all its challenges as quickly as possible [5].

C. Lane detection

Figure 3 displays the flow chart used for lane detection. Lane detection section takes a frame as input and outputs a line for each side of the lane if any can be found. Since all road

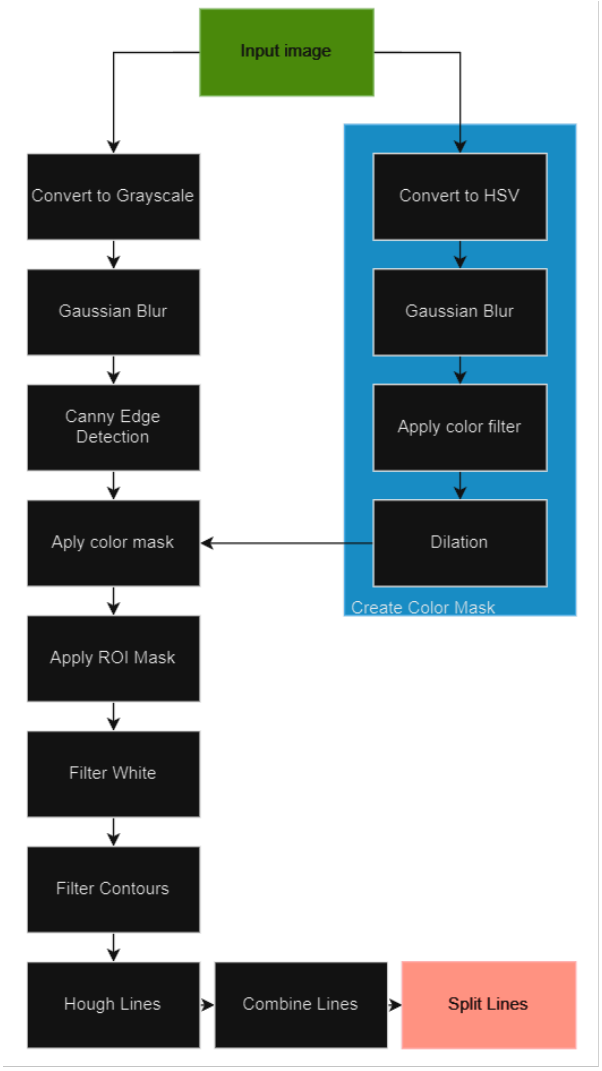

Fig. 3. The flow chart describes the lane detection, starting with an input images and outputting the left and right lane lines

markings are colored white, using a color mask is really useful in filtering out noise. First the image is converted to a HSV image. From here, a mask is created based on the brightness of the pixels. The threshold of the brightness filter is dependent on the exposure of the camera, which is automatically adjusted when starting up the cameras. A small dilation is applied to ensure all white edges are included within the mask. Besides being converted to a HSV image, the input image is also converted to a gray scale image. This is standard procedure, since edge detection does not care about color. The image is then put through a Gaussian filter to smoothen the image, reducing noise in the edge detection. Canny edge detection is then applied to find the regions of interest. Applying the color filter removes any edges that are not white and therefore do not belong to any road markings. A region of interest mask is then applied to remove any irrelevant edges. This is region of interest is determined by the expected position of the road and the previous location of the road lines.

Then a filter is applied that removes any dense white areas. This is necessary to filter out noise. An example of this is the remnants of the zebra crossing. This area caused a lot of lines to be detected due to its broken nature. A contour filter is then applied to filter out any small contours which are often caused by white stones in the road structure. This completes the edge detection stage of the lane detection. For line fitting Hough lines are used. This detects many different lines. Lines with similar angles in close proximity are then combined together to mend any broken lines. Finally the lines are split based on their angle and position, outputting the best line for each side of the road. all parameters were empirically tuned on several videos of the car driving on the track. These videos were taking at different times on different days to ensure robustness in multiple weather and lighting conditions.

D. Vanishing point based control

Instead of using the vanishing point to optimize the line detection, the vanishing point can also be used for control of the vehicle. For the RDW challenge the car always starts on a straight piece of road. This means that the starting lines are always parallel. This means that during initialization the lane detection lines can be used to determine the vanishing point and therefore the height of the horizon. Based of the orientation of the car and the road lines, the horizontal coordinate of the vanishing point in the front camera might differ, but the vertical coordinate remains the same. After initialization, even when one of the side lines is not visible, the vanishing point can still be determined based on the intersection point of the horizon and the remaining road line. This way the error can be calculated based on the horizontal distance between the vanishing point and the center of the frame. In reality, however, the camera is not perfectly facing forward, because the camera angle is not perfectly aligned with the car. In addition, when just using the vanishing point for control, the car does not care about its position in the lane. This would cause the car to slowly drift of the track in the opposite direction of the camera misalignment. This problem is fixed by taking into account the distance from the car to both sidelines. If the car is closer to one line, it will get a bias in the opposite direction. The error is thus calculated based on the vanishing point and the distance to the sidelines. The car is then controlled using a PID controller. To test the vanishing point based control of the car, the car will drive the track to see how well it stays within the lane. The radius of the car can be calculated as is described in figure 4. The error that is fed into the controller is the difference between the horizontal target position and the horizontal frame centre in frame pixels. The target is calculated like in equation 1. x_{hl} and x_{hr} are the x coordinate of the left and right line crossing the horizon respectively. d_L and d_R indicate the distance from the bottom centre to the left and right line and k is a constant that determines how much the distance compensation contributes. For the experiments k was set at 1.

$$target = ((x_{hl} + x_{hr})/2) + (d_L - d_R) * k \qquad (1)$$

E. Optical flow odometry

To track the car several solutions were attempted. All these options were tested on videos of the car driving on the race track. The car starts on the start line, then goes straight until it has to go through s-turn. Then there is another straight section until it reaches the sharp u-turn and finally finishes besides the parking spot. The overtaking of the parked car and the parking challenge are not included in this experiment. In total three methods to determine the position of the car were tested. First of all, the position was estimated based on the speed sensor and the steering angle sensor on the car itself. The steering angle is measured at the middle in the front at the axis of the steering wheel. The speed of the car is determined through counting the rotations of the rear axis. In figure 4 is a top view of the turning radius of the car. Note that the rear axis has a smaller turning angle than the front one. This needs to be taken into account when calculating the movement of the car. Before calculating the cars movement, the sensor information needs to be converted to SI units. The speed sensor outputs data in hm/h which is converted to m/s. Equation 2 is then used to find the distance traveled from one data point to the next. With maximal steering input, the car has a turning radius of 5.75 meter. Using the recorded data, a relation between the sensor output and the car angle could be established. To get the radius from the steering angle equation 3 is used and equation 4 then calculates the circumference. The angle change can then be calculated like in equation 5, here the percent sign is used to calculate the remainder, ensuring the equation holds after completing a full circle. The position of the car is then given by equations 6 and 7

$$s_1 = v_1/(t_1 - t_0) \tag{2}$$

$$r = l/sin(\delta) \tag{3}$$

$$c = 2\pi * r \tag{4}$$

$$angle = 2\pi * (s\%|c|)/c \tag{5}$$

$$x_1 = s_1 * sin(angle_1) \tag{6}$$

$$y_1 = s_1 * cos(angle_1) \tag{7}$$

Secondly, monocular visual odometry was attempted. Stereo visual odometry was unfortunately unattainable, as a result of the too small overlapping area between the three cameras. Instead of comparing keypoints between two cameras like for stereo visual odometry, monocular visual odometry compares keypoints between two frames. Again a combination of ORB feature detection and FLANN based keypoint matching was used. ORB is a feature detector. It stands for Orientated FAST and Rotated BRIEF. It is a combination of FAST feature detection and BRIEF descriptor with several enhancements. FLANN stands for Fast Library for Approximate Nearest

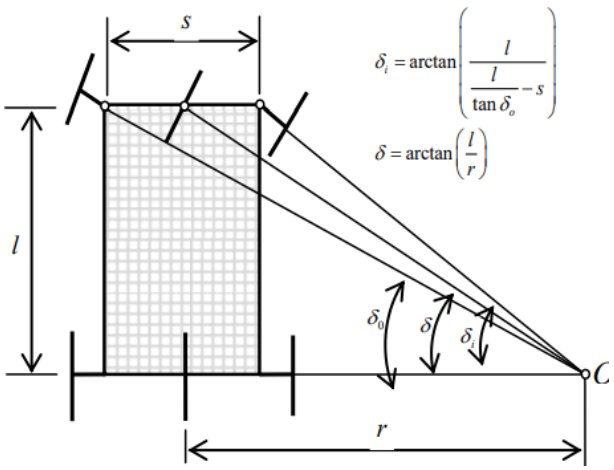

Fig. 4. top view schematic of the turning radius of the car [17]

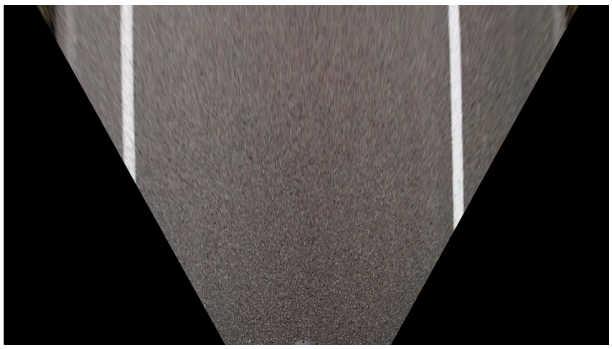

Fig. 5. A frame from the topview when driving on the track

Neighbours. It takes the features detected by the orb feature detector in two frames and matches them in pairs. From these pairs the essential matrix is determined, which is then used to find the transformation between the different frames and thus the taken path.

Lastly, odometry was attempted through optical flow in the birds eye view. Optical flow is often used to measure relative movement of objects in a camera frame. Instead of tracking objects, the movement of the road will be tracked, knowing that if the road moves in one direction on camera, the car moves in the opposite direction. For this approach keypoints were manually selected to generate an accurate birds eye view of the front camera. This can be observed in figure 5. The birds eye view transformation was manually calibrated to have equal horizontal and vertical pixel lengths. This means that the height and width of a pixel transposed into real world coordinates are the same. Since optical flow does not work as well near the borders of the frame, due to structures leaving and entering the frame, a window was used to select a region to use for analysis. The optical flow results are displayed in figure 6. Here the colors represent the direction in which the pixels move and the brightness shows the magnitude. The red square is the selected window used for odometry.

Since regions closer to the car showed more detail, this

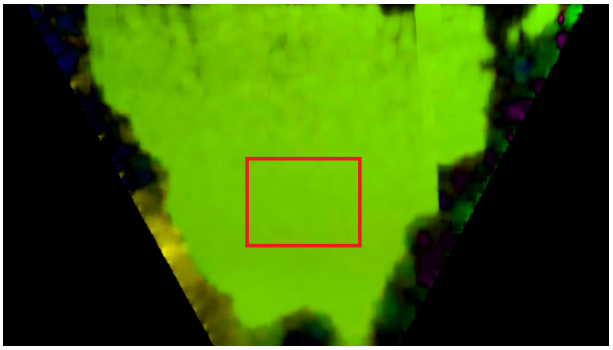

Fig. 6. Optical flow applied to the top view perspective. The colors indicate direction and the brightness displays speed. The red square is the area used for optical flow odometry.

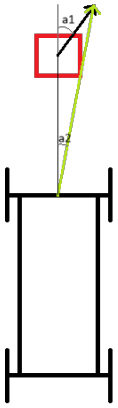

Fig. 7. the angle measured in the square (a1) compared to the actual angle of the car (a2).

region was chosen. The average vector of all vectors in this region was taken to estimate the movement of the camera and therefore the movement of the car represented in polar coordinates. Since the camera is slightly misaligned with the car, a small compensation is incorporated.

It is important to know that the optical flow movement is calculated slightly in front of the car and not on the car itself. The further away the selected area is from the car, the more the squared region will move with sideways movement of the car. This difference in angle can be observed in figure 7. To determine the position of the square relative to the car, a marker was placed on the road of which the distance to the car was measured. This is then used to find the size and position of the square relative to the car. From the angle at the red square, the angle of the car (a2) can be calculated as in equation 8. In this equation a1 and mag are the angle and the magnitude calculated by the optical flow section respectively. d is the distance from the camera to the square.

$$a2 = arctan((mag * sin(a1))/(mag * cos(a1) + d)) \quad (8)$$

F. Stitching

To utilize the full potential of the given cameras it is beneficial to use all three cameras. Camera images could be analysed individually. This way, however, one needs to deal with overlapping areas between cameras and different orientations. It would be easier if all camera images are stitched together first and then a larger image can be used for lane detection. The goal of stitching is to use all three cameras while only having to process a single image. It enlarges the closer and therefore more relevant lines, because of their presence in the side cameras. Due to the unstable nature of the cameras manually determining the camera parameters is not an option. To find the transformation matrix between the different cameras, three experiments were conducted. All experiments were done on a recording of the car driving on the racing track. The first experiment aims attempts live stitching by using keypoint detection and keypoint matching for every single frame.ORB detection is attempted in combination with FLANN based matching. Using the keypoint pairs, a the essential matrix is found. As a result the transformation matrix is used to transpose the images from the left and right camera into the front cameras frame. The advantages of this method is that it is more robust to camera movement and cart differences. However, it requires a decent camera overlap and good key points for accurate results. Furthermore, it is more computationally expensive due to constant key point detection and matching. The second experiment uses the same preset transformation matrix for every frame. This matrix is determined by manually selecting and matching keypoints in a video recording of the car. To do this a frame with a pedestrian crossing was selected. The corners of the crossing rectangles are manually determined to ensure good keypoints are found. Through the same process as the first method the matching keypoint pairs are then used to find the transformation matrix between cameras and transpose them all into the coordinate system of the front camera. This method allows for precise tuning and therefore a very accurate result. Furthermore, no live tuning is required making this the fastest method. The last experiment that will be run uses the same predetermined matrix, however, it also implements a bumper detection algorithm to detect bumper movement in the frame, allowing for compensation for the bouncing of the cameras. This is essentially the same method as used for the second experiment with an additional step. Using color filtering the bumper is extracted from the frame. Tracking these keypoints and transposing them to the coordinate of the manually tuned frame, allows to compensate for any movement of the camera in relation to the bumper. The bumper tracking extraction can be seen in figure

IV. Results

A. Line detection

To demonstrate the implemented line detection, a single frame was selected while autonomously driving the track. Figure 8, 9 and 10 correlate to the blocks shown in figure

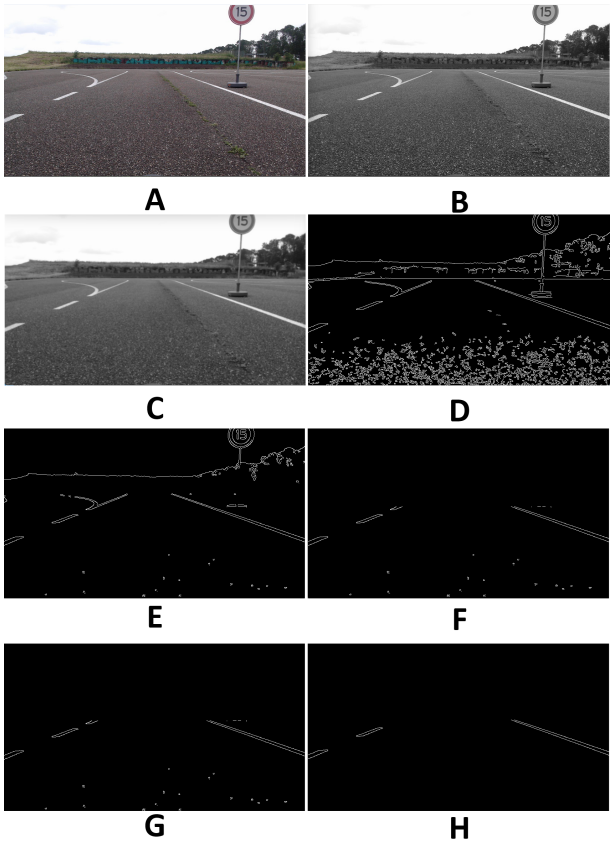

Fig. 8. The process of getting from the input image to the road line edges in eight steps. A: The input image. B: Grayscale conversion. C: Gaussian Blurring. D: Canny Edge Detection. E: Color filtering. F: Region of interest filtering. G: White Filtering. H: Contour Filtering.

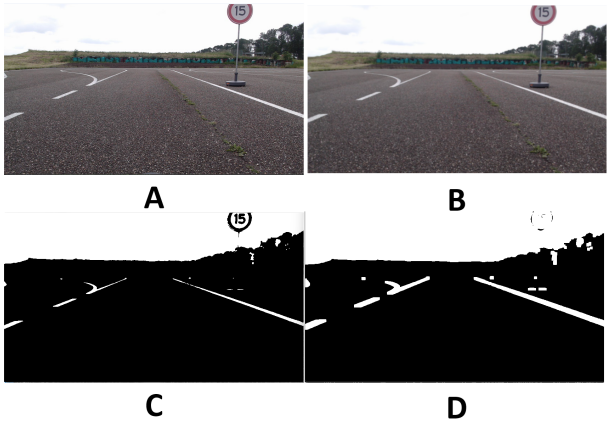

Fig. 9. The creation of the color mask in 4 steps. A: HSV conversion. B: Gaussian Blurring. C: Brightness thresholding. D: Dilation.

Fig. 10. Line fitting starting with Houghlines in A and the selected road lines in B.

3. Figure 8 shows the steps from the input images to the eventual edge detection. Figure 9 shows the process of creating the color mask and figure 10 demonstrates how the line fitting process gets from multiple Hough lines to two lane boundaries.

B. Vanishing Point based control

As previously stated the error for the vanishing based control is determined by the position of the vanishing point and the distance to the side lines. While driving on the track several frames were captured in different situations that show the middle of the frame in white and the target location in red. The left and right lane line are also included in the image, colored green and blue respectively. These results can be found in figure 11. The scenarios from top to bottom are: driving straight on a straight road, driving at an angle on a straight road, driving in a turn on one side of the road, driving over a pedestrian crossing and driving in a sharp turn. Note that in the last image the target dot is not in the frame. This is because it is too far left to fit the images size.

C. Odometry

Figure 12, 13 and 14 show the results of the the three odometry experiments compared to the orange line, which represents the shape of the track. All three images show the path of the car starting from the origin and following the line to the parking space. Figure 12 shows the worst results of the three. The track is unrecognizable in this figure. The orange line of the track is only partially visible here because of the major deviation of the estimation of the track. Figure 13 shows the path estimated by the speed and steering angle sensor. When comparing this estimated path to the actual track in figure 2, the first part of the track is pretty similar. The estimated path is pretty straight until the first turn at 150 meters, which was measured only slightly early. After the turn the line is rotated slightly more left than expected. In reality it should have been parallel to the start section. Similar to the first turn, the second turn is slightly early again and measured sharper than expected. This results in the slight crossing of lines in the last section.

In figure 14, just like in figure 13, the track can be seen in the estimated path. When comparing it to the actual track in figure 2, the estimation makes a straight start just as expected. Both turns, expected at 150 meter and 218 meter, however, are measured a bit later than then they are on the actual track. Furthermore, the last turn seems to be measured a little too

Fig. 11. The target point (red) and frame centre (white) for different situations.

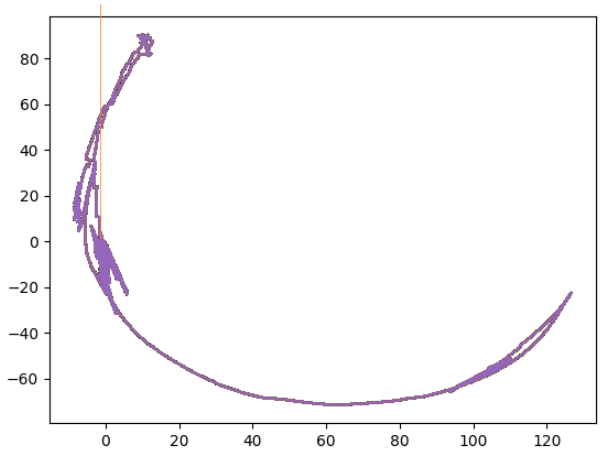

Fig. 12. The estimated path of the car determined through visual monocular odometry (purple) compared to the true path (orange). Both axis are in meters, where the car start on the start of the track looking in the positive y direction.

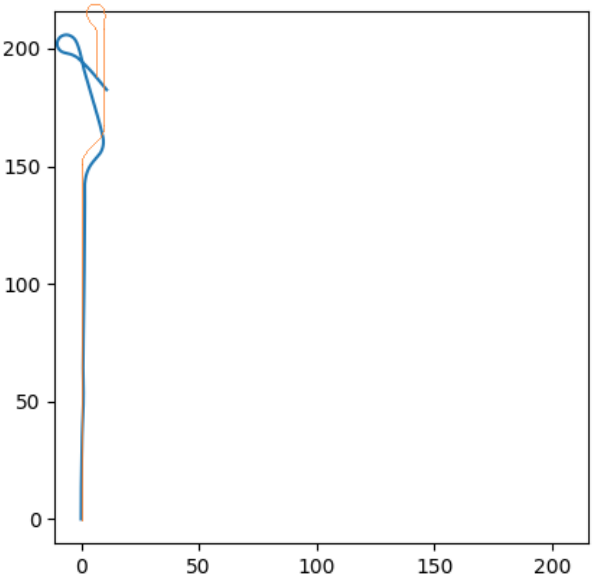

Fig. 13. The estimated path of the car determined through the speed and steering angle sensor of the car (blue) compared to the true path (orange). Both axis are in meters, where the car start on the start of the track looking in the positive y direction.

blunt. This is the case because the last section of the estimated path slowly moves away from the supposed to be parallel line besides it.

D. Image Stitching

For Image stitching three experiments conducted. Figure 15, 16 and 17 show the live keypoint stitching, preset transformation stitching and preset transformation stitching with bumper based correction respectively. Since the entire video could not be shown in a report format, two frames were selected from each method. In these images each channel of the RGB matrix represents a different camera. Live keypoint stitching is displayed in figure 15. Frame A is one of the

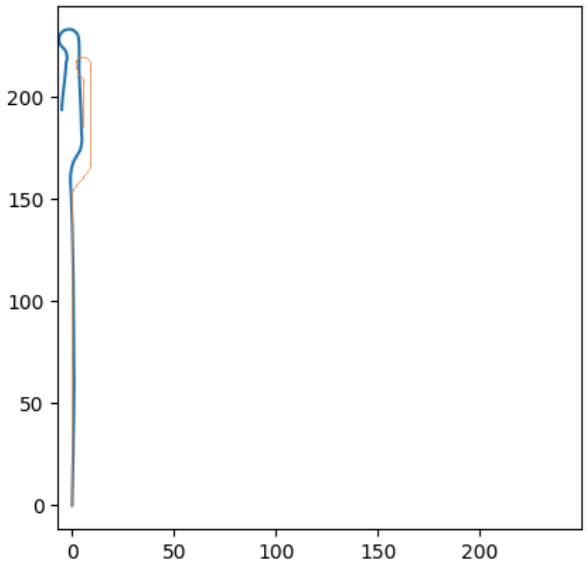

Fig. 14. The estimated path of the car determined through optical flow odometry (blue) compared to the true path (orange). Both axis are in meters, where the car start on the start of the track looking in the positive y direction.

better looking frames, while frame B looks a bit worse. This method showed really inconsistent results. The first frame has a slight misalignment of the lines while this error is way greater in frame B. Figure 16 and 17 show images from a video were due to a bump in the road, the right camera is displaced halfway through the drive. Figure 16 A and 17 A show the results before the displacement and figure 16 B and 17 B afterwards. Before the displacement both videos showed consistent results, with a bit more noise in the bumper correction experiment. After the camera displacement however, the bumper displacement corrected more for this error than the method with only a preset transformation.

V. Discussion

A. Lane detection

The lane detection algorithm was tested on many recordings with great results. For a single frame all the steps described in figure 3 can be seen in figure 8, 9 and 10. Looking at figure 8, the edge detection section does its job really well. After the plane canny edge detection result in 8B, all noise was filtered leaving only the road lines in figure 8H. Note that in the last image, the last striped road line on the left side of the road was filtered out. Since the line was still far away and therefore really small in the image, this is not a problem. Especially if the side cameras could be utilized, the bigger and more relevant lines are always detected.

Figure 10 shows the line fitting step. The Hough lines detected in the left image are combined to get both road lines. This is not just the case for the shown example. Over several runs no wrong lines could detected. Both sidelines could consistently be found as long as they were present in the camera frame. The only small problem occurs on the zebra

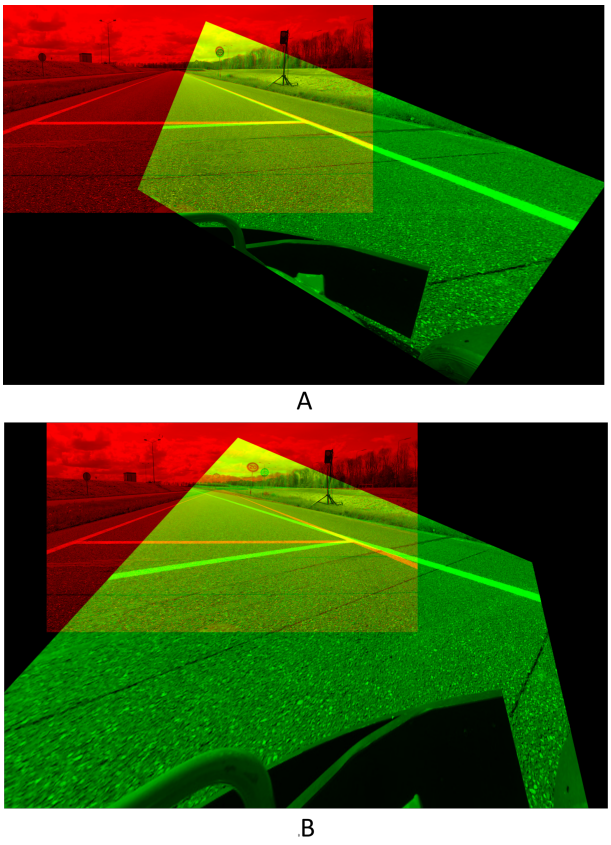

Fig. 15. Two different frames when using live keypoint detection and matching to stitch the centre and right camera together.

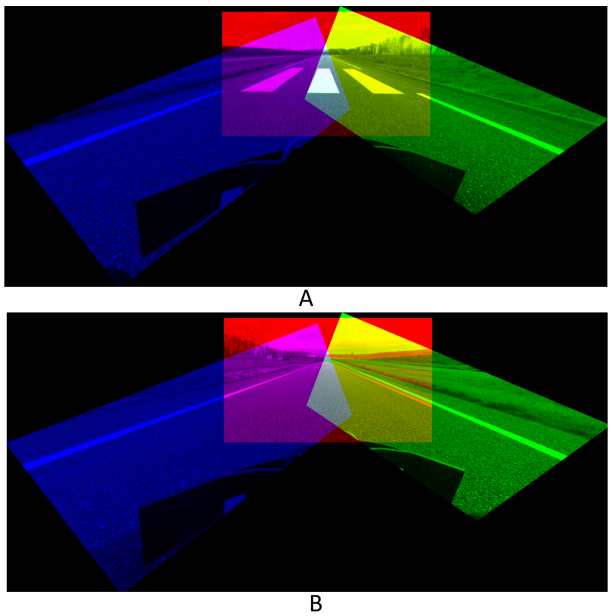

Fig. 16. Two image stitching frames with a preset transformation matrix. Each channel of the RGB matrix represents a different camera. image A was taken before the right camera was moved and image B after the camera movement.

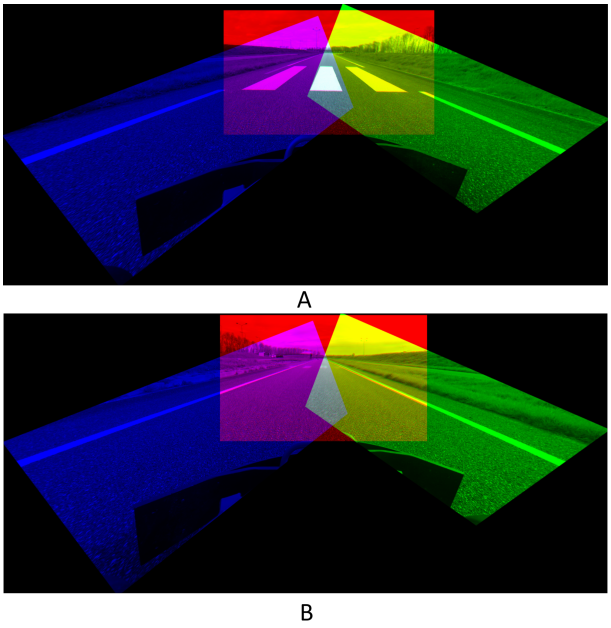

Fig. 17. Two image stitching frames with a preset transformation matrix and bumper based correction.Each channel of the RGB matrix represents a different camera. image A was taken before the right camera was moved and image B after the camera movement.

crossings. On these crossings the algorithm detects the crossing lines as road lines. For determining the vanishing point this is not necessarily a problem, since all these lines point to the vanishing point anyways. However, when determining the distance to the sidelines, it matters more which line is selected. On occasion a small reposition is done by the car to stay in the middle between the selected lines. This reposition is still far too small to displace the car from its lane. This could be fixed by filtering out the crossing stripes and focusing on the lines further along the road for lane detection, when such a crossing is encountered.

The main advantage of the implemented lane detection is that it is very accurate and consistent. There are however a few downsides to the design. One of the disadvantages is that it only considers the current that lane the car is driving in. For planning in real life scenarios it is often useful to not only track the current lane but all lanes on the road. Another disadvantage is that the lane detection algorithm is designed and fine-tuned for the RDW test track in Lelystad. Several adjustments may need to be made when driving in a different environment. For example, if yellow road lines were used, they would be filtered out by the white colour mask.

B. vanishing point based control

To test the vanishing point based control, the car was sent to drive the track autonomously. To give an idea of how it behaves, figure 11 shows the target location in several interesting situations. When driving straight in the middle of a straight road, like in the top image, the target is in the middle of the frame, meaning no steering is required. When the car drives at an angle like in the second image of figure 11, the target tries to take it back so it becomes straight again. When the car approaches the side of the road, even in the inside of a turn, the car is encouraged to move back to the middle of the road like in the third image of figure 11. The behaviour on a pedestrian crossing was already discussed in the lane detection section. To reiterate, the somewhat inconsistent choice of which lines that is used at the pedestrian crossing sometimes causes the car to reposition itself between these lines. This problem is more the responsibility of the lane detection than the control. The final image in figure 11 shows the behaviour of the car in a sharp turn. This is the most problematic part. Just before the displayed frame the car had no vision of the left road line for a while. For this reason, the car crossed the inner line of the road slightly. When the car loses vision of one of the lanes the lane centering does not work as effectively anymore, especially in a sharp turn. The best way to improve this is by increasing the vision of the car. This would have been done if image stitching was achieved in time. When both lane lines are correctly detected, however, the vanishing point based control performed really well. In these scenarios the car was kept in the middle of the lane in a really consistent and controlled way.

C. Odometry

From the three odometry experiments that were conducted, the monocular visual odometry clearly performs the worst. This is a result of having poor keypoints. The current keypoints rely on the environment, which for a big part, consists out of asphalt, grass and trees. All of these generally do not give many good keypoints, because they are not very distinctive. The latter two are especially bad because they can also be influenced by wind. When having a higher threshold to filter the quality of the keypoint matches, often frames occur with too few points to be analysed. To get usable results, the threshold had to be lowered. As a result of this too many poor keypoints were used and the estimated path became unrecognisable. To efficiently use this method several improvements could be made. First of all, better features are required. This would require a track with more solid structures like buildings close to the road. Secondly, it would benefit this method to have a greater overlap between the camera images, so that stereo visual odometry was possible.

The remaining two methods to track the cars movement, using the sensors and using optical flow, both showed some promise. The odometry based on the steering angle and speed sensor of the car showed the best results regarding distance estimation, with only a small error between actual and expected turning points. The inaccuracy is probably caused by the low accuracy of the speed sensor being rounded to full kilometers per hour. This rounding error adds up over time. The sensor based odometry method does however show some inaccuracies during the turns. Both turn are estimated too sharp. This is probably the case, because the model does not account for tyre slip. This means that some or all tyres of the car lose traction, causing the car to turn without movement of the steering wheel. Since the steering wheel is unaffected, the rotation can

not be picked up by the steering angle sensor, resulting in a tracking error. To reduce this error, a mathematical model could be implemented that predicts tyre slip based on factors like the speed and weight distribution of the car.

The odometery based on optical flow also produced a path from which the track can clearly be recognized. The estimated distance is however slightly greater than in reality. This is presumably caused by drift in the distance error due to inaccuracies in the optical flow measurements. Optical flow odometry does perform better, however, when it comes to the turn angle. The results seem to be almost in line with the real track. Although the performance of the optical flow odometry seems pretty good, there are still some inaccuracies caused by the optical flow calculations and in the conversion to the birds eye view. There are several improvements that could be made to increase the accuracy of optical flow odometry. A higher frame rate allows for a more accurate optical flow estimation and a higher quality camera for better quality optical flow and more accurate tracking.

Based on distance, the sensors outperformed the optical flow. However, on the turning sections optical flow shows better results. Ideally combining these methods would allow for a better solution, by combining the best of both worlds.

D. Image Stitching

Similar to the keypoint based odometry, the keypoint based image stitching yields poor results. The problem is again rooted in the poor available keypoints. Furthermore, this method was quite slow and therefore difficult to implement on the limited hardware.

The second experiment with manually initialised keypoints already shows better results. Since all the keypoints are calculated in advance this method works the fastest out of the three. Furthermore, it is the most noise resistant out of the three methods. The big disadvantage of this method is that it is vulnerable to camera movement. Calculating the transformation matrix beforehand assumes that the cameras will not move relative to one another. When this does happen like in figures 16 B, the method breaks. The last method with bumper detection is more robust to camera movement as can be seen in figure 17. This comes at a small computational cost as well as being influenced by any inaccuracies in the keypoint tracking on the bumper itself. One can argue if this trade off is worth it since camera stabilization should most likely not be done through software. This could be achieved by improving the camera mount, allowing for less camera movement. Unfortunately, Stitching could only be achieved on video, but not live on the track. This is because of the inconsistency of camera positioning between runs. Keypoints would have to mannually selected every run, which is a process that takes a lot of time to complete. This could be fixed through the introduction of an initialization mat. A big mat could be rolled out in front of the cart, similar to a chessboard pattern. Before driving, the transformation matrix can then easily be determined through matching of easily detectable key points like the chess board corners. This would fix the inconsistency between runs. Unfortunately, due to time limitations, this idea could not be executed.

VI. Conclusion

In conclusion autonomous driving based on visual input and traditional computer vision techniques was definitely achieved. The car stably drove on the majority of the RDW track. The only problem was encountered when vision was lost of the inner sideline in a sharp turn. Image stitching fixes this problem. Unfortunately, image stitching was only achieved in video analysis, but not live on the car. Based on video analysis, however, it is highly likely that its is possible to be achieved through a checkerboard mat initialization. Realising this would also allow for the live implementation of optical flow odometry, since it would allow a stable birds eye view to be consistently created. Optical flow is definitely a good option for odometry when no other sensors are available. It can also be used in combinations with other sensors to combine different methods. In real world applications however IMU units and GPS will likely still outperform it.

References

[1] C. Wild, ”Software defined vehicles: The path to autonomy”, Altran, 2016

[2] Coalition for greater mobility, ”Highly automated technologies, often called self-driving cars, promise a range of potential benefits.” Derived from: https://coalitionforfuturemobility.com/benefits-of-self-driving-vehicles/

[3] M. Hartwig, ”Self-driving and coope- rative cars”, Bayrische Motoren Werke AG, Muchich, Germany, 10-02-2020.

[4] RDW wiki

[5] RDW, ”Self driving challenge edition 2024 information document”, RDW, the Netherlands, 2024

[6] W. Hao, ”Review on lane detection and related methods” Cognitive Robotics 3 pages 135-141, 2023

[7] Q. Zao et al. ”Robust Lane Detection from Continuous Driving Scenes Using Deep Neural Networks”, 2020, Derived from: https://arxiv.org/pdf/1903.02193

[8] T. Mandlik, A.B.Deshmukh, ”A REVIEW ON LANE DETECTION AND TRACKING TECHNIQUES”, Skncoe, Vadgoan ,Maharashtra,India, 05-05-2016

[9] Y. Xiong and K. Pulli, ”Fast panorama stitching for high-quality panoramic images on mobile phones,” in IEEE Transactions on Consumer Electronics, vol. 56, no. 2, pp. 298-306, May 2010, doi: 10.1109/TCE.2010.5505931.

[10] C. Herrmann et al, ”Robust image stitching with multiple registration”, Proceedings of the European Conference on Computer Vision (ECCV), 2018, pp. 53-67

[11] S. A. K. Tareen and Z. Saleem, ”A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK,” 2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), Sukkur, Pakistan, 2018, pp. 1-10, doi: 10.1109/ICOMET.2018.8346440.

[12] T. Youjin et al, ”A Robust Lane Detection Method Based on Vanishing Point Estimation”, Procedia Computer Science, Volume 131, 2018, Pages 354-360.

[13] Moghadam, Peyman Starzyk, Janusz Wijesoma, W.. (2012). Fast Vanishing-Point Detection in Unstructured Environments. IEEE Transactions on Image Processing. 21. 425-430. 10.1109/TIP.2011.2162422.

[14] Mohamed Reda, Ahmed Onsy, Amira Y. Haikal, Ali Ghanbari, ”Path planning algorithms in the autonomous driving system: A comprehensive review”, Robotics and Autonomous Systems, Volume 174, 2024, 104630, ISSN 0921-8890.

[15] Chao cheng Li, Iun Wang, Xiaonian Wang, ”A Model based Path Planning Algorithm for Self-driving Cars in Dynamic Environment”, Department of Control Science and Engineering, Tongji University, Shanghai, P. R. China.

[16] Bohlmann, Karsten Marks, Henrik Zell, Andreas. (2012). Automated odometry self-calibration for car-like robots with four-wheel-steering. 2012 IEEE International Symposium on Robotic and Sensors Environments, ROSE 2012 - Proceedings. 168-173. 10.1109/ROSE.2012.6402609.

[17] VU, T[rieu] M[inh], "VEHICLE STEERING DYNAMIC CALCULATION AND SIMULATION", Annals of DAAAM for 2012 Proceedings of the 23rd International DAAAM Symposium, Volume 23, No.1, ISSN 2304-1382 ISBN 978-3-901509-91-9, CDROM version, Ed. B. Katalinic, Published by DAAAM International, Vienna, Austria, EU, 2012

[18] V.S.Bottazzi, P.V.Borges, B.Stantic, "Adaptive regions of interest based on HSV histograms for lane marks detection". In Robot Intelligence Technology and Applications 2, pp. 677-687, Springer International Publishing, 2014

[19] Jiang, Ruyi Klette, Reinhard Wang, Shigang Vaudrey, Tobi. (2011). Lane Detection and Tracking Using a New Lane Model and a Distance Transform. http://www.mi.auckland.ac.nz/tech-reports/MItech-TR-39.pdf. 22. 10.1007/s00138-010-0307-7.

[20] R. Muthalagu, A. Bolimera and V. Kalaichelvi, Lane detection technique based on perspective transformation and histogram analysis for self-driving cars, Computers and Electrical Engineering, https://doi.org/10.1016/ j.compeleceng.2020.106653

[21] Hu Y., Kim Y., Lee K. and Ko S. (2010). LANE DETECTION BASED ON GUIDED RANSAC. In Proceedings of the International Conference on Computer Vision Theory and Applications, pages 457-460 DOI: 10.5220/0002832204570460

[22] Ming, Yu Li, Yanqiang Zhang, Zihui Yan, Weiqi. (2021). A Survey of Path Planning Algorithms for Autonomous Vehicles. SAE International Journal of Commercial Vehicles. 14. 10.4271/02-14-01-0007.

[23] Zhang, Ji Singh, Sanjiv. (2014). LOAM : Lidar Odometry and Mapping in real-time. Robotics: Science and Systems Conference (RSS). 109-111.

[24] Zhang, Sen Zhang, Jing Tao, Dacheng. (2022). Towards Scale Consistent Monocular Visual Odometry by Learning from the Virtual World. 5601-5607. 10.1109/ICRA46639.2022.9812347.

[25] Bahnam, S., Pfeiffer, S., de Croon, G. C. H. E. (2021). Stereo Visual Inertial Odometry for Robots with Limited Computational Resources*. In IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021: Proceedings (pp. 9154-9159). Article 9636807 (IEEE International Conference on Intelligent Robots and Systems). IEEE. https://doi.org/10.1109/IROS51168.2021.9636807

[26] Bohlmann, Karsten Marks, Henrik Zell, Andreas. (2012). Automated odometry self-calibration for car-like robots with four-wheel-steering. 2012 IEEE International Symposium on Robotic and Sensors Environments, ROSE 2012 - Proceedings. 168-173. 10.1109/ROSE.2012.6402609.

[27] Hassaballah, M., Alshazly, H.A., Ali, A.A. (2019). Analysis and Evaluation of Keypoint Descriptors for Image Matching. In: Hassaballah, M., Hosny, K. (eds) Recent Advances in Computer Vision. Studies in Computational Intelligence, vol 804. Springer, Cham. https://doi.org/10.1007/978-3-030-03000-1$_5$