2018 IEEE Intelligent Vehicles Symposium (IV)
Changshu, Suzhou, China, June 26-30, 2018

Autonomous Car Navigation using Vector Fields

Zahra Boroujeni, Mostafa Mohammadi, Daniel Neumann, Daniel Goehring, Raul Rojas

Abstract— **In this paper, a method based on vector fields for the navigation of autonomous cars is developed. Vector fields—used to generate the desired heading angle of a vehicle toward a specified road lane—attract the car to the desired path and prevent the car from colliding with obstacles. Also, a control law is developed to define the velocity direction and the desired steering angle based on the angle between the car and the vector field. The efficacy of the proposed approach is investigated through several simulations and lab experimental tests.**

I. Introduction

The problem addressed in this paper is that of navigation of autonomous cars using vector fields in structured road maps. The ego car should follow the desired road lane while avoiding obstacles on the road. The proposed approach combines map data and path planning algorithms, and provides vector fields which navigate the ego car toward the road. The vector fields are calculated offline to obtain the force vector of each point in an efficient way. On-line path planning and trajectory generation are computationally heavy tasks. By making offline calculations, especially when the autonomous cars drive inside cities with restricted area, the calculation capacity of the car's computer might be saved.

A vector field is like a magnetic field around the desired road lane, so that the desired road lane attracts the ego car. A well known weakness of vector fields is that for each problem a specific vector field should be defined "ad hoc". A new 2D vector field definition based on the car velocity is proposed in this paper. In the vector field approach, it is important to run the controller in a high frequency. However, online calculation of vector fields for large areas makes it difficult to run the controller in a high rate. To cope with this problem, we calculate the force field for the discretized area, and save it in an efficient way, so that the online algorithm can be run in a very high frequency (200Hz). Repulsive forces around the obstacles are also defined based on the road lanes, which push the ego car to the adjacent lane. A discrete vector field navigation system could cause the chattering; this problem is solved in this paper by interpolations in on space and speed.

In this paper we present a novel approach for autonomous vehicle navigation in environments with a structured map, by creating offline force fields, which specify the desired heading angle of the vehicle to fulfill *path following* and *lane keeping* tasks. The force fields are augmented and modified locally by presence of obstacles, as result of the obstacles force field. In creating force fields we take into account the velocity of the vehicle along with the vehicle distance from the path, to find feasible force vectors.

The rest of the paper is organized as follows: The next subsection briefly reviews the related work, and highlights the contribution of the present paper. Section II describes the vector fields. In Section II-E the obstacle repulsive vectors are designed. Then, in Section III numerical simulation and experimental results are provided to show the effectiveness and efficiency of the proposed approach. Finally, conclusions and plans for future works are outlined in Section IV.

A. Related Work

The core idea of using vector fields for the navigation of autonomous vehicles is to define an artificial vector field which attracts the vehicle toward a desired point (goal), and prevents collision with obstacles [1]. The concept of using vector and potential fields for finding optimal paths in environments with static and dynamic obstacles for mobile robots is yet a developing research topic, and researchers suggest new techniques for path planning and collision avoidance. Vector fields for the robotic navigation are used in several applications ranging from mobile robots [2] to aerial vehicles [3], space crafts [4], and recently for autonomous cars [5], [6]. An informative comparison between different novel and well-established potential field based path planning approaches is presented in [7]. The application of the vector field techniques to path path planning and navigation of autonomous cars has recently gained a great attention. An integrated motion planning and control approach for autonomous car navigation based on potential fields is presented in [8], which aimed at reducing the control effort while maintaining a desired tracking error tolerance, and a smooth steering. A framework for path planning and tracking is suggested in [9], which concerns collision-free paths.

We aim at navigation of an autonomous vehicle in a structured road map. The desired path is initially planned by a path planning sub-system in accordance to the mission of the autonomous car. We utilize a force field to execute the path following task. The contribution of this paper with respect to the state of art is summarized as follows.

- Instead of considering a single point as goal, we consider a lane of goal points which specifies the path, and in calculation of the goal points the vehicle lateral distance from the path, and the ego car speed are considered.

The authors are all affiliated with Dahlem Center for Machine Learning and Robotics, Computer Science Institute, Freie Universität Berlin, Germany { zahra.boroujeni, mostafa.mohammadi, daniel.neumann, daniel.goehring, raul.rojas}@fu-berlin.de

This work was supported by the Bundesministerium für Bildung und Forschung (BMBF) - Intelligente und effiziente Elektromobilität der Zukunft (e-Mobilize) and by the Bundesministerium für Verkehr und digitale Infrastruktur (BMVI) - Automatisiertes und Vernetztes Fahren auf digitalen Testfeldern in Deutschland.

- In order to reach high speed, high rate, and computationally light navigation, we generate a force field offline, which is updated online in case of detecting obstacles in the vicinity of the car.
- In order to optimize the memory usage, and achieve a smooth motion, the vector fields are interpolated not only in the space domain, but also in the speed domain.

II. Navigation using vector fields

In order to perform the path following task using force field approach, let G be the desired path to be followed, *e.g.*, a lane of the road. This path is considered as a collection of goal points. We discretize the area around the path with an appropriate resolution. The idea is to define a force field F, such that—for each position (x,y)—the goal attracts the autonomous car, and detected obstacles must generate artificial repulsive forces to prevent collision. Thus, the attractive force vector $F_G(x,y) \in \mathbb{R}^2$ and repulsive force vectors $F_{O_i}(x,y) \in \mathbb{R}^2$ constitute in each position (x,y) the force vector $F(x,y)$ as:

$$F(x,y) = F_G(x,y) + \sum F_{O_i}(x,y) \tag{1}$$

In the sequel, we show how to calculate F_G and F_{O_i}.

In calculating F_G we consider a constant magnitude for it, in each point, and we calculate its orientation which specifies the appropriate vehicle heading angle in order to reach and follow the path. The direction of $F_G(x,y)$ is defined by a vector from the current position of the car to a goal point (P_g) on the path. In the following sub-sections, the goal points are defined first, and then the process of generating attractive force field is described.

A. Goal Points definition

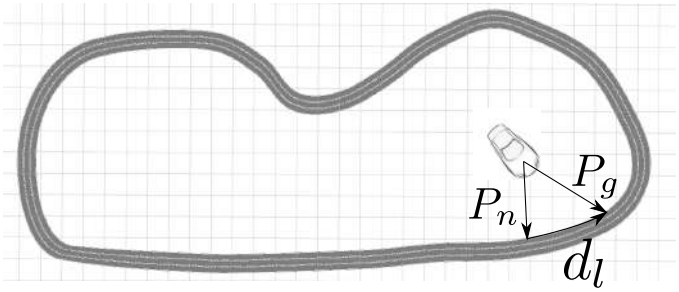
The main goal of the proposed approach is to follow a desired path, that without loss of generality we may consider a road lane as desired path. If the objective was only to reach the road lane, for each $(x,y)-$point, the choice of the nearest point (P_n) on the path as a goal point would be the best choice ($P_g = P_n$). However, to follow the lane, the goal point must be further ahead of the nearest point which pulls the car along the path. Also, when the autonomous vehicle reaches the path, the look-ahead point should be chosen based on the car's speed, in order to obtain smooth changes in desired heading angle of the car, and to prevent oscillation of its heading angle due to unfeasible commands.

$$P_g = P_n + d_l \tag{2}$$

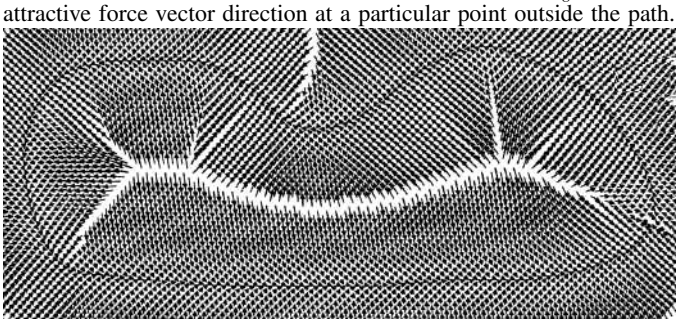
where d_l is a vector from the end point of P_n to a point l meters ahead along the path, and l is defined based on the velocity and the car distance to the path.

$$l = \alpha \min(|v|, \frac{|v|}{||P_n||}) \tag{3}$$

in which $v \in \mathbb{R}$ is the ego car's speed, $||P_n||$ is the distance between the ego car's position and its nearest point on the road path, and $\alpha \in \mathbb{R}$ is a tuning parameter. When the ego car is far from the path, l is approximately zero, while once the ego car is on the road, l reaches its maximum value. Fig. 1a depicts P_g, P_n, and d_l for a car outside the path. The vector field is saved in a two-dimensional matrix F of size of $l \times w$ where $l = \lfloor \frac{L}{r} \rfloor$ and $w = \lfloor \frac{W}{r} \rfloor$ in which r is the resolution of the vector field, and L and W are the length and the width of the surrounding area of the path. Each element of the matrix is a 2D force vector containing F_x in x-direction and F_y in y-direction.



(a) Tempelhof test area is discretized in position. Vector P_g shows the attractive force vector direction at a particular point outside the path.



(b) A vector field obtained for a constant velocity of 2 m/s.

Fig. 1: A typical map and path, and the corresponding vector field for a constant velocity.

As is evident from (3) the vector field depends on the car speed, *i.e.*, the car's ability to steer depends on its velocity: in the lower velocities the autonomous car can effectively and safely rotate more than in higher velocities. Therefore, we calculate the vector fields for minimum and maximum speeds offline, and save them as F_{min} and F_{max}, respectively. Then, for a specific velocity the appropriate force vectors will be interpolated online.

B. Interpolation

We calculated a force field for a discretized set of positions and in minimum and maximum velocities. In order to use it in any position in the field and in any velocity in range, we use interpolation twice:

- Interpolation in space domain, that calculates the force vector for any position based on the surrounding force vectors within a specified vicinity of the position.
- Interpolation in speed domain, that calculates the appropriate force vector for a specific velocity based on the force vectors for minimum and maximum velocities at the same position.

The space interpolation is firstly performed, and followed by the speed interpolation.

Interpolation in space is based on that the force vector in a specific position P is the weighted average of the force vectors belong to the elements of F which P located in between them. Table I depicts the interpolation in space algorithm. In the interpolation presented in Table I first the

TABLE I: The interpolation algorithm

Interpolation
For grid points around the car position:
$d_i[m]$=distance to the car position
$w_i = \frac{1}{d[i]+\varepsilon}$
normalization step $w_i = \frac{w_i}{\sum w_i}$
$f_p = \sum w_i F[i]$

grid points near the current car position (P) are selected, and their indices (i) are kept. Then, the distance of each grid point from the current car position (d_i) is calculated. The weight of each grid point is defined as the inverse of its distance to the car's position, and a small value (ε) is added to the denominator to avoid big values in case of very short distance. Next, a weight normalization step is performed to keep the result value in the range of the input elements; after the weight normalization step, the interpolated force vector (f_p) is calculated as the weighted average of the input force vectors. The next interpolation step is to be done in the velocity domain; we need the force vector ($f_v(P)$) in a specific velocity (v) in a specific position (P), which is expressed as:

$$f_v(P) = f_{min}(P) + \frac{f_{max}(P) - f_{min}(P)}{v_{max} - v_{min}}(v - v_{min}) \qquad (4)$$

where $f_{min}(P)$ and $f_{max}(P)$ are the force vectors of the minimum and maximum speeds (v_{min} and v_{max}), which are calculated in the previous interpolation step offline.

C. Motion direction and steering angle

The motion direction and the steering angle, in each instance, are obtained by projecting the force vector on the car frame. If the longitudinal element of force (f_x), w.r.t. the body frame of the vehicle, is negative then the car drives backward, and if f_x is positive, the car moves forward. The steering angle (ψ) is expressed as:

$$\psi = \begin{cases} \beta \ \text{atan2}(f_y, f_x) & \text{if } f_x \geq 0 \\ -sign(f_y)\max(|\psi|) & \text{if } f_x < 0 \end{cases}$$

where $\beta \in \mathbb{R}^+$ is a positive tuning parameter. Fig. 2 shows the force vector (f_P) in four different position of the car w.r.t. to the path. In each instance f_P is projected to the body frame of the car, if $f_x > 0$ the car moves forward, while if $f_x < 0$ means a backward motion of the car. The circles in Fig. 2 represent the steering corresponding to f_P and the motion of the car as result of that steering.

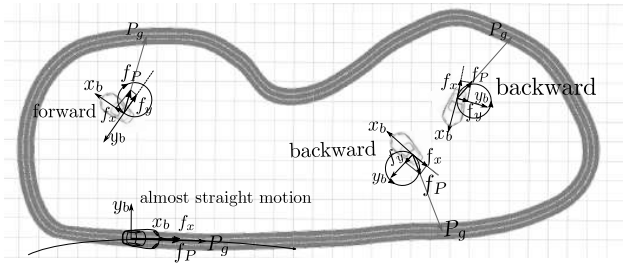


Fig. 2: Direction of motion and steering for different orientation of the car in different positions w.r.t. the path.

D. Lane changing

An autonomous car driving on a road quite often needs to change the driving lane, in order to overtake other cars, to change the speed, or to find a better path with less traffic. The motion planning subsystem of the autonomous car is in charge of selecting the lane. The force field for each lane is separately calculated, and according to the command from the motion planning subsystem, when lane changing action is needed, the system only need to load the vector field of the new lane; then, the ego car will change the lane automatically. Fig. 3 shows a schematic representation of the force field for a road of three lanes, and two consecutive lane changing in order to perform an overtaking action. Once Lane 1 is the desired lane, its corresponding force vector navigates the car along the the desired path, and by changing the desired lane to the second lane, the lane changing action takes place automatically by loading the force field of Lane 2 and using it as the navigation source.

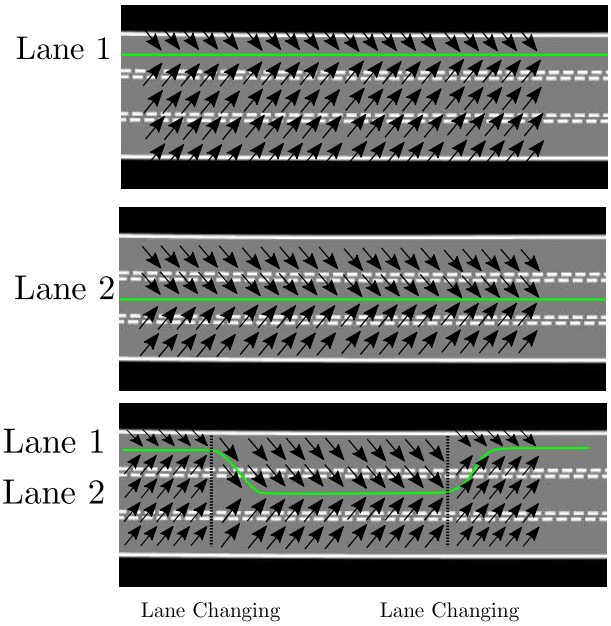


Fig. 3: For each lane the vector field is separately generated and saved, and when a lane changing command received, the appropriate force field is loaded, and lane changing takes place automatically by switching between two vector fields.

E. Obstacle repulsive force field

Collision avoidance is an essential demand for any path following approach in the navigation of autonomous cars. Keeping the current framework of using force fields to navigate the car along the desired path, we define a separate force field for each detected obstacle, in a predefined vicinity of the car, on the path. To avoid collision while overtaking the obstacles, for each obstacle a repulsive force field is defined, and the force element of each obstacle field in each point is added to the offline calculated attractive force field (F_G). The repulsive force vector of an obstacle is inversely proportional to the distance between the obstacle and the ego car. The force vector around an obstacle in each (x,y)-point is defined as:

$$F_{O_i} = \begin{cases} \frac{kd_o}{(||d_o||^2+\varepsilon)} & \text{if } \frac{(d_yS\theta+d_xC\theta)^2}{a^2} + \frac{(d_xS\theta-d_yC\theta)^2}{b^2} < 1 \\ 0 & \text{if } \frac{(d_yS\theta+d_xC\theta)^2}{a^2} + \frac{(d_xS\theta-d_yC\theta)^2}{b^2} \geq 1 \end{cases} \quad (5)$$

where d_o is a vector connecting the center of the obstacle to the center of the ego car, and $k \in \mathbb{R}$ is the tuning parameter to adjust the intensity of the repulsive force. We chose $k = ||f_P||$ to keep the intensity of the repulsive force in range of attractive force. As previously stated, the magnitude of the car velocity is constant, and the direction of velocity is controlled by the force field; therefore, the intensity of F_o is defined proportional to F_P. The area of influence for each obstacle locate at $P_O = [x_o, y_o]^\top$ is defined as an ellipse–that is rotated along the path with angle θ–that can be tuned by parameters $a, b \in \mathbb{R}$ depending on the velocity of the car and other general navigation policies. The obstacle influence threshold in the longitudinal direction of the path is longer than in lateral direction ($a > b$).

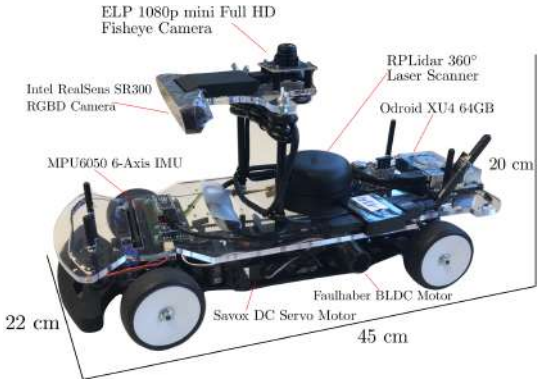
III. Simulation and Experiments

In order to evaluate the feasibility and to assess the efficiency of the proposed method, we performed simulations and experiments on a model car designed and fabricated at Freie University Berlin for research and educational purposes.

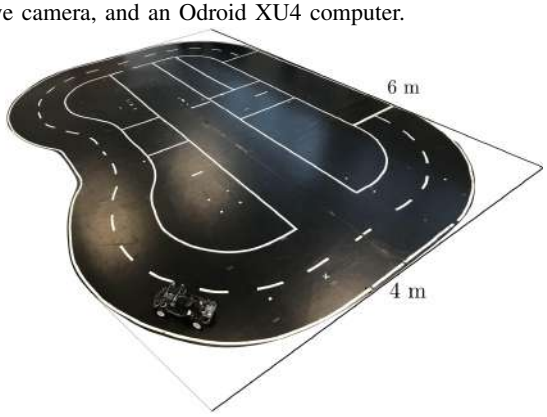
A. Experimental Setup

The model car is a four-wheel drive vehicle, which its size is one tenth of its actual counterpart, it is driven by a BLDC motor equipped with encoder, and its steering is controlled by a DC servo-motor. The perception system of the model car consists of different sensors: an IMU, a Lidar, an RGBD camera, and a fish-eye camera (Fig .4a). Data acquisition, navigation, and control are performed on-board using an Odroid XU4 computer. The model car can communicate with external systems, *e.g.,* PC, smart phone, or other model cars, via WiFi connection for monitoring, data collection, and supervisory control purposes. The whole system is powered by an Li-Po battery. The software is implemented using C++ and Python in the ROS framework installed on the Linux core of the onboard computer.

We tested the car in a test field shown in Fig .4b. In order to facilitate localization we installed four colored lamps on the ceiling over the test field. The upward looking fish-eye camera observes the lamps in each moment and using a real time range-based localization [10], localizes the vehicle on the test field. The Lidar can be used to detect and avoid static and dynamic obstacles. The RGBD camera can not only be used in conjunction with the Lidar to obstacle avoidance, but also to lane keeping.



(a) The experimental test-bed (model car), a four-wheel drive vehicle of one tenth of a real car, driven by a BLDC motor and steered by a DC servo-motor, equipped with IMU, Lidar, RGBD camera, and fish-eye camera, and an Odroid XU4 computer.



(b) The test field: a road circle with two lanes, that has connections and intersections between different its different sections, which be can inscribed in a rectangle of 6 m length and 4 m width.

Fig. 4: Model car in the test field.

The whole system, including the model car with all its subsystems and the test field, is simulated using the ROS framework and visualized by Rviz. The model car can be operated in either simulation or in real mode. The proposed approach for navigation is implemented using python [11], and validated in simulations and experiments, that are reported in the rest of this section.

B. Simulation

The proposed method is tested on a test field that is a road circle with two lanes and has connections and intersections between its different sections. The circumference of the road map can be inscribed in a 6×4 m^2 rectangle (Fig 4b). The same map is also used for simulation study. Each one of the two lanes might be the desired path.

We discretized the area to a 10 cm^2 grid. For each corner

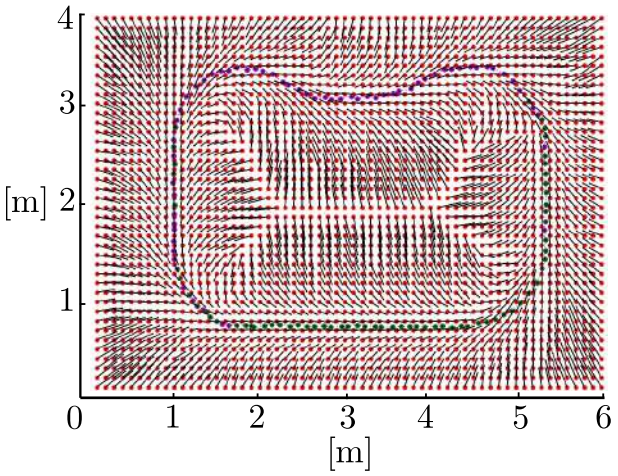


Fig. 5: Generated force field for the second lane of the road map used in simulations and experiments.

of the grids, the nearest point on the road lane is calculated based on KD-tree algorithm [12]; then, the vector field for each one of the lanes is calculated and saved separately. A visualization of the vector field for Lane 2 of the road circle (the inner lane) is shown in Fig. 5 in which the center of each grid cell is shown with a red dot and the outpointing arrow shows the appropriate heading of the car to follow the lane, which is shown by colored dots.

We evaluated the proposed approach in three test cases in order to assess the functionality of the method in different circumstances separately. The test scenarios are the followings: *a)* lane keeping with initial opposite heading with respect to the lane, *b)* lane changing, and *c)* overtaking and obstacle avoidance. In all cases the constant speed of the car is 0.6 m/s. In the following each of these simulations and their results are described.

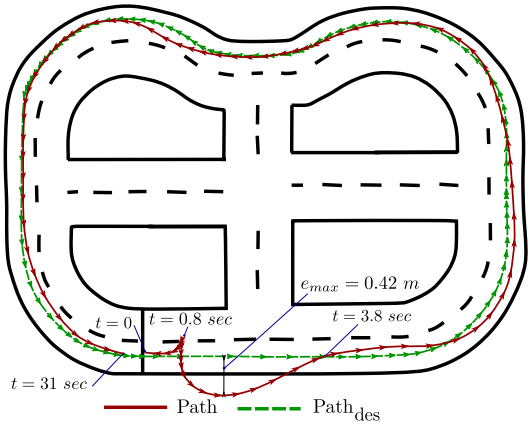


Fig. 6: Simulation result of lane keeping with initial opposite heading with respect to the lane

Lane keeping with initial opposite heading with respect to the lane: In this test, the vehicle is initially located on the lane, but with opposite heading with respect to the direction of the lane. As is shown in Fig. 6, the vehicle initially moves backward in the direction of the guiding force vectors till its heading becomes perpendicular to the path, then it takes the forward direction and the guiding force vectors attract it to the desired path. The average and maximum errors in this test were 0.04 m and 0.42 m distance from the desired path, respectively. The comparably big error from the moment the vehicle heading is perpendicular to the path till its heading becomes along the path, *i.e.*, between $t = 0.8$ to $t = 3.8$ s is due to the nonholonomic kinematic of the the vehicle and its constant velocity. In future work we consider changing the speed automatically to improve the path following performance in such cases.

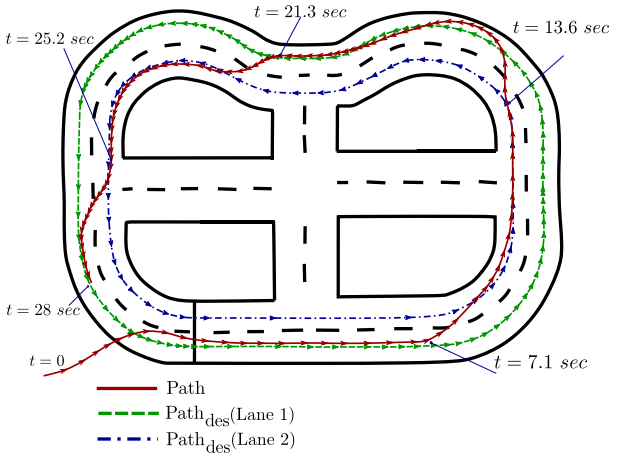


Fig. 7: Simulation result of lane changing during path following task.

Lane changing: In the real world, while driving a car, changing the driving lane is inevitable. Human drivers, considering the situation and objectives, *e.g.*, destination, time, traffic, decide to change the driving lane. As previously stated, the vector field for both driving lanes are separately generated and saved off-line. A mission planning system, that could be an AI system based on cognitive interpretation of the circumstances and objectives, could command our navigation system to change the driving lane. However, an automatic mission planning system is beyond the scope of this paper; therefore, to test the lane changing task, a human operator commands the vehicle to change the lane. The vehicle is initially located outside the road heading toward the direction of the lane, and the initial desired path is the outer lane; after reaching the lane, the human operator sends the lane changing command (using keyboard) four times, and as it can be observed from Fig. 7 the lane keeping and lane changing in constant speed is performed sufficiently accurate.

Overtaking and obstacle avoidance: Overtaking is another common action while driving a vehicle. We evaluated the overtaking task, which in effect is strongly related to obstacle avoidance. In this test, there exist two static obstacles, *e.g.*, parked vehicles, on the same lane on which the test car drives on. As is shown in the Fig. 8 our autonomous vehicle performs the overtaking maneuver and comes back to the desired path keeping its constant speed, by considering the obstacles' repulsive force vectors along with the guiding

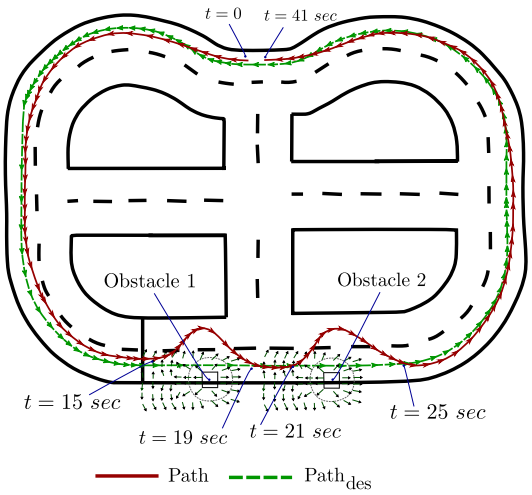


Fig. 8: Simulation result of overtaking and obstacle avoidance (lane keeping in presence of static obstacles).

vector field of the desired path. As it can be seen in Fig. 8, the overtaking actions were performed sufficiently smooth, and the path following on the rest of the path was accurate.

C. Experiment

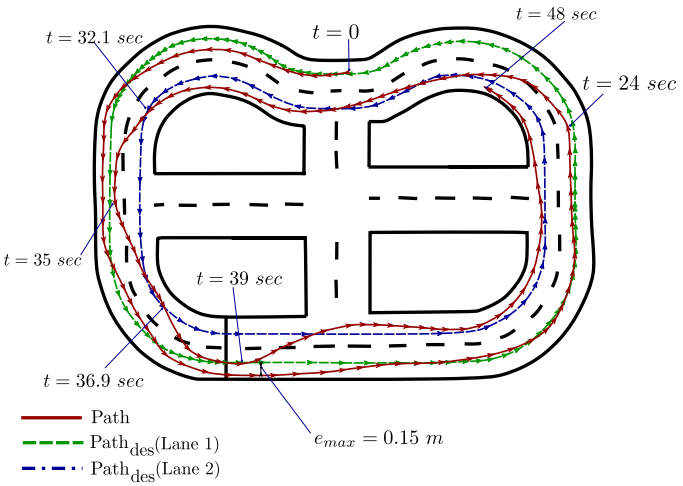


Fig. 9: Experimental result of lane changing during path following task.

In order to experimentally evaluate the proposed approach, we tested the proposed approach on the model car shown in Fig. 4a over the test field shown in Fig. 4b. We tested the lane changing task in which the lane changing commend, similarly to the simulations, is given by a human operator. The model car was initially located on the road heading along the path, and the desired constant speed of the car set to be 0.6 m/s. The human operator commanded the lane changing 6 times during the path following task, which took 48 seconds. As the result of the experiment in Fig. 9 shows, the lane changing actions took place smoothly and the lane keeping was sufficiently accurate such that in the first turn around the field, *i.e.,* between $t = 0$ to $t = 24\ s$ that no lane changing command is given, the average and maximum error were 0.045 m and 0.15 m distance from the desired path.

IV. Conclusion

We propose a vector field approach for navigation and path following of autonomous cars. The vector fields in this paper are calculated and saved offline, which allows to perform path following task in real time and with a very low computational load. In calculating the vector fields we take into account the distance from the path and the velocity of the vehicle. The vector field calculation are performed for a discretized area (area around the desired path), and for minimum and maximum velocity of the car, then in each moment and speed the appropriate force vector is calculated by interpolation on space and speed. Furthermore, we calculate online a repulsive force field in the vicinity of each observed obstacle along the path. The repulsive force from all obstacles is added to the attractive force field of the path to obtain the final force vector that specifies the desired heading of the car. This approach, as validated by simulations and experiments, is compatible with normal driving tasks such as lane keeping, lane changing, and overtaking.

References

[1] "Integrating behavioral, perceptual, and world knowledge in reactive navigation," *Robotics and Autonomous Systems*, vol. 6, no. 1, pp. 105 – 122, 1990, designing Autonomous Agents.

[2] F. Bounini, D. Gingras, H. Pollart, and D. Gruyer, "Modified artificial potential field method for online path planning applications," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, June 2017, pp. 180–185.

[3] D. R. Nelson, D. B. Barber, T. W. McLain, and R. W. Beard, "Vector field path following for miniature air vehicles," *IEEE Transactions on Robotics*, vol. 23, no. 3, pp. 519–529, 2007.

[4] N. Bloise, E. Capello, M. Dentis, and E. Punta, "Obstacle avoidance with potential field applied to a rendezvous maneuver," vol. 7, p. 1042, 10 2017.

[5] Y. Rasekhipour, A. Khajepour, S. K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1255–1267, May 2017.

[6] D. A. De Lima and G. A. S. Pereira, "Navigation of an autonomous car using vector fields and the dynamic window approach," *Journal of Control, Automation and Electrical Systems*, vol. 24, no. 1-2, pp. 106–116, 2013.

[7] "Path planning for mobile robots using bacterial potential field for avoiding static and dynamic obstacles," *Expert Systems with Applications*, vol. 42, no. 12, pp. 5177 – 5191, 2015.

[8] E. Galceran, R. M. Eustice, and E. Olson, "Toward integrated motion planning and control using potential fields and torque-based steering actuation for autonomous driving," in *2015 IEEE Intelligent Vehicles Symposium (IV)*, June 2015, pp. 304–309.

[9] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Transactions on Vehicular Technology*, vol. 66, no. 2, pp. 952–964, Feb 2017.

[10] W. Dargie and C. Poellabauer, *Fundamentals of Wireless Sensor Networks: Theory and Practice*. Wiley Publishing, 2010.

[11] AutoModelCar, neumann89, L. Freitag, pkorivi, Z. Boroujeni, juauer, and L. Sixt, "Automodelcar/catkin_ws_user: catkin_ws_user," Feb. 2018.

[12] S. Maneewongvatana and D. M. Mount, "On the efficiency of nearest neighbor searching with data clustered in lower dimensions," in *Computational Science — ICCS 2001*, V. N. Alexandrov, J. J. Dongarra, B. A. Juliano, R. S. Renner, and C. J. K. Tan, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001, pp. 842–851.