Kỷ yếu Hội nghị Quốc gia lần thứ X về Nghiên cứu cơ bản và ứng dụng Công nghệ thông tin (FAIR), Đà Nẵng, ngày 17-18/08/2017
DOI: 10.15625/vap.2017.0001

A BIDIRECTIONAL PATH PLANNING ALGORITHM FOR AUTONOMOUS PARALLEL CAR PARKING WITH SMOOTH MOVEMENT CONSIDERATION

Pham Quang Dung[1], Vu Thanh Hai[1], Nguyen Tuan Anh[1], Tu Minh Phuong[2]

[1] FPT Software, Hoa Lac Hi-tech Park, Km29, Thanglong Freeway, Thach That District, Hanoi, Vietnam

[2] Posts and Telecommunications Institute of Technology Km10, Nguyen Trai Rd., Ha Dong Dist., Hanoi, Vietnam

dungpq1@fsoft.com.vn, haivt2@fsoft.com.vn, anhnt66@fsoft.com.vn, phuongtm@ptit.edu.vn

ABSTRACT: *We consider in this paper the problem of path planning for autonomous parallel car parking in partially observable environments. In this scenario, a car is suppose to have sensors to detect obstacles around within a limited distance. We propose an algorithm based on bidirectional search for computing the path with respect to kinematic constraints to drive the car from a current position to the specified parking lot. We aim at computing a path plan for the car so that the movement is smooth because changing the control profile (turning angle, move direction) too much may result in longer execution time and more energy used. We conduct simulation experiments on different positions of the car. Simulation results show that our algorithm can find smooth itineraries to drive the car to the parking lot and the itineraries are natural as those conducted by human experiences.*

Keywords: *Autonomous vehicles, parallel parking, path planning, breadth-first search.*

I. INTRODUCTION

Research in advanced driver assistance system has received much attention from scientists of artificial intelligence community. One of the problem this domain is the path planning for autonomous car parking. The objective of this problem is to compute a path to drive the given car from a current location to a specified parking lot avoiding obstacles with respect to the kinematic model. Different technical approaches have been proposed for solving this problem. Search-based methods reply on a discrete search space [2], [7], [6]. In this approach, a neighborhhood of a state (or position) of the car is defined. This neighborhood is a set of neighboring states of the current state generated by applying a set of discretized control parameters. Then, a search heuristic, i.e., A* algorithm [3], applied to find a path from the initial state to a target state. Empirically, the path computed by such heuristics is often not smooth that it can contain unnatural swerves and unnecessary steering. Dolgov et al. in [2] proposed a post-processing algorithm with two-stage optimization method the make the path more smooth. This post-processing algorithm is based on non-linear optimization program and non parametric interpolation which is quire complicated to implement. Li et al. in [4] formulated the path planning for car parking problem as a dynamic optimization problem and proposed an algorithm based on interior point method for solving this problem. The parallel car parking problem was considered in [1, 5, 8, 9]. In those papers, algorithms based on geometry computation were proposed that find the path from a given position to the parking lot.

In this paper, we consider a case-study of parallel parking and propose an algorithm based on breadth-first search for computing the path from the current position to one of the given parking lot. In our proposed algorithm, we apply a bi-directional search strategy in order to speedup the search. More precisely, we generate in advance a set of positions that can drive the car to the target parking lot and then perform a breadth-first search from the current position to one of these generated positions. Simulation results on some generated scenarios show that the computed path is smooth and natural as human drive.

The paper is organized as follows. Section II describes the path planning in autonomous parallel car parking problem. Section III presents our proposed algorithm. Simulation results are presented in Section IV. Section V concludes the paper and gives some future works.

II. PROBLEM DESCRIPTION

Position space and position transition

Each position of the car is represented by the position of the middle of the rear axle and is represented by a tuple (x, y, θ) where x, y are coordinate in 2D surface and θ is the orientation. We denote X the set of positions of the car in the considered surface. The movement of the car must satisfy the kinematic model which specifies a new position from the current position under a control profile represented by a speed v and a steering angle ϕ (see Figure 2.1). Its dynamic is described by following equations:

- $\dot{x} = v\cos(\theta)$
- $\dot{y} = v\sin(\theta)$

$$\dot{\theta} = v\frac{\tan(\phi)}{L}$$

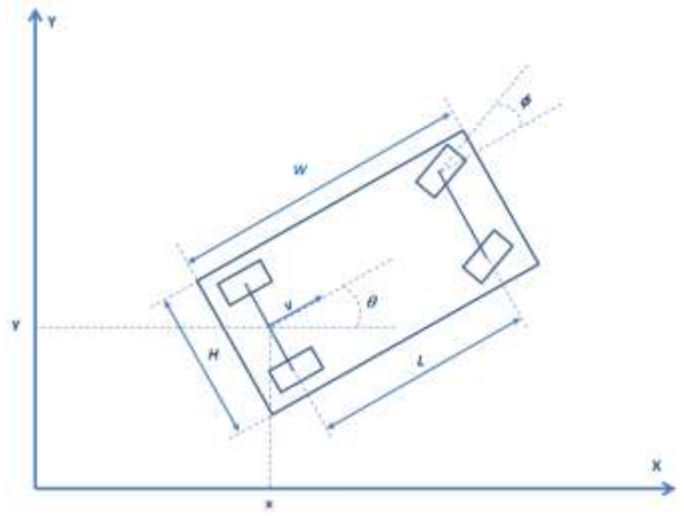


Figure 2.1. Illustration of car models

where L is the distance between the front and rear axles. Most of the research on path planning for cars in the literature discreterizes the space by considering a position transition function to specify the set of neighboring positions of the current position p with some discrete values of the parameters of the control profile. In this paper, we consider a transition function $f(p, s, \phi)$ that receives a current position p and two control parameters s and ϕ representing the move distance and the level of steering angle and returns a neighboring position of p. Values of s and ϕ can be positive and negative and are taken from a discrete set. Positive values of s mean moving forward, and negative values of s mean moving backward. Positive values of ϕ mean turning left and negative values of ϕ mean turning right. This transition function can be specified by experiments for a specific car-like vehicle. We denote $N(p, S, \Phi)$ the neighborhood function that specifes the set of neighboring position of p.

$$N(p, S, \Phi) = \{f(p, s, \phi) \mid s \in S \wedge \phi \in \Phi\}$$

Given two positions $p_1 = (x_1, y_1, \theta_1)$ and $p_2 = (x_2, y_2, \theta_2)$, $p_1, p_2 \in X$, we say that p_1 and p_2 are identical and denote $p_1 \equiv^a p_2$ if $|x_1 - x_2| < \varepsilon_1$, $|y_1 - y_2| < \varepsilon_1$, and $|\theta_1 - \theta_2| < \varepsilon_2$. Given a set of position $P = \{p_1, \ldots, p_K\}$, $(P \subseteq X)$ and a position p, we say that p belongs to P and denote $p \in^a P$ if there exists i $\in \{1, \ldots, K\}$ such that $p \equiv^a p_i$.

Problem statement

We make assumption that the car is equipped a sensor that is able the detect obstacles-free region R (for example RPLIDAR 360^o Laser Scanner). Given an initial position (x_0, y_0, θ_0) and a position of the parking lot (x^*, y^*, θ^*), the path planning problem consists of finding a sequence of positions $(x_0, y_0, \theta_0), \ldots, (x_N, y_N, \theta_N)$ such that:

- $(x_N, y_N, \theta_N) \equiv^a (x^*, y^*, \theta^*)$,
- $(x_{i+1}, y_{i+1}, \theta_{i+1}) \in N((x_i, y_i, \theta_i), S, \Phi)$; $\forall\, i = 0, 1, \ldots, N\text{-}1$
- $(x_i, y_i, \theta_i) \in R$, $\forall\, i = 0, 1, \ldots, N$

III. PROPOSED ALGORITHM

We describe in this section the proposed algorithm. We aim at finding a path with smooth movement consideration in the sense that the car will not change the steering angle or the direction too much.

Elementary movement sequences

Given a position p_0 and a drive control (s, ϕ), the sequence of positions $seq(p_0, s, \phi, k) = \langle p_0, \ldots, p_k\rangle$, where $p_i = f(p_{i\text{-}1}, s, \phi)$, $\forall i = 1, 2, \ldots, k$ is called an elementary movement sequence (or elementary path). Intuitively, an elementary movement sequence is a sequence of positions in which the drive control is kept unchanged. Figure 3.1 illustrates elementary movement sequences. Algorithm 1 depicts the procedure of generating an elementary movement sequence within a obstacles-free region R from a position p using drive control (s, ϕ) with a length constrained.

The objective of path planning algorithm proposed in this paper is to compute a path that drives the given car from the its current position to a specifed parking lot will be a sequence of elementary movement sequences in which the number of elementary movement sequences is as small as possible.

```
Input:
  • $p$: current position
  • $maxLen$: maximal length of path
  • $(s,\phi)$: drive control (move distance and steering angle)
  • $R$: obstacle-free region detected by a sensor
Output: An elementary movement sequence from the position $p$ by the drive control $(s, \phi)$
1.  $P \leftarrow \langle p \rangle$;
2.  $p_i \leftarrow p$;
3.  while Length($P$) < $maxLen$ do
4.  $p \leftarrow f(p_i, s, \phi)$;
5.  if $p_i \in R$ then
6.      $P \leftarrow P::p_i$;
7.  else
8.      BREAK;
9.   endif
10. endwhile
11. return $P$;
```

Algorithm 1. GeneratePath(*p*, *maxLen*, *s*, *ϕ*, *R*)

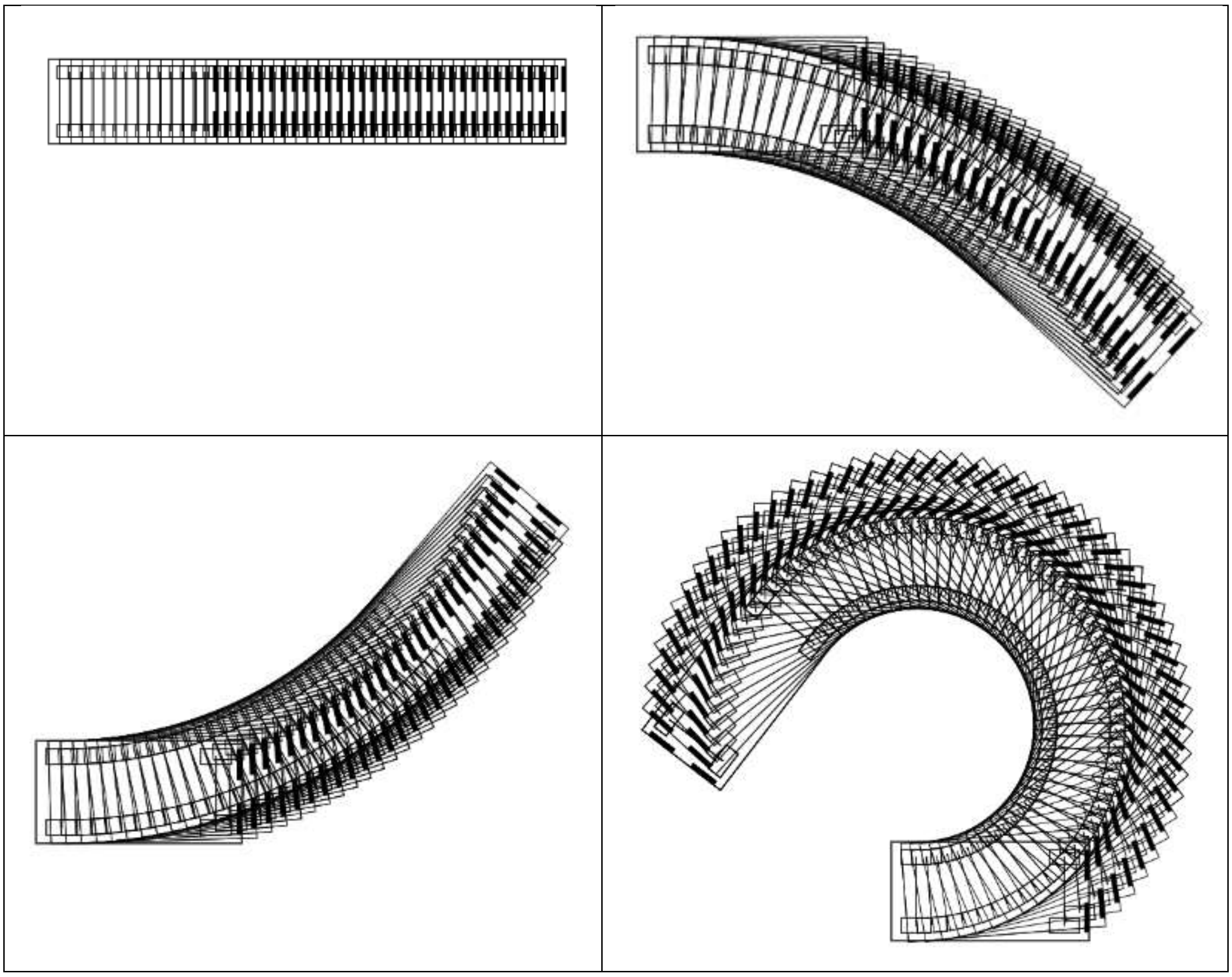


Figure 3.1. Illustration of elementary movements

```
Input:
  • $p_0 = (x_0, y_0, \theta_0)$: initial position
  • $P = \{(x_1^p, y_1^p, \theta_1^p), \ldots, (x_K^p, y_K^p, \theta_K^p)\}$: intermediate target
    positions
  • $R$: obstacle-free region detected by a sensor
  • $\Phi$: set of discretized steering angles
Output: path from $p_0$ to one of position of $P$
1.  $Q \leftarrow \varnothing$;
2.  foreach $s \in S$ do
3.      foreach $\phi \in \Phi$ do
4.          $P^e \leftarrow$ GeneratePath($p_0$, $maxLen$, $s$, $\phi$);
5.          foreach position $p$ of $P^e$ do
6.              if $p \in^a P$ then
7.                  return FormPath($p$ , $p_0$);
8.              endif
9.          endfor
10.         level($P^e$) $\leftarrow$ 1;
11.         Enqueue($Q$, $P^e$);
12.     endfor
13. endfor
14. while $Q \neq \varnothing$ do
15.     $P \leftarrow$ Dequeue($Q$);
16.     foreach $i \in \{1, 2, \ldots,$ length($P$)-1$\}$ do
17.         $p \leftarrow P[i]$;
18.         foreach $s \in S$ do
19.             foreach $\phi \in \Phi$ do
20.                 $P^e \leftarrow$ GeneratePath($p$, $maxLen$, $s$, $\phi$);
21.                 foreach position $p$ of $P^e$ do
22.                     if $p \in^a P$ then
23.                         return FormPath($p$, $p_0$);
24.                     endif
25.                 endfor
26.                 if level($P$) < $maxLevel$ -1 then
27.                     level($P^e$) $\leftarrow$ level($P$) + 1;
28.                     Enqueue($Q$, $P^e$);
29.                 endif
30.             endfor
31.         endfor
32.     endfor
33. endwhile
34. return $\varnothing$;
```

Algorithm 2. Path(p_0, $maxLen$, $maxLevel$, P, S, Φ, R)

The algorithm

We apply a bi-directional search strategy to find the path from the given initial position (x_0, y_0, θ_0) to a target parking lot (x^*, y^*, θ^*). In the first step, we find a set of positions $P = \{(x_1^p, y_1^p, \theta_1^p), \ldots, (x_K^p, y_K^p, \theta_K^p)\}$, in which from a position (x_i^p, y_i^p, θ_i^p) of P the car can move to (x^*, y^*, θ^*) by a path P_i containing no more than λ elementary movement sequences. In the second step, we find the path from (x_0, y_0, θ_0) to one of the positions in P, let say position (x_i^p, y_i^p, θ_i^p), and then, the car continues following the path P_i computed in the first step to the target position (x^*, y^*, θ^*). The remaining of the paper describes the method to solve the problem of finding a path from an initial position $p_0 = (x_0, y_0, \theta_0)$ to one of the positions of the set P.

The key idea of our algorithm is based on the breadth-first search fashion and is described as follows. First, we try to find a path from p_0 to P by one elementary movement sequence. If no such path is found, we try to find a path from (x_0, y_0, θ_0) to P by two elementary movement sequences, etc. Our proposed method is depicted in Algorithm 2. Lines 1-13 try to find a path from p_0 to P by one elementary movement sequence in which lines 2-3 scan all drive controls (s, ϕ) in the discretized S and Φ. For each control (s, ϕ), we generate a path P^e from p_0. Lines 5-9 return a path if there exists a position of P^e that belongs to P. Otherwise, we set the level of P^e by 1 and push it into the queue Q (lines 10-11). In the loop (lines 14-33), at each step, a path P in the front of the queue is popped (line 15). Lines 16-20 generate an elementary movement sequence P^e from each position p of P. If there exists a position p^e of P^e that belongs to P, then we return a path from p_0 to p^e (lines 21-25). Otherwise, we set of the level of P^e and push it into the queue Q (lines 26-29) if the level does not exceed the maximum upper bound *maxLevel*.

IV. SIMULATION

Settings

The configuration of the car in this experiments is described in Table 4.1 in which two gap parameters specifying whether two positions are the same (see Section 2.1) are $\varepsilon_1 = 0.5$ and $\varepsilon_2 = 0.005$. Table 4.2 presents the neighboring position of the position (0,0,0) under different drive control (the angle is measured in rad).

Table 4.1. Car configuration

W	H	L	ε_1	ε_2
10	20	15	0.5	0.005

Table 4.2. Transition profile: neighboring positions of position (0, 0, 0)

s	ϕ	Neighboring positions
2	0	(1.33, 0, 0)
2	0.2	(1.33, 0.02, 0.3)
2	-0.2	(1.33, -0.02, -0.3)
2	0.52	(1.33, 0.04, 0.8)
2	-0.52	(1.33, 0.04, -0.8)
-2	0	(-1.33, 0, 0)
-2	0.2	(-1.33, 0.02, -0.3)
-2	-0.2	(-1.33, -0.02, 0.3)
-2	0.52	(-1.33, 0.04, -0.8)
-2	-0.52	(-1.33, 0.04, 0.8)

In this setting, we set two cars at positions (40,35,0) and (95,35,0) and drive the given car at different positions to the parking lot between these two cars as illustrated in Figure 4.1. The car is equipped a sensor located in the front of the car. We generate 321 different positions (x_0, y_0, θ_0) of the given car where $x_0 \in \{45, 50, 55, 60, 65, 70\}$, $y_0 \in \{16, 17, 18, 19, 20, 21, 22\}$ and $\theta_0 \in \{-0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3\}$. We only consider the case that the position of the sensor is inside the dash square in Figure 4.1 so that the sensor can detect two square corners A and B from which the parking lot can be detected (composed positions from this sampling that are out of scope of of the dash square region will be dropped out).

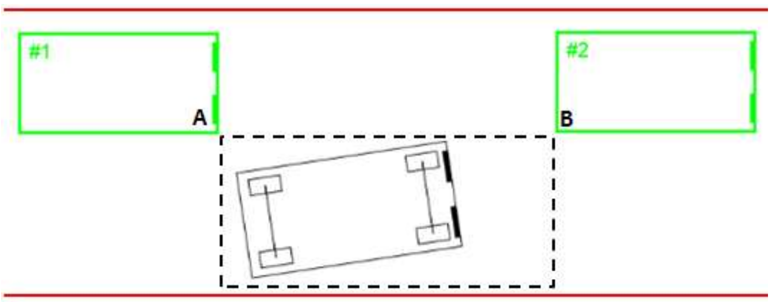
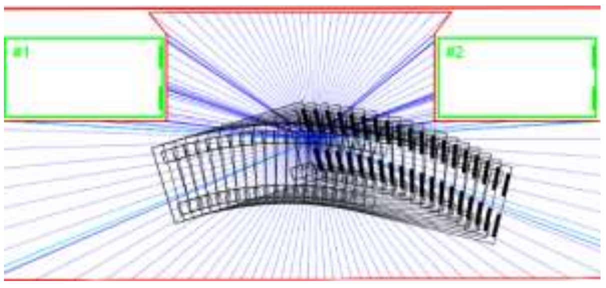
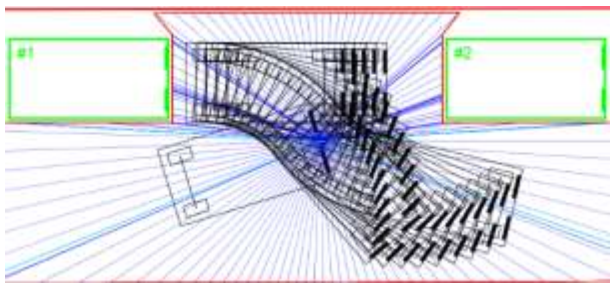


Figure 4.1. Parallel parking scenario

For generating intermediate position P, we compute paths with lengths more than 20 so that the intermediate positions generated are out of the parking lot region between two cars. Each path from these intermediate position to the parking lot contains 3 elementary movement sequences: moves backward - backward - forward (as human experiences). The simulation presented in Section 4.2 will show the path from the initial position of the car to one of the generated intermediate positions P. All the experiments are conducted on the machine Intel(R) Core(TM) i7-4600U CPU, 2.10GHz and 8GB memory.

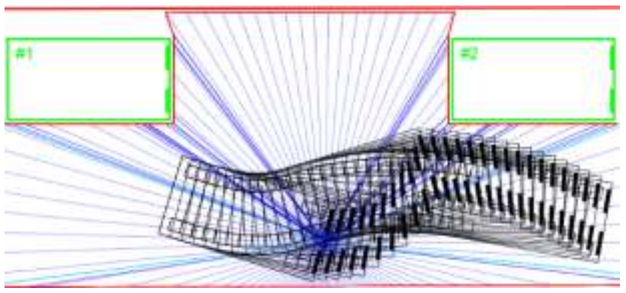


a. Step 1: Path from initial position to an intermediate position

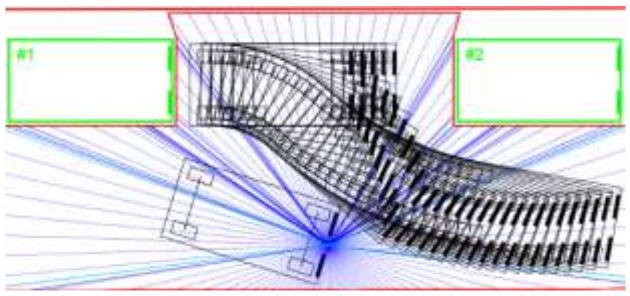


b. Step 2: Path from the intermediate position to the parking lot

Figure 4.2. Illustration of smooth parallel parking



a. Step 1: Path from initial position to an intermediate position



b. Step 2: Path from the intermediate position to the parking lot

Figure 4.3. Illustration of smooth parallel parking

Simulation results

Partial simulation results are presented in Tables 4.3, 4.4 which show the path from the initial position (x_0, y_0, θ_0) of the car to one of the intermediate position P. Columns 4 and 5 respectively present the number of position generated and the number of elementary paths for reaching the intermediate positions. Columns 6 and 7 present the time (in seconds) and the length of the path found. The results show that our algorithm is able to find the path from the initial position to intermediate positions with no more than 3 elementary paths within 8 seconds. For some easy cases, the algorithm find the path very fast with only 1 elementary path and generate only 21 positions. For some more difficult cases, the algorithm must generate up to 216131 positions in 7.6 seconds and find the path with 3 elementary paths. Figures 4.2 and 4.3 show 2 examples of the paths. We observe that the paths are smooth and natural as human experience.

Implementation in a toy car

We have implemented our algorithm on a toy car with a board Jetson TX1 equiped by a RPLIDAR. The proposed algorithm has successfully in finding a path for the toy car to move into the parking lot which is the space formed between two boxed as illustrated in Figure 4.4).

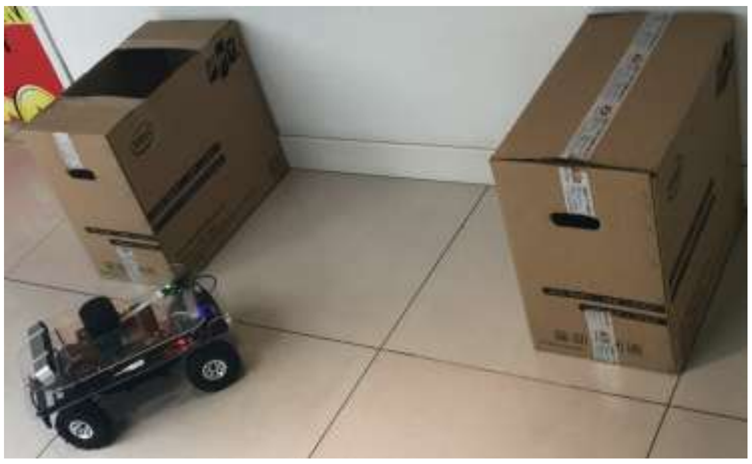


Figure 4.4. The toy car

Table 4.3. Simulation result (part I)

x_0	y_0	θ_0	*P*	*L*	*time*	*m*
50	16	0	2609	2	0.078	33
50	16	0.1	30548	3	0.942	28
50	16	0.2	16934	3	0.495	28
50	16	0.3	1203	2	0.047	31
50	17	-0.1	113203	3	3.319	32
50	17	0	19735	3	0.579	32
50	17	0.1	54459	3	1.68	28
50	17	0.2	2737	2	0.1	27
50	17	0.3	1195	2	0.027	31
50	18	-0.2	83468	3	2.741	34
50	18	-0.1	103712	3	3.334	27
50	18	0	3241	2	0.114	32
50	18	0.1	28076	3	0.858	27
50	18	0.2	1748	2	0.068	28
50	18	0.3	1125	2	0.015	30
50	19	-0.2	40396	3	1.326	32
50	19	-0.1	34948	3	1.144	27
50	19	0	3287	2	0.11	31
50	19	0.1	6137	2	0.187	26
50	19	0.2	1692	2	0.06	27
50	19	0.3	933	2	0.02	26
50	20	-0.3	2011	2	0.07	35
50	20	-0.2	51782	3	1.657	33
50	20	-0.1	90242	3	2.938	32
50	20	0	3372	2	0.109	29
50	20	0.1	6062	2	0.204	24
50	20	0.2	25476	3	0.813	26
50	20	0.3	34948	3	1.094	29
50	21	-0.3	2239	2	0.08	34
50	21	-0.2	19676	3	0.683	28
50	21	-0.1	3504	2	0.11	27
50	21	0	3306	2	0.113	27
50	21	0.1	25325	3	0.825	23
50	21	0.2	2926	2	0.104	25
50	21	0.3	21625	3	0.712	35
50	22	-0.3	2306	2	0.09	33
50	22	-0.2	32282	3	1.241	28
50	22	-0.1	3547	2	0.126	24
50	22	0	2962	2	0.075	29
50	22	0.1	2402	2	0.088	27
50	22	0.2	43773	3	1.618	34
50	22	0.3	1041	2	0.034	27
50	23	-0.3	2395	2	0.086	32
50	23	-0.2	4686	2	0.161	27
50	23	-0.1	3363	2	0.11	25
50	23	0	1892	2	0.059	30

Table 4.4. Simulation result (part II)

x_0	y_0	θ_0	*P*	*L*	*time*	*m*
70	16	0	127947	3	3.624	29
70	16	0.1	99042	3	3.235	39
70	16	0.2	12200	3	0.388	21
70	16	0.3	410	2	0.029	21
70	17	-0.1	36091	3	1.14	23
70	17	0	141910	3	4.057	23
70	17	0.1	36817	3	1.11	28
70	17	0.2	2945	2	0.078	20
70	17	0.3	383	2	0.01	20
70	18	-0.2	85639	3	2.733	30
70	18	-0.1	2873	2	0.11	22
70	18	0	177693	3	5.7	22
70	18	0.1	31424	3	1.02	27
70	18	0.2	2888	2	0.111	22
70	18	0.3	266	2	0.01	19
70	19	-0.2	115830	3	3.759	32
70	19	-0.1	87166	3	3.112	36
70	19	0	36361	3	1.145	39
70	19	0.1	1635	2	0.062	25
70	19	0.2	8453	2	0.279	22
70	19	0.3	211	2	0.007	19
70	20	-0.3	36212	3	1.188	23
70	20	-0.2	8123	2	0.288	29
70	20	-0.1	109099	3	3.536	36
70	20	0	27363	3	0.883	23
70	20	0.1	1431	2	0.046	19
70	20	0.2	9094	2	0.301	23
70	20	0.3	31	1	0.001	19
70	21	-0.3	23118	3	0.77	21
70	21	-0.2	46837	3	1.525	32
70	21	-0.1	10410	2	0.42	25
70	21	0	1791	2	0.074	24
70	21	0.1	37781	3	1.249	20
70	21	0.2	2388	2	0.063	19
70	21	0.3	2601	2	0.092	21
70	22	-0.3	23576	3	0.984	22
70	22	-0.2	83962	3	2.783	30
70	22	-0.1	9601	2	0.344	26
70	22	0	1829	2	0.052	24
70	22	0.1	2673	2	0.108	18
70	22	0.2	1663	2	0.05	13
70	22	0.3	1873	2	0.038	16
70	23	-0.3	2019	2	0.089	18
70	23	-0.2	2963	2	0.11	15
70	23	-0.1	2048	2	0.065	28
70	23	0	559	2	0.031	15

V. CONCLUSION

We consider in this paper the path planning problem in autonomous parallel car parking taking into account the smooth movements criteria. We proposed a bi-directional algorithm based on breadth-first search for solving this problem. The proposed algorithm is easy to implement. Experimental results on different scenarios show that our algorithm is able to find a smooth path to drive the car from the given position to the parking lot. We will consider the path planning for autonomous cars on partially observable environments.

REFERENCES

[1] Sungwoo CHOI and Clement Boussard and Brigitte d'Andrea-Novel. Easy Path Planning and Robust Control for Automatic Parallel Parking. 18th IFAC World Congress pages 656-661, 2011.

[2] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. The international Journal of Robotics Rearch, 29(5):485-501, 2010.

[3] P. E. Hart, N. J. Nilsson, and B. Raphael. Correction to "a formal basis for the heuristic determination of minimum cost paths". ACM SIGART Bulletin, 37:28-29, 1972.

[4] B. Li and Z. Shao. A uni_ed motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles. Knowledge-Based Systems, 86:11-20, 2015.

[5] Y. K. Lo, A. B. Rad, C. W. Wong, and M. L. Ho. Automatic parallel parking. In IEEE Conf. Intelligent Transportation Systems, pages 1190{1193, 2003.

[6] C. Pradalier, S. Vaussier, and P. Corke. Path planning for a parking assistance system: implementation and experimentation. In Australasian Conference on Robotics and Automation (ACRA 2005), 2005.

[7] A. Vorashompoo, B. Panomruttanarug, and K. Higuchi. Bidirectional best _rst based autonomous parallel parking system. In The 8th Conference o Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology (ECTI-CON), 2011, pages 593-596, 2011.

[8] H. Vorobieva, N. Minoiu-Enache, S. Glaser, and S. Mammar. Geometric continuouscurvature path planning for automatic parallel parking. In Proceedings of 10th IEEE International Conference on Networking, Sensing and Control, ICNSC, pages 418-423, 2013.

[9] A. Zhdanov, D. Klimov, V. Korolev, and A. Utemov. Modeling parallel parking a car. Int. J. Comput. Syst. Sci, 47(6):907-917, 2008.

THUẬT TOÁN TÌM ĐƯỜNG ĐI DỰA TRÊN TÌM KIẾM 2 CHIỀU CHO BÀI TOÁN ĐẬU XE TỰ ĐỘNG VÀO BÃI TÍNH ĐẾN YẾU TỐ MƯỢT CỦA DI CHUYỂN

Phạm Quang Dũng, Vũ Thanh Hải, Nguyễn Tuấn Anh, Từ Minh Phương

TÓM TẮT: *Chúng tôi khảo sát bài toán tìm đường đi cho xe di chuyển tự động vào bãi đỗ. Trong ngữ cảnh này, chiếc xe được trang bị một cảm biến có thể dò được các chướng ngại vật xung quanh với khoảng cách hữu hạn. Chúng tôi đề xuất thuật toán tìm đường đi dựa trên tìm kiếm 2 chiều để cho xe di chuyển từ vị trí hiện tại vào bãi đậu xe tuân theo luật di chuyển của xe. Trong thuật toán đề xuất, chúng tôi tính đến tính mượt của di chuyển vì nếu lệnh điều khiển (góc đánh lái, hướng di chuyển) thay đổi quá nhiều và vụn vặt thì quãng đường di chuyển không tối ưu và tiêu hao nhiều nhiên liệu. Chúng tôi đã cài đặt thử nghiệm thuật toán đề xuất, kết quả chạy mô phỏng cho thấy trong đa số các trường hợp, thuật toán tìm ra đường đi giúp xe di chuyển tự động vào bãi đỗ một cách mượt mà như đường đi do lái xe điều khiển dựa trên kinh nghiệm.*