Local Path Planning Based On Particle Filter Optimization

Langcheng Huo†
Shenzhen Institute of Artificial Intelligence and Robotics for Society, China.
Shenzhen, China
elenhuo@gmail.com

Yuncai Wu†
Shenzhen Institute of Artificial Intelligence and Robotics for Society, China.
Shenzhen, China
LSXHShen@163.com

***Abstract*—To improve the running efficiency of vehicles in navigation, we present a novel algorithm for local path planning combined with the global reference path(consists of discrete points) to safely navigate the differential wheel vehicle in unstructured environments. Each local path consists of two-segment approximate arcs and the particle filter is used to optimize the cost function to ensure two-segment approximate arcs fit the global reference path and avoid obstacles. Besides, a set of Fibonacci Numbers is used to generate the particle swarms to make particle filter converge faster. We design several path following experiments to prove that our algorithm follows the arbitrary global path and avoids the obstacle more efficiently. Our algorithm can also be applied to other mobile robots.**

***Index Terms*—local path planning, particle filter, obstacles avoidance, path following.**

I. Introduction

Path planning is crucial in navigation. It is made up of global path planning and local path planning. Global path planning generates a global path from the current position of vehicle to the goal in a known map. There are plenty of global path planning algorithms that are often used, such as Dijkstra [1], A star [2], RRT [3].

Local path planning plays an important role in avoiding obstacles and following a global path during navigation. Many scholars research local path planning, including Genetic algorithm [4], Artificial potential field [5], Bug algorithm [6]. In 1997, Fox proposed *dynamic window approach* (DWA) [7]. It considers the physical constraints of the vehicle, environmental constraints, and other factors. Because of its reliability and stability, it is extensively applied in the local path planning. However, there are some drawbacks when in a complex environment. 1) The trajectory is not smooth. 2) In the intensive obstacles areas, the vehicle may not choose the shortest route, from which the vehicle needs to go through the obstacles so that the total route is much longer. 3) The vehicle may be stuck in the local minimum. Its primary cause is that the fixed parameters can not adapt to each dynamic and complex environment.

Based on those problems, some scholars put forward improvement methods. In 1999, the Global Dynamic Window Approach [8] uses an NF1 navigation function to tackle the local minimum. In 2005, Ogren presented a version of the DWA [9] in a model predictive control that is tractable and convergent. To adapt to various environments, [10] applied the fuzzy logic controller to alter the weight parameters of DWA to fit different circumstances. To enhance the dynamic obstacle avoidance, [11] proposed a prediction method to acquire the motion of other objects in the future. It regards the dynamic obstacles as polygons with velocities and predicts its future collisions so that it can avoid the obstacles in advance. The algorithm implementation is prominent at reducing collisions and fosters navigational performance.

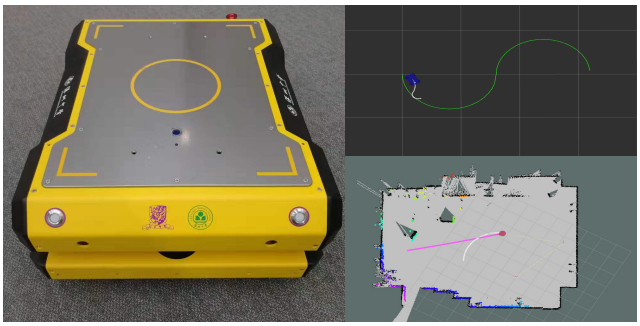

Fig. 1. a local path made up of two-segment approximate arcs fits the global reference path by using particle filter. It's denoted by the white arcs. We have tested our method in the simulation and authentic context with an AGV.

While the angle $\Delta\phi$ (see in Fig. 3) between the orientation of vehicle and orientation of a global path is too large, DWA turns a big circle and performs poorly. Those improvements above do not fix it. Based on this, in this paper, we propose a novel local path planning method. We generate the particle swarm randomly to produce local paths that are made up of two-segment approximate arcs. After that, we use the particle filter to optimize the swarm and cost function we define to select the best local path. Our algorithm mainly takes into account the smoothness of moving trajectory, motion constraint of vehicle, the fitting degree between local path planning and global path planning. By integrating these, the vehicle can move along with the global path towards the goal safely and quickly. The contribution of this paper is proposing a algorithm using two-segment approximate arcs and particle swarm to fix the problem above rather than using controller judgement. Besides, it fits the global path better.

† The authors are contributed equally to this work.

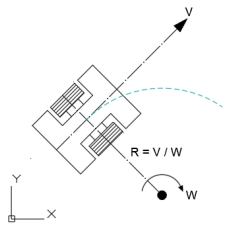

Fig. 2. The kinematic model of vehicle

To validate the effectiveness of our algorithm, we design experiments of path following and obstacle avoidance to compare with two state-of-art planners, namely pure pursuit [12] and DWA that the local planner in ROS navigation stack. And we turn off the orientation controller judgement of the DWA. Experimental results show that our algorithm has the shortest path and has a better performance than others about running efficiency. We also use our algorithm to replace the local path planner of the ROS navigation stack [13] to drive an AGV in an authentic environment, see Fig. 1.

The rest of this paper will be organized as follows. Sec. II illustrates the definitions of the vehicle's kinematic model, environmental map, global reference path, and cost function. Sec. III introduces our algorithm flow and implementation. Finally, we show the experiments and results in Sec. IV, conclusion and future work in Sec. V.

II. Preliminaries

A. Vehicle Description

The differential wheel vehicle is a non-holonomic constraint system, see Fig. 2, the kinematic model [14] is shown as follows:

$$\begin{aligned} \dot{x}(t) &= v\cos\theta \\ \dot{y}(t) &= v\sin\theta \\ \dot{\theta}(t) &= w \end{aligned} \tag{1}$$

As describes in Eq. (1), the vehicle is nonlinear. Let $z(t) = (x(t), y(t), \theta(t))^T$ denotes a vehicle on the plane $W = R^2$. The v denotes the linear velocity of the vehicle, w denotes the angular velocity of the vehicle and θ denotes the orientation of the vehicle.

B. Environmental Map

In this paper, the static obstacle is captured in an occupancy grid map, which is constructed by a SLAM algorithm—gmapping. The map including unknown space is denoted by $\Theta \subset W$, free space is denoted by $F \subset W$, occupied space is denoted by $\overline{F} \subset W$ and obstacles space is denoted by $O \subset W$.

C. Global Reference Path

A global reference path is the key for the vehicle to arrive at the goal. The global reference path can be generated by a global planner and artificially designed. We consider a global reference path $P \subset F$ consisting of a sequence of path segments connecting M waypoint $p_m^T = [x_m^p, y_m^p]^T \in W$ with $m \in M = \{1, \ldots, M\}$. To ensure that the vehicle reaches the goal, the global reference path is collision free generally.

D. Problem Formulation

To guarantee that the local path planner generates the best path, we define a cost function J to evaluate a local path. The J includes the penalty for $\frac{v_1}{w_1}$, deviations between the local path and the global reference path, the period time t_1. This is formulated in the optimization problem

$$\begin{aligned} J = \min_{u_1,u_2} \sum_{k=1}^{N-1} & J_{tra}(z_k, u_1, u_2, t_1, t_2) + J_{smooth}(t_1, t_2, \frac{v_1}{w_1}) \\ & + J_{dist}(z_k, u_1, u_2, t_1, t_2, m_l) \end{aligned} \tag{2}$$

$$\text{s.t.} \quad z_k = f(z_{k-1}, u_1), \tag{2a}$$

$$v_1, w_1 \in u_1,\ u_1, u_2 \in U,\ z_k \in Z,\ z_0\ given. \tag{2b}$$

$$z_k \cap O \cap \overline{F} = \emptyset, \tag{2c}$$

$$0 < t_1, t_2 < t_{const},\ t_1 + t_2 = t_{const}. \tag{2d}$$

k is the k_{st} reference point(behind the waypoint) on the path. In order to ensure the local path is close to the global reference path, we select 4 path points as reference points, see Fig. 4. u_1 and u_2 are the set of states and control inputs. t_1 is the interval first stage that duration of first control input, t_2 is the interval second stage that duration of second control input, m_l denotes the local map that is generated by the lidar point clouds. Because the Eq. (2) is nonlinear, the particle filter is used to solve the cost function.

$J_{tra}(z_k, u_1, u_2, t_1, t_2)$ is related to the deviation between the local path and the global reference path, $J_{smooth}(t_i, \frac{v_1}{w_1})$ is related to the sampling time of the first stage and the reference value in control command, $J_{dist}(z_k, u_1, u_2, t_1, t_2, m_l)$ is related to the distance between the local path and obstacles. The details of the cost function are explained in Sec.III.

III. Method

In this section, we describe the algorithm of local path planning and solve the general problem of Eq. (2) by particle filter.

A. Generate the particle swarm

Each particle $c_j(r_1, r_2, t_1, t_{const})$ in particle swarm $C(c_1, c_2, \ldots, c_n)$ is generated by randomly sampling, and t_{const} is constant. $r_1, r_2 \in R$ denotes the rotation radius of the vehicle. In the experiment, in order to make the particle filter converge much faster, R is a set of Fibonacci Numbers, $R = \{-23.3, -14.4, -8.9, -5.5, -3.4, -2.1, -1.3, -0.8, -0.5, -0.3, -0.2, -0.1, 0, 0.1, 0.2, 0.3, 0.5, 0.8, 1.3, 2.1, 3.4, 5.5, 8.9, 14.4, 23.3\}$. t_1 denotes the time when the vehicle moved in the first stage($r_1 = \frac{v_1}{w_1}$), ($t_2 = t_{const} - t_1$) denotes the time when the vehicle moved in the last stage($r_2 = \frac{v_2}{w_2}$). We set $t_1 < \frac{\pi}{w_1}$ so that it satisfies the uniqueness of optimization results on particle filter.

For each particle, there are three parameters(r_1, r_2, t_1) generated by randomly sampling, it is used to generate a local path.

B. Generate the local path

Each particle c_j generates a local path $P_l = \{p_0^l, p_1^l, \ldots, p_n^l\}$ consisting of a series of discrete points. Each point contains location information at different times, such as $p_t^l(x_t, y_t) \in P_l$. We constrain (v_{const}, w_{const}) as a speed limit for the vehicle and base on (r_1, r_2) of each particle to generate $(v_1, w_1)^T$ and $(v_2, w_2)^T$, which will create two-segment standard arcs respectively on a local path.

$$(v_1, w_1)^T = \begin{cases} (v_{const}, \frac{v_{const}}{r_1})^T & \frac{v_{const}}{r_1} < w_{const} \\ (w_{const} * r_1, w_{const})^T & \frac{v_{const}}{r_1} > w_{const} \end{cases} \tag{3}$$

$$(v_2, w_2)^T = \begin{cases} (v_{const}, \frac{v_{const}}{r_2})^T & \frac{v_{const}}{r_2} < w_{const} \\ (w_{const} * r_2, w_{const})^T & \frac{v_{const}}{r_2} > w_{const} \end{cases} \tag{4}$$

The larger (v_{const}, w_{const}) is, the longer the local path is. If v_{const} is so large that the local path is too scattered, which will affect the calculation of distance deviation from the local path to a global reference path. If v_{const} is so small that the local path is too short to fit the global reference path well. In our experiment, $(v_{const}, w_{const}) = (0.3, 0.3)$.

To satisfy the dynamic constraint of the vehicle, We consider the acceleration $(\dot{v}, \dot{w})$ of the vehicle in the process of generating the trajectory. The acceleration is $(\dot{v}, \dot{w}) = (\dot{v}_{max}, \dot{w}_{max})$. In each sampling time Δt_1, the velocity command (v_t, w_t) is:

$$v_t = \begin{cases} v_{t-1} + \frac{v_1 - v_{t-1}}{|v_1 - v_{t-1}|}\dot{v}_{max} \cdot \Delta t_1, & \frac{|v_{t-1} - v_1|}{\Delta t_1} > \dot{v}_{max}, \\ & t \le t_1; \\ v_{t-1} + \frac{v_2 - v_{t-1}}{|v_2 - v_{t-1}|}\dot{v}_{max} \cdot \Delta t_1, & \frac{|v_{t-1} - v_2|}{\Delta t_1} > \dot{v}_{max}, \\ & t_1 \le t \le t_{const}; \\ v_1, & \frac{|v_{t-1} - v_1|}{\Delta t_1} < \dot{v}_{max}, \\ & t \le t_1; \\ v_2, & \frac{|v_{t-1} - v_2|}{\Delta t_1} < \dot{v}_{max}, \\ & t_1 \le t \le t_{const}; \end{cases} \tag{5}$$

$$w_t = \begin{cases} w_{t-1} + \frac{w_1 - w_{t-1}}{|w_1 - w_{t-1}|}\dot{w}_{max} \cdot \Delta t_1, & \frac{|w_{t-1} - w_1|}{\Delta t_1} > \dot{w}_{max}, \\ & t \le t_1; \\ w_{t-1} + \frac{w_2 - w_{t-1}}{|w_2 - w_{t-1}|}\dot{w}_{max} \cdot \Delta t_1, & \frac{|w_{t-1} - w_2|}{\Delta t_1} > \dot{w}_{max}, \\ & t_1 \le t \le t_{const}; \\ w_1, & \frac{|w_{t-1} - w_1|}{\Delta t_1} < \dot{w}_{max}, \\ & t \le t_1; \\ w_2, & \frac{|w_{t-1} - w_2|}{\Delta t_1} < \dot{w}_{max}, \\ & t_1 \le t \le t_{const}; \end{cases} \tag{6}$$

Because of the dynamic constraint, the vehicle's speed can not jump all at once. The trajectory of the local path transforms two-segment standard arcs to two-segment approximate arcs.

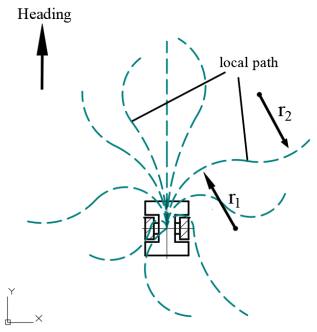

Fig. 3. Trajectories of the local path. Each particle generates a local path made up of discrete points p_t^l. There are ten local paths of different lengths in the picture. Each local path can be viewed as a two-segment approximate arc. r_1, r_2 are sampling parameters of one particle.

The vehicle is a differential wheel structure, for each velocity command (v_t, w_t), it moves in a circle around the center $p_o = (x_t^o, y_t^o)^T$, and the radius of the arc is $r_o = \frac{v_t}{w_t}$. $p_o = (x_t^o, y_t^o)^T$ is generated by:

$$\begin{bmatrix} x_t^o \\ y_t^o \\ 1 \end{bmatrix} = \begin{bmatrix} cos\theta_{t-1} & -sin\theta_{t-1} & x_{t-1} \\ sin\theta_{t-1} & cos\theta_{t-1} & y_{t-1} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ \frac{v_t}{w_t} \\ 1 \end{bmatrix} \tag{7}$$

Simultaneously, via Eq. (3), (4), (5), (6), (7), we can obtain a local path P_l made up of discrete points $p_t^l = (x_t, y_t, \theta_t)^T$ by:

$$\begin{aligned} x_t &= x_t^o + \frac{v_t}{w_t} sin(\theta_{t-1} + w_t \cdot \Delta t_1) \\ y_t &= y_t^o - \frac{v_t}{w_t} cos(\theta_{t-1} + w_t \cdot \Delta t_1) \\ \theta_t &= \theta_{t-1} + w_t \cdot \Delta t_1 \end{aligned} \tag{8}$$

In Eq. (8), the vehicle's trajectory is a circle about the center p_o with radius r_o duration the Δt_1 (in the experiment, $\Delta t_1 = 0.1s$). Overall, the local path can be viewed as two-segment approximate arcs, because the vehicle is restricted by an acceleration in Eq. (5), (6).

Compared to the global reference path, the points of the local path are much denser, and the local path is satisfied with the dynamics constraint of the differential wheel vehicle. The trajectory of the local path can be seen in Fig. 3.

C. Set the cost function

The cost function Eq. (2) is used to judge the performance of each local path. It consists of three parts. In order to keep the local path close to the global path, an error term is defined, see Fig. 4.

The error term of the cost function is:

$$\begin{aligned} &\sum_{k=1}^{N-1} J_{tra}(z_k, u_1, u_2, t_1, t_2) = \alpha(e_1 + e_2 + e_3 + e_4) \\ &\text{s.t.} \quad |p_m p_{m+1}| = |p_0^l p_a^l| \\ &\qquad |p_m p_{m+2}| = |p_0^l p_b^l| \\ &\qquad |p_m p_{m+3}| = |p_0^l p_c^l| \\ &\qquad |p_m p_{m+4}| = |p_0^l p_d^l| \end{aligned} \tag{9}$$

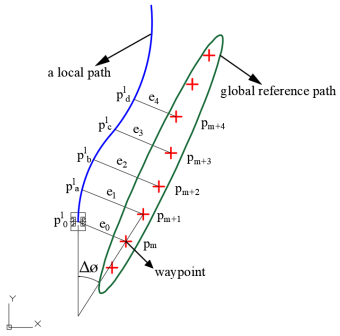

Fig. 4. The deviation between the local path and a global reference path is exaggerated in the figure. p_0^l not only denotes the beginning of the local path but also the current position of the vehicle. $p_a^l, p_b^l, p_c^l, p_d^l \in P_l$ denotes discrete points of a local path, $p_m \in P$ denotes the waypoint that is nearest point to the vehicle on the global path. $p_{m+1}, p_{m+2}, p_{m+3}, p_{m+4}$ denotes the reference points. e_0 denotes the bias between p_0^l and p_m, e_1, e_2, e_3, e_4 are the same above. $\Delta\phi$ denotes the angle between the orientation of vehicle and orientation of global reference path.

If Eq. (9) is much smaller, it drives the vehicle to follow the global reference path as much as possible, α is a weight parameter.

To make the vehicle move more smoothly, we expect the vehicle not to rotate as much as possible so that the time when the vehicle move in the first stage t_1 is as large as possible(a small t_1 makes the vehicle more likely to rotate, which results in an unsmooth behavior). Thus, the smooth cost function is defined by:

$$J_{smooth}(t_1, t_2, \frac{v_1}{w_1}) = \beta \exp\left(\frac{-t_1}{t_1 + t_2}\right) + \gamma \exp\left(-\left|\frac{v_1}{w_1}\right|\right) \tag{10}$$

In Eq. (10), β and γ are weight parameters. $r_1 = \frac{v_1}{w_1}$ denotes that the vehicle moves around a circle of radius r_1 in the first stage t_1.

The obstacle avoidance is important to vehicles, so we define a cost function $J_{dist}(z_k, u_1, u_2, t_1, t_2, m_l)$ about the distance of the vehicle to obstacles, which is related to the surrounding information m_l (local map) and the local path. We estimate the discrete points p_i^l of the local path whether in the obstacle space O or occupied space $\overline{F}$, which can decrease the calculation time, so the obstacle cost function is:

$$J_{dist} = \begin{cases} \sigma & p_t^l \in \overline{F} \cup p_t^l \in O \\ 0 & else \end{cases} \tag{11}$$

To ensure that the vehicle does not collide with obstacles, obstacles should be inflated in the map, otherwise, the vehicle is likely to have a collision.

To sum up, there are 4 parameters can be adjusted in the cost function J, in our experiment, $(\alpha, \beta, \gamma, \sigma) = (1, 0.1, 1, 3)$.

D. Solve the cost function by particle filter

The cost function Eq. (2) related to the local map is nonlinear, it is difficult to get an analytical solution, so we solve the cost function by particle filter.

It is simple for the optimization process of the particle filter. First, generate the particle swarms $C(c_1, c_2, \ldots, c_n)$ that satisfy the condition(Sec. III-A) by randomly sampling. Finally, multiple resampling performs according to the particle of the smallest cost value that is calculated by the cost function.

For particle filter, theoretically, a larger particle swarm along with a larger number of samples would have a better effect, but it will spend too much time to meet the real-time performance. Considering that only three parameters (r_1, r_2, t_1) need to be optimized and Fibonacci sequence is adopted for coding, the size of initial particle swarm is 400 and the iteration times of resampling is six that are suitable for our experiment.

E. Control

According to the optimized local path, select the best particle $c_{best}(r_1, r_2, t_1, t_{const})$ as the control input, the control method is:

$$v_t^c = \begin{cases} v_{t-1}^c + \frac{kv_1 - v_{t-1}^c}{|kv_1 - v_{t-1}^c|}\dot{v}_{max} \cdot \Delta t_c & \frac{|kv_1 - v_{t-1}^c|}{\Delta t_c} > \dot{v}_{max} \\ kv_1 & \frac{|kv_1 - v_{t-1}^c|}{\Delta t_c} \leq \dot{v}_{max} \end{cases} \tag{12}$$

$$\text{s.t.} \quad |kv_1| < |v_{max}|$$

$$w_t^c = \begin{cases} w_{t-1}^c + \frac{kw_1 - w_{t-1}^c}{|kw_1 - w_{t-1}^c|}\dot{w}_{max} \cdot \Delta t_c & \frac{|kw_1 - w_{t-1}^c|}{\Delta t_c} > \dot{w}_{max} \\ kw_1 & \frac{|kw_1 - w_{t-1}^c|}{\Delta t_c} \leq \dot{w}_{max} \end{cases} \tag{13}$$

$$\text{s.t.} \quad |kw_1| < |w_{max}|$$

Where (v_{t-1}^c, w_{t-1}^c) is the current velocity of the vehicle, (v_t^c, w_t^c) is the output of the controller, Δt_c denotes the controller period, k denotes the adjustment coefficient that determines the actual velocity of the vehicle, $k < \frac{\Delta t_1}{\Delta t_c}$ is suitable for the experiment. (v_{max}, w_{max}) denotes the maximum linear velocity and maximum angular velocity of the vehicle.

Algorithm 1 Summary of local path planning by particle filter

1: Get the map and local map m_l ;
2: Get the global reference path P;
3: Get the state $(x, y, \theta, \dot{x}, \dot{y}, \dot{\theta})$ of vehicle;
4: Generate the particle swarm by randomly sampling;
5: Generate the trajectory of local path for each particle by Eq. (3), (4), (5), (6), (7), (8);
6: Calculate the cost function of each particle by Eq. (2);
7: **for** $i \in \{1, \ldots, n\}$ **do**
8: Resampling according to the optimal particle;
9: Generate the trajectory of local path for each particle;
10: Calculate the cost function of each particle;
11: **end for**
12: Select the best particle and calculate the (v_t^c, w_t^c) by Eq. (3), (12), (13);
13: Apply (v_t^c, w_t^c) to system;

Algorithm 1 summarizes our method. It should be noted that Algorithm 1 is only one control period. The vehicle will take its state, local map and global reference path as the input of the system and get the output (v_t^c, w_t^c) in each period.

IV. Experiment Results

In order to test the performance of our algorithm, we do experiments in the simulation environment based on ROS [15]. We compare DWA that the local planner in ROS navigation stack, Pure pursuit [12] and our algorithm with the same restriction of velocity and acceleration in three different scenarios(straight line, S-shape curve and W-shape curve) that are shown in Fig. 5 to test the performance of vehicle following arbitrary paths. Meanwhile, We consider the influence of the different initial angle deviation $\Delta\phi$(see Fig. 4) between the initial orientation of the vehicle and the path direction. The authentic trajectory, distance deviation e_0(see Fig. 4) between vehicle's current location and the waypoint that is nearest to the vehicle in global reference path, and average time (complete a task) will be used to judge the experiment results. The distance deviation e_0 is oscillating (see Fig. 6) since the waypoint is discrete as well as the global reference path.

Different algorithms generate different trajectories to drive the vehicle to move, shown in Fig. 5. Although Pure pursuit performs well when $\Delta\phi$ is small, it performs poorly when the initial direction of the vehicle is opposite to the direction of the global reference path ($\Delta\phi = \pi$). Because the radius of rotation of the vehicle is so large that the vehicle moves around in a big arc, see Fig. 5(b). In contrast, our algorithm performs much better than others, see Fig. 5(b), (e). In Fig. 6, we

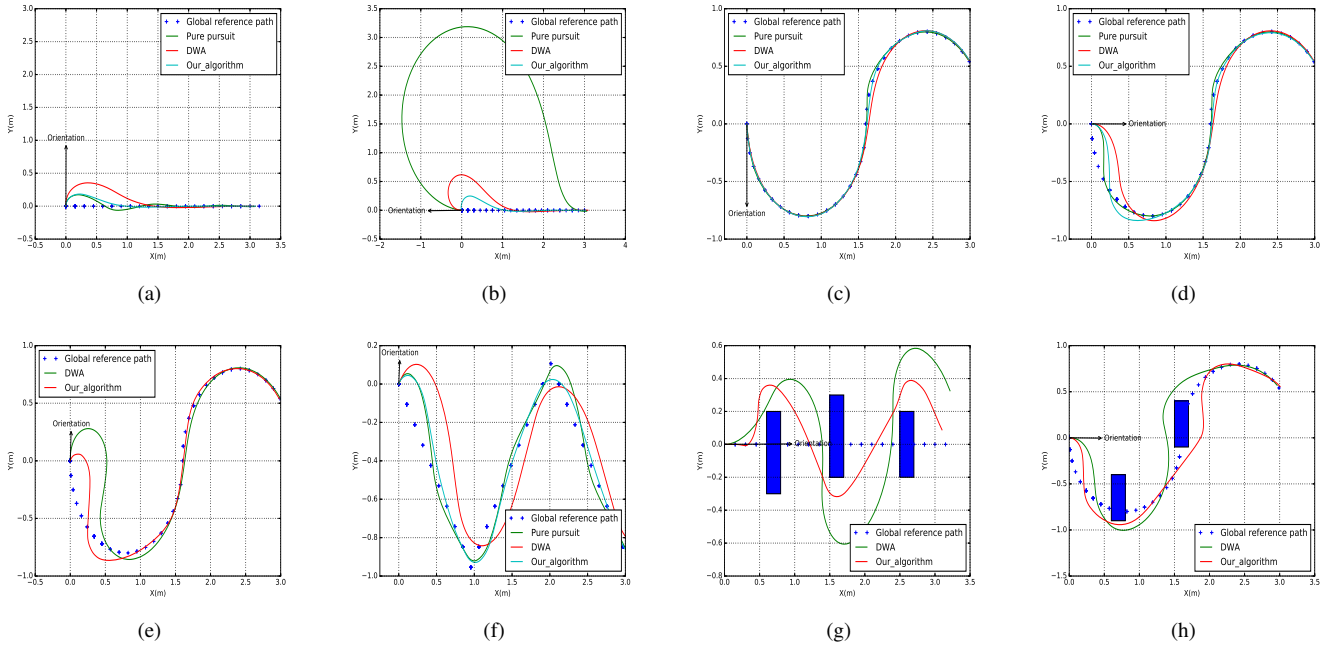

Fig. 5. (a)-(f) are real trajectories of the vehicle in three different scenarios. The initial angle $\Delta\phi = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$ in the figure respectively. (g)-(h) are static obstacle avoidance experiments. Blue rectangles are obstacles we set on the global reference path. Trajectories of the vehicle using different algorithms to avoid obstacles are shown above.

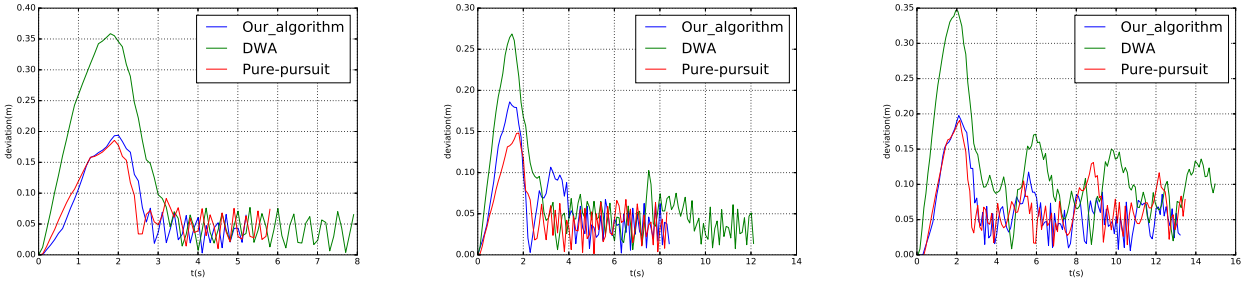

Fig. 6. The distance deviation e_0 between vehicle and global reference path. These pictures correspond to (a), (d) and (f) respectively in Fig. 5. The initial angle $\Delta\phi = \pi/2$. These figures illustrate that our algorithm has the least deviation so that the vehicle can move along with the global reference path well and improve running efficiency.

shape of path	straight line											
path length(m)	3.00											
initial direction $\Delta\phi$ (rad)	0			$\pi/2$			π			$3\pi/2$		
algorithm	M	DWA	P	M	DWA	P	M	DWA	P	M	DWA	P
average run time(s)	3.4	7.4	3.6	5.2	7.9	5.8	6.8	10.8	10.7	5.0	7.9	6.3
shape of path	S-shape curve											
path length(m)	5.10											
initial direction $\Delta\phi$ (rad)	0			$\pi/2$			π			$3\pi/2$		
algorithm	M	DWA	P	M	DWA	P	M	DWA	P	M	DWA	P
average run time(s)	6.0	10.7	6.3	8.4	12.1	8.3	10.6	14.4		8.2	12.8	8.2
shape of path	W-shape curve											
path length(m)	5.65											
initial direction $\Delta\phi$ (rad)	0			$\pi/2$			π			$3\pi/2$		
algorithm	M	DWA	P	M	DWA	P	M	DWA	P	M	DWA	P
average run time(s)	11.9	13.3	11.5	13.0	14.9	13.1	14.9	17.0		12.6	14.7	14.0

TABLE I

THE AVERAGE RUNNING TIME IN DIFFERENT SCENARIOS USING DIFFERENT ALGORITHMS. M DENOTES OUR ALGORITHM, P DENOTES PURE PURSUIT.

record the distance deviation at each moment to demonstrate our algorithm has the least deviation and improve running efficiency.

As shown in Tab. I, the pure pursuit doesn't work when the $\Delta\phi = \pi$, see Fig. 5(b), so We don't record that data. In terms of average running time, compared to DWA, our algorithm improves at least 34.1% running efficiency in a straight line, 26.3% in the S-shape curve, 10.5% in the W-shape curve.

Besides, to test the obstacle avoidance of our algorithm, We assume that there are some unknown obstacles on the global reference path, the path can not update. Although the global reference path is invalid that our algorithm can still move along with a global reference path and avoid obstacles, as well as DWA, see Fig.5(g) 5(h). What's more, the accumulated deviation from our algorithm is much smaller than DWA, which shows that our algorithm has a better effect, see in Fig. 5(g), (h).

Therefore, our algorithm has a better performance than others, especially the angle $\Delta\phi$ is too large. It improves running efficiency, speeds up the vehicle, and fits the global path better.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we have presented the local path planning based on particle filter optimization and combined with the global reference path to safely navigate the differential wheel vehicle in unstructured environments. We compare our algorithm with two approaches (DWA and Pure pursuit). The experimental results demonstrate that our algorithm is good at following arbitrary paths made up of discrete points and avoiding the obst. Especially, while the angle $\Delta\phi$ is too large, our algorithm has faster average speed and a shorter path than DWA. In the future, we intend to expand our approach for moving obstacles and multi-vehicle scenarios [16] by using lidar to estimate the state(position and velocity) of the moving obstacles(such as humans, vehicles and other robots) like [17] so that the danger zone can be predicted in advance.

REFERENCES

[1] H. Wang, Y. Yu, and Q. Yuan, "Application of dijkstra algorithm in robot path-planning," in *2011 second international conference on mechanic automation and control engineering*. IEEE, 2011, pp. 1067–1069.

[2] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path planning with modified a star algorithm for a mobile robot," *Procedia Engineering*, vol. 96, pp. 59–69, 2014.

[3] S. M. LaValle, "Rapidly-exploring random trees: A new tool for path planning," 1998.

[4] K. H. Sedighi, K. Ashenayi, T. W. Manikas, R. L. Wainwright, and H.-M. Tai, "Autonomous local path planning for a mobile robot using a genetic algorithm," in *Proceedings of the 2004 Congress on Evolutionary Computation (IEEE Cat. No. 04TH8753)*, vol. 2. IEEE, 2004, pp. 1338–1345.

[5] M. C. Lee and M. G. Park, "Artificial potential field based path planning for mobile robots using a virtual obstacle concept," in *Proceedings 2003 IEEE/ASME International Conference on Advanced Intelligent Mechatronics (AIM 2003)*, vol. 2. IEEE, 2003, pp. 735–740.

[6] N. Buniyamin, W. W. Ngah, N. Sariff, and Z. Mohamad, "A simple local path planning algorithm for autonomous mobile robots," *International journal of systems applications, Engineering & development*, vol. 5, no. 2, pp. 151–159, 2011.

[7] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.

[8] O. Brock and O. Khatib, "High-speed navigation using the global dynamic window approach," in *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No. 99CH36288C)*, vol. 1. IEEE, 1999, pp. 341–346.

[9] P. Ogren and N. E. Leonard, "A convergent dynamic window approach to obstacle avoidance," *IEEE Transactions on Robotics*, vol. 21, no. 2, pp. 188–195, 2005.

[10] O. A. Abubakr, M. A. K. Jaradat, and M. A. Hafez, "A reduced cascaded fuzzy logic controller for dynamic window weights optimization," in *2018 11th International Symposium on Mechatronics and its Applications (ISMA)*. IEEE, 2018, pp. 1–4.

[11] M. Missura and M. Bennewitz, "Predictive collision avoidance for the dynamic window approach," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8620–8626.

[12] J. Morales, J. L. Martínez, M. A. Martínez, and A. Mandow, "Pure-pursuit reactive path tracking for nonholonomic mobile robots with a 2d laser scanner," *EURASIP Journal on Advances in Signal Processing*, vol. 2009, p. 3, 2009.

[13] ROS.wiki, http://http://wiki.ros.org/navigation.

[14] W. E. Dixon, D. M. Dawson, E. Zergeroglu, and A. Behal, *Nonlinear control of wheeled mobile robots*. Springer, 2001, vol. 175.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "Ros: an open-source robot operating system," in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.

[16] J. Alonso-Mora, S. Baker, and D. Rus, "Multi-robot formation control and object transport in dynamic environments via constrained optimization," *The International Journal of Robotics Research*, vol. 36, no. 9, pp. 1000–1021, 2017.

[17] Y. Peng, D. Qu, Y. Zhong, S. Xie, J. Luo, and J. Gu, "The obstacle detection and obstacle avoidance algorithm based on 2-d lidar," in *2015 IEEE International Conference on Information and Automation*. IEEE, 2015, pp. 1648–1653.