IMPROVED EXECUTION METHOD OF TENTACLE ALGORITHM FOR INTELLIGENT VEHICLE

Zhang Minghuan(张明环), *Zhang Ke*(张科)

(College of Astronautics, Northwestern Polytechnical University, Xi'an, 710072, P. R. China)

Abstract: In order to solve some deficiencies in tentacle execution, an improved execution method of tentacle algorithm is presented. The method uses a short trajectory to match the curvature between the path of vehicle and tentacle, rather than computing a whole steady state. To control vehicle motion via wheel force and steering angle, two parameters should be discretized under certain area and these discrete values can form 18×20 groups. Then the curvature between the trajectory and tentacle should be matched, and the corresponding group of wheel force and steering angle can be found. The flow chart of the improved execution method is given, and simulation is performed on a platform named "pro-sivic". The simulation results show that the improved method can maintain the advantage of the tentacle algorithm in terms of computation speed, and avoid the errors such as endless loop and data overflow, which proves the method more efficient.

Key words: intelligent vehicle; tentacle algorithm; execution method

CLC number: TP242.6 **Document code**: A **Article ID**: 1005-1120(2013)04-0355-06

INTRODUCTION

With the development of the information technology, people pay more and more attention to the control techniques for intelligent vehicle in these years[1]. The research on collision avoidance for intelligent vehicle, which transfers the information of the environment to the central control system with the technology of information and sensing, is particularly prominent[2]. According to the environment information, the control system detects obstacle and then commands the vehicle to avoid it, so it can make the driving safe. Collision avoidance system for intelligent vehicle improves the security of the vehicle, thus it is extremely valuable[3].

These years people have got many achievements in this field, such as the artificial potential field method of Khatib[4] and the grid method of Howden[5]. However, there are some deficiencies such as the simultaneous localization and mapping (SLAM) problem and the complexity of the executing system[6]. In order to solve the two problems, Ref. [7] presents a tentacle algorithm strategy. This strategy is very simple because instead of accumulating data to create a precise map of the environment in each step, only an ego-centered occupancy grid is created. During the process, only one lidar sensor fixed at the centroid of the vehicle is used to ensure that the vehicle drives safely. Before the movement of the vehicle, all the potential paths based on different vehicle speed are computed, from which the best one is selected via the data from the lidar in each step to ensure the safety of driving according to the current speed. Therefore the computation will be efficient.

Though tentacle algorithm has got many good results, there are some drawbacks in tentacle execution. For example, in the final race of the Urban Challenge 2007, the AnnieWay car stopped at the entry of a sparse zone, which is based on the tentacle algorithm mentioned in Ref. [7]. The probable reason is that there is a computation problem in tentacle execution. This paper proposes an improved execution method to

Received date: 2012-11-20; **revision received date**: 2013-02-30

Corresponding author: Zhang Ke, Professor, E-mail: zhangke@nwpu.edu.cn.

solve this problem.

1 TENTACLE ALGORITHM

1.1 Tentacle structure

As the potential paths for the vehicles, all 81 tentacles are created (Fig. 1), and their corresponding groups of wheel force and steering angle are pre-computed.

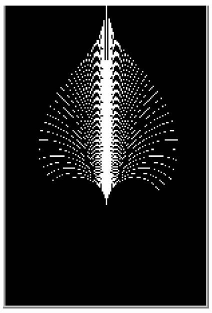


Fig. 1 All 81 tentacles (from left to right, the numbers of the tentacles are $k=0,\cdots,40,80,\cdots,41$)

1.2 Obstacle detection

Judge the speed section that the vehicle speed belongs to, and then get the speed set j . Based on the speed set j, an occupied grid map can be created (Fig. 2) via the data of the lidar. Check all the grids in the classification area according to each tentacle, and record all the occupied grids and their positions.

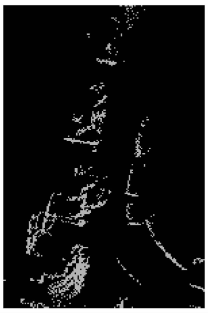


Fig. 2 Occupied grid map

1.3 Tentacle selection and execution

For each tentacle, compute the distance to the first obstacle l_o via the positions of the occupied grids. Then select the tentacle from the drivable area (Fig. 3), which has the maximal l_o , as the trajectory. Finally choose the corresponding group of pre-computed wheel force and steering angle to follow the selected tentacle.

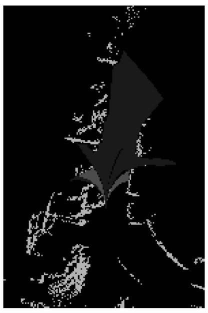


Fig. 3 Drivable area

2 IMPROVED EXECUTION METHOD

This section intends to control the vehicle to travel along the selected tentacle more efficiently. In other words, the corresponding groups of wheel force and steering angle will be pre-computed with an improved method.

2.1 Motion equations

Ref. [7] uses the well-known "bicycle model"[8] to create and simplify the motion equations of the vehicle.

In bicycle model shown in Fig. 4, F_{Ax} and F_{Ay} are the lateral aerodynamic forces, F_{xR} and F_{xF} the longitudinal forces, F_{yF} and F_{yR} the forces acting at the wheels, l_F and l_R the distances between the wheels and CG, α_R and α_F the slip angles of the wheels, CG shows the center of gravity, PP the pressure point, e_{CG} the distance between CG and PP, l the distance between front wheel and rear wheel, v the velocity of the vehicle, $\frac{v^2}{\rho}$ the centripetal acceleration, β the sideslip angle, ψ the angle of the heading direction measured against the x-axis, and M the center of curvature.

By using the mass of the vehicle m, the moment of inertia around the z-axis J_z and the steering angle δ_F, we can get the motion equations through the relationship in Fig. 4, shown as

$$\ddot{\Psi}=\frac{1}{J_z}[(F_{yF}\cos\delta_F+F_{xF}\sin\delta_F)l_F-F_{yR}l_R+F_{Ay}e_{CP}]$$

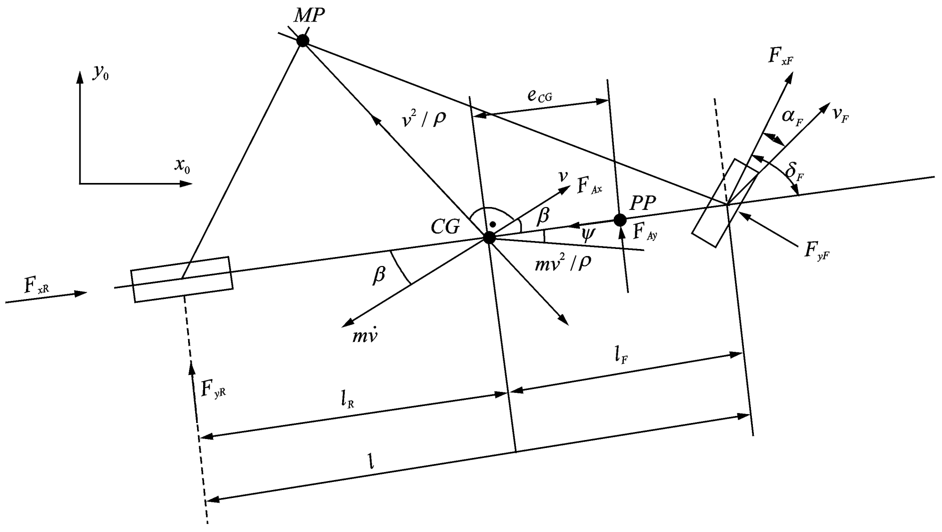


Fig. 4 Bicycle model

$$\dot{\beta} = -\dot{\psi} - \frac{\sin\beta[F_{xR} - F_{Ax} + F_{xF}\cos\delta_F - F_{yF}\sin\delta_F]}{mv} - \frac{\cos\beta[-F_{yR} - F_{Ay} - F_{xF}\sin\delta_F - F_{yF}\cos\delta_F]}{mv}$$

$$\dot{v} = \frac{\cos\beta[F_{xR} - F_{Ax} + F_{xF}\cos\delta_F - F_{yF}\sin\delta_F]}{m} - \frac{\sin\beta[-F_{yR} - F_{Ay} - F_{xF}\sin\delta_F - F_{yF}\cos\delta_F]}{m} \quad (1)$$

Then compute the trajectory of the vehicle as

$$y(t) = y(t_0) + \int v(t)\sin(\beta(t) + \psi(t))\,\mathrm{d}t$$

$$x(t) = x(t_0) + \int v(t)\cos(\beta(t) + \psi(t))\,\mathrm{d}t \quad (2)$$

where $x(t)$ and $y(t)$ mean the positions of the vehicle along the trajectory. The coordinate system is shown in Fig. 4.

2.2 Problem in current tentacle execution

It is noticed that all the parameters are related to wheel force and steering angle. It means that once a group of wheel force and steering angle is certain, the trajectory of the vehicle can be fixed. That to say, a group of wheel forces and steering angles refers to one certain tentacle. Therefore, the vehicle can be controlled to follow the selected tentacle through the two parameters.

Ref. [7] presented a method to execute the selected tentacle. This method pre-computed all the corresponding groups of wheel force and steering angle by creating a steady state (Fig. 5) according to each tentacle, and then stored them. So that once a tentacle is selected, the corresponding wheel force and steering angle can be acquired at the same time.

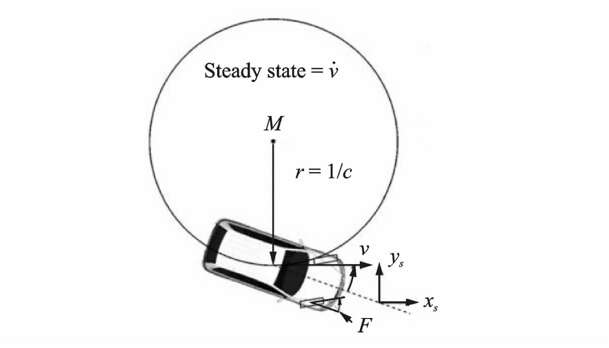
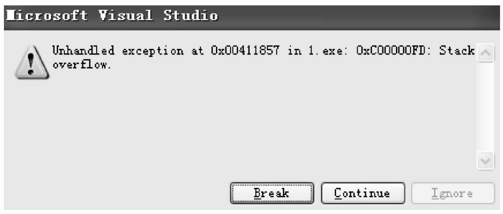


Fig. 5 Steady state in tentacle algorithm

Though this method is correct in theory, there are some errors in programming which will cause the tentacle algorithm invalid, such as data overflow and endless loop (Fig. 6). It is because that when the radius of a tentacle is too large, matching this tentacle with the corresponding steady state will cost plenty of time, and at worst a steady state can not be even achieved.


Microsoft Visual Studio

Unhandled exception at 0x00411857 in 1.exe: 0xC00000FD: Stack overflow.

Fig. 6 Simulation error in tentacle algorithm

2.3 Improved method

In order to solve this problem, an improved method is presented as follows.

(1) Discretization of control parameters

For one tentacle, the corresponding group is found through two loops related to wheel force and steering angle. At first, the step size and the upper and lower limitation of the loops should be fixed.

Because each tentacle is a part of a circle, in order to drive as a circle, the sum of the external forces should be 0, that is

$$\sum F = 0 \tag{3}$$

For 16 speed sets in tentacle algorithm, because of the differences of resistance, there should be 16 different wheel forces F_j ($j=0,\cdots,15$) according to the 16 values of the speed sets v_j ($j=0,\cdots,15$) to keep the vehicle drive with a constant speed.

The vehicle can be accelerated when the selected tentacle is $k=40$ (the straight one), and be decelerated when there is no available tentacle. Therefore two other wheel forces can be set as $F_a = ma$ and $F_d = -ma$, where $a=1.5$ m/s^2.

So far, there are 18 different wheel forces to be used. Therefore, the step size of wheel force is 18, and the upper and lower limitation are F_0 and F_{15}. For the steering angle, the region $(-0.5\pi, 0.5\pi)$ can be separated into 20 sections equally, so there are 18×20 groups of wheel force and steering angle to be used.

(2) Tentacle matching

Now the loops can be started. As the period of the lidar is 0.1 s, the temporal step size $\mathrm{d}t$ (Eq. (2)) is set as 0.001 s. And we only need to compute 100 steps according to one group of wheel force and steering angle. There will be a trajectory according to the 100 steps through Eq. (2). Because the origin of each tentacle is $(r_k, 0)$, a value φ can be set to show the difference between the tentacle and the trajectory

$$\varphi = \sum_{i=1}^{100} \left| (x_i - r_k)^2 + y_i^2 - r_k^2 \right| \tag{4}$$

where (x_i, y_i) shows 100 positions of the vehicle according to 100 steps, and r_k the radius of the tentacle. By comparing the curvature between the tentacle and the trajectory, it is obvious that the minimal value of φ shows the best group of wheel force and steering angle, which can ensure the vehicle follows this tentacle.

Fig. 7 shows the whole flow of the improved methd.

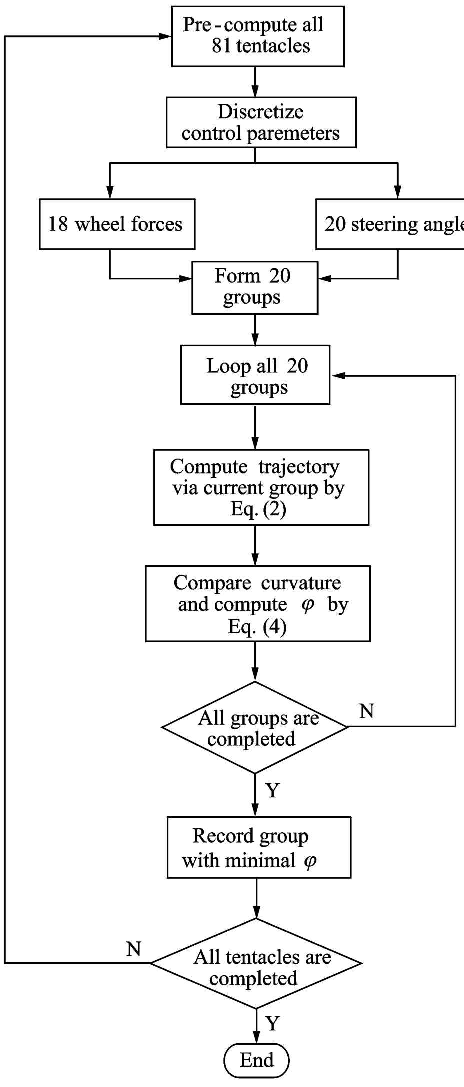

Pre-compute all 81 tentacles
Discretize control paremeters
18 wheel forces
20 steering angles
Form 20 groups
Loop all 20 groups
Compute trajectory via current group by Eq. (2)
Compare curvature and compute φ by Eq. (4)
All groups are completed
N
Y
Record group with minimal φ
All tentacles are completed
N
Y
End

Fig. 7 Flow chart of improved method

3 SIMULATION RESULTS

In order to validate the validity of the improved method, a simulation is implemented on a platform named "pro-sivic". The project is that the vehicle drives around a rectangular wall. The initial scene is shown as Fig. 8.

The results are shown in Tables 1, 2 and Figs. 9,10. Meanwhile, in order to observe all the

Table 1　Records of driving situations

Time/10^{-9} s	Rotation X/m	Rotation Z/m	Rotation Y/m	Translation X/m	Translation Z/m	Translation Y/m	Steering angle/rad
27 968 309	0. 000 000	0. 279 922	0. 000 000	−0. 456 519	0. 000 000	−1. 917 042	0. 000 000
28 088 289	0. 000 000	0. 280 923	0. 000 000	−0. 518 232	0. 000 000	−2. 131 284	−0. 015 708
28 188 272	0. 000 000	0. 282 257	0. 000 000	−0. 584 346	0. 000 000	−2. 359 622	−0. 015 708
28 308 252	0. 000 000	0. 283 591	0. 000 000	−0. 654 945	0. 000 000	−2. 602 291	0. 000 000
28 418 233	0. 000 000	0. 284 926	0. 000 000	−0. 729 956	0. 000 000	−2. 858 791	0. 031 416
28 518 217	0. 000 000	0. 286 594	0. 000 000	−0. 802 210	0. 000 000	−3. 104 365	0. 015 708

Table 2　Time cost for path planning (five continuous sets)　　s

Method	Time 1	Time 2	Time 3	Time 4	Time 5
Improved tentacle algorithm	0. 023 755	0. 023 211	0. 021 982	0. 021 497	0. 023 328
Tentacle algorithm	0. 026 536	0. 025 471	0. 022433	0. 022 584	0. 023 997
Artificial potential field method	0. 046 742	0. 046 635	0. 042 276	0. 042 587	0. 044 569
Grid method	0. 050 287	0. 049 872	0. 049 247	0. 048 973	0. 051 846

Fig. 8　Initial scene in pro-sivic

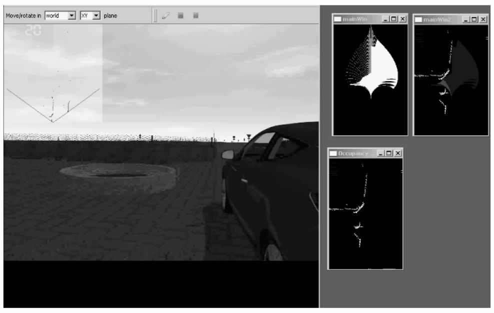


Fig. 9　3-D flash and tentacle selection in simulation

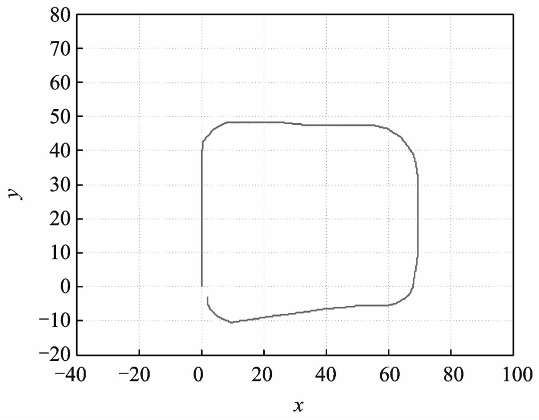


Fig. 10　Trajectory throughout whole simulation

drivable tentacles in each step, the tentacles are shown with a boundary.

Table 1 shows a part of the records which are collected from the driving situation, including the simulation time, the position (x, y, z) of the vehicle, the rotation around each axis, and the steering angle. It should be noted that the value of Translation Z in position (x, y, z) is constantly equal to 0 while the steering angle changing. Meanwhile, both the values of Rotation X and Rotation Y are also equal to 0 as well, and there is only one moment called Rotation Z around Z-axis. It is because that Z-axis here is vertical to the ground and the vehicle is always driving on a horizontal plane, so the vehicle would not turn over while driving, which proves that the "bicycle model" of the vehicle in this paper is credible.

In Fig. 9, left part shows the 3-D flash so the environment can be observed more clearly, and right part is the boundary of the first obstacle in each tentacle, drivable area, and occupancy grid in turn. This is one step of the whole simulation. As it is figured out, once the occupancy grid is created, the drivable tentacles can be computed rapidly and the one with maximal value of l_o will be selected as the trajectory immediately. Moreover, the corresponding pre-computed wheel force and steering angle can be acquired to control the vehicle at the same time.

Accordingly, Table 2 shows five sets of time costs for path planning by different algorithms sequentially. The improved method costs about 0.02 s in each set, and the tentacle algorithm costs almost the same time, while the other two methods cost about 0.04 s, which is almost twice as much as the time cost of the improved method. Results show that the improved method is much faster than the artificial potential field method and the grid method, and no longer than the original algorithm.

Fig. 10 shows the trajectory of the vehicle throughout the whole simulation. The vehicle achieves its purpose with high quality, and no errors occur during the simulation. That is because a whole steady state does not need to be computed in each step, instead, a short trajectory is used.

To be general, all the simulation results show that the improved method can avoid the errors such as endless loop and data overflow, while maintaining the advantage of the original algorithm in the form of computation speed, so the improved method is proved to be more efficient and effective compared with the original algorithm.

4 CONCLUSION

This paper presents an improved tentacle execution for tentacle algorithm. In the improved method, a short trajectory is used rather than computing a whole steady state in each step. Therefore it can avoid some errors such as endless loop and data overflow.

The simulation results indicate that the improved algorithm can not only maintain the advantage of the tentacle algorithm in the form of computation speed, but also avoid the errors, thus proves this method to be more efficient.

It should be noted that the model of the intelligent vehicle is very simple, and some factors are also ignored, such as air resistance and sideslip of the vehicle. Therefore, the further work will be focused on solving these problems.

Acknowledgement

The author would like to thank Axis department in University of Paris 11 for the support of the research work.

References:

[1] Paz L M, Tardos J D, Neir J. Divide and conquer: EKF SLAM in O(n)[J]. IEEE Transactions on Robotics, 2008, 24(5): 1107-1120.

[2] Moravec H, Elfes A. High resolution maps from wide angle sonar[C]// International Conference on Automation Proceeding. [S. l]:IEEE,1985: 116-121.

[3] Moras J, Cherfaoui V, Bonnifait P. A lidar perception scheme for intelligent vehicle navigation[C]// 11th International Conference on Control, Automation, Robotics and Vision. Singapore:[s. n.], 2010: 1809-1814.

[4] Sariff N, Buniyamin N. An overview of autonomous mobile robot path planning algorithms[C]// 4th Student Conference on Research and Development (Scored 2006). Malaysia: IEEE, 2006: 183-188.

[5] Feng X W, Guo S H, Li X H, et al. Robust mobile robot localization by tracking natural landmarks[C]// Artificial Intelligence and Computational Intelligence. Berlin, German: Springer, 2009: 278-287.

[6] Thomas G, Karsten B, Antje W. Feature-based camera model identification works in practice: Results of a comprehensive evalutation study[C]// 11th International Workshop on Information Hiding. Darmstadt, Germany: Springer-Verlag, 2009: 262-276.

[7] Hundelshausen F, Himmelsbach M, Muller A, et al. Driving with tentacles-integral structures of sensing and motion[J]. International Journal of Field Robotics Research, 2009(56): 393-440.

[8] Mitschke M, Wallentowitz H. Dynamik der Kraftfahrzeuge[M]. Heidelberg: Springer, 2004.