Proceedings of The 2017 IEEE International
Conference on Real-time Computing and Robotics
July 14-18, 2017, Okinawa, Japan

Optimization of Robot Path Planning Parameters Based on Genetic Algorithm

Yuhong Liang, Fating Hong, Qinjie Lin ,Sheng Bi*, Liqian Feng
South China Universiy of Technology

*Abstract*_**The paper introduces a method which is based o n genetic algorithm to select more properly parameters fo r local path planning of mobile robot. In our experiments, according to experience, we choose properly parameter domain as chromosomes in Genetic algorithm(GA). Then we will extract several different individual cost function from DWA algorithm, adopted by Local path planning, and merge them into a fitness function. Experiments show that the automatic tuning method, proposed in this paper, can generate smoother and faster trajectories than manual tuning method and perform better in avoiding obstacle. This method can also be used for other platforms.**

I. Introduction

Mobile robot is a mobile platform that can be used in unknown or known, outdoor or indoor autonomous positioning, fixed-point navigation, dynamic obstacle avoidance. An important sign of mobile robot intelligence is autonomous navigation, and the realization of robot navigation has a basic requirement - local path planning. There are a lot of researches on the dynamic obstacle avoidance of mobile robots in recent years. The algorithm of local path planning mainly has intelligent bionic algorithm, which includes artificial neural network algorithm, genetic algorithm, ant colony optimization algorithm and particle swarm Algorithm, using these algorithms to achieve global optimization to get the global optimal path. And heuristic search algorithms, such as D *, Focussed D *, use the robot to collect the environmental data to quickly correct and select the optimal path, greatly reducing the local planning time for high real-time platform. Based on the dynamic window algorithm (DWA), the basic idea is to collect real-time data from the environment and update the planning dynamically.

In the open source robot operating system (Robot Operating System, ROS), the base_local_planner function package in the navigation package provides implementation of the dynamic window method (The Dynamic Window Approach). A local plan is generated by DWA as follow: Firstly, a number of the velocity samples will be specified by given controller parameters. Next for each velocity, the controller will perform forward simulation from the current state of the robot to foresee the situation from applying each velocity for a short period of time. Then, the controller will evaluate each resultant trajectory generated from the simulation and finally pick the lowest-scoring trajectory and send associated velocity to the mobile base. So in DWA the parameters which is manually configured by us according to practical environment have great impact on sampling velocity and evaluating trajectory. However, there are some unstable factors in environment such as quality of robots, the size of the robots, the shape and so on. Such instability caused a great challenge to manual setting and as a result setting the proper value of key parameters will be a challenge in applying DWA to practical environment.

We adopt the genetic algorithm to solve the problem of artificial parameter tuning. Genetic algorithm can overcome the problem of local minimum, and can find the optimal or suboptimal solution of minimum granularity. And genetic algorithm calculation is moderate, we can achieve while planning side tracking, in other words its real-time performance is good. Genetic algorithm is used to optimize the parameters and find the optimal parameters. The robot can avoid the obstacle more accurately when using the local programming algorithm to find the best path, so as to solve the problem that the DWA algorithm is applied in the complex environment.

II. Experimental Platform

In this paper, the platform we use is TurtleBot, as shown in Figure1. TurtleBot is an open source hardware project and consists of a mobile base. And with TurtleBot we can build a robot that can navigate with 2D map, generate a global path, generate local path planning and so on.

Figure1. TurtleBot platform

TurtleBot's base can be seen as a circular, which plays an important role in the stability of the robot. For example, while generating local path plan, a circular mobile base can reduce the computational complexity of the algorithm.

In this intelligent mobile platform, we established a mobile robot system based on laser sensor, whose shape is

Yuhong Liang, Fating Hong, Qinjie Lin ,Sheng Bi and Liqian Fen are with the School of Computer Science&Engineering, South China University of Technology, Guangzhou, 510640. Sheng Bi is also recently with Postdoctoral Fellow in Emerging Technologies Institute, The University of Hong Kong. (corresponding author: Sheng Bi, e-mail: picy@scut.edu.cn)

shown in Figure2 .The laser sensor on the robot platform is an all-angle two-dimensional scanning ranging radar, which uses a triangulation method, as shown in Figure3. The laser sensor can emit a laser beam to detect the position and velocity of the target, and can quickly measure the distance from the front obstacle to itself, then create a map for the mobile robot. Through this sensor we can measure the distance with fast speed, high precision, large range, anti-light, strong electrical interference and other advantages, which is why we choose it.

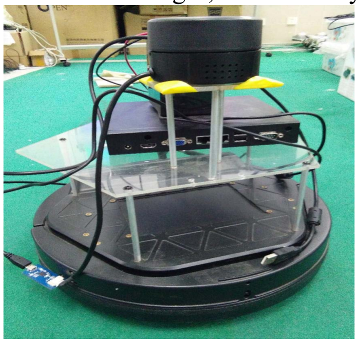
Figure 2. Mobile robot system based on laser sensor

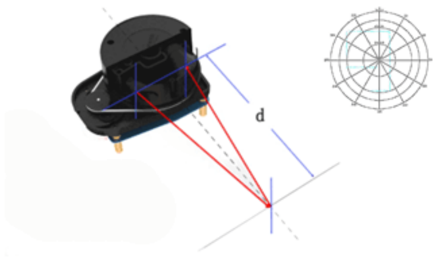

Figure 3. 360 ° two-dimensional scanning ranging radar

III. CONTROL PARAMETERS OF DWA

A. Search Space:

In DWA, search space is a set of sampled velocities based on the current speed that are used to perform forward simulation. Firstly, it calculates the sampled velocities which is reachable within the next time interval given the current translational and rotational velocities and accelerations. Then the samples are reduced to satisfied the limit of translational and rotational velocities .In this case, the definition of the speed constraints is

$$
\begin{aligned}
\min[0] &= \max\begin{pmatrix} \min\_vel\_x, \\ vel[0] - acc\_lim\_x \times sim\_period \end{pmatrix} \\
\max[0] &= \min\begin{pmatrix} \max\_vel\_x, \\ vel[0] + acc\_lim\_x \times sim\_period \end{pmatrix} \\
\min[2] &= \max\begin{pmatrix} -1 * \max\_rot\_vel, \\ vel[2] - acc\_lim\_th \times sim\_period \end{pmatrix} \\
\max[2] &= \min\begin{pmatrix} \max\_rot\_vel, \\ vel[2] + acc\_lim\_th \times sim\_period \end{pmatrix}
\end{aligned}
\tag{1}
$$

Where max_*vel_x* , min_*vel_x* is respectively the maximum speed and minimum speed in the x-direction of the robot coordinate system , max_*rot_vel* is the absolute value of minimum translational velocity and max_rot_*vel* is the absolute value of the maximum translational. *acc_lim_x* is the given acceleration and *acc_lim_th* is the given sampling time. These are our controlled parameters. Also, min [1] and max [1] is useless because the robot in our experiments can only forward and rotate but cannot move vertically, and as a result, there is no velocity in the y-direction.

After obtaining the constraints of velocity, in other words, the maximum and minimum values of the velocity, the discrete velocity samples in search space are defined as

$$
\begin{aligned}
\mathrm{Vel} &= \left\{ (V_x, V_y, V_{th}) \middle| \begin{matrix} V_x \in Vel_x \wedge V_y = 0 \wedge V_{th} \in Vel_{th} \wedge \\ (V_x < \mathrm{min\_trans\_vel}) \wedge \\ (V_{th} > \mathrm{min\_rot\_vel}) \end{matrix} \right\} \\
\mathrm{Vel}_x &= \left\{ V_x \middle| \begin{matrix} V_x = \min[0] + i \times \left(\dfrac{(\max[0] - \min[0])}{vxsamples} \right), \\ 0 \le i \le vxsamples \end{matrix} \right\} \\
\mathrm{Vel}_{th} &= \left\{ V_{th} \middle| \begin{matrix} V_{th} = \min[2] + i \times \left(\dfrac{(\max[2] - \min[2])}{vthetasamples} \right), \\ 0 \le i \le vthetasamples \end{matrix} \right\}
\end{aligned}
\tag{2}
$$

Where (V_x, V_y, V_{th}) represents the velocity of the robot in the x-direction, y-direction and orientation-direction of the robot's coordinate system. $vxsamples$ and $vthetasamples$ are the number of samples set and the two are the parameters we need to set.

B. Create Trajectory

Giving the robot's velocity samples, one of the samples that are reachable will be sent to the mobile base. However, the robot will suffer impact and damage such as deviation from the global path, hitting obstacles and so on. So we will generate a trajectory for each velocity sample, and evaluate the trajectory using some method to find the best sample velocity to send to the mobile base.

Based on the analysis of the *sim_time, sim_granularity* and $\left(V_x(t_0), V_y(t_0), V_{th}(t_0)\right) \in Vel$, the generated trajectories are defined as

$$
\begin{aligned}
\mathrm{x}(t_i) &= \mathrm{x}(t_0) + \sum_{j=0}^{i} \begin{pmatrix} V_x(t_0) \times cos \begin{pmatrix} \theta(t_0) + \\ j \times V_{th}(t_0) \end{pmatrix} * \\ s \\ sim\_granularity \end{pmatrix} \\
\mathrm{y}(t_i) &= \mathrm{y}(t_0) + \textstyle\sum_{j=0}^{i} \begin{pmatrix} V_x \times sin(\theta(t_{i-1})) \\ \times sim\_granularity \end{pmatrix} \\
\theta(t_i) &= \theta(t_0) + \sum_{j=0}^{i} \begin{pmatrix} V_{th}(t_0) \times \\ sim\_granularity \end{pmatrix}
\end{aligned}
\tag{3}
$$

Where $(x(t_i), y(t_i), \theta(t_i))$ represent the pose of robot in robot's coordinate.$(x(t_n), y(t_n), \theta(t_n))$ $(\mathrm{n = sim\_time \div sim\_granularity})$ is the last point of a trajectory. Especially, sim_time and sim_granularity are our controlled parameters.

C. Cost Functions

1) *Alignment_cost* and *Path_cost*

In DWA, the values of *alignment_cost* and *path_cost* are based on how closely the trajectory follows a global path. For path_cost, the closer the last point of the trajectory is to the global path, the lower value the cost function will return. And for *alignment_cost*, the closer the last point of the shifted trajectory is to the global path, the lower value cost function will return. Therefore, the definition of *path_cost* and *alignment_cost* is:

$$alignment\_cost = costmap\_path\begin{pmatrix} x(t_n) + \\ x\_shift \times \theta(t_n), \\ y(t_n) + \\ y\_shift \times \theta(t_n) \end{pmatrix}$$

$$\text{path\_cost} = \text{costmap\_path}\big(x(t_n), y(t_n)\big) \quad (4)$$

Where *x_shift* and *y_shift* are our controlled parameters which are the distance from the centre point of the robot to place an additional scoring point. The $\text{costmap\_path}(x, y)$ returns the value of an element whose index is $x \times resolution + y$ in array *costmap_path*. In DWA, the *costmap_path* where the value of the element is assigned to the distance from the current position to the global path is a one-dimensional array.

2) *Goal_cost* and *Goal_front_cost.*

In DWA, the values of *goal_cost* and *goal_front_cost* depend on how closely the point of trajectory $\big(x(t_n), y(t_n), \theta(t_n)\big)$ approaches a goal point. .For *goal_cost,* the closer the last point of the trajectory is to the global goal, the lower value cost function will return. And for *goal_front_cost*, the closer the last point of the shifted trajectory is to the global goal, the lower value it returns. Therefore, the follow equations can be used to represent *goal_cost* and *goal_front_cost:*

$$goal\_front\_cost = costmap\_goal\begin{pmatrix} x(t_n) + \\ x\_shift \times \theta(t_n), \\ y(t_n) + \\ y\_shift \times \theta(t_n) \end{pmatrix}$$

$$\text{goal\_cost} = \text{costmap\_goal}\big(x(t_n), y(t_n)\big) \quad (5)$$

Where x_shift and y_shift are our controlled parameters, which are the distance from the centre point of the robot to place an additional scoring point. $cosmap\_goal(x, y)$ returns the value of an element whose index is $x \times resolution + y,$in array *costmap_goal.* In DWA, the *costmap_goal* where the value of the element is assigned to the distance from the current position to the global goal is a one-dimensional array.

3) *Obstacle_cost* .

In DWA, if the robot doesn't run into obstacles the cost function will return a value, based on the distance from the robot to obstacles around. If the robot passes through an obstacle, the associated velocity will be abandoned. The equations can be represented as follow:

$$obstacle\_cost = max\begin{pmatrix} footprint\big(x(t_i), y(t_i), \theta(t_i)\big), \\ cosmap\big(x(t_n), y(t_n)\big) \end{pmatrix}$$

$$i = 1,2, \dots \dots, n \quad (6)$$

Where *cosmap(x,y)* returns the value of an element whose index is $x \times resolution + y,$ in array *costmap*. In DWA, the *costmap* where the value of the element is assigned to the distance from the current position to the nearest obstacle is a one-dimensional array. And the value which the function would return is associated with the distance from the mobile base of the robot to the nearest obstacle.

4) *Oscillation_cost.*

This cost function helps reduce certain oscillations. The definition of Oscillation_cost is as follow::

$$oscillation\_cost = cost\_dist + cost\_angle$$

$$\text{cost\_dist} = \begin{cases} 1000 & \begin{pmatrix} (V_x(t_0) \times pre\_V_x < 0) \wedge \\ \big((pre\_x - x(t_0)) < dist\big) \end{pmatrix} \\ 0 & else \end{cases}$$

$$\text{cost\_angle} = \begin{cases} 1000 & \begin{pmatrix} (V_{th}(t_0) \times pre\_V_{th} < 0) \wedge \\ \big((pre\_theta - \theta(t_0)) < angle\big) \end{pmatrix} \\ 0 & else \end{cases}$$

(7)

Where $V_x(t_0)$is the sample velocity in the x-direction of Robot's Coordinate system and $V_{th}(t_0)$ is the associated velocity in rotation. $pre_x$is the last position of the robot where it changed direction in x-direction and pre_theta is the last position of the robot where it rotated direction is different from $V_{th}(t_0)$. And *dist* and *angle* are our controlled parameters.

D. *Selected Parameters*

We choose the parameters having significant influence on these steps as our controlled parameters. This includes *maxx_vel_x, min_vel_x, max_rot_vel, min_rot_vel, acc_lim_x, acc_lim_th, vxsamples* and *vthetasamples* which have influence on reducing search space. Also *Sim_time* and *sim_granurality* influencing the generation on trajectory and *x_shift,y_shif, dist* and *angle* influencing the costs of generated trajectory can be chosen as our controlled parameters.

IV. SEARCHING THE BEST PARAMETER GROUP AS A SINGLE-OBJECTIVE PROBLEM

The DWA(Dynamic Window Approach) is put forward to generate local plan which can avoid obstacles , follow global plan as closely as possible ,approach global goal as closely as possible ,and stay stable while navigating. Therefore the Objective Function includes *obstacle_cost, oscillation_cost, goal_cost, goal_front_cos*t and *alignment_cost, path_cost.* It can be measured as follow:

$$\begin{aligned} &cost\big(V_x, V_y, V_{th}\big) \\ &= path\_distance\_bias \times (alignment\_cost + path\_cost) \\ &+ goal\_distance\_bias \times (goal\_cost + goal\_front\_cost) \\ &+ occdist\_scale \times obstacle\_cost + oscillation\_cost \end{aligned}$$

(8)

Where *path_distance_bias* is the weight for how much the controller should stay close to the given path, *goal_distance_bias* is the weight for how much the controller should attempt to reach its local goal and *occdist_scale* is the weight for how much the controller should attempt to avoid obstacles.

In this paper, searching for the best parameter group can be abstracted to a Single-Objective problem. Our purpose is to find the optimal parameter group, which can minimize the value of objective function returns. Single-Objective Optimizing by Genetic Algorithm

Now we apply genetic algorithms to solve single-objective problems. Based on the previous study of genetic algorithm, the (8) is used as the fitness function, and the minimum cost is obtained. Genetic algorithm is used to find the global optimal solution. Our algorithm is implemented as follows steps:

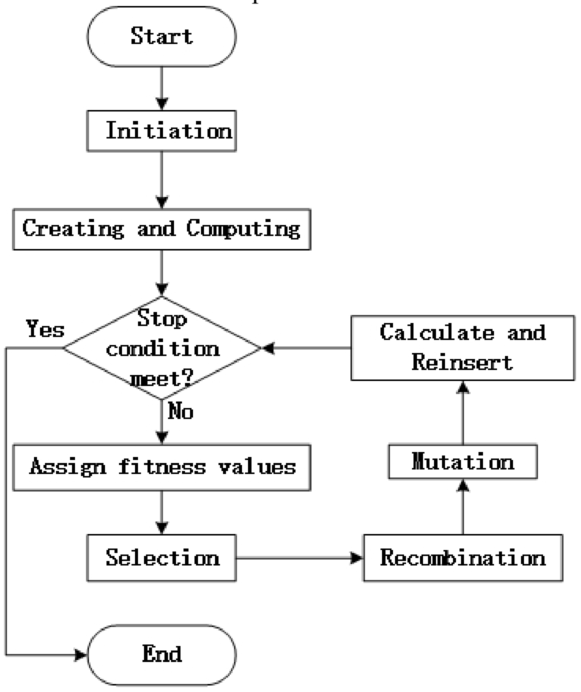

Figure 4. Steps of our algorithm

Initiation: Given the user-defined number of individuals *NIND*, the number of variables *NVAR*, the precision of variables *PRECI* and the field descriptor *FieldD*, the initial population can be represented as fixed size, real-valued and randomly initiated in the range of respective variables. We can change a decimal coded into binary code. Therefore the initial population is a binary vector.

Creating and Computing: In this step, the initial population can be defined as a vector with the size of *NIND* x vector of realtors (*NVAR* * *PRECI*). Because we use the binary encoding, the vector of each element is either 1 or 0. The value of the variety defined in each step is determined by region descriptor defined in the last step, so the value that each rank of the vector represents is in a domain. Then the objective function value of the initial population will be calculated.

Assign fitness values: Each individual will be sorted in order of target *ObjV*, and a column vector will be returned including *FitnV* corresponding to individual fitness value.

Selection: In this step, the individual who has fitter solutions are more likely to breed a new generation while the less fit are less likely to breed a new generation. Fitness value of group is close to the value of the constant approximation optimal. The selection operator used in this paper is the random traversal sampling, similar to the roulette selection operator. The difference between then is shown as follow Figure5:

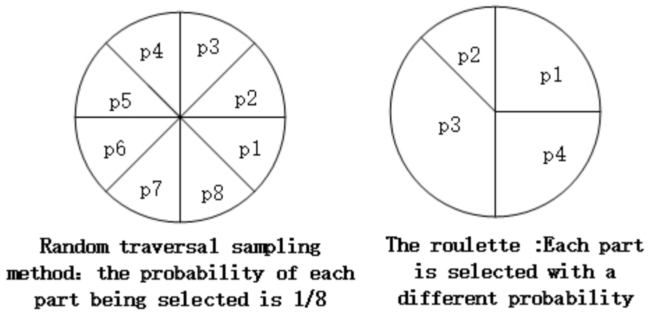
p1
p2
p3
p4
p5
p6
p7
p8
Random traversal sampling method: the probability of each part being selected is 1/8
p1
p2
p3
p4
The roulette :Each part is selected with a different probability

Figure 5. Random traversal sampling method and the roulette

Assuming that *n_point* is the number of individuals needed to be selected, then choice individuals in equal distance, choosing 1 / *n_point*, as a pointer distance. The position of first pointer depends on the random one of [0 , 1/*n_point*].

Recombination: In the intersection of genetic algorithm, two individuals are selected who have more probability, and some new individuals are created by handing over some of their parts and they inherits the characteristics of the parent. We use a single-point crossover operator, also called simple cross. Firstly, a cross point is randomly placed in the individual binary string where part of the chromosome of individuals is exchanged. It is shown as Figure 6. The process is as follow:

1) Mating. Pairs of individuals are selected to mate and if the amount of population is *M*, then there will be $\lfloor M/2 \rfloor$ pairs of combinations.

2) Placing a cross point. The cross points are placed randomly behind part of the chromosome. Assuming that the length of the chromosome is M, then there will be M-1 possible cross-point positions in it.

3) For each pairing pair, two chromosomes of the two individuals are handed over at the intersection according to the exchange probability set by the user, resulting in two new individuals.

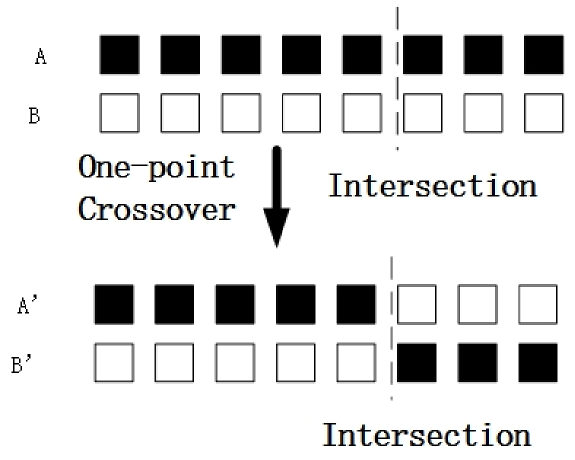

Figure 6. Single-point Crossover

Mutation: This process in the genetic algorithm is to change some of the bit values of the individual code string with a smaller probability, resulting in a new individual. Crossover is a common method of generating new individuals, which determines the global search ability of genetic algorithm, and mutation operation is a method to generate new individuals, which determines the local search ability of genetic algorithm. The two are complementary to each other, so that the genetic algorithm performs better in a search to solve the optimization problem optimization process. In this step, the discrete variation method is used because it improves the local search of genetic algorithms and maintains the diversity of population genes to prevent premature phenomenon.

Calculate and Reinsert: After the three steps selection, crossover, and mutation, we get the offspring produced by these three stages, then calculate the value of these offspring objective function returns, and then insert offspring into the parent group according to a given probability. As a result, the parents with lower value which objective function returns are abandoned, and a new population forms.

Stop condition: the maximum generation of the GA is a user-defined variable used to ensure that the algorithm terminates. We should also consider the stability of the algorithm, if compared with the previous generation, the offspring of the objective function value has not changed, we can think that the algorithm has found the optimal solution.

V. EXPERIMENT AND RESULT

A. Experiment Description

Then to verify the model and the algorithm, an experiment is carried out using Matlab R2016A in a PC (Intel Core i5-2520M CPU @ 2.50GHz with Memory 4G).

We apply Matlab Genetic Algorithm Toolbox gatbx developed at University of Sheffield to experiment. In the experiment, a set of optimal combination of parameters is selected by the genetic algorithm, and all we need to do is to collect the cost of each grid of the grid map and the robot footprint, and then pass these data to the cost function. So cost function is just affected by the variables which we need to manipulate and it makes influence on the machine's local planning path. After obtaining the complete cost function, we decide the upper and lower bounds of each parameter according to experience, so that the genetic algorithm can optimize the parameters in this region, and finally get the parameters we want.

B. Map data acquisition

The collection of map data is one of the most important parts of our experiments. Genetic algorithm selection of parameters is based on specific map data. In different environments, the same set of parameters of the robot are not ideal. Therefore, depending on different map environment, selecting different parameters can obtain better path planning and make the robot more adaptable.

In the process of experiment, we modify the base_local_planner code of the TurtleBot platform and let the system output the current map information to a file and then we use these information to restore the raster map containing the cost of grid. After getting the map, we infer some relevant information in advance to reduce the amount of calculation, as shown in Table 1:

Table 1: THE PART CAN BE CALCULATED DIRECTLY

goal_cost	path_cost
costmap_goal$(x(t_n), y(t_n))$	costmap_path$(x(t_n), y(t_n))$

C. Parameter Setting for Genetic Algorithm

The Genetic algorithm for Single-Objective optimization we introduced in this paper is applied to solve path planning problem.

1) Genetic Algorithm parameters setting:

Table 2 :THE PARAMETER USED IN GENETIC ALGORITHM

G_{max}	NIND	NVAR	PRECI	GGAP
500	40	20	20	0.9

2) The search space of parameters: In the process of parameter optimization, we use a region descriptor to define the range of optimization parameters. The representation of the region descriptor is FieldD = (len, lb, ub, code, scale, lbin, ubin)^T. In this vector, len is the length of each individual in the population, lb and up are a row vector representing the lower and upper bounds of each variable. Code is a binary row vector, indicating the encoded form of the substring. Scale is also a binary row vector, which is used to indicate whether the chromosome coding of each individual uses a logarithmic or arithmetic scale, lbin, ubin is a binary row vector indicating that the range is a lagged inclusion boundary.

We can specify lb, ub vector to specify the scope of the parameters, with lbin, ubin to indicate whether there are upper bounds and lower bounds. The range of parameters are decided by our experience, as shown in Table 3:

Table 3: THE SEARCH RANGE OF OPTIMIZATION PARAMETERS

acc_lim_x	min_vel_x	max_vel_x
[0.1,0.3]	[0.0,0.05]	[0.1,0.2]
vxsamples	vtheta_samples	min_trans_vel
[3,9]	[10,30]	[0.03,0.05]
acc_lim_th	max_rot_vel	min_rot_vel
[2,3]	[0.5,0.6]	[0.4,0.5]

D. The Simulation Result

As shown in Figure 7, given the range of parameters, there will be multiple sets of parameters which make the algorithm obtain the optimal trajectory. Though in a set of parameters, most of the trajectories are useless, there are a few ideal trajectories which have a low cost and the algorithm will choose the lowest cost as fitness of the set of parameters. In a population, the average fitness of the population is not far from the optimal value. In other word, the cost of optimal trajectory of parameters set is almost the same. However, there is still a small gap between the cost of optimal trajectory of parameters sets and these gaps may affect the smoothness of the robot trajectory. With the increase of algebra, the number of optimal solutions in the population increases, as shown in Figure 8.

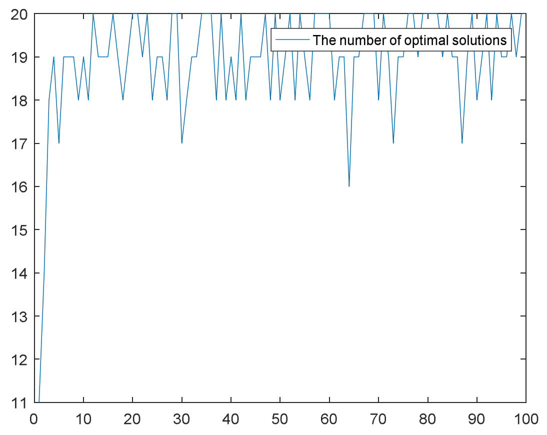

Figure 7. The change of the number of solutions of the population

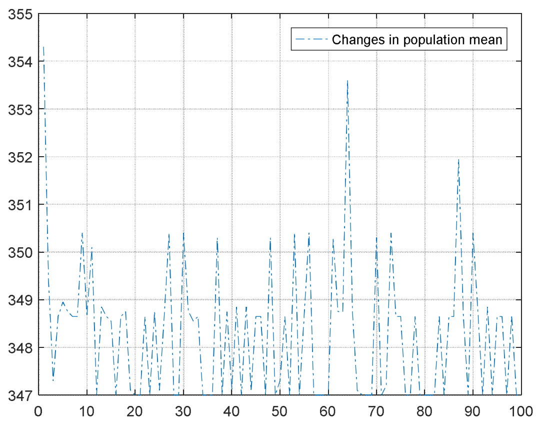

Figure 8. The change of the population mean

VI. Conclusions

In this paper, we set up a mobile robot platform based on laser sensors. With this platform, the local path planning parameters of Turtlebot mobile robots is optimized. Since considering the searching of optimized parameter group as a single-objective optimization problem in our experiment, we use gatbx , the Sheffield University Matlab genetic algorithm toolbox integrated with local path planning algorithm DWA and make it possible for robots to search optimized parameters automatically with given the domain of parameters. In order to get the smooth and usability of the trajectory, we have chosen the following parameters: acc_lim_x, min_vel_x, max_vel_x, acc_lim_th, max_rot_vel, vxsamples, vtheta_samples, min_trans_vel, min_rot_vel. The result reveals that searching optimized parameters in a given range through GA is more convenient and effective than searching it manually and the generated local plan in our experiment is smoother. In addition, Genetic Algorithm shows that, after 10 generations the population tends to be stable and the optimized parameters is more likely to be found, which means that we can shorten the number of iterations of the genetic algorithm and reduce the computational complexity by this way.

Further research on this way may include following aspects:

1) In addition to searching optimized parameter in a single map, you can also search for most of the parameters which can apply in other maps..

2) Besides considering the parameters of the algorithm, the robot hardware parameters can also be considered, such as the radius of the robot.

3) By optimizing algorithm and reducing the complexity, the algorithm can optimize the parameters in real time and then the robot can move more smoothly.

Acknowledge

This research work is supported by National Undergraduate Innovative and Entrepreneurial Training Program (No.201610561123).Guangdong province science and technology plan projects (2015A020219001, 2017A010101031). the Fundamental Research Funds for the Central Universities (2015ZM140,2017MS048). Guangzhou Key Laboratory of Robotics and Intelligent Software (15180007).Shenzhen peacock project (KQTD20140630154026047). Shenzhen basic research projects(JCYJ20160429161539298).

References

[1] F. Dellaert, D. Fox, W. Burgard and S. Thrun, "Monte Carlo localization for mobile robots," *Proceedings 1999 IEEE International Conference on Robotics and Automation (Cat. No.99CH36288C)*, Detroit, MI, 1999, pp. 1322-1328 vol.2.

[2] D. Fox, W. Burgard, F. Dellaert, and S. Thrun, "Monte Carlo localization: Efficient position estimation for mobile robots," in *In Proc. of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, (Orlando, Florida), pp. 343-349, 1999.

[3] Gerkey, Brian P., and Kurt Konolige. "Planning and control in unstructured terrain." *ICRA Workshop on Path Planning on Costmaps*. 2008.

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

[5] Kelly, Alonzo. *An intelligent predictive controller for autonomous vehicles*. No. CMU-RI-TR-94-20. CARNEGIE-MELLON UNIV PITTSBURGH PA ROBOTICS INST, 1994.

[6] Fox, Dieter. "KLD-sampling: Adaptive particle filters." *Advances in neural information processing systems*. 2002.

[7] Khanh, Doan VK, et al. "Single-objective optimization of thermo-electric coolers using genetic algorithm." *AIP Conference Proceedings*. Eds. Sarat Chandra Dass, et al. Vol. 1621. No. 1. AIP, 2014.

[8] Thrun, S., W. Burgard, and D. Fox. "Probabilistic Robotics. Cambridge, Massachusets." (2005).

[9] J. Xiong, Y. Liu, X. Ye, L. Han, H. Qian and Y. Xu, "A hybrid lidar-based indoor navigation system enhanced by ceiling visual codes for mobile robots," *2016 IEEE International Conference on Robotics and Biomimetics (ROBIO)*, Qingdao, 2016, pp. 1715-1720.

[10] Yi-Chun Lin, Chih-Chung Chou and Feng-Li Lian, "Indoor robot navigation based on DWA*: Velocity space approach with region analysis," *2009 ICCAS-SICE*, Fukuoka, 2009, pp. 700-705.