Design of Genetic Algorithm-based Parking System for an Autonomous Vehicle

Xing Xiong and Byung-Jae Choi*

School of Electronic Engineering, Daegu University
Jillyang, Gyeongsan, Gyeongbuk 712-714, Korea
Tel : +82-53-850-6633, Fax : +82-53-850-6619, E-mail : bjchoi@daegu.ac.kr

Abstract

A Genetic Algorithm (GA) is a kind of search techniques used to find exact or approximate solutions to optimization and searching problems. This paper discusses the design of a genetic algorithm-based intelligent parking system. This is a search strategy based on the model of evolution to solve the problem of parking systems. A genetic algorithm for an optimal solution is used to find a series of optimal angles of the moving vehicle at a parking space autonomously. This algorithm makes the planning simpler and the movement more effective. At last we present some simulation results.

Key Words: Genetic Algorithm, Evolutionary Algorithm, Autonomous Vehicle, Parking System, Fitness

1. Introduction

In recent years, autonomous parking problems have attracted a great deal of attention and more intelligent technologies are being applied to automobiles. Automatic parallel parking is an important capability for autonomous ground vehicles (AGVs). It allows the vehicles to be efficiently stored or placed when they are not in operation. It is even more important in military applications since it can be used to hide AGVs in the battlefield, for example between trees or in crevices or small buildings, hence preventing them from being detected or attacked. Several researchers studied path planning methods for the parallel parking control of AGVs, where the vehicles follow a designed sinusoidal path or a curve formed by two circular arcs ([1]-[3]).

There are many approaches suggested by researchers to solve the parking problem of autonomous vehicles in presence of static, based on soft computing. The basic method is to design a control algorithm that makes an automobile follow a reference trajectory via a tracking method. Soft computing consists of fuzzy logic, neural networks and evolutionary computation. There have been many attempts to use fuzzy logic for parking [1, 2]. Recently, it has been widespread using evolutionary computations which are robust, global and mose straightforward to apply in situations where there is little or no prior knowledge about the problem to solve. GA and EA require no derivative information or formal initial estimates of the solution, and because they are stochastic in nature, they are capable of searching the global optimum.

A Genetic Algorithm is a kind of search techniques used to find exact or approximate solutions to optimization and searching problems. GAs is categorized as global search heuristics. They are a particular class of Evolutionary Algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. GAs constructs a population of probable solutions and evolves it over the generations to find a better individual that is close to the optimal solution [11].

GAs are a part of evolutionary computing, which is a rapidly growing area of artificial intelligence. GAs searches the solution space of a function through the use of simulated evolution, i.e., the survival of the fittest strategy. In general, the fittest individuals of any population tend to reproduce and survive in the next generation, thus improving successive generations. As global optimization methods do not require gradient evaluation, GAs is applicable to solving a great range of optimization problems, including determination of the optimal combination of membership functions, and scaling factors reasoning rules ([4]-[8]).

In this paper our approach is based on the evolution programming principle: appropriate data structures and specialized "genetic" operators should do the job of taking care of constraints. The controlled process is the four-wheeled car. We assume that the wheels are fixed parallel to car body and allowed to roll or spin but have no side-slipping. The front wheels can turn to the left or right, however the left and right front wheels must be parallel.

This paper is organized as follows. In section II, we introduce a way for a kinematic equation to move cars. We design the autonomous parking system based on genetic algorithms in Section III. Computer simulation results are given to show the validity of the genetic control algorithm in Section IV. We discuss and conclude in Section V.

2. Autonomous Vehicle and Moving Trajectories

The motion of an autonomous vehicle is firstly discussed [1]. The controlled process of the four-wheeled car is shown in Fig.1.

Manuscript received Nov. 6, 2009; revised Dec 1, 2009.
* Corresponding author
This research was supported (in part) by the Daegu University Research Grant.

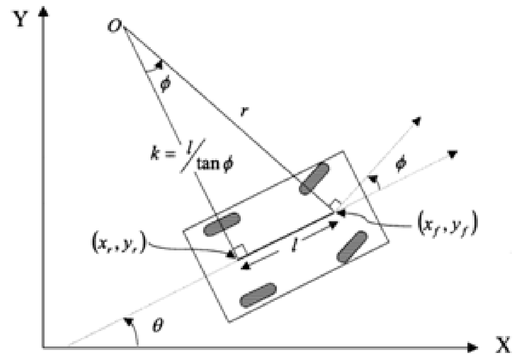


Fig. 1 Kinematic model of an autonomous vehicle

Considering the model shown on Fig.1, the rear wheel of the car are fixed parallel to car body and allowed to roll or spin but not slip. The front two wheels can turn to left or right, but the front wheels must be parallel. All the corresponding parameters of the autonomous vehicle depicted are defined in the Table 1.

Parameter	Meaning
(x_r, y_r)	Position of the front wheel center
(x_f, y_f)	Position of the rear wheel center
Φ	Orientation of the steering-wheels with respect to the frame
θ	Angle between vehicle frame orientation and X-axis
l	Wheel-base of the mobile car
O	Center of curvature
r	Distance from point O to point
k	Curvature of the fifth-order polynomial
V	Velocity
v_i	i^{th} Chromosome
pop_size	Population of chromosomes
p_{sz}	Probability of applying simple crossover
p_{ac}	Probability of applying arithmetical crossover
p_m	Probability of mutation
F	Total fitness of the population
p_i	Probability of a selection
q_i	Cumulative probability

The rear wheel is always tangent to the orientation of the vehicle. The no-slipping condition mentioned previously requires that the autonomous vehicle travels in the direction of its wheels. Thus, we have

$$\dot{y}_r \cos\theta - \dot{x}_r \sin\theta = 0 \tag{1}$$

This is the so-called nonholonomic constraint. The front of the autonomous vehicle is fixed relative to the rear, thus the coordinate (x_r, y_r) is related to (x_f, y_f)

$$\begin{aligned} x_r &= x_f - l\cos\theta \\ y_r &= y_f - l\sin\theta \end{aligned} \tag{2}$$

Thus, differentiating (2) with respect to time gives

$$\begin{aligned} \dot{x}_r &= \dot{x}_f + \dot{\theta} l \sin\theta \\ \dot{y}_r &= \dot{y}_f - \dot{\theta} l \cos\theta \end{aligned} \tag{3}$$

Substituting (3) to (1), we can get

$$\dot{x}_f \sin\theta - \dot{y}_f \cos\theta + \dot{\theta} l = 0 \tag{4}$$

From Fig. 1, we have

$$\begin{aligned} \dot{x}_f &= v\cos(\theta + \phi) \\ \dot{y}_f &= v\sin(\theta + \phi) \end{aligned} \tag{5}$$

Substituting (5) to (4), we can derive

$$\dot{\theta} = v\frac{\sin\phi}{l} \tag{6}$$

Equations (5) and (6) are the kinematic equations of autonomous vehicle with respect to the axle center of the front wheels. We rewrite them in the following:

$$\begin{aligned} \dot{x}_f &= v\cos(\theta + \phi) \\ \dot{y}_f &= v\sin(\theta + \phi) \\ \dot{\theta} &= v\frac{\sin\phi}{l} \end{aligned} \tag{7}$$

Equations (7) are used to generate the next forward state position of the vehicle when the present states and control input are given. Following these equations describing the motion kinematics, we can easily obtain the kinematic equations of the vehicle motion described by the position of the center of the front wheel (x_r, y_r).

3. Design of GA-based Parking System

The central problem in applications of genetic algorithms is the constraints – few approaches to the constraint problem in genetic algorithms have previously been proposed. One of these uses penalty functions as an adjustment to the optimized objective function, other approaches use "decoders" or "repair" algorithms, which avoid building an illegal individual, or repair one, respectively. However, these approaches suffer from the disadvantage of being tailored to the specific problem and are not sufficiently broad to handle a variety of problems.

Our approach is based on the evolution programming principle: appropriate data structures and specialized "genetic" operators should do the job of taking care of constraints.

In this section we present an evolution program GA-based system for this class of parking problems. This class includes the typical application of the optimal control problem.

3.1 Encoding

To solve the optimization problems by a genetic algorithm, an encoding mechanism is very important, as well is as its data structure. For this situation, the variable is a real number to solve the problem, relative to the required special decoding algorithm to rears encoding mechanism; it has superiority to adopt elative decimal encoding system, which ensures searching intelligently for the solution directly in the solution space.

The GA-based system maintains a population of float number vectors. Each vector $v = (v_0, v_1, ..., v_{N-1})$ is a control vector for the parking problem. So mechanism represents the angle of the moving car.

3.2 Initializing species

As discussed at the beginning of this section, there are few constraints which may appear for the given problem.

The GA-based system initializes a population in a way that can satisfy some of these constraints. According to our driving experience, we always turn the steering wheel to the left and right in the car.

The procedure initialization used by the GA-based system is described below:

```
Procedure initialization;
Begin
Random generates,
Repeat
Compute the distance R between the position of the front wheel center ($x_r$, $y_r$) and the centre of a circle
Compute the difference L of the R and radius r
If (L <width/5) then vi=L*/width or vi=sign(L)*/4
i=i+1
until |O+270|<π *5/180
until achieve pop_size
end
```

3.3 Fitness Function

The fitness function plays the role of the environment, rating potential solutions in terms of their fitness. The choice of a fitness function is usually very specific to the problem under conditions. The fitness function of a chromosome measures the objective cost function.

A path can be either feasible (collision free) or infeasible because intermediate nodes can fall on any of the parking area. The evaluation should be able to distinguish feasible and infeasible paths and tell the difference of path qualities within either category. The evaluation function is presented below [8]:

$$f(x) = \begin{cases} 1/((2 * x_f - x_{right\_limit} \\ \quad - x_{left\_limit}) + (\theta - 270°)) & \text{Feasible path} \\ 0 & \text{Infeasible path} \end{cases} \quad (8)$$

The definition of individual fitness is according to the following rule:

1) If the generated path is not feasible (obstacle collision), its fitness is equal to zero.

2) If the generated path is feasible, its fitness is as follows:

- The shorter the distance between the front wheel center and the goal-point is, the higher the fitness.
- The smaller the angle between the car and the goal-point is, the higher the fitness.

3.4 Choice

The choice of the algorithm is carried out according to the manner that the individual suitability is proportional to choice probability, the optimal parent individuals keep directly to the son generation at the same time, so that it can ensure the convergence of the genetic algorithm.

For the selection process (selection of a new population with respect to the probability distribution based on fitness values), a roulette wheel with slots sized according to fitness is used. We constructed such a roulette wheel as follows (we assume here that the fitness values is positive, otherwise, we can use some scaling mechanism).

$$F = \sum_{i=1}^{pop\_size} f(x) \quad (9)$$

$$p_i = f(x)/F \quad (10)$$

$$q_i = \sum_{j=1}^{i} p_j \quad (11)$$

The selection process is based on spinning the roulette wheel *pop_size* times, each time we select (as in chromosome) a single chromosome for a new population.

Now we are ready to spin the roulette wheel *pop_size* times. Each time we select (as in chromosome) a new population.

If the random number r_i and $q_{10}<r<q_{11}$, $v_i=v_{11}$.

Obviously, some chromosomes would be selected more than once. This is in accordance with the Schema Theorem: the best chromosomes get some more copies, the average stay even, and the worst die off.

3.5 Crossover

Crossover is the operator that randomly chooses a node from Parent 1 and the other node from Parent 2. Exchange the parts after these two nodes.

Now we are ready to apply the recombination operator, crossover, to the individuals in the new population (vectors v'_i).

The probability of crossover is p_c=0.25 so we expect that (on average) 25% of chromosomes undergo crossover. For each chromosome in the (new) population we generate a random number r from the range [0…1]; if $r<p_c$, we select a given chromosome for crossover. For each of these two pairs, we generate a random integer number *pos* from the range [1…45] (45 is the total length - number of bits - in a chromosome). The number *pos* indicate the position of the crossing point.

A simple crossover is defined as follows: if $s_v^t =< v_1, v_2, \cdots, v_m >$ and $s_w^t =< w_1, w_2, \cdots, w_m >$ are crossed after the k-th position, the resulting offsprings are:

$$s_v^{t+1} =< v_1, v_2, \cdots, v_m > \text{and } s_w^{t+1} =< w_1, w_2, \cdots, w_m >$$

Arithmetical crossover is defined as a linear combination of two vectors: if s_v^t and s_w^t are to be crossed, the resulting offspring are:

$$s_v^{t+1} = a \bullet s_w^t + (1-a) \bullet s_v^t$$

And

$$s_w^{t+1} = a \bullet s_v^t + (1-a) \bullet s_w^t .$$

This operator uses a simpler static system parameter a∈ [0…1], as it always guarantees closure.

3.6 Mutation

Mutation is performed on a bit-by-bit basis. The mutation process is also applied to flip a bit position randomly in the chromosome.

The probability of mutation is p_m=0.01, so we expect that (on average) 1% of the bits would undergo mutation. There are 45**pop_size* bits in the whole population. Every bit has an equal chance to be mutated, so, for every bit in the population, we generate a random number *r* from the range [0…1].

If $r_{40}<p_m$, we mutate the *40th* bit.

Following selection, crossover, and mutation, the new population is ready for its next evaluation. This evaluation is used to build the probability distribution for the next selection process. The rest of the evolution is just a cyclic repetition of these steps.

Now we apply GA to the algorithm of the autonomous vehicle parking. The summary of the algorithm is as follows: First, initial solutions are randomly generated and evaluated by the fitness function in Equation (8); two parents are selected according to a particular selection mechanism. Then genetic operators are selected and applied to the parents according to probabilities. The whole generation is replaced by children. The best solution so far is updated in each generation, and it will be the final solution when the stop criteria are satisfied. The stop criteria can be when the preset maximum generation is exceeded. The algorithm of parking is based on the genetic algorithm and is summarized:

```
Procedure GA-based Paring System
Begin
◈ Generate initial solutions v randomly
◈ Action
    ★ Compute the fitness for each chromosome in the current population using the fitness function
        ★ Update the best-so-far
    ★ Calculate the fitness value f(x) for each chromosome v_i(i=1,...pop_size)
    ★ Find the total fitness of the population $F = \sum_{i=1}^{pop\_size} f(x)$
    ★ Compute the probability of a selection $p_i$ for each chromosome $v_i$ (i=1,...pop_size): $p_i = f(x)/F$
    ★ Compute a cumulative probability $q_i$ for each chromosome $v_i$ (i=1,...pop_size): $q_i = \sum_{j=1}^{i} p_j$
    ★ Generate a random (float) number r from the range [0…1].
    ★ If $r<q_1$ then select the first chromosome $v_i$; otherwise select the $i^{th}$ chromosome $v_i$ (i=1,2,...pop_size) such that $q_{i-1} < r \le q_i$.
    ★ Generate a random (float) number r from the range [0…1].
    ★ If $r<q_{ac}$ then select a given chromosome for arithmetical crossover.
    ★ Generate a random (float) number r from the range [0…1].
    ★ If $r<q_{sc}$ then select a given chromosome for single crossover.
    ★ Generate a random (float) number r from the range [0…1].
        ★ If $r<q_m$, the mutate the bit.
        ★ Generation = generation + 1.
        ★ End
◈ While (generation<maximum)
◈ End
```

4. Simulation Results

The proposed algorithm has been tested in the MATLAB simulation environment. The simulation of the autonomous vehicle is given to demonstrate the effectiveness of the proposed scheme. Taking real life into account, the length of the garage is about 1.5 times longer than that of the car, and the width of the garage is 1.5 to 2 times wider than that of the car.

In this simulation, we assumed that the length of the garage is 5[m] and the width is 3.65[m]. We exploited the following two cases with several situations to test the effectiveness of the proposed GA-based scheme.

4.1 Ordinary car parking case

The autonomous vehicle used in the simulation has the following dimensions: length L=3.5[m] and width W=2[m]. The simulation result is shown in Fig. 3.

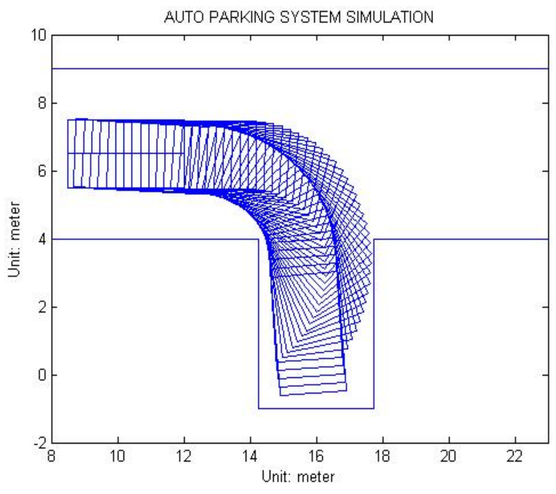


Fig. 2 Simulation result of an ordinary car parking

4.2 Longer car parking case

The autonomous vehicle used in the simulation has the following dimensions: length L=4[m] and width W=2[m]. Simulation result is shown in Fig. 4.

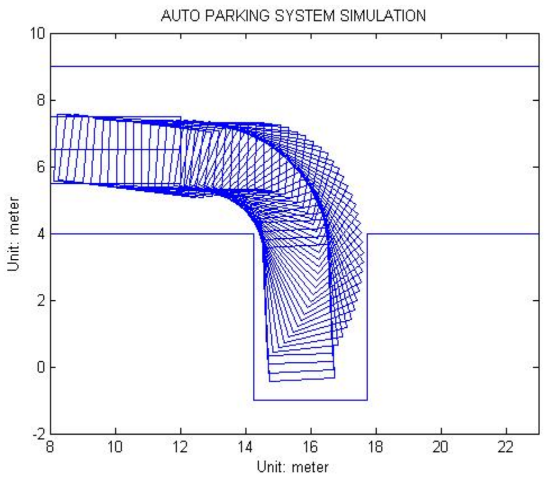


Fig. 4 Simulation result of a longer car parking

5. Concluding Remarks

In this paper, we have explained the moving equations of an autonomous car-like vehicle, designed its parking path using a genetic algorithm for optimal performance and a trajectory tracking controller. We incorporate problem-specific knowledge into our systems, using evolution programs. The GA-based parking system is a stronger method than the conventional parking system and can be applied to a function with linear constraints only. The advantage of the proposed method is that it yields an effective parking path and further extends the controlled workspace to a larger region. By the use of a GA, the performance of the parallel parking algorithm is tuned to behave well for both types of steering systems in **tight** parking spaces. We proved for the same results when we examine more specific classes dealing with the optimal control problem.

References

[1] T.-H S. Li and S.-J. Chang, “Autonomous fuzzy parking control of a car-like mobile robot”, *Proc. IEEE Int. Conf. Systems, Man and Cybernetics,* Part A, vol. 3, pp. 415-465, 2003.

[2] Y.-H Hao, T.-K Kim and B.-J. Choi, “Design of fuzzy logic based parking systems for intelligent vehicles”, *J. of Korean Institute of Intelligent Systems*, vol. 18, pp. 1-6, 2008.

[3] Y. Zhao, E.-G Collins and D. Dunlap, “Design of genetic fuzzy parallel parking control systems”, *Proc. The American Control Conference, Denver.* pp. 4107-4112, 2003.

[4] O. Bühler and J. Wegener, “Automatic testing of an autonomous parking system using evolutionary computation”, *SAE Int.*, 2004.

[5] H.-D. Zhang, B.-H. Dong, Y.-W. Cen and R. Zheng, “Path planning algorithm for mobile robot based on path grids encoding novel mechanism”, *in Int. Conference on Natural Computation,* vol. 3, 2007.

[6] X.-Z. Hu and C.-X. Xie, “Niche genetical agorithm for robot path planning”, *Int. Conference on Natural Computation,* vol. 3, 2007.

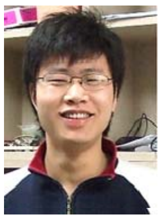
[7] T.-R. Wan, H. Chen and R. Earnshaw, “Real-time path planning for Navigation in unknown environment”, *Proc. Theory and Practice of Computer Graphics,* vol. 3, 2003.

[8] B.-R. Geiger, J.-F Horn, A.-M. Delullo and L.-N. Long, “Optimal path planning of UAVs using direct collocation with nonlinear programming”, *AIAA GNC Conference*, pp. 1-13, 2006.

[9] B. J. Choi, S. W. Kwak and B. K. Kim, "Design and stability analysis of single-input fuzzy logic controller," *IEEE Trans. on SMC(B),* vol. 30, no. 2, pp. 303-309, 2000.

[10] Zbigniew Michalewicz, “Genetic Algorithms + DataStructures = Evolution Programs”, *Springer-Verlag*, 1994

[11] A. Munawar, M. Wahib, M. Munetomo, and K. Akama, "A Survey: Genetic Algorithms and the Fast Evolving World of Parallel Computing", *The 10th IEEE Int. Conference on High Performance Computing and Communications*, pp.897-902, Sept. 2008.



Xing Xiong
He received his B.S. degree in automation engineering at NanJing Institute of Technology, and M.S. degree from Daegu University. He is currently Doctor Course in the School of Electronic Engineering from Daegu University. His research interests include intelligent control and embedded systems.

Phone : 053-850-4432
E-mail : xiongxing@nate.com

Byung-Jae Choi
He received his B.S. degree in electronic engineering at Gyeongbuk National University, and M.S. degree and Ph.D. from KAIST. Since 1999 he is working at Daegu University. His research interests include intelligent control and systems.

Phone : 053-850-6633
Fax : 053-850-6619
E-mail : bjchoi@daegu.ac.kr