A Quantum PSO Algorithm for Feedback Control of Semi-Autonomous Driver Assistance Systems

Che-Cheng Chang, Jichiang Tsai and Shi-Jia Pei

Abstract

The development of automotive technology has become increasingly important for preventing car accidents. Hence, as a basic research of driver assistance systems, we propose a novel control method for steering support in this paper. In particular, we investigate the stability and limitation of such a system for the safe and comfortable drive. First, we use the particle swarm optimization (PSO) based algorithm to search the optimal feedback gain under practical constraints for achieving tracking control. Moreover, to reduce the convergence time further, the particle swarm optimization algorithm is combined with the technique of *quantum computing*. Simulation results indicate that the proposed feedback controller based on quantum particle swarm optimization (QPSO) has the ability to provide efficient computational performance for trajectory tracking and stabilization.

Index Terms

Driver Assistance Systems, Particle Swarm Optimization, Quantum Computing, Feedback Control.

C.-C. Chang is with the Department of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan. Email: d9864001@mail.nchu.edu.tw .

J. Tsai is with the Department of Electrical Engineering, National Chung Hsing University, Taichung, Taiwan. Email: jichiangt@nchu.edu.tw .

S.-J. Pei is with Philips & Lite-On Digital Solutions Corp., Taipei, Taiwan. Email: pagandog@gmail.com .

Address of Correspondent: Jichiang Tsai. Department of Electrical Engineering, National Chung Hsing University, Taichung 40227, Taiwan. TEL: (886-4)2284-0688 ext. 821. FAX: (886-4)2285-1410. Email: jichiangt@nchu.edu.tw .

I. INTRODUCTION

An ultimate goal of automatic vehicle control is the ability of being automatically driven by machines in order to reduce accidents caused by human errors and to improve safety as well as drive comfort. In recent years, there has been considerable progress in developing such *Automatic Vehicle Control System (AVCS)* [1]-[4]. Particularly, these researches focused on how to design the AVCS in different ways as well as showing the efficiency of their methods. Moreover, upon implementing these methods in practice, a large number of sensors is required to establish the systems. Unfortunately, even with these sensors, there still exist some critical problems necessary to be considered for these systems. For example, when sensors detect some obstacles in the blind spot of the driver, the car turns suddenly without noticing the driver first, or drivers find obstacles but sensors do not detect anything. Both foregoing scenarios may incur accidents such that drivers tend be panic due to lack of confidence on the AVCS.

For the above reasons, the driving support system where part of control depends on the driver has been extensively studied in the literature [5]-[9]. More specifically, these systems do not directly control the vehicle, but just try to assist the driver to control the vehicle. For instance, by monitoring the health of the driver, the system can realize whether the driver is suitable for driving. Among these systems, the most important one is the semi-autonomous driver assistance system. Such a system will automatically control the steering wheel to follow the designed path while the driver steps on the accelerator [5], [6]. Now that the velocity cannot be influenced by any feedback controller, the traditional kinematic vehicle model cannot be applied to achieving tracking control here. Thus to solve the aforementioned problem, the authors in [6] proposed a new differentially flat model accompanied with the time-scaling input. However, the optimal feedback gain becomes difficult to decide in this model. Furthermore, because information of the velocity cannot be obtained in advance and there are some physical constraints in a real system, the Linear Quadratic Regular (LQR) optimization method introduced in [10] is also unsuitable for dealing with the problem.

In this paper, to solve the above problem, we use the Particle Swarm Optimization (PSO) based method to search the optimal feedback gain under real constraints. By doing so, tracking control of the vehicle with the time-scaling input becomes able to be realized at any velocity of the car and any constraint of the steering angle. Moreover, to reduce the convergence time further,

our algorithm is combined with the technique of *quantum computing*. Simulation results show that our semi-autonomous driver assistance system equipped with the QPSO-based feedback gain controller can issue instructions to control the steering wheel autonomously when the driver steps on the accelerator. Also, our system can reduce the tracking error along the path exponentially.

This paper is structured as follows. Section II describes the preliminary knowledge about the semi-autonomous driver assistance system. In Section III, we introduce how to utilize the PSO and QPSO methods to design the feedback gain controller for our system, and the simulation results for evaluating our methods are presented in Section IV. At last, we make a conclusion for our work as well as presenting some possible future work in Section V.

II. Theory and Applications

A. Flatness and Time-Scaling

A notable approach for designing the controller of a nonlinear system is the input-output linearization algorithm [11]. First, to extend the notion of controllability from linear systems to nonlinear dynamical systems, the flatness theory has been introduced by using the formalism of differential algebraic methods [12], [13]. A system is said to be *flat* if it has a flat output that can be used to explicitly express all states, all inputs and a finite number of its derivatives [14]. On the other hand, time-scaling is widely used in the control theory due to its convenience for system analysis. Particularly, a new time scale τ depending on the state x is introduced and the state equation of the system is rewritten in this time scale such that the relation between the time t and τ is defined as the continuous function shown below:

$$\frac{dt}{d\tau} = s(x, \mu), \tag{1}$$

where μ is the time-scaling input and the function $s(x, \mu)$ is called the *time-scaling function*.

B. Semi-Autonomous Driver Assistance System

Like the works in [5], [6], for our system, we suppose the longitudinal velocity v, denoted by v_{car}, is generated by the driver in the Ackermann steering geometry. Since there is only one input from the controller, i.e. the steering angle φ of the front wheel, we have an affine system with no drift term. However, such a system is flat only when there are more than two inputs.

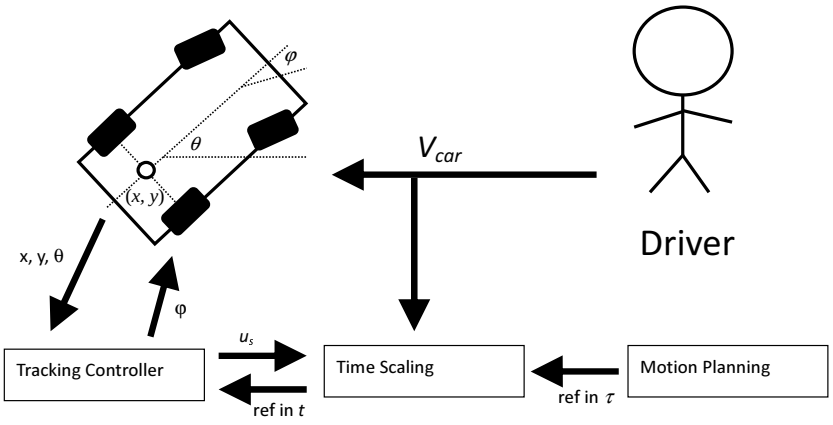

Fig. 1. The scheme of the closed loop with time-scaling.

Therefore, one way to obtain a differentially flat model is using time-scaling to add an extra input variable. More specifically, doing so can lead to the following time-scaling function:

$$\frac{dt}{d\tau} = \frac{u_s}{v_{car}}. \tag{2}$$

Then we can derive the following equations for the kinematic vehicle model by using the previous time-scaling function:

$$x' = u_s \cos\theta. \tag{3}$$

$$y' = u_s \sin\theta. \tag{4}$$

$$\theta' = \frac{u_s}{l} \tan\varphi. \tag{5}$$

Here, we use Figure 1 to illustrate our semi-autonomous driver assistance system. In particular, such a system is derived from three major components: *Motion planning*, *Time-scaling of the reference trajectory* and *Tracking feedback*, for tracking the trajectory that has been planned in advance while the driver generates the longitudinal velocity v_{car} to the car. Note that the task of motion planning is to define the geometry of the reference path. There are several ways for realizing the motion planning scheme [6], [15]. Although a higher degree polynomial can represent a more complicated path, a seventh degree polynomial is sufficient for most paths in practice. Hence, just like the work proposed in [6], the trajectories are defined as:

$$x_{\tau,ref} = \sum_{i=0}^{7} a_{x,i}\tau^i, \qquad y_{\tau,ref} = \sum_{i=0}^{7} a_{y,i}\tau^i. \tag{6}$$

According to [6], we can get the resultant scaling for the x-direction of the reference path:

$$x_{ref}(t) = x_{\tau,ref}(\tau(t)), \tag{7}$$

$$\dot{x}_{ref}(t) = x'_{\tau,ref}(\tau(t))\dot{\tau}, \tag{8}$$

$$\ddot{x}_{ref}(t) = x''_{\tau,ref}(\tau(t))\dot{\tau}^2 + x'_{\tau,ref}(\tau(t))\ddot{\tau}, \tag{9}$$

$$\dddot{x}_{ref}(t) = x'''_{\tau,ref}(\tau(t))\dot{\tau}^3 + 3x''_{\tau,ref}(\tau(t))\dot{\tau}\ddot{\tau} + x'_{\tau,ref}(\tau(t))\dddot{\tau}, \tag{10}$$

The other y-direction can be obtained in a similar manner. Finally, also according to [6], if we choose the control inputs u_1 and u_2 to be the following form:

$$\begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\frac{z_1^2 \sin\theta}{l\cos^2 z_3} \\ \sin\theta & \frac{z_1^2 \cos\theta}{l\cos^2 z_3} \end{bmatrix}^{-1} \begin{bmatrix} v_x - f_1 \\ v_y - f_2 \end{bmatrix}, \tag{11}$$

we can get a linear relationship between the outputs and the new inputs v_x and v_y as below:

$$x''' = v_x, \qquad y''' = v_y. \tag{12}$$

As a result, the design of the tracking controller becomes simple in light of existing linear control techniques. Here, let $e_x = x - x_{\tau,ref}$ and $e_y = y - y_{\tau,ref}$ be the tracking errors. Then we define the two new inputs v_x and v_y as:

$$v_x = x'''_{\tau,ref} - k_{x,2}e''_x - k_{y,1}e'_x - k_{x,0}e'_x, \tag{13}$$

$$v_y = y'''_{\tau,ref} - k_{y,2}e''_y - k_{y,1}e'_y - k_{y,0}e'_y, \tag{14}$$

where $k_{a,i}$, for $a \in \{x, y\}$ and $i = 0, 1, 2$, is the feedback gain. By choosing these gains such that the corresponding characteristic polynomials have all their roots in the left part of the complex plane, the tracking errors of the close loop system are given as follows:

$$e'''_x + k_{x,2}e''_x + k_{x,1}e'_x + k_{x,0}e_x = 0, \tag{15}$$

$$e'''_y + k_{y,2}e''_y + k_{y,1}e'_y + k_{y,0}e_y = 0. \tag{16}$$

III. QPSO-Based Feedback Gain Controller

A. Determination of Feedback Gains

First, we explain how to determine the feedback gains by using Routh-Hurwitz stability criterion [16], [17]. Particularly, this criterion states that the necessary and sufficient condition for stability is that all elements in the first column of the Routh array must be positive. Here, with the operation of Laplace transformation to transform equation (15), we obtain the following characteristics equation for x:

$$s^3 + k_{x,2}s^2 + k_{x,1}s + k_{x,0} = 0, \tag{17}$$

where s is the Laplace operator. Then we continue to construct the Routh array:

$$\begin{array}{lll} s^3: & 1 & k_{x,1} \\ s^2: & k_{x,2} & k_{x,0} \\ s^1: & -\frac{k_{x,0}-k_{x,1}k_{x,2}}{k_{x,2}} & \\ s^0: & k_{x,0} & \end{array} \tag{18}$$

After applying the same procedure to (16) as well, we can obtain the conditions on the feedback gains for a system to be stable below:

$$k_{x,0} > 0, \qquad k_{x,1} > 0, \qquad k_{x,2} > 0, \qquad k_{x,1}k_{x,2} > k_{x,0}, \tag{19}$$

$$k_{y,0} > 0, \qquad k_{y,1} > 0, \qquad k_{y,2} > 0, \qquad k_{y,1}k_{y,2} > k_{y,0}. \tag{20}$$

B. Particle Swarm Optimization

PSO is an optimization technique based on the movement and intelligence of swarms [18], [19], [20], [21]. Its concept is in light of the simulation on the behavior of bird flocking. In the particle swarm algorithm, the trajectory of each individual in the search space is adjusted by dynamically altering the velocity of each particle, according to its own flying experience and the flying experience of other particles in the search space. In the algorithm, each particle keeps track of its coordinates in the hyperspace that are associated with the best solution ($pbest$), and the global particle swarm optimizer keeps track of the overall best value obtained by any particle ($gbest$) [21]. Particularly, some researches suggested that dynamic parameters can achieve better control on the search space [22], [23]. For example, in [22], the authors have observed that the optimal solution can be improved by varying the value of the inertia weight factor ω from 0.9 at

the beginning of the search to 0.4 at the end of the search for most problems. Moreover, in [23], for PSO with time-varying acceleration coefficients, by changing $c1$ from 2.5 to 0.5 and $c2$ from 0.5 to 2.5, the algorithm can control the local search more efficiently and thus can converge more quickly. Here, the position vector x and the velocity vector v of the whole n particles in the D dimensional search space are represented as:

$$x = [x_1^d, x_2^d, \cdots, x_i^d, \cdots, x_n^d], \qquad i \in \{1, 2, \cdots, n\}, \qquad d \in \{1, 2, \cdots, D\}, \tag{21}$$

$$v = [v_1^d, v_2^d, \cdots, v_i^d, \cdots, v_n^d], \qquad i \in \{1, 2, \cdots, n\}, \qquad d \in \{1, 2, \cdots, D\}, \tag{22}$$

Furthermore, according to a user-defined fitness function, the best position $pbest$ of each particle and the fittest particle $gbest$ found so far are denoted below:

$$pbest = pbest_i^d, \qquad i \in \{1, 2, \cdots, n\}, \qquad d \in \{1, 2, \cdots, D\}, \tag{23}$$

$$gbest = pbest_g^d, \qquad g \in \{1, 2, \cdots, n\}, \qquad d \in \{1, 2, \cdots, D\}, \tag{24}$$

Then, the new velocities and new positions of the particles for the next fitness evaluation are calculated from the following two equations:

$$v_i^d(t+1) = \omega v_i^d(t) + c_1 rand_1(pbest_i^d(t) - x_i^d(t)) + c_2 rand_2(pbest_g^d(t) - x_i^d(t)), \tag{25}$$

$$x_i^d(t+1) = x_i^d(t) + v_i^d(t+1) * T_{int}, \tag{26}$$

where the inertia weight factor ω and the acceleration coefficients c_1 and c_2 are given by the designer, $rand_1$ and $rand_2$ uniformly distributed random numbers in the range $[0, 1]$ to represent the stochastic behaviors, and T_{int} the time interval, which equals 1 in Equation (26).

In summary, the steps of our particle swarm optimization algorithm are stated as follows:

1) Initialize each of n particles with a random position x_i and a velocity v_i on D dimensions in the search space, $\forall i \in [1, n]$.
2) For every particle, evaluate the desired optimization fitness function in D variables.
3) Compare the current evaluation with the previous best value ($pbest_i$) of each particle:

   If the current value $<$ $pbest_i$, then

   $pbest_i =$ the current value, and

   $pbest_i^d =$ the current position in D dimensions.
4) Compare the current evaluation with the previous best value ($pbest_g$) of the group:

If the current value $< pbest_g$, then

g = the array index of each particle.

5) Change the velocity and position of the particle according to equations (25) and (26).
6) Change the inertia weight factor ω and the acceleration coefficients c_1 and c_2.
7) Repeat from Step 2 until satisfying the terminative condition.

C. Quantum Particle Swam Optimization

To further reduce the convergence time and obtain better feedback gains, we incorporate the technique of quantum computing into our PSO algorithm. Particularly, in a quantum PSO algorithm, a particle has to be defined based on a quantum bit [24]. Therefore, we define the quantum particle vector $P = [P_1, P_2, \cdots, P_m]$, where m is the particle size, with the coding based on the probability amplitude. Here, we have:

$$P_i = \left[\left| \begin{matrix} \cos\theta_{i1} \\ \sin\theta_{i1} \end{matrix} \right| \left| \begin{matrix} \cos\theta_{i2} \\ \sin\theta_{i2} \end{matrix} \right| \left| \begin{matrix} \cdots \\ \cdots \end{matrix} \right| \left| \begin{matrix} \cos\theta_{in} \\ \sin\theta_{in} \end{matrix} \right| \right], \tag{27}$$

where n is the degree of dimensions and $\theta_{ij} = 2\pi * rand$, $\forall i \in [1, m]$ and $\forall j \in [1, n]$, where $rand$ is a random number generator that generates a uniformly distributed random number in the range $[0, 1]$. Since every particle has two positions separately corresponding to quantum states $|0>$ and $|1>$ in the search space, a particle can be expressed as follows:

$$P_{ic} = (\cos\theta_{i1}, \cos\theta_{i2}, \cdots, \cos\theta_{in}), \tag{28}$$

$$P_{is} = (\sin\theta_{i1}, \sin\theta_{i2}, \cdots, \sin\theta_{in}), \tag{29}$$

where P_{ic} is the cosine position vector and P_{is} is the sine position vector. Besides, each particle size in the search space is $[-1, 1]$ in QPSO. Hence, to obtain a fitness value from the designed fitness function, we have to do a transformation for the solution space such that the positions of P_{ic} and P_{is} can be confined to the search space of the real problem:

$$X_{ic} = \frac{1}{2}[b_i(1 + \alpha_i^j) + a_i(1 - \alpha_i^j)], \tag{30}$$

$$X_{is} = \frac{1}{2}[b_i(1 + \beta_i^j) + a_i(1 - \beta_i^j)], \tag{31}$$

where α_i^j is the probability amplitude of quantum state $|0>$ and β_i^j is that of state $|1>$.

Next, the movement of a particle position is implemented by the quantum rotation gate. Specifically, we assume that the best position searched from the particle P_i is the cosine position:

$$P_{il} = (\cos\theta_{il1}, \cos\theta_{il2}, \cdots, \cos\theta_{iln}), \tag{32}$$

and the global best position is:

$$P_g = (\cos\theta_{g1}, \cos\theta_{g2}, \cdots, \cos\theta_{gn}). \tag{33}$$

Furthermore, the updating rules of the quantum state are described as follows:

1) The update of the quantum argument:

$$\Delta\theta_{ij}(t+1) = \omega\Delta\theta_{ij}(t) + c_1 r_1(\Delta\theta l) + c_1 r_1(\Delta\theta g), \tag{34}$$

where

$$\begin{aligned}
\Delta\theta_l &= \begin{cases} 2\pi + \theta_{ilj} - \theta_{ij}, & \theta_{ilj} - \theta_{ij} < -\pi \\ \theta_{ilj} - \theta_{ij}, & -\pi < \theta_{ilj} - \theta_{ij} < \pi \\ \theta_{ilj} - \theta_{ij} - 2\pi, & \theta_{ilj} - \theta_{ij} > \pi \end{cases} \\
\Delta\theta_g &= \begin{cases} 2\pi + \theta_{gj} - \theta_{ij}, & \theta_{gj} - \theta_{ij} < -\pi \\ \theta_{gj} - \theta_{ij}, & -\pi < \theta_{gj} - \theta_{ij} < \pi \\ \theta_{gj} - \theta_{ij} - 2\pi, & \theta_{gj} - \theta_{ij} > \pi. \end{cases}
\end{aligned} \tag{35}$$

2) The update of the quantum probability amplitude based on the quantum rotation gate:

$$\begin{aligned}
\begin{bmatrix} \cos(\theta_{ij}(t+1)) \\ \sin(\theta_{ij}(t+1)) \end{bmatrix} &= \begin{bmatrix} \cos(\Delta\theta_{ij}(t+1)) & -\sin(\Delta\theta_{ij}(t+1)) \\ \sin(\Delta\theta_{ij}(t+1)) & \cos(\Delta\theta_{ij}(t+1)) \end{bmatrix} \begin{bmatrix} \cos(\theta_{ij}(t)) \\ \sin(\theta_{ij}(t)) \end{bmatrix} \\
&= \begin{bmatrix} \cos(\theta_{ij}(t) + \Delta\theta_{ij}(t+1)) \\ \sin(\theta_{ij}(t) + \Delta\theta_{ij}(t+1)) \end{bmatrix}.
\end{aligned} \tag{36}$$

As a result, the new positions of particle P_i become:

$$P_{ic} = (\cos(\theta_{i1}(t) + \Delta\theta_{i1}(t+1)), \cdots, \cos(\theta_{in}(t) + \Delta\theta_{in}(t+1))), \tag{37}$$

$$P_{is} = (\sin(\theta_{i1}(t) + \Delta\theta_{i1}(t+1)), \cdots, \sin(\theta_{in}(t) + \Delta\theta_{in}(t+1))). \tag{38}$$

Moreover, a lack of particle diversity in a search process is often the reason of making PSO converge at a local minimum. So we add a mutation operator in our QPSO algorithm and then decide whether to change according to a mutation probability to avoid premature convergence:

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \cos(\theta_{ij}) \\ \sin(\theta_{ij}) \end{bmatrix} = \begin{bmatrix} \sin(\theta_{ij}) \\ \cos(\theta_{ij}) \end{bmatrix} = \begin{bmatrix} \cos(\frac{\pi}{2} - \theta_{ij}) \\ \sin(\frac{\pi}{2} - \theta_{ij}) \end{bmatrix}. \tag{39}$$

Last but not least, the steps of the proposed QPSO algorithm are summarized below:

1) Initialize the quantum particle vector for particles according to (27).
2) Do a transformation of the solution space based on (28) and (29) for calculating the fitness value from the defined fitness function, and then update the previous best value of the particle along with the previous best value of the group.
3) Update the particle states according to (35) and (36).
4) With a mutation probability, perform a mutation based on (39).
5) Repeat from Step 2 until satisfying the terminative condition.

D. Fitness Function

To solve an optimization task with given constrained conditions, one of the approaches to handle this problem has been developed in [25]. However, such an approach is under the assumption that the fitness function is differentiable. Also, the algorithm is complex due to the augmented Lagrangian. Therefore, the authors in [26] proposed another simple way to handle PSO algorithms under various constraints. More specifically, the fitness function with multiple constraint conditions is mathematically formulated as follows:

$$\underbrace{\text{minimize } f(x)}_{x\in F}, \qquad F = \{x \in R^n |\ h(x) < 0\}, \tag{40}$$

where F denotes the feasible region and $h(x)$ is the constraint function. Note that if F is not empty, the minimum value of the fitness function is the optimal solution under the constrained conditions. Hence, to achieve this end, it is necessary to find the virtual objective function $f_v(x)$ that satisfies the two properties below:

1) $f_v(x) < 0$, $\forall x \in F$.
2) $f_v(x_a) < f_v(x_b)$ for $x_a < x_b$.

There are several ways to choose $f_v(x)$, one possible candidate is:

$$f_v(x) = \tan^{-1} f(x) - \pi/2. \tag{41}$$

With the above virtual objective function, the fitness function with multiple constraint conditions shown in (40) is accordingly modified as:

$$\underbrace{\text{minimize } f_m(x)}_{x\in R^n}, \qquad f_m(x) = \begin{cases} h_{max}(x), & \text{if } h_{max}(x) \geq 0 \\ f_v(x), & \text{otherwise} \end{cases} \tag{42}$$

where $h_{max}(x)$ is the maximum value among all entries of $h(x)$ in (40).

In light of the foregoing technique, we begin to design the required fitness function for our algorithm. First, according to the discussion in subsection II.B, we know that we need the position vector x, which contains six feedback gains. Second, we incorporate such six feedback gains into the semi-autonomous driver assistance system to obtain values of the steering angles and tracking errors between the reference path and tracking path by simulation. Third, we define the constraint function as:

$$h(x) = |\varphi(x)| - \varphi_{lim}, \tag{43}$$

where $\varphi(x)$ denotes the value of the steering angle gotten by simulation and φ_{lim} is the limit of the steering angle of the real car. Therefore, the original fitness function $f(x)$ modulated by the tracking errors becomes:

$$f(x) = sum((e_x(x))^2 + (e_y(x))^2), \tag{44}$$

where function $sum()$ is used to calculate the sum, and $e_x(x)$ and $e_y(x)$ denote values of the tracking errors between the reference path and tracking path in the x and y directions gotten by simulation, respectively. Then we can calculate by (42) the best fitness value for changing the velocity and position of the particle. Below, the steps of the designed fitness function are listed:

1) Set the limit of the steering angle.
2) Obtain values of the steering angles and the tracking errors by simulation.
3) Calculate the maximum value of the constrain function $h(x)$ in (43).

    If $h_{max}(x) > 0$, then

        The fitness value = $h_{max}(x)$.

    Else

        Calculate the value of the original fitness function $f(x)$ by (44), and
        Calculate the value of the virtual objective function $f_v(x)$ by (41), and
        The fitness value = $f_v(x)$.

4) Return the fitness value to the PSO algorithm.

Although we have not added the constraint function on feedback gains yet here, the fitness values can be obtained in a situation that the corresponding feedback gains almost satisfy the conditions of (19) and (20). The reason is that the PSO algorithm will search the feedback

gains where $f(x)$ has the minimum value. However, the previous situation does not hold all the time. This means that if there does not exist any solution under the constraint condition, the fitness values will not be calculated from $f(x)$. Therefore, the feedback gain may not satisfy the stable conditions at this case. To guarantee the system stability, it is necessary to incorporate the constraint function of feedback gains into the fitness function. Moreover, the feedback gains are sometimes unstable in the beginning because the range of the position vector x is not limited. Since in simulation an error will occur when the system is unstable, we have to identify whether the feedback gains satisfy the stable conditions of (19) and (20) to avoid such a situation at first.

According to the previous discussions, we modify the foregoing fitness function. In particular, if the feedback gains make the system unstable, we do not feed it to the simulation. Instead, we use the worse fitness values from the stability fitness function $f_s(x)$ below:

$$f_s(x) = 90 - \varphi_{lim} + |min(x)|. \tag{45}$$

For example, if we set the limit value of the steering angle to 30 degrees, the maximum value of $h_{max}(x)$ will be 60 degrees because the maximum value of $\varphi(x)$ is smaller than 90 degrees. In the previous equation, the term $|min(x)|$ means that the smaller the minimum value of the minus feedback gain is, the worse the fitness value becomes. Subsequently, we add the stability conditions to the fitness function with multiple constraint conditions presented in (42):

$$\text{minimize } f_{ms}(x), \qquad x \in R^n, \qquad f_{ms}(x) = \begin{cases} f_s(x), \text{if x does not satisfy stability conditions} \\ f_m(x), \text{otherwise} \end{cases} \tag{46}$$

As a result, the steps of the final fitness function are summarized as follows:

1) Set the limit value of the steering angle.
2) Identify whether the feedback gain satisfies conditions of (19) and (20) or not.

   If not, then

   Calculate values of the stability fitness function $f_s(x)$ by (45), and

   The fitness value = $f_s(x)$, and jump to Step 5.
3) Obtain values of the steering angles and the tracking errors by simulation.
4) Calculate the maximum value of the constraint function $h(x)$ in (43).

   If $h_{max}(x) > 0$, then

   The fitness value = $h_{max}(x)$.

Else

Calculate the value of the original fitness function $f(x)$ by (44), and
Calculate the value of the virtual objective function $f_v(x)$ by (41), and
The fitness value = $f_v(x)$.

5) Return the fitness value to PSO algorithm.

IV. Simulation Study

In the beginning, we describe the settings of our simulation experiments below:

1) The initial conditions of the vehicle:
   a) The speed v_{car} is set at $2.8m/s$.
   b) Three cases about the distance l between the front and rear axles are considered: $0.256m$, $2.56m$ and $5.12m$.
   c) Three cases about the limit of the steering angle are considered: $\pi/2$, $\pi/3$ and $\pi/6$.
   d) The initial positions are $x(0) = -1.5m$, $y(0) = 2m$, and $\theta(0) = \pi/4$.
2) The parameters of PSO:
   a) The particle size is 20 and each particle contains 6 feedback gain values with $x_i = [k_{x,0}, k_{x,1}, k_{x,2}, k_{y,0}, k_{y,1}, k_{y,2}]$ and $x = [x_1, x_2, \cdots, x_{20}]$.
   b) The inertia weight ω is 0.9 at the beginning of the search and then down to 0.4.
   c) The acceleration coefficient c_1 is 2.5 at the beginning of the search and then down to 0.5.
   d) The acceleration coefficient c_2 is 0.5 at the beginning of the search and then up to 2.5.
   e) The iteration step is 60.
3) The reference trajectories:
   a) The simple straight line: $x = \tau m$, $y = 1m$.
   b) The quadratic curve: $x = \tau m$, $y = \tau^2 m$.
   c) The polynomial curve, as shown in (6), with $a_{x,0} = 0$, $a_{x,1} = 1$, $a_{x,2} = 0$, $a_{x,3} = 0$, $a_{x,4} = 0$, $a_{x,5} = 0$, $a_{x,6} = 0$, $a_{x,7} = 0$, $a_{y,0} = 0.000411483579975819$, $a_{y,1} = -0.0239451269963952$, $a_{y,2} = 0.0686056335377441$, $a_{y,3} = -0.0339753187056231$, $a_{y,4} = 0.0184493748597237$, $a_{y,5} = -0.00337671401022544$, $a_{y,6} = 0.000249281347661342$ and $a_{y,7} = -6.52498884905689 * 10^{-6}$.

Now we present the simulation results for different kinds of reference trajectories. First, the simulation results for tracking the simple straight line with different limits on the steering angle are shown in Figure 2. In particular, the reference trajectory starts from the point $(x_{\tau,ref}(0) = 0, y_{\tau,ref}(0) = 1)$ and arrives at the point $(x_{\tau,ref}(T) = 10, y_{\tau,ref}(T) = 1)$, where $T = 10s$ is the total driving time. Next, the simulation results for tracking the quadratic curve with different limits on the steering angle are shown in Figure 3. Here, the reference trajectory starts from the point $(x_{\tau,ref}(0) = 0, y_{\tau,ref}(0) = 0)$ and arrives at the point $(x_{\tau,ref}(T) = 10, y_{\tau,ref}(T) = 100)$. Finally, the simulation results for tracking the polynomial curve with different limits on the steering angle are shown in Figure 4. For this curve, the reference trajectory starts from the point $(x_{\tau,ref}(0) = 0, y_{\tau,ref}(0) = 0)$ and arrives at the point $(x_{\tau,ref}(T) = 10, y_{\tau,ref}(T) = 3.5)$. Note that in our system, after the vehicle has caught the reference trajectory, the errors produced by our system will soon become nearly zero. Hence, by imposing deviation between the reference and the track trajectories on the simulation at the beginning, we can show the ability of our system that can help the vehicle catch the reference trajectory quickly. As expected, we can find that all the tracking actions along the considered three reference trajectories are achieved for each limit of the steering angle, with similar geometries to the corresponding real paths. Particularly, in these simulations, the time that the vehicle needs to catch the reference trajectories is about one second when the limit of the steering angle is $\pi/2$. Even with the strictest steering angle limit $\pi/6$, the time required to catch the reference trajectories is only about three seconds. In summary, the larger the limit of the steering angle is, the better the performance is. Still, for a smaller steering angle limit, the tracking trajectory is smooth. On the other hand, we also show the convergence times of the PSO and QPSO algorithms for the foregoing three cases in Figure 5. We can see that all results demonstrate that the QPSO algorithm has better performance than the PSO one. In other words, the QPSO method can find the optimal feedback gain in a more efficient manner than the PSO one.

Last but not least, we utilize two simulation results to show two more critical properties of our system. First, the simulation results presented in Figure 6 demonstrate that our system can tolerate finite disturbances very well. Specifically, in each of these experiments, the vehicle starts at the beginning point of the reference trajectory. Then we add a disturbance on the vehicle, causing it to turn 90 degrees from the current direction suddenly. Note that that if the vehicle is with a larger steering angle limit, the deviation between the reference and track trajectories can be

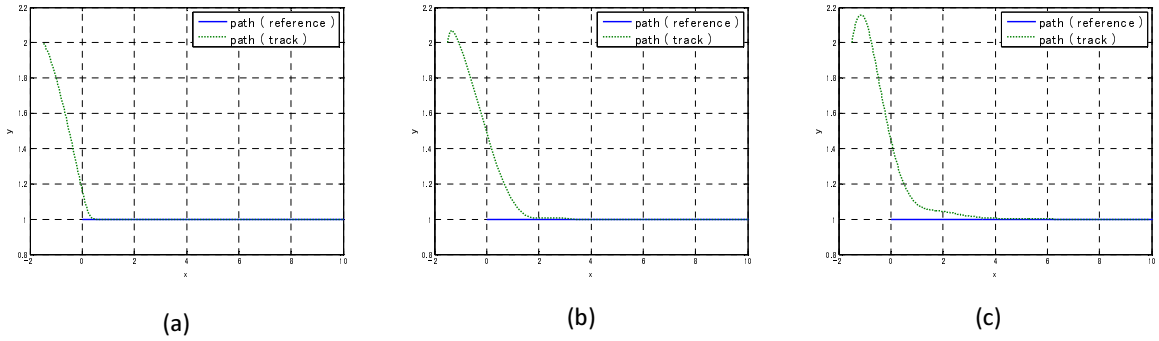

Fig. 2. (a) The tracking performance of the simple straight line for $\varphi_{lim} = \pi/2$; (b) The tracking performance of the simple straight line for $\varphi_{lim} = \pi/3$; (c) The tracking performance of the simple straight line for $\varphi_{lim} = \pi/6$.

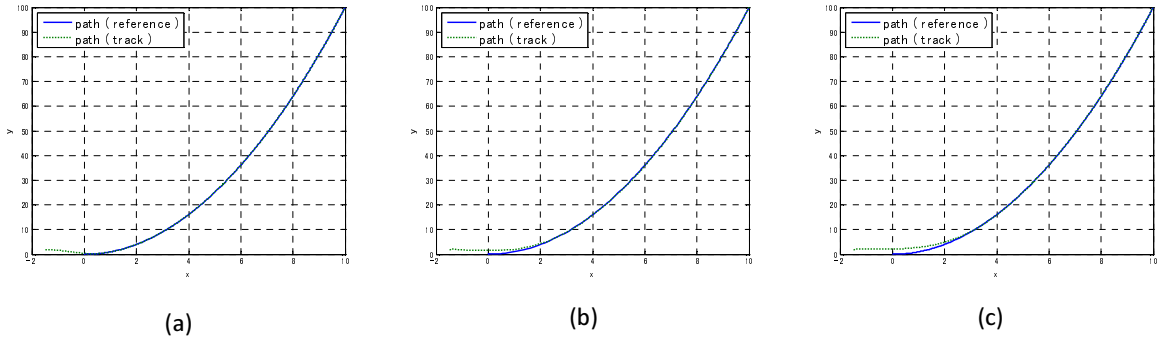

Fig. 3. (a) The tracking performance of the quadratic curve for $\varphi_{lim} = \pi/2$; (b) The tracking performance of the quadratic curve for $\varphi_{lim} = \pi/3$; (c) The tracking performance of the quadratic curve for $\varphi_{lim} = \pi/6$.

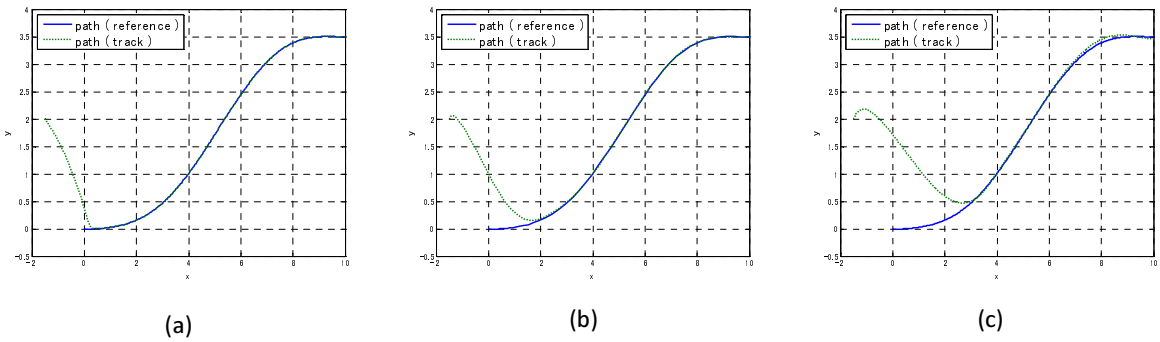

Fig. 4. (a) The tracking performance of the polynomial curve for $\varphi_{lim} = \pi/2$; (b) The tracking performance of the polynomial curve for $\varphi_{lim} = \pi/3$; (c) The tracking performance of the polynomial curve for $\varphi_{lim} = \pi/6$.

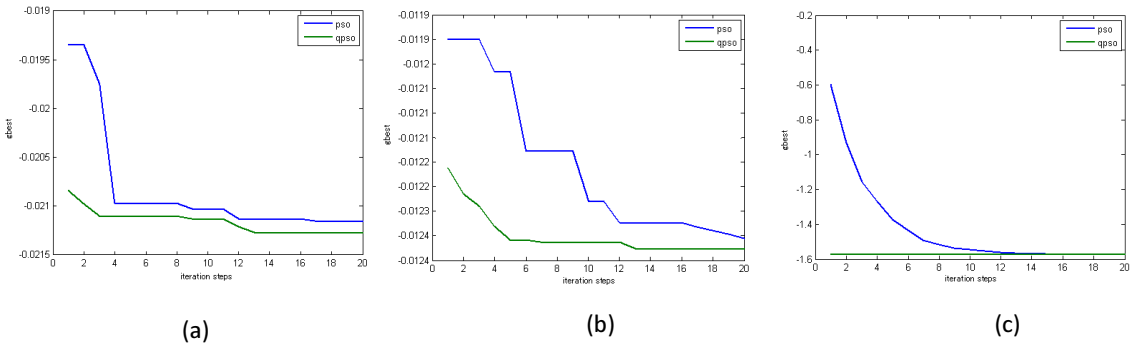

Fig. 5. (a) Convergence times for the simple straight line; (b) Convergence times for the quadratic curve; (c) Convergence times for the polynomial curve.

adjusted much sooner. Hence, we only consider here that the vehicle is with the strictest steering angle limit, i.e. $\pi/6$. Based on the simulation results, we can know that after the vehicle deviates from the reference trajectory due to a disturbance, our algorithm can direct the vehicle back to the reference trajectory efficiently, even with the strictest steering angle limit. On the other hand, the simulation results shown in Figure 7 demonstrate that our system can be applied to vehicles of different sizes. Normally, the distance l between the front and rear axles of most sedans is between $2.4m$ and $3m$. Therefore, beside the original scenario $l = 0.256m$, we further consider two more scenarios: the ten times and the twenty times of the original scenario. Obviously, such a range can cover most real vehicles. Also, we only consider here that the vehicle is with the strictest steering angle limit $\pi/6$. Based on the simulation results, we can see that under these three different values of the distance l between the front and rear axles, our algorithm can direct the vehicle to catch the reference trajectory efficiently, even with the strictest steering angle limit.

V. Conclusions

In the paper, based on the techniques of flatness and particle swarm optimization, we have introduced a steering support controller for the semi-autonomous driver assistance system that can help drivers improve safety and drive comfort. With our controller, the vehicle can catch the designed trajectory quickly, even upon a deviation from the designed trajectory. Furthermore, after the vehicle has caught the designed trajectory, our system will not deviate from it nearly. This means that our controller can guarantee the system stability as well as achieving tracking

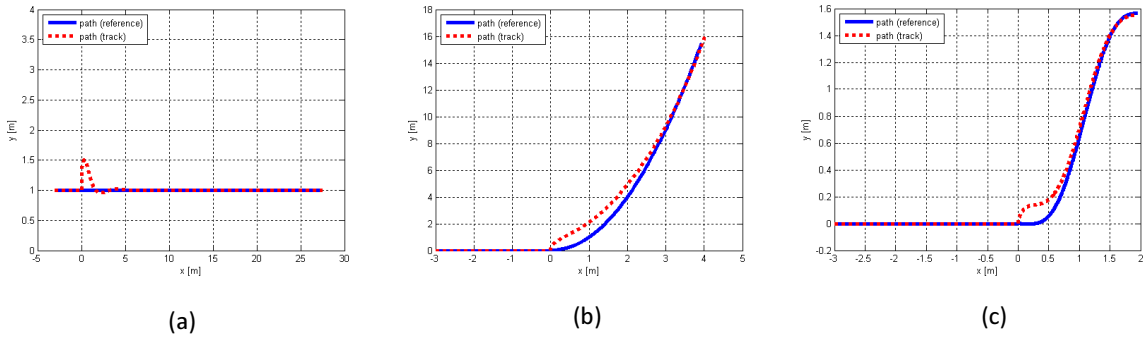

Fig. 6. (a) The tracking performance of the simple straight line with the disturbance for $\varphi_{lim} = \pi/6$; (b) The tracking performance of the quadratic curve with the disturbance for $\varphi_{lim} = \pi/6$; (c) The tracking performance of the polynomial curve with the disturbance for $\varphi_{lim} = \pi/6$.

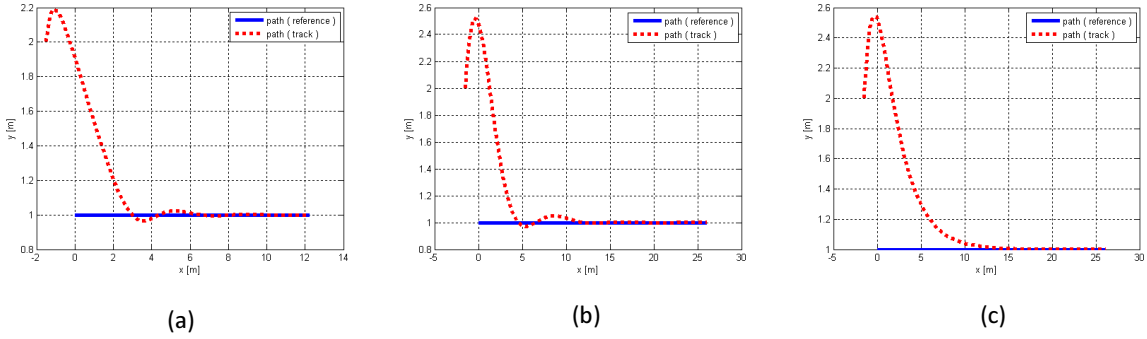

Fig. 7. (a) The tracking performance of the simple straight line for $l = 0.256m$ and $\varphi_{lim} = \pi/6$; (b) The tracking performance of the simple straight line for $l = 2.56m$ and $\varphi_{lim} = \pi/6$; (c) The tracking performance of the simple straight line for $l = 5.12m$ and $\varphi_{lim} = \pi/6$.

control for the system. Particularly, from the simulation experiments, we can see that even with the strictest steering angle limit, the time required by the vehicle to catch the reference trajectory is still small and the tracking trajectory is also smooth. Besides, our system can tolerate finite disturbances very well and can also be applied to vehicles of different sizes. More importantly, according to the simulation results, the QPSO algorithm has better performance than the classical PSO one in finding the optimal feedback gains for the controller. As for the future work, since our current control system can work only when the driver is independent from the system, it is interesting to develop a driver assistance system that can estimate the behavior of the driver to handle conflicts between both sides.

ACKNOWLEDGMENT

The authors wish to express their sincere thanks to the anonymous referees for their valuable comments. Moreover, this work was supported by the National Science Council, Taiwan, ROC, under Grant NSC 101-2221-E-005-034.

REFERENCES

[1] Inagaki, T.: 'Driver support systems and sensing technology', *Journal of the Society of Automotive Engineers of Japan*, 2007, **61**, (2), pp. 16-21.

[2] Huang, R.-W., Lai, C.-H.: 'Development of fuzzy-based automatic vehicle control system in a virtual reality environment'. International Conference on Emerging Technologies, Islamabad, Nov., 2007, pp. 184-189.

[3] Naranjo, J. E., Gonzalez, C., Garcia, R., Pedro, T.: 'Using fuzzy logic in automated vehicle control', *IEEE Intelligent Systems*, 2007, **22**, 1, pp. 36-45.

[4] Wu, C.-Y., Huang, S.-H., Peng, P.-L., Jong, G.-J.: 'Automatic vehicle parking system using gyroscope'. Proceedings of the 3rd International Conference on Ubiquitous Information Management and Communication, 2009, pp. 168-171.

[5] Kiss, B., Szadeczky-Kardoss, E.: 'Design of a semi-autonomous parking assist system'. European Control Conference. 2009, pp. 4949-4954.

[6] Kiss, B., Szadeczky-Kardoss, E.: 'Tracking control of the orbitally flat kinematic car with a new time-scaling input'. IEEE Conference on Decision and Control, New Orleans, L.A., Dec., 2007, pp. 1969-1974.

[7] Nikiforos, S.: 'Is ITS ready for the older driver?'. IEEE Proceedings of the Intelligent Transportation Systems, Oakland, Aug., 2001, pp. 315-318.

[8] Xu, L., Zhao, Q., Ma, C., Lu, F.: 'Systematic methodology of multi-source information fusion for improving older drivers' safety'. IEEE Intelligent Vehicles Symposium, Xi'an, Jun., 2009, pp. 1009-1014.

[9] Son, J.-W., Suh, J.: 'Design considerations for the older population: A human-vehicle interface perspective'. International Symposium on Humanities, Science and Engineering Research, Kuala Lumpur, Jun., 2011, pp. 8 - 12.

[10] Reid R., Mears, B.: 'Design of the steering controller of a supertanker using linear quadratic control theory: a feasibility study', *IEEE Trans. Autom. Control*, 1982, **27**, (4), pp. 940-942.

[11] Isidori, A.: 'Nonlinear control systems' (Springer, Berlin, 1995).

[12] Fliess, M.: 'Generalized controller canonical forms for linear and nonlinear dynamics', *IEEE Trans. Autom. Control*, 1990, **35**, (9), pp. 994-1001.

[13] Fliess, M., Levine, J., Martin, P., Rouchon, P.: 'A lie-backlund approach to equivalence and flatness of nonlinear systems', *IEEE Trans. Autom. Control*, 1999, **44**, (5), pp. 922-937.

[14] Martin, P., Murray, R.M., Rouchon, P.: 'Flat systems, equivalence and trajectory generation'. Technical report, Ecole des Mines de Paris, France, 2003.

[15] Cuesta F., Ollero A.: 'Intelligent Mobile Robot Navigation' (Springer, Heidelberg, 2005).

[16] Hurwitz, A.: 'On the conditions under which an equation has only roots with negative real parts', *Mathematische Annalen*, 1895, **46**, pp. 273-284.

[17] Routh, E.J.: 'A treatise on the stability of a given state of motion' (Macmillan, London, 1877).

[18] Kennedy, L.: 'The particle swarm: social adaptation of knowledge'. IEEE International Conference on Evolutionary Computation, Indianapolis, IN., Apr., 1997, pp. 303-308.

[19] Kennedy, J.: 'Minds and cultures: particle swarm implications'. Technical report, AAAI Fall Symposium, Menlo park, CA., 1997.

[20] Kennedy, J., Eberhart, R.: 'Particle swarm optimization'. Proc. IEEE Int. Conf. Neural Networks, Perth, WA., Nov., 1995, pp. 1942-1948.

[21] Eberhart, R., Kennedy, J.: 'A new optimizer using particle swarm theory'. Proceedings of the sixth international symposium on micro machine and human science, Nagoya, Oct., 1995, pp. 39-43.

[22] Shi, Y., Eberhart, R.C.: 'Empirical study of particle swarm optimization'. Proc. IEEE Int. Congr. Evolutionary Computation, Washington, DC., Jul., 1999, pp. 101106.

[23] Ratnaweera, A., Halgamuge, S.K., Watson, H.C.: 'Self-organizing hierarchical particle swarm optimizer with time varying acceleration coefficients', *IEEE Trans Evol Comput*, 2004, **8**, (3), pp. 240-255.

[24] Yang, Y., Wang, M., Jiao, L.: 'A quantum particle swarm optimization'. Proc. IEEE Congr. Evol. Comput., Jun., 2004, pp. 320324.

[25] Sedlaczek, K., Eberhard, P.: 'Using augmented lagrangian particle swarm optimization for constrained problems in engineering', *Structural and Multidisciplinary Optimization*, 2006, **32**, (4), pp. 277-286.

[26] Maruta, I., Kim, T.H., Sugie, T.: 'Fixed-structure H_∞ controller synthesis: a meta-heuristic approach using simple constrained particle swarm optimization', *Automatica*, 2009, **45**, (2), pp. 553-559.