Path Following Control of Bike Robot

Ryotaro Miyahara and Masaki Yamakita(✉)

Tokyo Institute of Technology, Meguro, Tokyo 152-8552, Japan
yamakita@ctrl.titech.ac.jp
http://www.ac.ctrl.titech.ac.jp/

Abstract. In this paper we consider about path following control of a bike robot which have been studied for a long time in our laboratory and propose a method to generate paths to avoid obstacles based on the sensor information. In the proposed path generation algorithm, BiRRT, smoothing method, and RRT* are combined to generate paths on line. The validity of the proposed method is demonstrated by numerical simulations.

Keywords: Bike robot · Path following · Output zeroing control

1 Introduction

In our laboratory, automatic bike robots have been studied for more than a decade [1–12] since such bike robots have a narrow bodies compering to 4 wheel cars and they have higher potential mobility as those in motocross races.

In [1–4,6] basic control model and control algorithm for an automatic bike robot were proposed using output zeroing controller and the validity of the controller were shown by experiments. In [5] an acrobatic motion was realized with 2 D.O.F. balancer, and in [6–8] a new type of flywheel balancer was introduced and the advantages of it were demonstrated. In [9,10] control algorithm and experimental results were summarized, and new type of control algorithms were tested in [11,12].

In this paper we propose to install a laser range finder to a bike robot which have been studied for a long time and a method to generate paths to avoid obstacles based on the sensor information. In the proposed path generation algorithm, BiRRT, smoothing method, and RRT* are combined to generate paths on line. The validity of the proposed method will be demonstrated by numerical simulations.

This paper consists of the following sections. In Sect. 2 the considered bike robot and its mathematical model and control methods are explained. In Sect. 3 path generation algorithm is proposed and the validity of the method is demonstrated in Sect. 4 followed by concluding remarks in Sect. 5.

V.H. Duy et al. (eds.), *AETA 2017 - Recent Advances in Electrical Engineering and Related Sciences: Theory and Application*, Lecture Notes in Electrical Engineering 465,
https://doi.org/10.1007/978-3-319-69814-4_57

2 Bicycle Robot

In Fig. 1 the considered bike robot in this research is shown. It consists of an electric bicycle, flywheel balancer, gyro sensor, laser range finder, and control unit. It has four motors for rotating a flywheel balancer, shifting CG of the balancer, steering and driving a rear tire. The flywheel balancer can be configured as a flywheel or an inverted pendulum by changing the position of CG as in Fig. 2.

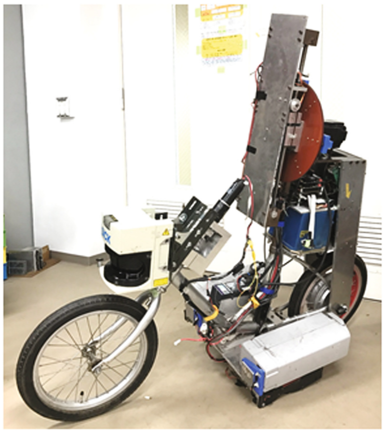

Fig. 1. Unmanned bicycle robot with variable configured balancer.

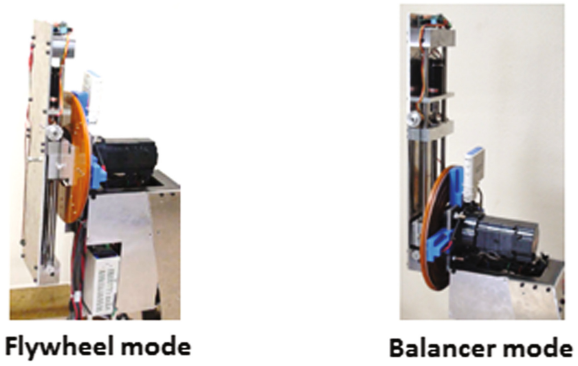

Fig. 2. Flywheel mode and balancer mode

For the balancing control, the system is considered as a two-link system where a first link consists of bike body, rear tire, steering, and front tire, and the second link is the flywheel balancer. The schematic mathematical model of the system is shown in Fig. 3. For the model, we use an input-output linearization with 1 d.o.f. zero dynamics using a special output function is used. For the details, please refer to [2,9].

To control the system and conduct numerical simulations, we construct a 3D model as shown in Fig. 4. Since a precise model becomes very complex, the model is simplified by introducing several assumptions. Using the simplified model, we constructed a path following controller using PVFC. Please refer the details of PVFC and the modifications to [12]. Please notice that if the desired trajectory is defined using a scalar parameter like a time, we can define a desired velocity field by introducing a virtual time and can realize a path following control using PFVC.

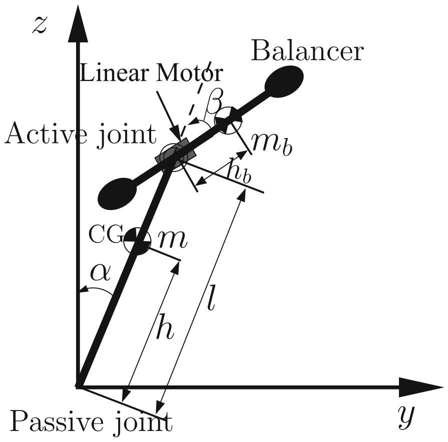

Fig. 3. Two-link system.

3 Path Generation

In this section, we propose a fast method to generate a local path for the bike robot by combining BiRRT (Bidirectional Rapidly exploring Random Trees) method, heuristic smoothing algorithm, RRT* (Rapidly exploring Random Trees Star)[14] with considering the non-holonomic nature of the system.

3.1 Proposed Method

RRT (Rapidly exploring Random Trees)[13] is used to generate paths for systems as mobile robots with non-holonomic constraints in high dimensional spaces. RRT randomly generates branches of a tree to search paths, and it is simple and fast and does not give a local minimum for a cost function. RRT* is a modified version of RRT in which branches are re-connected so that the cost function becomes smaller. BiRRT is a fast version of RRT in which paths are generated

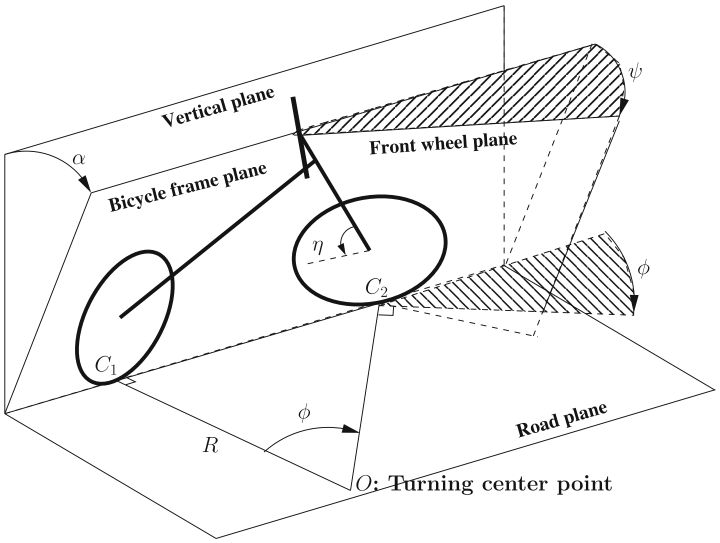

Fig. 4. Coordinate system of the bike with steering angle in a plane.

from both starting and goal positions. Palmieri et al. proposed a heuristic method in which a candidate path is modified by RRT [15]. In the proposed method their method is combined with a method in which a path is represented by an intermediate variable and is modified to minimize a cost function of the length and the curvature to generate paths for the bike robot.

3.2 Algorithm

First, by BiRRT a path P_{BiRRT} which connects start point x_{init} and goal point x_{goal} is generated. In this phase the non-holonomic constraint of the system is not considered. Second, two points on the path P_{BiRRT} is randomly selected and connected by a straight line. If the new path does not collide with an obstacle on a field, the path is remained as a new smoothed path. Finally, within a distance of w from the new path it is modified by considering the non-holonomic constraint by RRT*. The algorithm is shown in Algorithm 1.

In 1st line `BiRRT`$(x_{init}, x_{goal}, \mathcal{X})$ generates a path P_{BiRRT} by BiRRT where x_{init} is an initial position, x_{goal} is a goal position, $\mathcal{X}$ is a search area, $\mathcal{X}_{obst}$ represents obstacles where $\mathcal{X}_{obst} \in \mathcal{X}$. In 5th line `Smooth`$(P_{BiRRT})$ generates a smoothed version of P_{BiRRT} as P_{smooth}. In the process it is repeated that two points on P_{BiRRT} are randomly generated and if there is no obstacle on the line connecting the two points, the line is becomes a part of the path. In 6th line, x_{init} is added to a node set X and initialize edge set E. From 7th line a searching loop is initiated (Fig. 5).

In 8th line `PathWidthSampling`$(P_{smooth}, \mathcal{X})$ randomly samples a candidate in a near neighbor of P_{smooth} within a distance w where the candidate consists

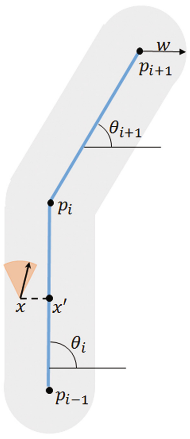

Fig. 5. Path based sampling

of a position of the robot (x_x, x_y) and a moving direction x_θ. Let assume that θ_i is a moving direction at x' which is a nearest point on the path P_{smooth} from the sample point x. x_θ is sampled randomly within between $\theta_i - \delta\theta$ and $\theta_i + \delta\theta$ where $\delta\theta$ is a bias angle and if the distance between a reference point P_i and x' is less or equal to w, θ is calculated using the distance d_i between P_i and x' and as

$$\theta = \frac{d_i}{w}\theta_i + \frac{1-d_i}{2w}(\theta_i + \theta_{i+1}). \tag{1}$$

Similarly, if a distance between a reference point P_{i-1} and x' is less or equal to w, θ is calculated using the distance d_{i-1} P_{i-1} and x' and as

$$\theta = \frac{d_{i-1}}{w}\theta_i + \frac{1-d_{i-1}}{2w}(\theta_i + \theta_{i-1}). \tag{2}$$

In 9th line $\texttt{Nearest}(X, x_{rand})$ picks up the nearest node $x_{nearest}$ in a node set x_{rand}. In 10th line if a distance between x_{rand} and $x_{nearest}$ is larger than e, $\texttt{Steer}(x_{rand}, x_{nearest})$ define a point x_{new} which is located on a line connecting x_{rand} and $x_{nearest}$ and whose distance from $x_{nearest}$ is e. If the distance between x_{rand} and $x_{nearest}$ is less or equal to e, x_{rand} is defined as x_{new}.

In 11th line $\texttt{SplineCurve}(x_{new}, x_{nearest})$ generates a spline curve connecting x_{new} and $x_{nearest}$. Since the desired velocity field for PVFC should be defined in term of an intermediate parameter τ, Ferguson-Coons curve is used. When the initial position is x_0, initial velocity is v_0, terminal position is x_1, and terminal velocity is v_1, the function can be represented using an intermediate parameter T as

$$P(T) = \begin{bmatrix} T^3 \; T^2 \; T \; 1 \end{bmatrix} \begin{bmatrix} 2 & -2 & 1 & 1 \\ -3 & 3 & -2 & -1 \\ 0 & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x_0 \\ x_1 \\ v_0 \\ v_1 \end{bmatrix} \tag{3}$$

$$v = k \left[\cos(x_\theta) \sin(x_\theta) \right] \tag{4}$$

where T is within $[0, 1]$ and the maximum velocity is k, and *path* is a set of points on the curve, σ_{max} is a maximum curvature, and c is a cost for the curve connecting x_{new} and $x_{nearest}$. The cost is a weighted sum of the curve length and an integration of square of change of curvature along the curve and is represented by

$$c = w_p L + w_\sigma \int_0^1 \left| \frac{1}{L} \frac{d\sigma}{ds} \right|^2 \frac{ds}{dT} dT, \tag{5}$$

where s is a segment length and L is the length of the curve, and

$$s(T) = \int_0^T \left\| \frac{dP}{dT} \right\| dT \tag{6}$$

$$L = s(1), \tag{7}$$

and w_p and w_σ are weights and $\sigma(s)$ represents a curvature at s.

In 12th line if the curvature is larger than a critical curvature which the robot can track, the following loop is started. In 15th line `Near`(X, x_{new}) defines a set X_{near} whose distances from x_{new} are less than r. During 15th line and 33th line new connections to points X_{near} are searched so that the new connections gives lower cost, which is equivalent to RRT* where `Cost`(x) is a total cost from x_{init} to x and `Parent`(x) returns a parent node of x.

4 Numerical Simulation

4.1 Path Generation

In this section the proposed path generation algorithm in the previous section is applied for a test field and the validity of it is verified.

Parameters in the algorithm are shown in Table 1.

The considered field is represented as a square area defined by 100×100 grids as shown in Fig. 6, and the yellow color area corresponds to obstacles. The numerical simulation was conducted by a personal computer with Intel Pentium B940 (2.00 GHz) CPU and the software was written by MATLAB.

In the table Roughness means a nature of jaggy of the path which is calculated as the integration of the change of curvature and the path is better if the value is smaller. When a random process is used in an algorithm, the value is an average one over 10 executions (Table 2).

From the results, it can be seen that the proposed method gave the shortest and the second smoothest path next to the one by MPC. The computational time is 1.41 [s] which is quick enough for online generation of the path though the computation of MPC is quite long (188 [s]).

Algorithm 1. BiRRT + Smooth + RRT*

```
1: $P_{BiRRT} \Leftarrow \texttt{BiRRT}(x_{init}, x_{goal}, \mathcal{X})$
2: **if** $P_{BiRRT} = \emptyset$ **then**
3:     **return** *failure*
4: **end if**
5: $P_{smooth} \Leftarrow \texttt{Smooth}(P_{BiRRT})$
6: $X \Leftarrow \{x_{init}\}; E \Leftarrow \emptyset$
7: **for** $i = 1$ to $n$ **do**
8:     $x_{rand} \Leftarrow \texttt{PathWidthSampling}(P_{smooth}, \mathcal{X})$
9:     $x_{nearest} \Leftarrow \texttt{Nearest}(X, x_{rand})$
10:    $x_{new} \Leftarrow \texttt{Steer}(x_{rand}, x_{nearest})$
11:    $path, \sigma_{max}, c \Leftarrow \texttt{SplineCurve}(x_{new}, x_{nearest})$
12:    **if** $path \in \mathcal{X}_{obst} \vee \sigma_{max} > \texttt{MAX\_CURVATURE}$ **then**
13:        continue
14:    **end if**
15:    $X_{near} \Leftarrow \texttt{Near}(X, x_{new})$
16:    $X \Leftarrow X \cup \{x_{new}\}$
17:    $x_{min} \Leftarrow x_{nearest}; c_{min} \Leftarrow \texttt{Cost}(x_{nearest}) + c$
18:    **for all** $x_{near} \in X_{near}$ **do**
19:        $path, \sigma_{max}, c \Leftarrow \texttt{SplineCurve}(x_{new}, x_{near})$
20:        **if** $path \in \mathcal{X}_{obst} \vee \sigma_{max} > \texttt{MAX\_CURVATURE} \vee c_{min} \leq \texttt{Cost}(x_{near}) + c$ **then**
21:            continue
22:        **end if**
23:        $x_{min} \Leftarrow x_{near}; c_{min} \Leftarrow \texttt{Cost}(x_{near}) + c$
24:    **end for**
25:    $E \Leftarrow E \cup \{(x_{min}, x_{new})\}$
26:    **for all** $x_{near} \in X_{near}$ **do**
27:        $path, \sigma_{max}, c \Leftarrow \texttt{SplineCurve}(x_{new}, x_{near})$
28:        **if** $path \in \mathcal{X}_{obst} \vee \sigma_{max} > \texttt{MAX\_CURVATURE} \vee \texttt{Cost}(x_{near}) \leq \texttt{Cost}(x_{new}) + c$
           **then**
29:            continue
30:        **end if**
31:        $x_{parent} \Leftarrow \texttt{Parent}(x_{near})$
32:        $E \Leftarrow (E \setminus \{(x_{parent}, x_{near})\}) \cup \{(x_{new}, x_{near})\}$
33:    **end for**
34:    **if** $x_{new} \in \mathcal{X}_{goal}$ **then**
35:        **return** Path(X,E)
36:    **end if**
37: **end for**
38: **return** *failure*
```

4.2 Path Following

The generated path is applied for PVFC and the performance is evaluated where the parameters of PVFC are same as in [12]. The initial configuration is (1 [m], 1 [m], 0 [deg]) and the desired configuration is set (8 [m], 8 [m], 90 [deg]) and obstacles are located as in Fig. 7. The generated path and trajectory of the

Table 1. Parameters of path generation

Parameter	Symbol	Value
Max. distance between nodes	e	5
Max. repeat number for smoothing	i_{smooth}	200
Search distance from path P	w	2
Angle bias	$\delta\theta$	22.5 [deg]
Max. velocity on curve	k	1
Max. tracking curvature	σ_{MAX}	1
Reach tolerance	l	3
Weight for path length	w_p	1
Weight for curvature	w_σ	1

Table 2. Path planning result

Limit of curvature	Algorithm	Planning time [s]	Path length [m]	Roughness
Not apply	LP	16.0	198.1	7.61e − 04
	RRT	0.651	270.2	3.40e − 02
	RRT*	1.30	263.6	1.02e − 03
	BiRRT	0.304	261.2	1.45e − 03
	BiRRT + Smooth	0.372	178.7	3.70e + 00
Apply	MPC + LP	2.70e + 3	188.2	6.08e − 07
	BiRRT + Smooth + RRT*	1.41	177.2	5.61e − 05

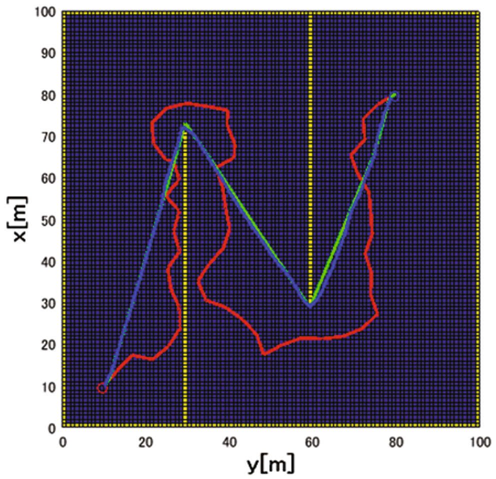

Fig. 6. Proposed path (red: RRT, green: RRT + Smooth, blue: RRT + Smooth + RRT*)

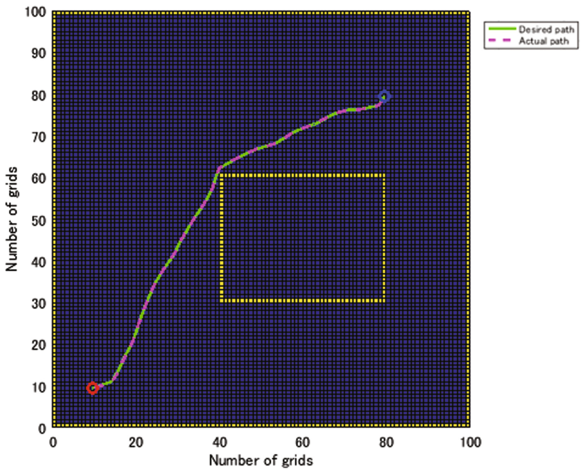

Fig. 7. Desired path and actual path

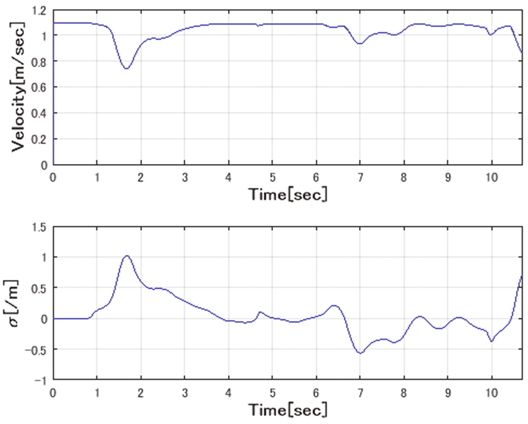

Fig. 8. Velocity and curvature

reference point are shown in Fig. 7, and the history of velocity changes of the reference point is shown in Fig. 8.

From Fig. 7 it can be seen that the bike robot followed the path, and from Fig. 8 it can be also confirmed that the velocity at corners became slow and side slippages were avoided.

5 Concluding Remarks

In this research we have proposed a path generation algorithm for a bike robot by combining BiRRT, smoothing, and RRT*. By a numerical simulation it was confirmed that practical paths can be generated for the robot so that the robot can track the path and avoid collisions against obstacle online.

Future problem is to apply the proposed method for an experimental robot.

References

1. Yamakita, M., Utano, A.: Automatic control of bicycles with a balancer. In: Proceedings of AIM2005,1245/1250 (2005)
2. Yamakita, M., Utano, A., Sekiguchi, K.: Experimental study of automatic control of bicycle with balancer. In: Proceedings of IROS2006 (2006)
3. Murayama, A., Yamakita, M.: Development of autonomous bike robot with balancer. In: Proceedings of SICE2007 (2007)
4. Keo, L., Yamakita, M.: Trajectory control for an autonomous bicycle with balancer. In: Proceedings of AIM2008 (2008)
5. Okawa, S., Keo, L., Yamakita, M.: Realization of acrobatic turn via wheelie for a bicycle with a balancer. In: Proceedings of ICRA09 (2009)
6. Keo, L., Yamakita, M.: Controller design of an autonomous bicycle with both steering and balancer controls. In: Proceedings of MSC09 (CCA09) (2009)
7. Lychek, K.E.O., Yamakita, M., Ito, K.: Stabilizing of an unmanned bicycle with flywheel balancer. In: Proceedings of NOLCOS 2010 (2010)
8. Kawaguchi, M., Yamakita, M.: Stabilizing of bike robot with variable configured balancer. In: Proceedings of SICE2011 (2011)
9. Lychek, K., Yamakita, M.: Control of an autonomous electric bicycle with both steering and balancer controls. Adv. Robot. **25**(1), 1–22 (2011)
10. Lychek, K.E.O., Yoshono, K., Kawaguchi, M., Yamakita, M.: Experimental results for stabilization of a bicycle with a flywheel balancer. In: Proceedings of ICRA 2011 (2011)
11. Noda, Y., Sumioka, T., Yamakita, M.: An application of fast MPC for bike robot. In: Proceedings of SICE Annual Conference (2012)
12. Yin, S., Yamakita, M.: Passive velocity field control to bicycle robot path following. In: Proceedings of SICE 2016 (2016)
13. LaValle, S.M., Kuffner, J.J.: Randomized kinodynamic planning. Int. J. Robot. Res. **20**, 378–400 (2001)
14. Karaman, S., Frazzoli, E.: Sampling-based algorithms for optimal motion planning. In: Robotics: Science and Systems (RSS) (2010)
15. Palmieri, L., Koenig, S., Arras, K.O.: RRT-based nonholonomic motion planning using any-angle path biasing. In: International Conference on Robotics and Automation (2016)