ICIC Express Letters
Part B: Applications
Volume **14**, Number **12**, December **2023**
 ISSN 2185-2766
pp. **1259–1266**

INDUSTRIAL APPLICATION PLATFORM FOR TWO-WHEELED ROBOTS

Bao Shi[1], Guoliang Zhao[2,*], Sharina Huang[3] and Hongxing Li[1]

[1]School of Control Science and Engineering
Dalian University of Technology
No. 2, Linggong Road, Ganjingzi District, Dalian 116024, P. R. China
dutlion@mail.dlut.edu.cn; lihx@dlut.edu.cn

[2]College of Electronic Information Engineering
Inner Mongolia University
No. 235, West College Road, Saihan District, Hohhot 010021, P. R. China
[*]Corresponding author: guoliangzhao@imu.edu.cn

[3]College of Science
Inner Mongolia Agricultural University
No. 306, Zhaowuda Road, Saihan District, Hohhot 010018, P. R. China
huangsharina@imau.edu.cn

Received April 2023; accepted June 2023

Abstract. *The large-scale design of industrial robots and the innovation of artificial intelligence technology have promoted the development of industrialization and modernization. Theoretical analysis shows that its universality and technological progress effect are conducive to the transformation and upgrading of industrial structure, especially the two-wheeled robot. Two-wheeled robot has the structure of wheeled mobile robot and inverted pendulum, which is an unstable nonlinear system. In industry, it can be used to perform tasks in some special environments. Firstly, in order to improve the adaptability of the controller, this paper takes the variable universe interval type II fuzzy control method as an example to realize the path tracking control of the mobile robot. Secondly, in addition to the design of the controller, the development platform of the two-wheeled robot is also worth studying. Different development platforms have a significant impact on the efficiency and stability of industrial robots. Two application methods of development platform are proposed in this paper, namely Matlab/Simulink platform and OtoStudio platform. Finally, the effectiveness of the controller is verified by simulation, and the conversion method between different platforms is given, which has the advantage of wide application.*

Keywords: Mobile two-wheeled robot, Fuzzy control, OtoStudio platform, Industrial application

1. **Introduction.** In recent years, the application of robots has become more and more extensive. Mobile two-wheeled robot is often used to test the performance of control theory, and because of its own particularity, it has a good application prospect in industry, for example, [1-5]. In [1], the PID algorithm is used to control the speed of the two DC motors to achieve precise control of the speed of the mobile robot. Model predictive controller is designed based on the state space, and the control method is compared with the conventional control method [2]. Wheeled robots exhibit fast and stable motion on smooth roads, but lack the ability to overcome obstacles and rough terrain. A two-wheeled jumping robot combining wheel motion and bouncing motion is proposed to solve this deficiency [3]. A robot with deformable wheels that could steer in hard and soft soils is proposed [4]. The linear velocity of the slave mobile robot follows the position command

DOI: 10.24507/icicelb.14.12.1259

from the haptic master robot, while the slip-induced velocity error acts as a haptic force feedback felt by the human operator [5].

There are also many studies on the path tracking control of robots, and different control methods are continuously proposed. For two-wheeled robots with different configurations, a population control design is proposed [6]. Under the leadership of the leader, the leader performs the task of tracking the trajectory and can avoid obstacles to reach the specified position. A fault-tolerant dynamic control method was developed for accurate trajectory tracking of redundantly actuated mobile robots, which utilizes a two-level structure to cover wheel-ground interactions and possible actuator failures [7]. A control strategy is proposed to tune the robot's power output along the two DOF directions, which can improve the robot's trajectory-following control, especially on rough terrain [8]. Modular deformable wheels capable of overcoming obstacles are proposed for indoor service robots or service platforms that often move back and forth on thresholds or sidewalks [9]. According to the requirements of simultaneous position and force control of ankle rehabilitation robot, a prototype of 3-PRS ankle rehabilitation robot based on impedance control and its force/position control strategy are proposed [10].

As a programming software, OtoStudio is not only advanced in function and structure, but also easy to master and control. It has become a leading programming tool in the automation market. For OtoStudio platform, it is especially that the use of basic C language function library and Windows dynamic link library can well realize complex control functions. Relatively speaking, the platform is simple in design and fast in operation. At the same time, under Windows system, any development tool supporting dynamic link library can be used to develop programs, which is convenient to use.

This paper mainly realized the following aspects:

1) The controller design of two-wheeled robot with path tracking function, namely variable universe interval type II fuzzy logic controller (VUIT2FLC);

2) Path tracking control based on Matlab/Simulink platform;

3) Path tracking control based on OtoStudio platform.

In this paper, the first section introduces the research status at home and abroad; The second section introduces the physical model of the balancing robot; The third section is the design of controller based on Matlab platform; The fourth section is the design of controller based on OtoStudio platform; The fifth section is system simulation, through the form of simulation to verify the performance of the controller; The last is the summary.

2. **Mobile Two-Wheeled Robot Model.** The Lagrangian modeling method is used to complete the system modeling of the mobile two-wheeled robot. The system is used as the controlled object. The quasi-linear parametric variable model of the system can be written as [11]

$$\dot{\mathbf{x}} = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & a_1 & 0 & 0 & a_2 & 0 & b_1 & b_1 \\ 0 & a_3 & 0 & 0 & a_4 & 0 & b_2 & b_2 \\ 0 & 0 & 0 & 0 & 0 & 0 & b_3 & -b_3 \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{u} \end{bmatrix} \tag{1}$$

where

$$\begin{aligned} a_1 &= -3g \cos x_2 \operatorname{sinc}(x_2/\pi)/(4h_1) \\ a_2 &= Lh_2/h_1 \\ a_3 &= \left(3/(4L) + 9\cos^2 x_2/(16h_1L)\right) g \operatorname{sinc}(x_2/\pi) \\ a_4 &= -3h_2 \cos x_2/(4h_1) \\ b_1 &= (3R\cos x_2 + 4L)/(4LRM_ph_1) \end{aligned}$$

$$
\begin{aligned}
b_2 &= -\left(3h_3 \cos x_2/(4h_1 L) + 3/\left(4M_p L^2\right)\right) \\
b_3 &= 6/((M_p + 9M_r)DR) \\
h_1 &= 1 + 3M_r/M_p - 3\cos^2 x_2/4 \\
h_2 &= x_5 \sin x_2 \\
h_3 &= 1/(M_p R) + 3\cos x_2/(4M_p L)
\end{aligned} \tag{2}
$$

Among them, $\mathbf{x} = [x_1, x_2, x_3, x_4, x_5, x_6]^T = \left[x_p, \theta, \delta, \dot{x}_p, \dot{\theta}, \dot{\delta}\right]^T$, where x_1 represents the displacement state of the robot, x_2 represents the pitching state of the robot, x_3 represents the yaw state of the robot, and x_4, x_5 and x_6 represent the state change rate of the robot, respectively. Various physical parameters can be found in [11].

3. **Matlab Platform Design.** Matlab/Simulink is used to build the simulation platform, and the platform interface of VUIT2FLC's block diagram is shown in Figure 1.

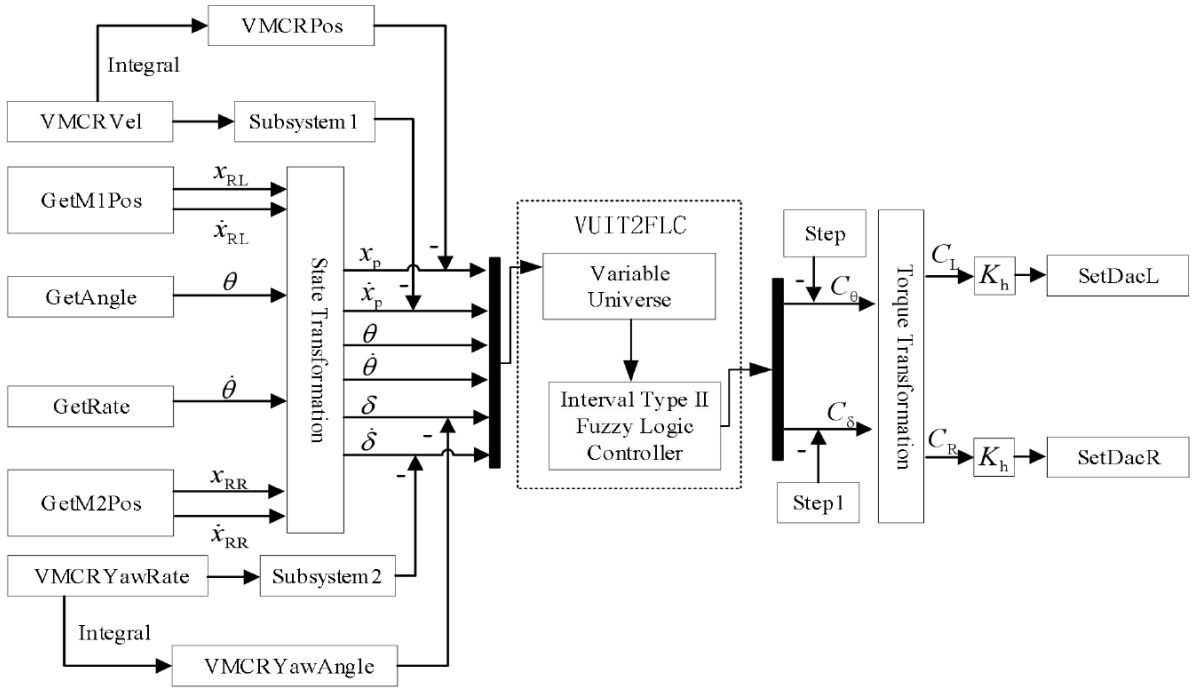

FIGURE 1. Block diagram of VUIT2FLC

Among them, the six states acquired by the hardware are represented through six modules, namely GetM1Pos, GetAngle, GetRate, GetM2Pos, VMCRVel and VMCRYawRate. The two states output to the motor are represented by two modules SetDacL and SetDacR, respectively. GetM1Pos, GetAngle, GetRate, GetM2Pos are used to represent the actual values of the left motor encoder, the angle sensor on the gyroscope and the right motor encoder, respectively, which are transformed into six state quantities by the state conversion module, namely Pos, Vel, PitchAngle, PitchRate, YawAngle, and YawRate.

The state transformation satisfied

$$
\begin{aligned}
&x_p = \frac{x_{RL} + x_{RR}}{2} + \theta L, \quad \dot{x}_p = \frac{\dot{x}_{RL} + \dot{x}_{RR}}{2} + \dot{\theta} L, \quad \theta = \theta, \quad \dot{\theta} = \dot{\theta}, \\
&\delta = \frac{x_{RL} - x_{RR}}{D}, \quad \dot{\delta} = \frac{\dot{x}_{RL} - \dot{x}_{RR}}{D}
\end{aligned} \tag{3}
$$

Before the six state variables enter the controller through feedback, the speed offset of the forward channel input of the remote control is superimposed on Pos and Vel. The

voltage VMCRVel is transformed into the forward speed of the robot by subsystem 1 and superimposed on the Vel signal. The speed is integrated and superimposed on the Pos signal. In the same way, the yaw angular velocity offset of yaw channel input of the remote control is superimposed on YawAngle and YawRate. The voltage VMCRYawRate is transformed into the yaw angular velocity of the robot by subsystem 2 and superimposed on YawRate signal. The voltage VMCRYawAngle is integrated and superimposed on the YawAngle signal. Among them, the subsystem 1 and the subsystem 2 only have different product factors, and they are all composed of different sub-modules, that is, blind zone, saturation and speed limiter modules are included. The parameters are set as blind zone: $[-2, 2]$; saturation zone interval: $[-5, 5]$; speed limiter: $[-0.1, 0.1]$.

The six state variables of the superimposed remote control signal are calculated by the controller VUIT2FLC to obtain the pitch and yaw torque C_θ and C_δ, which is subtracted from the step input of the system pitch channel and the step input of the yaw channel to obtain the error signal of the C_θ and C_δ. Among them, C_θ via hysteresis module Backlash, its main purpose is to protect the motor from frequent switching directions and damage the DC brushed motor. The error signal is transformed into the torque C_θ and C_δ of the left and right motors through torque transformation, which meets the relationship: $C_\theta + C_\delta = C_L$, $C_\theta - C_\delta = C_R$. C_L and C_R enter the safety protection switches, respectively. The pitch angle range of the two-wheeled robot is limited. In this design, $\pm\pi/6$ is selected. With the help of the safety switch, the control is started within the angle range, and the left and right motor torque output outside the angle range is 0. The transformation between the input voltage and torque of the left and right wheel motors meets the K_h times relationship, that is

$$K_h = -\frac{U_m}{I_m \times \tau \times \gamma} \tag{4}$$

Among them, the maximum voltage U_m is set to 10 V, the maximum current corresponding to the maximum voltage I_m is set to 6 A, the motor torque coefficient τ is set to 0.1, the reduction ratio of the reducer γ is set to 8, and a saturation module is added with a range of ± 10 V.

After the Matlab/Simulink simulation platform was built, the "Level-2 MATLAB S-Function" module was used to edit the program of the controller. The Matlab help file can be referenced for the parameter definition in the specific module. The edited controller program is translated into C code by Matlab "Coder". The reason is that the C language runs fast, so that the response speed of the controller is increased. The "S-Function Builder" module is used to call the C file. It should be noted that the header files and function files in the C file are totally added to the "S-Function Builder", so that the functions in the C file can be directly used by the "S-Function Builder" module call. The help file of Matlab can be referenced for the specific use of "S-Function Builder" module function.

It should be noted that

1) During transformation, the input type of C language needs to be defined, while the block in "Level-2 MATLAB S-Function" belongs to the structure, and the C form language cannot be directly constructed. However, the input of the output subroutine of "Level-2 MATLAB S-Function" can be directly edited as a C function, and the input type can be defined as double (6×1);

2) In Matlab, load statement functionality is used to load "mat" type data, but the load statement cannot be used during tensor product model transformation stage. Since "Load" function is a unique function of Matlab, then the "coder.load" statement is used to load the "mat" data set when the translation program is initialized, and the resulted statement is assigned to a new variable for subsequent calls;

3) If the function of the external m file is called in the "Level-2 MATLAB S-Function", it is necessary to be ensured that there is no common variable name among the functions,

because the global variable is defined when the C language is initialized. If the variable name used in the function is the same as the other function used, and the type is predefined different, then the transformation will fail to finish the job;

4) The transformed C language is divided into 32-bit and 64-bit, and the number of bits of the controllers is selected according to the need. The C code can also be obtained through the following methods.

Method 1: The dynamic link library of the generated C file is generated by Microsoft Visual Studio 2008 version. In addition, the functions encapsulated in the C file can be directly called through the dynamic link library added to the "S-Function Builder" module.

Method 2: It is compiled into a C-Mex file and the controller algorithm is invoked in Simulink.

Method 3: The C file or m file is called by "S-Function" module, but they both need to cooperate with the TLC package program; otherwise the real-time object cannot be found during hardware compilation.

4. **OtoStudio Platform Design.** As shown in Figure 2, the OtoStudio platform construction is divided into four steps:

Step 1: Compile the Simulink instruction program of Matlab into a C file;

Step 2: Establish an external library in OtoStudio;

Step 3: Establish a dynamic link library project in Microsoft Visual Studio to generate a dynamic link library;

Step 4: Join the GTC controller.

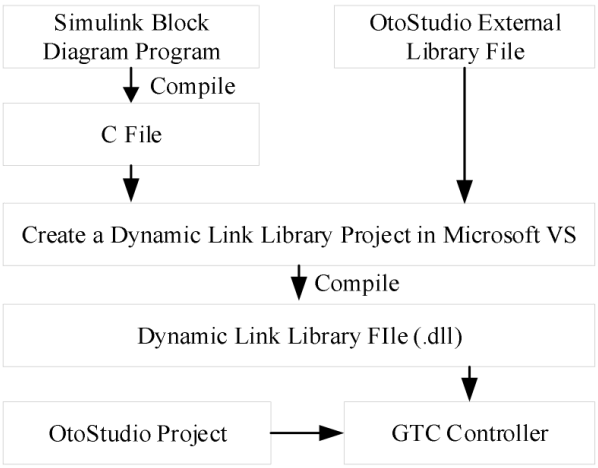

FIGURE 2. Simulink simulation algorithm transfer to OtoStudio external library block diagram

i) Convert C file

Parameters are configured in Matlab/Simulink according to Table 1.

After the parameter setting is completed, it is necessary to check the accuracy of the model parameter setting. After clicking "Modify Parameters" on the "Code generation Advisor" interface, the parameters to be automatically updated and set. Then click "Run This Check" to make sure the result is "Passed".

Notation:

1) Before detecting the model, you need to ensure that the current path of Matlab is consistent with the storage path of the Simulink model, otherwise an error will occur and the model cannot be found. At the same time, the parameter settings cannot be automatically updated.

Table 1. Parameter setting

Attribute	Parameter name	Parameter value
Solver	Type	Fixed-step
	Size	0.006
	Solver	Ode1(Euler)
	Stop time	10
Code Generation	System target le	ert.tlc
	Language	C
	Generate code only	√
	Prioritied objectives	Traceability
Code Generation=Report	Create code generation report	√
Code Generation=Interface	continuous time	√
	non-inlined S-functions	√

2) Because the C file obtained after compilation is a cyclically executed file, the value of the parameter "Stop time" only needs to be set larger than the value of the parameter "Fixed-step size". In this paper, this parameter is set to 10.

3) Since the robot system is a real-time tracking system, the parameter "Traceability" must be selected, and other targets can be selected as needed. In the "Code generation" interface, the program generated by "Generate Code", the generated file can be seen in the current path of Matlab, and the new file path can also be viewed in the generated report. The generated C file mainly includes four main functions, namely main function, initialization function, running function and termination function. The function of each function in the model and the meaning of each parameter can be viewed in the report generated by the program. At the same time, the input and output of the model are redefined in the C file to provide an interface for the later use of the program.

ii) Build external library

In OtoStudio, a project is created. The target system configuration is selected as CPAC-GUC-X00-TPX, and the input parameters are defined in VAR_INPUT. The input includes two parts: one is the six system input values; the other is the parameters in the Simulink block diagram algorithm that needs to be modified in real time in the OtoStudio project. The output parameters are defined in VAR_OUTPUT, and the output quantity also contains two parts: one is the two system output quantities; the other is the state quantity that needs to be monitored.

Three files are generated. "libGbot.c" contains two functions, namely the initialization function "GBOTinit" and the running function "GBOT". The implementation of the two functions is not defined. In the dynamic link library project of Microsoft Visual Studio, the function "slxGbot_initialize()" and the function "slxGbot_step()" in the C file converted by the Simulink block diagram algorithm can be called, respectively.

iii) Generate dynamic link library

In Microsoft Visual Studio, a new Win32 smart device project is created, the GTx86 platform is selected, and the C file generated in the first section and the external library file generated by OtoStudio are put into the project. Since Simulink block diagram algorithm converts C files, it includes some built-in functions of Matlab, and the folder under Matlab needs to be attached to the directory of Microsoft Visual Studio when converting. In order to test whether the converted C file is wrong, it is set to not use the precompiled header, and the solution is regenerated. The program is error-free and the rebuild is successful. It proves that the C program converted from the Simulink block diagram algorithm has been compiled successfully.

Next, OtoStudio external library program needs to be tested for errors. For OtoStudio external library program, generated by ST program, its data type definition is different

from that of C language data type, and it needs to be modified accordingly. After editing the program, the interface of dynamic link library is defined. In "dllGbot.cpp", the path "#include 'libGbot/libGbot.h'" is added, and OtoStudio's "libGbot.h" is added. In "dllGbot.h", the declaration of the defined GetFunctionByName function was added. Similarly, the solution needs to be regenerated, the program is error-free, and the regeneration is successful.

Finally, the dynamic link library "dllGbot.dll" generated in Microsoft Visual Studio is copied to the GTC controller "/HardDisk/CPAC" folder, which replaces the control program in the original controller.

5. **Simulation.** In this paper, the variable universe interval type II fuzzy controller is used to complete the displacement tracking control of the two-wheeled robot. The initial state is given as $x_0(t) = \left[0, \frac{\pi}{6}, 0, 0, 0, 0\right]^T$, the expected tracking displacement is $x_r = 1$ m and the yaw angle is $\delta_r = 0.8$ rad.

The simulation results are shown in Figures 3 and 4. Among them, state x_1 represents the displacement of the robot, state x_2 represents the pitch angle of the robot, state x_3 represents the yaw angle of the robot, and states x_4, x_5 and x_6 represent the change rate of each state of the robot, respectively. The initial pitch angle of the given robot is $\pi/6$, the expected displacement is 1 m, and the expected yaw angle is 0.8 rad. In order to ensure the stability of the robot, with the change of pitch angle, the robot is gradually

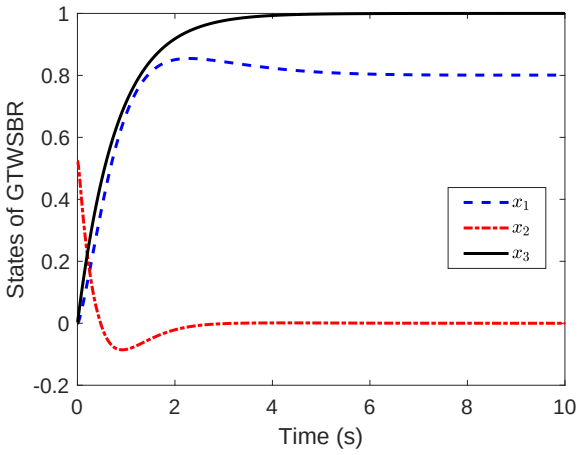

FIGURE 3. State curve of robot

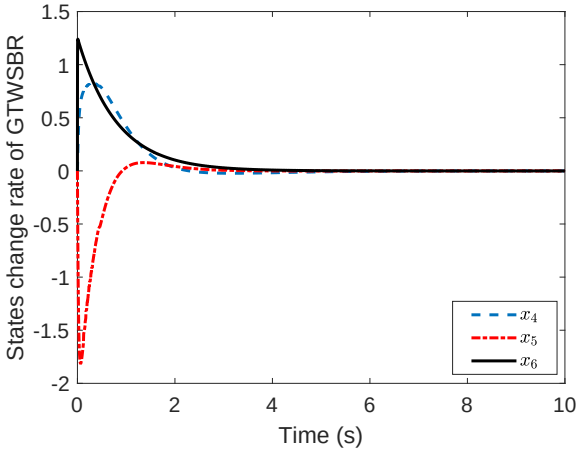

FIGURE 4. State change rate curve of robot

producing displacement. The displacement tracking controller designed in this paper is expected to control the balance robot to reach the specified position and still maintain the upright state. According to the simulation curve, the robot maintains a steady state while reaching the specified position 1 m and the yaw state 0.8 rad, that is, the horizontal rotation angle is 45.84°. At this time, the pitch angle of the robot is 0, indicating that the robot is in an upright state. To sum up, the two-wheeled robot realized the control of displacement and yaw state while stabilizing, and the balanced robot is well controlled, which proves the effectiveness of the controller.

6. **Conclusions.** With the popularity of mobile two-wheeled robots, people began to study different design platforms. In this paper, the balanced robot is successfully controlled by variable universe interval type II fuzzy controller. In order to further promote the application of balanced robot in industry, the design method of Matlab and OtoStudio platform is introduced in detail. First, the initialization module of the programming part is built through Simulink. Next, the transformation of C language and the generation of dynamic link library are completed. Finally, the program was successfully ported to OtoStudio platform. In the later research process, the functions of vision and obstacle avoidance can be added to the robot, so as to enhance the practical application value of the system.

Acknowledgment. This work is partially supported by Natural Science Foundation of Inner Mongolia (2019MS01005, 2020MS06016), National Natural Science Foundation of China under number 61603126, and the work is also supported by the Research Foundation for Advanced Talents Research Foundation of Inner Mongolia University (21700-5185130). The authors also gratefully acknowledge the helpful comments and suggestions of the reviewers, which have improved the presentation.

REFERENCES

[1] M. J. Meng, A. Liu, Y. Yang, Z. Wu and Q. Xu, Two-wheeled robot platform based on PID control, *2018 5th International Conference on Information Science and Control Engineering (ICISCE)*, 2018.

[2] S. Khatoon, D. K. Chaturvedi, N. Hasan and M. Istiyaque, Optimal controller design for two wheel mobile robot, *2018 3rd International Innovative Applications of Computational Intelligence on Power, Energy and Controls with Their Impact on Humanity (CIPECH)*, pp.1-5, 2018.

[3] Y. Zhang, L. Zhang, W. Wang, Y. Li and Q. Zhang, Design and implementation of a two-wheel and hopping robot with a linkage mechanism, *IEEE Access*, vol.6, pp.42422-42430, 2018.

[4] J. V. Salazar Luces, S. Matsuzaki and Y. Hirata, RoVaLL: Design and development of a multi-terrain towed robot with variable lug-length wheels, *IEEE Robotics and Automation Letters*, vol.5, no.4, pp.6017-6024, 2020.

[5] W. Li, L. Ding, H. Gao and M. Tavakoli, Haptic tele-driving of wheeled mobile robots under nonideal wheel rolling, kinematic control and communication time delay, *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol.50, no.1, pp.336-347, 2020.

[6] J. L. Mata Machuca, L. F. Zarazua and R. Aguilar-López, Experimental verification of the leader-follower formation control of two wheeled mobile robots with obstacle avoidance, *IEEE Latin America Transactions*, vol.19, no.8, pp.1417-1424, 2021.

[7] X. Zhang, Y. Xie, L. Jiang, G. Li, J. Meng and Y. Huang, Fault-tolerant dynamic control of a four-wheel redundantly-actuated mobile robot, *IEEE Access*, vol.7, pp.157909-157921, 2019.

[8] H. Qi et al., Control strategy for the pseudo-driven wheels of multi-wheeled mobile robots based on dissociation by degrees-of-freedom, *IEEE Access*, vol.8, pp.155477-155491, 2020.

[9] Y. Lee, S. Ryu, J. H. Won, S. Kim, H. S. Kim and T. Seo, Modular two-degree-of-freedom transformable wheels capable of overcoming obstacle, *IEEE Robotics and Automation Letters*, vol.7, no.2, pp.914-920, 2022.

[10] G. Chen, H. Zhou and P. Yang, Force/position control strategy of 3-PRS ankle rehabilitation robot, *International Journal of Innovative Computing, Information and Control*, vol.16, no.2, pp.481-494, 2020.

[11] B. Shi, S. Xu and G. Zhao, Variable universe type-II fuzzy logic control design for the Googol's two-wheeled self-balancing robot, *2020 Chinese Automation Congress (CAC)*, pp.1500-1505, 2020.