TELKOMNIKA Telecommunication, Computing, Electronics and Control
Vol. 18, No. 6, December 2020, pp. 3088~3095
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v18i6.16431

Design of a controller for wheeled mobile robots based on automatic movement sequencing

Holman Montiel Ariza, Fredy H. Martínez S., Fernando Martínez S.
Facultad Tecnológica, Universidad Distrital Francisco José de Caldas, Colombia

Article Info

Article history:

Received Apr 19, 2020
Revised Jul 06,2020
Accepted Jul 26, 2020

Keywords:

Controller
Genetic algorithm
Optimization algorithm
Sensor
Wheeled mobile robots

ABSTRACT

There are many kinds of robots and among them the wheeled mobile robots (WMR) stand out, because they are relatively cheap and easy to build. These features make WMRs the test prototypes for control strategies or motion generation. In general, the controllers developed are based on sensory schemes that give an WMR the ability to travel through flat or obstructed environments. However, these strategies are highly reactive, i.e. they are based on the control-action scheme and are not adaptive; or, they are motion schemes built from simulations that assume the environmental conditions to determine the robot's path. In both cases, WMRs do not adapt perfectly to the change of environment, since the controller does not find appropriate movements for the robot to move from one point to another. Therefore, this article proposesa partial solution to this problem, with a controller that generates sets of adaptive movements for an WMR to travel around its environment from the sensory perception information.

Corresponding Author:

Holman Montiel Ariza
Facultad Tecnológica,
Universidad Distrital Francisco José de Caldas,
Cll 68 D Bis A Sur No. 49F–70, Bogotá D.C, Colombia.
Email: hmontiela@udistrital.edu.co

1. INTRODUCTION

A robot is basically a system that performs one or more tasks automatically, which increases the level of precision and accuracy that a human being achieves when performing high-risk, heavy or repetitive activities. For example, virtual assistants that help online users or robotic manipulators that perform welding tasks in automotive manufacturing lines [1-4]. As can be seen, the robot definition is not only applied to designate prototypes with mechanical and electrical parts, but also to designate virtual prototypes that have a digital representation or are available in computer systems. Therefore, there is many robots and ways to classify them, such as mobile robots that are transport machines with a certain degree of autonomy and locomotion to travel around their environment [3, 5, 6].

Wheeled mobile robots (WMR) are a case of a mobile robot and have advantages such as; the ease of physical implementation or the ability to travel long distances in a short time. The ease of implementation has increased the mass of this type of robot and the frequency in industrial or research applications (e.g. the "Mars Exploration Rover" used in space missions) [6-8]. Another feature of WMRs is that they incorporate groups of instruments, sensors or accessories to interact with their environment, which attempts to reduce or eliminate human intervention and increase the level of autonomy during the execution of certain tasks. Although the autonomy level of an WMR is high enough to solve a problem, there are electrical (energy supply),

mechanical (repair or replacement of parts), computer (generation of control instructions) or electromechanical (integration of components) limitations that do not allow it to achieve this capacity in its entirety [9, 10].

Some proposals that achieve a certain level of autonomy of an WMR are related to the capacities to perceive, shape, plan and act, which have been worked on from different points of view. One of these proposals is the reactive scheme based on perception and action, i.e. the WMR implements a sensor that, on receiving a stimulus from the environment, modifies the state of the actuators. Although this scheme reduces the processing speed of the robot controller (it requires few resources to operate), it limits the amount of movements of the actuators because they are reduced to the indications given at the time of generating the controller [11-13].

Increasing the number of motions available on an WMR controller is not a simple task, since increasing the robustness of the controller reduces the robot's operating speed when performing a task. Therefore, motion and control schemes have been widely studied and there are remote or local implementation algorithms for operating an WMR. Some of the most used algorithms currently are the route mapping algorithms, machine vision algorithms and local operation algorithms. First, there are the routing algorithms that from a defined environment create a sequence of instructions that are sent to the robot, to go from one place to another. In second place, there are the machine vision algorithms that by means of an image processing algorithm determine the location of obstacles so that the robot avoids them with the advantage that the environment can be partially known. Finally, the local operation algorithms are operating routines that are implemented directly on the robot controller and depend on the response of the sensors to the environment [12-14].

In summary, the range and speed of response of an WMR depends on the robustness of the controller. However, the efficiency of a controller is a subject under study, since, current controllers are subject to variables that limit the amount of movement of the actuators and the ability to explore their environment. Therefore, this article proposes a control strategy for WMR that generates instruction libraries without a defined depth using an optimization algorithm so that the robot performs movements in an adaptive way. The algorithm developed and the results obtained are reported in the following sections, which are organized as follows: section 2 describes the concepts required to understand the functioning of the algorithm, section 3 presents the proposed algorithm and describes its operation. Finally, section 4 presents the results obtained.

2. MATERIALS AND METHODS

The WMR controller proposed in this article codes the movements of the actuators of a unicycle robot using the ROS (robot operating system) language, from the motion sets built using an optimization algorithm in a simulator set up for that purpose. These concepts are explained in detail in this section.

2.1. Unicycle robot

The combination of different types of wheels or their location within the structure of an WMR changes the kinematics of the robot, so they have been grouped in different ways. Although the functionality of the WMR allows it to be classified according to its specialty (e.g., search or rescue), the arrangement of its wheels is the most common form of classification, since it differentiates robots with respect to their maneuverability, and among them are: omnidirectional, unicycles, tricycles and quadricycles [9-13].

Omnidirectional robots have a greater number of degrees of freedom (number of axes in which a movement is made) than other types of robot, since they move in almost any direction without the need to re-orient themselves when they are going to rotate. Unicycles have a simple kinematics; their structure is composed of two fixed wheels aligned on the same axis and a support wheel that allows them to turn on the turning axis of the platform as shown in Figure 1. Then there are the tricycles and quadricycles that incorporate three or four wheels (respectively) in their structure trying to keep the center of gravity in the center of the structure when increasing the robot speed. In this case, the TurtleBot unicycle robot was used, which consists of a modular chassis to place electronic components, such as: control cards, cameras, sensors or accessories. In addition, the controller of this robot interprets ROS locally, which allows the user to use a standard language form to control the robot movements [15].

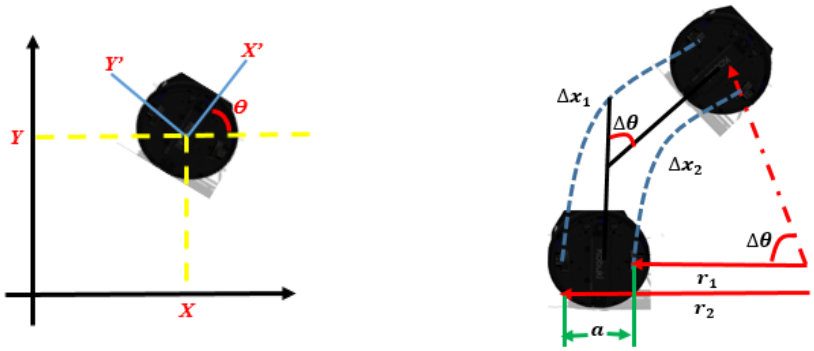
On the one hand, the robotic operating system or ROS is a standardized work environment for developing applications for robots and implements some basic services that contemplate: the abstraction of the hardware, the control systems, the way of communication and the transmission of messages between the robot and the controller. This environment is based on networks where the processor receives, sends and multiplexes the messages from the sensors, actuators and states of the platform. Furthermore, it is entirely free and can be implemented in a UNIX operating system (e.g. Ubuntu) [16, 17].

On the other hand, the TurtleBot generates motion schemes from an absolute reference system, which indicates that finding a robot position is equivalent to finding a coordinate with translation and rotation components $[x, y, \theta]$. This means that the robot has three degrees of freedom, shown in Figure 2 (a). Also, the position and orientation of the robot is determined using a technique called odo-metry, which achieves it through a partial time

integration of movements [15-17]. One of the advantages of odo-metry is that it is simple, easily implemented and requires relatively few computational resources, which allows for an increase in the sampling time to determine the orientation of the mobile. However, the margin of error increases as the wheels wear or slide, so periodic calibrations are required. This means that the relative orientation of the robot depends on the speed of the wheels, which when rotating at the same speed do not generate a variation in position and when increasing the speed in one of them also does it the tendency towards a direction, as shown in Figure 2 (b) [18].

(a) (b)

Figure 1. TurtleBot mobile robot; (a) font view, and (b) bottom view



(a) (b)

Figure 2. TurtleBot reference frames; (a) reference system, and (b) variablea involved in a turn

More specifically, to determine the robot position, mathematical transformations based on the robot reference frames are used, where the path of wheel 1 and 2 are represented by (1), the form of the average path by (2) and the orientation by is constructed by subtracting two coordinates and taking into account the wheel spacing in (3) [19]. It is worth noting that there are other ways to determine the orientation of a robot, such as; measurement with a GPS, digital goniometers or gyroscopes. However, the scheme implemented by ROS for the handling of the TurtleBot is based on the odo-metric technique and thus determines an orientation and relative position [20]. This information is relevant for the generation of the controller using an optimization algorithm, which is described in the following section.

$$\Delta x_i = r_i \Delta\theta \tag{1}$$

$$\Delta x = \frac{\sum_1^2 \Delta x_i}{2} \tag{2}$$

$$\Delta\theta = \frac{\sum_1^2 \Delta x_i}{a} \tag{3}$$

2.2. Optimization algorithm

An optimization algorithm is a technique that, when implemented in a software, finds approximate values to an optimal value. Usually, this value can be the largest or the smallest of a set of values and to find it the algorithm performs a series of iterations that alter the input values according to an aptitude value. Where, the proficiency value represents the performance of the variable when modifying the input variables (individual) of the function [21, 22].

A case of optimization algorithms is genetic algorithms (GA), which iteratively updates a population of individuals to select the best among them. In a similar way to natural selection, GA implement selection, crossing and mutation methods, which from an aptitude value assigned to everyone determine whether it survives when updating the population (generational change) [23]. GAs are very versatile tools and can be adapted to almost any need, so there are many varieties and classes. In its simplest form is the conventional GA that has no restrictions on the application of selection, crossing and mutation operations, and a variant of its computational implementation converts individuals into binary numbers to facilitate crossing and mutation operations (Algorithm 1) [24-27].

```
Algorithm 1. Conventional genetic algorithm
i=0
P0=0
While i < condition of termination do
    f ← Evaluating the aptitude function(P0)
    P1 ← Selection(P0, f)
    P1 ← crossing(P1, f)
    P1 ← mutation(P1)
    i++
```

Selection operators rank individuals based on their skill value, including roulette and tournament methods. First, the roulette method assigns everyone a probability value based on his or her skill value, i.e. a probability distribution is constructed by adding up all the skill values of the individuals and dividing them by that result. Then the method generates and searches for the individual with the closest probability, to that of a randomly generated number. Secondly, the tournament method selects four individuals at random and compares their skill value, among them it selects two and from the two resultants it selects one.

The binary crossing operator takes two individuals from the population, codes them into a binary number and cuts their resulting binary strings at a random point and recombines them to generate two new individuals, see Figure 3 (a). Similarly, the mutation operator selects an individual, converts it into a binary number and changes the value of a bit, see Figure 3 (b). The proposed controller builds motion sets using a modified GA, in which the number of crossing operators is increased and together with the mutation operator they are selected with a predefined threshold value. These values and the implementation of the modified GA are described in detail in the following section.

Figure 3. Genetic operators; (a) crossing operator, and (b) mutation operator

3. DEVELOPMENT AND IMPLEMENTATION

Initially, the controller generates a set of 20 individuals, which are sets of parameters coded in ROS language that allow the robot to execute movement routines, where, the suitability value is the distance the robot travels with the indications given by the individual during 10 seconds, see Figure 4 (a). Each set of parameters is composed by groups of up to 5 components (generated at random), that can be forward movements (duration between 0 and 5 s), backward movements (duration between 1 and 5 s) and turning (between $-\pi$ and π), see Figure 4 (b).

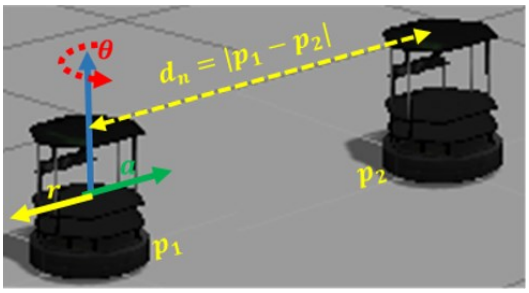
Like the conventional GA, the selection method incorporated in this version of the GA is tournament and the selection in each phase is made following a Bernulli distribution, shown in Figure 5. That is, everyone has a fifty percent (50%) chance of advancing to the next phase. In addition, this method is complemented by the selection and crossing methods, since thresholds were set for the algorithm to select among the genetic operators of crossing and mutation.

On the one hand, there are three options for selecting a genetic cross operator. The first option shown in Figure 6 (a), the combination of characteristics between two randomly selected individuals. The second shown in Figure 6 (b), the combination of similar parameters (if any) between individuals, e.g. if both individuals have the turn attribute their turn angles are exchanged. The third shown in Figure 6 (c), operator simply reverses the components of a selected individual. On the other hand, the genetic mutation operator is simpler, see Figure 6 (d), since it changes an individual in the population for a new one.

I	P1	P2	P3	P4	P5	f
1	θ	a	r	θ	r	d_1
2	a	r	-	-	-	d_2
3	a	r	θ	-	-	d_3
...						
20	θ	θ	θ	θ	-	d_{20}

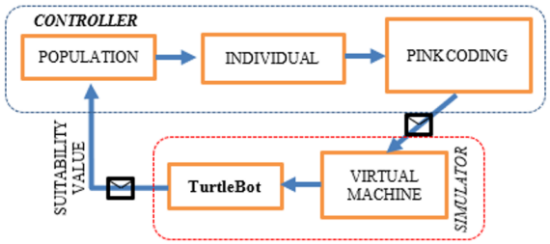
$a = forward, r = reverse,$
$\theta = turn, d_n = distance\ traveled$

(a)

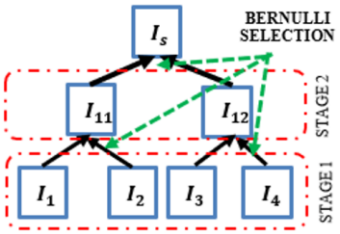


(b)

Figure 4. General diagram of an individual; (a) composition of an individual and the population, and (b) view of the parameters to be evaluated and calculation of the suitability value

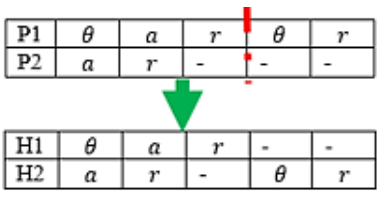


(a)

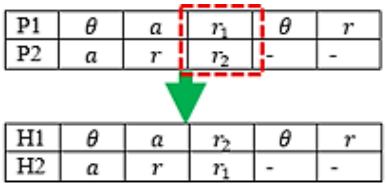


(b)

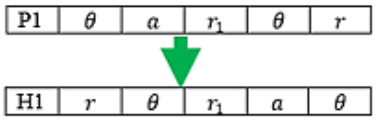
Figure 5. Functional diagrams of an individual's assessment method and selection method; (a) individual assessment scheme, and (b) representation of the selection method



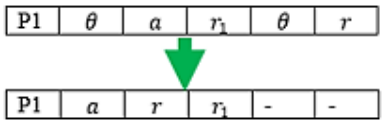
(a)



(b)



(c)



(d)

Figure 6. Genetic cross and mutation operators; (a) genetic crossing operator 1, (b) genetic crossing operator 2, (c) genetic crossing operator 3, and (d) genetic crossing operator 4

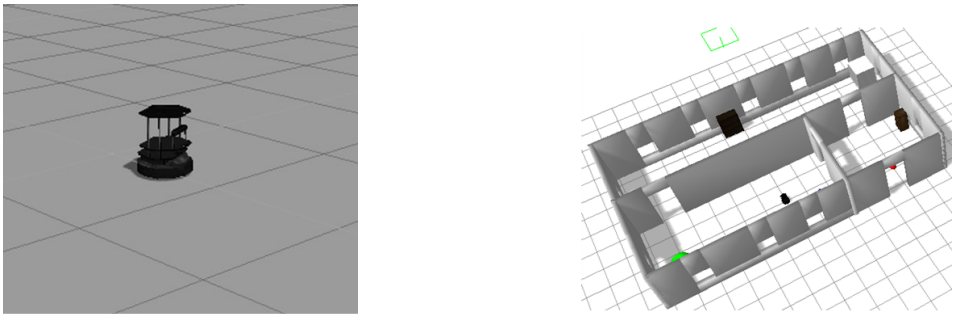
Genetic crossing and mutation operators are combined with a conventional GA, which incorporates a fixed threshold for the selection of each individual operator. That is, in each iteration of the modified GA the execution of an operator depends on a random number that follows a uniform distribution. It should be noted that when an individual is evaluated, the execution of the modified GA is stopped until the simulator responds with a value for the distance travelled (Algorithm 2).

```
Algorithm 2. Modified genetic algorithm
Function Genetic algorithm ()
   P0[20][5] =initial population;
   Generations=0;
   P0←Generate 20 individuals at random;
   While Generations<Termination condition do
      d←Evaluate (P0)
      P1←Tournament (P0, d)
```

```
        S~U [0,1]
        If S< 0.25 then
            P1←Crossing operator 1(P1)
        Else If S>=0.25 and S<0. 5 then
            P1←Crossing operator 2 (P1)
        Else If S>=0.5 and S<0. 75 then
            P1←Crossing operator 3 (P1)
        Else
            P1←Mutation Operator (P1)
       P0←P1
       Generations++

Function Evaluation (P)
 T←0
 Establishing a connection with the virtual machine
 If T<20 then
       Encode Individual with ROS tags (P0[T] [1:5])
       p_1=read current position ()
       Send motion parameters to the simulator (P0[T] [1:5])
       Wait 10 seconds
       p_2=read current position ()
       d[T]=|p_1-p_2 |
       T++
 Return d
```

In order to evaluate the performance of the modified GA, an implementation of the conventional GA was carried out. This implementation was done according to Algorithm 1, where the selection method is roulette and the cross and mutation operator is presented in Figure 6 (a) and Figure 6 (d) respectively. In both cases, the test environment is the working space of the gazebo simulator and the main objective of the TurtleBot is to travel the greatest distance possible, in a flat environment with and without obstacles, see Figure 7. This workspace is installed in a virtual machine with LINUX operating system, which ran on a computer with WINDOWS 10 operating system, an Intel inside TM core i3 processor, 8Gbs of RAM memory and a 240Gbs hard disk.



(a) (b)

Figure 7. Test scenarios; (a) unobstructed environment, and (b) obstacle environment

4. RESULTS AND ANALYSIS

The AGs mentioned in section 2 and section 3 carried out 1000 aptitude value evaluations over 50 algorithm runs. That is, each algorithm had a generational change over 50 generations in both scenarios. In the unobstructed environment, the information collected allowed the construction of the graphs shown in Figure 8 that show the standard deviation, the average, the best and the worst with respect to the distance traveled. In the environment with obstacles, the information collected allowed the construction of the graphs in Figure 9, where the behavior of the robot when trying to avoid obstacles and travel through its environment during a conventional and modified GA execution respectively is presented. Considering that 50 GA runs were performed in the obstacle environment, the best suitability value found by each algorithm is presented in Table 1. Finally, the interpretation of the results obtained is presented in the following section.

Table 1. Best fit value found in the obstacle environment by both GA

	Conventional GA	Modified GA
Distance travelled [cm]	43.12 ± 5.76	35.15 ± 6.88

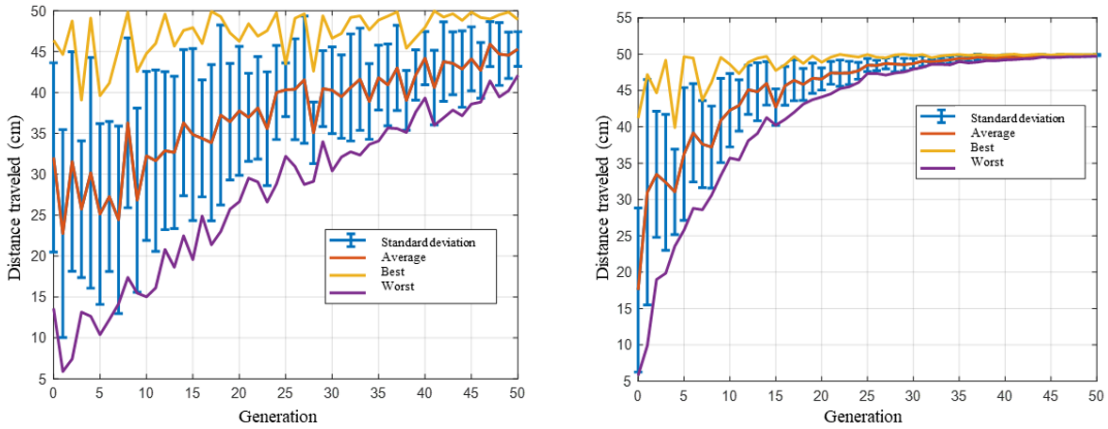


Figure 8. Trend of conventional and modified GA in an unhindered environment;
(a) conventional GA (X50), and (b) modified GA (X50)

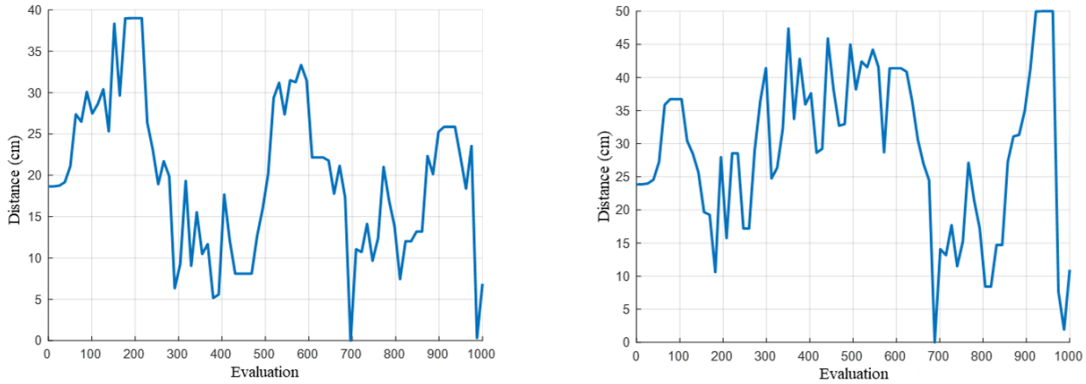


Figure 9. Trend of conventional and modified GA in a hindered environment;
(a) conventional GA, and (b) modified GA

5. CONCLUSIONS

The conventional GA is versatile enough to adapt to the needs of each situation, which indicates that this type of optimization algorithms is not exclusively used for the solution of complex mathematical problems. As shown in Figure 4 (b) the function to be optimized has only one arithmetic operator and depends on two values to fix the aptitude value of an individual, which indicates that determining an aptitude value can be achieved from simple mathematical transformations. In addition, this type of transformation requires a low computational cost, which gives the user the flexibility to implement it in a conventional desktop computer.

The graphs in Figure 8 show that both GAs find individuals that allow the robot to travel up to 50 cm in a straight line, which indicates that during the evolution the turning and backward movements disappear. Although, both GA find relatively good configurations of individuals, in conventional GA it has a lower speed of convergence, since, the mutation operator makes very strong changes to the population increasing in convergence time. However, by limiting the effect of the mutation operator on the modified GA, the effect of the mutation operator is reduced and so is the convergence time. In addition, increasing the number of crossing operators takes advantage of the characteristics of good individuals in the population.

The graphs in Figure 9 were not made in a similar way to the unobstructed environment, as the robot makes random movements when it encounters an obstacle, which does not allow a convergence of the proficiency value. As shown in the lower part of the graph, sometimes the proficiency value is less than 10 cm which means that the robot is trapped or cannot avoid the obstacle for a certain time. However, when dodging the obstacle, the aptitude value increases rapidly, which leads both GAs to follow that path and thus the robot travels through much of the environment. Although, the same characteristics of Figure 8 were not measured, Table 1 was constructed showing the best individual found by both GA, of which the modified GA found some partially better ones.

REFERENCES

[1] T. Tokunaga, K. Oka and A. Harada, "1segment continuum manipulator for automatic harvesting robot - prototype and modeling," *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*, pp. 1655-1659, 2017.

[2] Y. Wan and H. Xu, "On dynamics simulation of 3 DOF manipulator," *2016 35th Chinese Control Conference (CCC)*, pp. 6290-6294, 2016.

[3] S. Kucuk and B. D. Gungor, "Inverse kinematics solution of a new hybrid robot manipulator proposed for medical purposes," *2016 Medical Technologies National Congress (TIPTEKNO)*, pp. 1-4, 2016.

[4] 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)*, pp. 1011-1014, 2018.

[5] H. Watanabe, "Development of Wafer Transfer Simulator Based on Cellular Automata," *IEEE Transactions on Semiconductor Manufacturing*, vol. 28, no. 3, pp. 283-288, Aug. 2015.

[6] Y. Ninomiya, Y. Arita, R. Tanaka, T. Nishida and N. I. Giannoccaro, "Automatic Calibration of Industrial Robot and 3D Sensors using Real-Time Simulator," *2018 International Conference on Information and Communication Technology Robotics (ICT-ROBOT)*, pp. 1-4, 2018.

[7] B. Mu, J. Chen, Y. Shi and Y. Chang, "Design and Implementation of Nonuniform Sampling Cooperative Control on A Group of Two-Wheeled Mobile Robots," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 6, pp. 5035-5044, 2017.

[8] O. Calla, S. Mathur and K. L. Gadri, "Possible Landing site for Chandrayaan-2 Rover," *2016 International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, pp. 1-5, 2016.

[9] 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)*, pp. 1011-1014, 2018.

[10] K. Piemngam, I. Nilkhamhang and P. Bunnun, "Development of Autonomous Mobile Robot Platform with Mecanum Wheels," *2019 First International Symposium on Instrumentation, Control, Artificial Intelligence, and Robotics (ICA-SYMP)*, pp. 90-93, 2019.

[11] D. Cui, X. Gao, W. Guo and H. Dong, "Design and Stability Analysis of a Wheel-Track Robot," *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*, pp. 918-922, 2016.

[12] H. Wang, B. Li, J. Liu, *et al*., "Dynamic modeling and analysis of Wheel Skid steered Mobile Robots with the different angular velocities of four wheels," *Proceedings of the 30th Chinese Control Conference*, pp. 3919-3924, 2011.

[13] Z. Fan, Q. Qiu and Z. Meng, "Implementation of a four-wheel drive agricultural mobile robot for crop/soil information collection on the open field," *2017 32nd Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 408-412, 2017.

[14] K. Priandana, *et al*., "Design of A Task-Oriented Autonomous Wheeled- Robot for Search and Rescue," *2018 International Conference on Advanced Computer Science and Information Systems (ICACSIS)*, pp. 259-263, 2018.

[15] H. Aagela, *et al*., "An Asus_xtion_probased indoor MAPPING using a Raspberry Pi with Turtlebot robot Turtlebot robot," *2017 23rd International Conference on Automation and Computing (ICAC)*, pp. 1-5, 2017.

[16] R. Mishra and A. Javed, "ROS based service robot platform," *2018 4th International Conference on Control, Automation and Robotics (ICCAR)*, pp. 55-59, 2018.

[17] A. Koubâa, *et al*., "Turtlebot at Office: A Service-Oriented Software Architecture for Personal Assistant Robots Using ROS," *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 270-276, 2016.

[18] M. M. Kassir and M. Palhang, "Novel qualitative visual odometry for a ground: Vehicle based on funnel lane concept," *2017 10th Iranian Conference on Machine Vision and Image Processing (MVIP)*, pp. 182-187, 2017.

[19] Q. Lin, X. Liu and Z. Zhang, "Mobile Robot Self-LocalizationUsing Visual Odometry Based on Ceiling Vision," *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, pp. 1435-1439, 2019.

[20] M. Galli, R. Barber, S. Garrido and L. Moreno, "Path planning using Matlab-ROS integration applied to mobile robots," *2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 98-103, 2017.

[21] M. Patrascu and A. Ion, "Self-Adaptation in Genetic Algorithms for Control Engineering: The Case of Time Delay Systems," *2017 21st International Conference on Control Systems and Computer Science (CSCS)*, pp. 19-25, 2017.

[22] A. V. Mokshin, V. V. Mokshin and L. Sharnin, "Adaptive genetic algorithms used to analyze behavior of complex system," *Communications in Nonlinear Science and Numerical Simulation*, vol. 71, pp. 174-186, 2019.

[23] A. Hussein, "Improve the Performance of K-means by using Genetic Algorithm for Classification Heart Attack," *International Journal of Electrical and Computer Engineering*, vol. 8, no. 2, pp. 1256-1261, 2018.

[24] E. B. Asmae, B. Benhala, and I. Zorkani, "A genetic algorithm for the optimal design of a multistage amplifier," *International Journal of Electrical and Computer Engineering*, vol. 10, no. 1, pp. 129-138, 2020.

[25] S. Harun and M. F. Ibrahim, "A genetic algorithm-based task scheduling system for logistics service robots," *Bulletin of Electrical Engineering and Informatics*, vol. 8 no. 1, pp. 206-213, 2019.

[26] S. Shan, *et al*., "An adaptive genetic algorithm for demand-driven and resource-constrained project scheduling in aircraft assembly," *Information Technology and Management*, vol. 18, no.1, pp. 41-53, 2017.

[27] M. J. Varnamkhasti, "A genetic algorithm rooted in integer encoding and fuzzy controller," *International Journal of Robotics and Automation*, vol. 8 no. 2, pp. 113-124, 2019.