Proceedings of the 25th International Conference on Automation & Computing, Lancaster University, Lancaster UK, 5-7 September 2019

Ant Colony Optimization Algorithm for Industrial Robot Programming in a Digital Twin

Ridhi Bansal, Mojtaba Ahmadieh Khanesar, David Branson
Faculty of Engineering,
Department of Mechanical, Materials and Manufacturing Engineering,
University of Nottingham, Nottingham, UK
Emails: {eayrr4, ezzma5, ezzdtb}@exmail.nottingham.ac.uk

Abstract— **Advanced manufacturing that is adaptable to constantly changing product designs often requires dynamic changes on the factory floor to enable manufacture. The integration of robotic manufacture with machine learning approaches offers the possibility to enable such dynamic changes on the factory floor. While ensuring safety and the possibility of losses of components and waste of material are against their usage. Furthermore, developments in design of virtual environments makes it possible to perform simulations in a virtual environment, to enable human-in-the-loop production of parts correctly the first time like never before. Such powerful simulation and control software provides the means to design a digital twin of manufacturing environment in which trials are completed at almost at no cost. In this paper, ant colony optimization is used to program an industrial robot to avoid obstacles and find its way to pick and place objects during an assembly task in an environment containing obstacles that must be avoided. The optimization is completed in a digital twin environment first and movements transferred to the real robot after human inspection. It is shown that the proposed methodology can find the optimal solution, in addition to avoiding collisions, for an assembly task with minimum human intervention.**

***Keywords-** manufacturing; artificial intelligence; programming robot; digital twin; ant colony optimization,*

I. INTRODUCTION

Usage of artificial intelligence approaches in a manufacturing environment is quite challenging because of inevitable iterations required to be performed before success. In a manufacturing environment, such iterations are highly expensive and may cause serious damages which make them impossible to implement. A digital twin is a virtual representation of real world which is designed for various purposes such as product design, smart manufacturing, usage monitoring and maintenance, repair and overhaul [1] of which smart manufacturing applications fell into the scope of this paper. The complete production line from raw material to market can be designed and optimized in a factory floor [1], [2]. Such virtual environment make trial and errors required by machine learning and artificial intelligence approaches possible. An exact and realistic digital twin is desirable to make sure that the iteratively found solution is applicable to the system right from the first time.

Many applications require robots to move through obstacles to reach a desired position. Path planning is therefore commonly used in navigation for autonomous cars, UAVs, mobile robots, and satellites to enable collision free motion [3][4]. Similarly, in manufacturing tasks autonomous path planning of industrial robots is required due to product changes. Particularly in assembly tasks where changing the product design often necessitates changes in robot movement to accommodate new variations in production methods, currently done using manual reprogramming.

Path planning algorithms can be broadly classified as conventional and heuristic approaches [5]. Conventional approaches which are old algorithms require manual programming at some places and are computationally expensive as compared to heuristic approaches [6]. Examples of conventional approaches [5] are BUG algorithms, roadmap, cell decomposition, potential fields, and mathematical programming. Such approaches may fall in local minima more frequently than heuristic optimization algorithms. Heuristic path optimization approaches are autonomous path planning methods that have recently been developed. A few of the commonly used heuristic methods [5] are neural network [7], [8] and [9], genetic algorithms, particle swarm optimization [10], ant colony optimization (ACO) [11] and [12], stigmergy, wavelet theory, fuzzy logic [12] and [13], and Tabu Search. Heuristic approaches are preferred over conventional methods as they are better in dealing with complicated problems and local minimum solutions. Ant colony optimization was used for path planning in this study as it is easy to employ and adapt to a new problem [14].

In case of industrial robots, it is highly desirable to make robot autonomously find an optimal collision free path. This became more important as in new industrial systems manufacturing customizable products emerge flexible factory line. Soft computing approaches have been previously applied to such applications. For instance, continuous Genetic Algorithms is used to find the Collision-Free Cartesian Path for robotic manipulators [15]. Optimal positioning of surgical robot and path planning of its end-effector in a crowded operating room is another challenging task to which genetic algorithm is applied [16]. A 3D working model including surgical object is designed and the optimization is performed in such an environment.

In this paper, the ACO is used in a digital twin environment to find the shortest collision free path in 3D space for an industrial robot undergoing a pick and place operation within an environment containing obstacles that must be avoided. To reduce risks and number of experiments done with the robot, in this case a UR5, a digital twin of the environment is designed in Gazebo software. The obstacles designed in the digital twin have been created to represent the existence of obstacles in a production environment. The digital twin model has been created to detect possible collisions and are used to penalize the cost function when obtaining a collision free solution. Possible robot movements in 3D space are then encoded in terms of an array that represent translation and rotation of the end factor in the virtual environment. Inverse kinematics is then used to make robot joints move to the required position, where collisions will result in displacement of obstacles in the digital twin. By comparing the initial and final position of obstacles, a possible collision is detected automatically without carrying out manual calculations. As multiple calls of the cost function are required during the optimization process, the position of the robot and other components in the digital twin are required to reset to their initial conditions after each cost function execution. An optimal path was generated in virtual environment by integrating ACO algorithms to find a collision free route to fulfill the pick and place task. The resulting digital twin therefore provides a means to test obtained movement in a visual interface by a human expert to guarantee implementability and safety. In the simulation section, it is shown that the proposed method is a successful optimization algorithm to find the optimal path in the presence of obstacles.

This paper is organized as follows. The general ACO technique is described in Section II. Then the overall methodology and description of steps taken to perform the optimization in a digital twin are given in Section III, simulation results are presented in Section IV, and concluding marks are provided in Section V.

II. ANT COLONY OPTIMIZATION

ACO is a nature inspired optimization algorithm. It is based on ant's behavior to use swarm intelligence to find food [17]. ACO is a heuristic path planning method used to find optimal path. In this method, ants deposit pheromones on the path while travel that are sensed by other ants. Figure 1 shows a group of ants finding an optimal path from nest to food. The changes paths taken can be easily accommodated in ACO as pheromones evaporate with time. Pheromones deposition speed is faster than the evaporation rate in the paths visited more frequently by ants due to that path's short length. ACO is easy to implement, adaptable to new problems and can achieve both local and global minimum solutions [14]. This means that whenever a new feature is added to the path pheromone levels can be locally initialized by just changing the pheromone values near the object added or can be globally initialized by setting all the pheromones back to original level. ACO equations were used to employ a group of ants and update pheromone levels to find desired solution. Equations used (equation 1-4) are shown below [17]. The probability of ant k located at node i to go to node j depends on heuristic information and pheromone update in Equation (1).

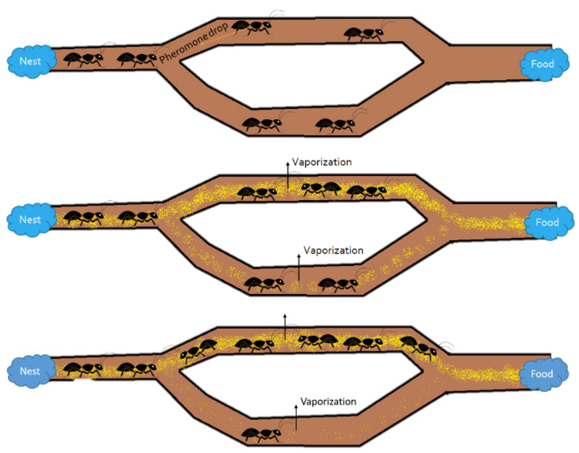

Figure 1. Ants finding optimal path to food

$$p_{ij}^{k} = \begin{cases} \frac{(\tau_{ij}^{k})^{\alpha}(\eta_{ij}^{k})^{\beta}}{\sum_{l\in N_i^k}(\tau_{ij}^{k})^{\alpha}(\eta_{ij}^{k})^{\beta}} & \text{if } j \in N_i^K \\ 0 & \text{if } j \notin N_i^K \end{cases} \quad (1)$$

Heuristic information depends on movements given to reach target and position number of ways ant can move. Pheromones remaining due to evaporation are determined as:

$$\tau_{ij}\,(t+\Delta t) = (1-\rho)\times\,\tau_{ij}(t) \quad (2)$$

where, τ_{ij} represents pheromones deposited on the path from nodes i to node j, ρ is pheromone evaporation rate, and is selected as $0 \le \rho \le 1$. Pheromone levels after a new pheromone is deposited then become:

$$\tau_{ij}\,(t+\Delta t) = \,\tau_{ij}\,(\mathrm{t}) \,+\sum_{k=1}^{m}\Delta\,\tau_{ij}{}^{k} \quad (3)$$

$$\Delta\,\tau_{ij}{}^{k} = \frac{1}{c^{k}} \quad (4)$$

where k represents ant doing path planning and c^k is associated with cost or reward ant k gets for choosing this path. The algorithm evolves for a few iterations to find the optimal path with highest concentration of pheromones.

III. METHODOLOGY

A. *Cost function to be optimized*

The digital twin designed in this paper is a replica of real industrial process which includes industrial components and machinery. The main goal is to automatically design the path for a robot end effector to move from its initial condition to a designated target position and orientation, without having any collision with obstacles in the environment. Collision detection is done automatically in the digital twin using specially written software in Python. Collision between the robot and components make the objects move from their initial position. By comparing initial position of components and their final position, a flag is returned as a consequence and can then be used to penalize the cost function. Collisions during movement are captured using software with the cost function defined as:

$$J = (x - x_d)^2 + (y - y_d)^2 + (z - z_d)^2 + (a - a_d)^2 + (b - b_d)^2 + (c - c_d)^2 + P.C \quad (5)$$

where x_d, y_d, z_d, a_d, b_d and c_d represent the desired values of position and orientation of robot, P represent the penalty term and C which is assigned a discrete value representing the absolute displacement value of any of investigated components. The parameters x, y, z, a, b and c are the position and orientation of robot before the last movement to the destination. Displacement values larger than software tolerance are considered to be collision. Two values are assigned for small displacements and large displacements in components. Following formula summarizes this logic.

$$C = \begin{cases} 0 & d < tol \\ C_1 & tol \le d < V_1 \\ C_2 & V_1 \le d < V_2 \end{cases} \quad (6)$$

where C_1 and C_2 represent penalties given for the case when the sum of absolute values of displacements are larger than V_1 and V_2 and tol is the tolerance of software.

B. Encoding movements

To optimize movements, they need to be encoded into an evolvable array. The ACO then evolves this array to find the optimal solution minimizing the cost function. The python code is then integrated with a gazebo model and the end effector of the robot moves in 3D space using an array containing integer values from the set $S = \{0, 1, \dots, 11\}$. To move the robot, joint angles corresponding to movements are required. Positions of the end effector were used to find rotation matrix (equation 8-11) then used to find joint angles of robot using inverse kinematics equations. The interpretation of movements is presented in Table I.

TABLE I. INTERPRETATION OF VALUES OF ARRAY GENERATED BY ACO

Movements		Rotations	
Value	**Interpretation**	**Value**	**Interpretation**
0	$x = x + 0.1$	6	$a = a + 5°$
1	$x = x - 0.1$	7	$a = a - 5°$
2	$y = y + 0.1$	8	$b = b + 5°$
3	$y = y - 0.1$	9	$b = b - 5°$
4	$z = z + 0.1$	10	$c = c + 5°$
5	$z = z - 0.1$	11	$c = c - 5°$

C. Kinematic and inverse kinematics of UR5

The values obtained from the movement array need to be translated in terms of robot movements. To enable this a transformation matrix that represents position and orientation of robot needs to be obtained [18] [19]. The transformation matrix for frame 6 with respect to frame 0 is as:

$$^{6}_{0}T(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = \begin{bmatrix} ^{0}_{6}R & ^{0}_{6}P \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} ^{0}_{6}\hat{X}_x & ^{0}_{6}\hat{Y}_x & ^{0}_{6}\hat{Z}_x & ^{0}_{6}P_x \\ ^{0}_{6}\hat{X}_y & ^{0}_{6}\hat{Y}_y & ^{0}_{6}\hat{Z}_y & ^{0}_{6}P_y \\ ^{0}_{6}\hat{X}_z & ^{0}_{6}\hat{Y}_z & ^{0}_{6}\hat{Z}_z & ^{0}_{6}P_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (8)$$

where $^{0}_{6}\hat{X}$, $^{0}_{6}\hat{Y}$ and $^{0}_{6}\hat{Z}$ are unit vectors defining the axis of frame 6 in relation to frame 0 and:

$$^{0}_{6}R = R_x(a) R_y(b) R_z(c)$$

in which the rotational matrices are obtained as follows:

$$R_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos a & -\sin a \\ 0 & \sin a & \cos a \end{bmatrix} \quad (9)$$

$$R_y = \begin{bmatrix} \cos b & 0 & \sin b \\ 0 & 1 & 0 \\ -\sin b & 0 & \cos b \end{bmatrix} \quad (10)$$

$$R_z = \begin{bmatrix} \cos c & -\sin c & 0 \\ \sin c & \cos c & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (11)$$

Inverse kinematics is then used to find joint angles of robot to move to position given by ACO. The angles of robot are then obtained as [18] [19]:

$$\theta_1 = atan2\left({}^{0}_{5y}P, {}^{0}_{5x}P\right) + \text{acos}\left(\frac{d_4}{\sqrt{{}^{0}_{5x}P^2 + {}^{0}_{5y}P^2}}\right) + \frac{\pi}{2}$$

$$\theta_2 = atan2(-{}^{1}_{4z}P, -{}^{1}_{4x}P) - asin\left(\frac{-a_3 \sin\theta_3}{|{}^{1}_{4xz}P|}\right)$$

$$\theta_3 = \pm acos\left(\frac{|{}^{1}_{4xz}P|^2 - a_2{}^2 - a_3{}^2}{2a_2a_3}\right)$$

$$\theta_4 = atan2\left({}^{3}_{4y}\hat{X}, {}^{3}_{4x}\hat{X}\right)$$

$$\theta_5 = \pm acos\left(\frac{{}^{0}_{6x}P \sin\theta_1 - {}^{0}_{6y}P \cos\theta_1 - d_4}{d_6}\right)$$

$$\theta_6 = atan2(M_1, M_2)$$

where:

$$M_1 = \frac{-{}^{6}_{0y}\hat{X} \sin\theta_1 + {}^{6}_{0y}\hat{Y} \cos\theta_1}{\sin\theta_5}$$

$$M_2 = \frac{{}^{6}_{0y}\hat{X} \sin\theta_1 - {}^{6}_{0y}\hat{Y} \cos\theta_1}{\sin\theta_5}$$

D. Collision detection

To evaluate the cost function, it is required to detect collisions and penalize the optimization routine accordingly. The position of components are published from Gazebo 7.0, which can be used to detect collision by comparing final positions with initial positions of the components. This is automatically done within the software as the position of components are reported as *rostopic* which can be used to detect possible collisions. Figure 2 illustrates a possible collision generated by robot moving according to a solution generated by ACO. As can be seen from the figure, Gazebo completely simulates such collisions and considers physical interaction between components including friction and gravitational force.

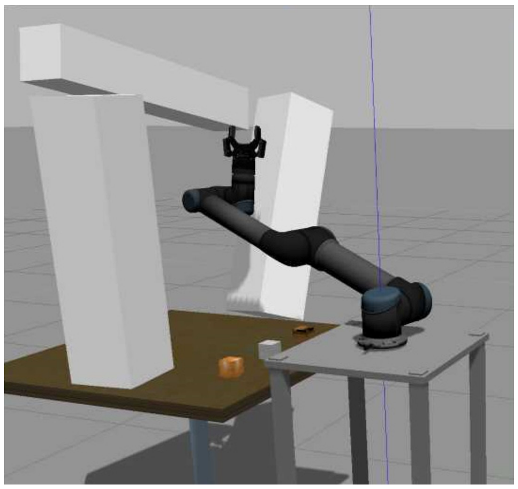

Figure 2. Collision due to movement generated by an ACO solution in digital twin

E. Methodology

Figure 3 demonstrates the steps taken to perform automatic path planning for UR5 in digital twin using ACO. The green box in this figure represent the steps related to the evaluation of the cost function.

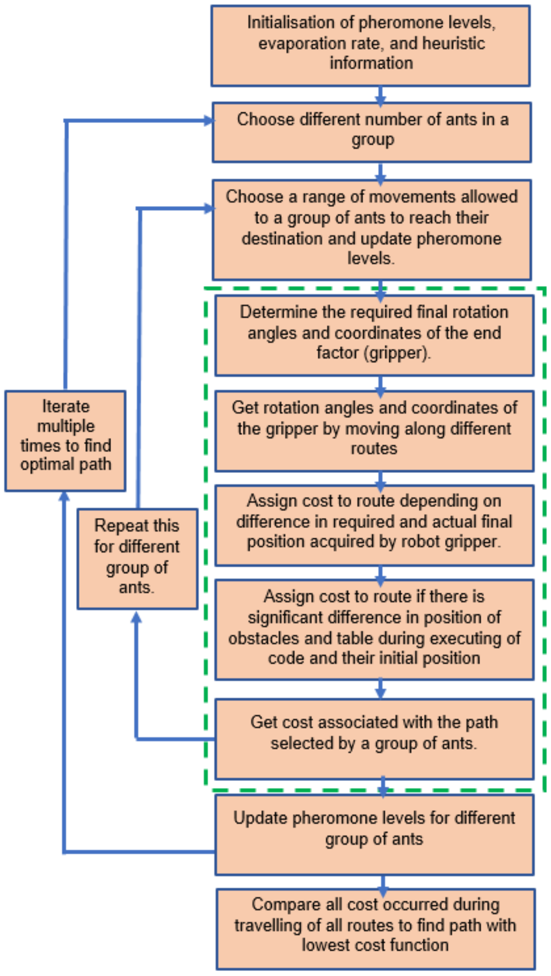
Initialisation of pheromone levels, evaporation rate, and heuristic information
Choose different number of ants in a group
Choose a range of movements allowed to a group of ants to reach their destination and update pheromone levels.
Determine the required final rotation angles and coordinates of the end factor (gripper).
Get rotation angles and coordinates of the gripper by moving along different routes
Assign cost to route depending on difference in required and actual final position acquired by robot gripper.
Assign cost to route if there is significant difference in position of obstacles and table during executing of code and their initial position
Get cost associated with the path selected by a group of ants.
Update pheromone levels for different group of ants
Compare all cost occurred during travelling of all routes to find path with lowest cost function
Iterate multiple times to find optimal path
Repeat this for different group of ants.

Figure 3. Steps taken for 3D path planning in simulation environment

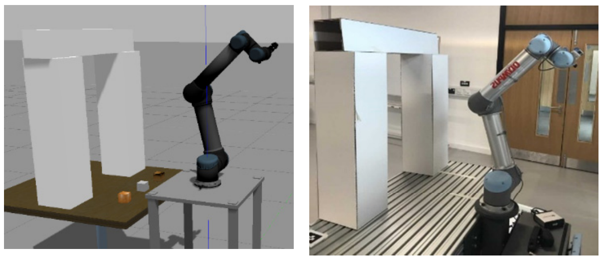

Figure 4. The digital twin of environment including UR5 robot vrrsus real robot

SIMULATION RESULTS

The results obtained for 3D path planning of industrial robots in digital twin using ACO is presented in this section. The cost function and the movements are as defined in Section III. Number of movements are considered to be equal to 20, number of ant agents considered is equal to 10 and the optimization algorithm is iterated for 100 times.

Figure 4 compares the real world and the digital twin designed for that. Figure 5 illustrates the method used for this simulation. The ACO is implemented under python software that communicates with ROS using *rostopics.* The results of simulations in terms of final distance between the end effector and desired position as well as possible displacement in obstacles are reported back in terms of corresponding *rostopics* that are then used in Gazebo to evaluate movements. The whole algorithm is iterated for 100 iterations before its convergence. The

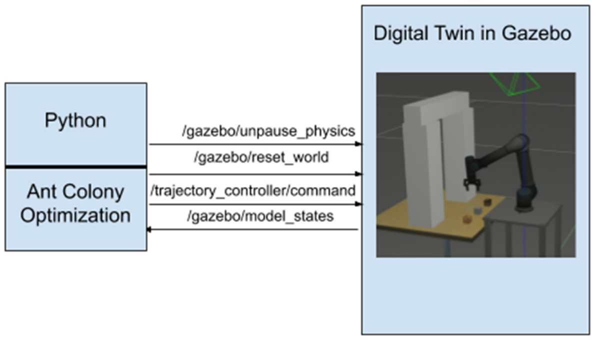

Figure 5. Communication between Python and ROS

movements are implemented in digital twin using /trajectory_controller/command rostopic in terms of robot joint angles. The positions of obstacles are reported back using /gazebo/modelstates rostopic. The rostopic /gazebo/reset_world is used to reset the position of objects after each evaluation of the cost function.

Figure 6 illustrates the evolution of cost function versus iterations. As can be seen from this figure, ACO has effectively been successfully in reducing the value of cost function. As can be seen from the figure ACO has successfully reduced the value of cost function from 1.37 to 0.31 which is 77% of improvement. The result of doing optimization in terms of motion time-lapse of robot is presented in Fig. 7. This figure shows that the final motion obtained is collision free and end effector successfully hits its target destination.

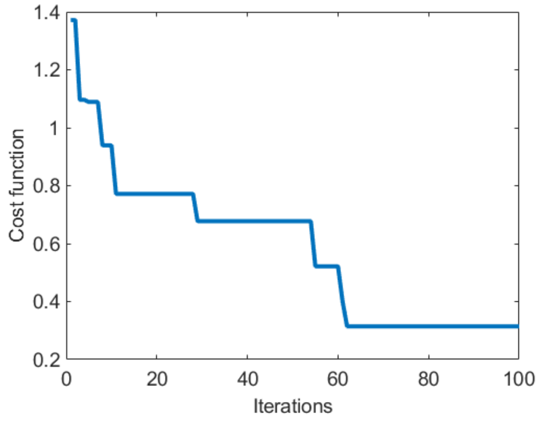

Figure 6. The evolution of cost function versus iterations

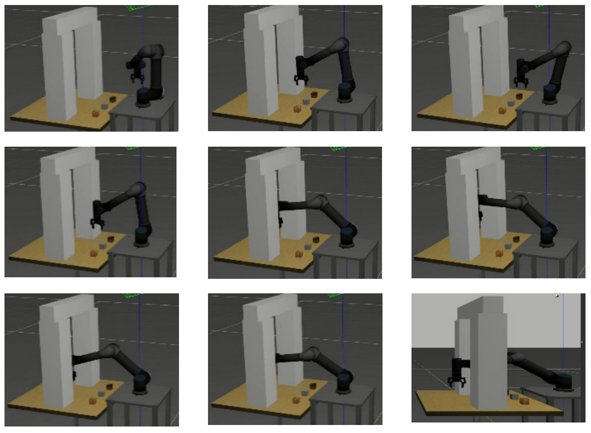

Figure 7. Motion time-lapse of robot for a successful run (every 3 seconds)

IV. CONLUSIONS

To implement product changes rapidly on the factory floor, manufacturing elements are required to learn to perform given tasks autonomously with minimum human intervention. Machine learning typically requires trial and error approaches that in most cases cannot be performed directly on the factory floor due to factors such as health and safety and to reduce waste. The utilization of a digital twin of a robotic manufacturing system provides the mean for performing multiple learning iterations at almost no cost. In such ACO as a successful, nature inspired, method to deal with path planning problems is utilized here in the planning of a pick and place task. The task is encoded for ACO and a cost function is considered to plan a path to the desired position without any collisions with environmental objects. Simulation results demonstrate that ACO can successfully perform the optimization in the digital twin using the proposed procedure.

ACKNOWLEDGMENT

This work is funded and supported by the Engineering and Physical Sciences Research Council (EPSRC) under grant number: EP/R021031/1 - New Industrial Systems: Chatty Factories.

REFERENCES

[1] H. Zhang, Q. Liu, X. Chen, D. Zhang, and J. Leng, "A Digital Twin-Based Approach for Designing and Multi-Objective Optimization of Hollow Glass Production Line," *IEEE Access*, vol. 5, pp. 26901–26911, 2017.

[2] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *IFAC-PapersOnLine*, vol. 28, no. 3, pp. 567–572, 2015.

[3] M. A. P. Garcia, O. Montiel, O. Castillo, R. Sepúlveda, and P. Melin, "Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation," *Appl. Soft Comput. J.*, vol. 9, no. 3, pp. 1102–1110, 2009.

[4] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser, "Heuristic approaches in robot path planning: A survey," *Rob. Auton. Syst.*, vol. 86, pp. 13–28, 2016.

[5] M. K. A. A. S. H. Tang, W. Khaksar, N. B. Ismail, "A Review on Robot Motion Planning Approaches," *Pertanika J. Sci. Technol.*, vol. 20, no. 1, pp. 15–29, 2012.

[6] B. Beklisi, K. Ayawli, R. Chellali, A. Y. Appiah, and F. Kyeremeh, "An Overview of Nature-Inspired , Conventional , and Hybrid Methods of Autonomous Vehicle Path Planning," vol. 2018, 2018.

[7] O. Avoidance, "Neural Network Dynamics for Path Planning and Obstacle Avoidance," 1995.

[8] H. Li, S. Member, S. X. Yang, S. Member, and M. L. Seto, "Neural-Network-Based Path Planning for a Multirobot System With Moving Obstacles," vol. 39, no. 4, pp. 410–419, 2009.

[9] E. Engineering, "Neural network and genetic algorithm based global path planning in a static environment," vol. 4, no. 6, pp. 549–554, 2005.

[10] J. Han and Y. Seo, "Mobile robot path planning with surrounding point set and path improvement," *Appl. Soft Comput. J.*, vol. 57, pp. 35–47, 2017.

[11] U. Cekmez, M. Ozsiginan, and O. K. Sahingoz, "Multi Colony Ant Optimization for UAV Path Planning with Obstacle Avoidance," *2016 Int. Conf. Unmanned Aircr. Syst.*, pp. 47–52, 2016.

[12] M. Fakoor, A. Kosari, and M. Jafarzadeh, "Humanoid robot path planning with fuzzy Markov decision processes," *Rev. Mex. Trastor. Aliment.*, vol. 14, no. 5, pp. 300–310, 2016.

[13] M. Wang and J. N. K. Liu, "Fuzzy logic based robot path planning in unknown environment," vol. 1, no. August, pp. 18–21, 2005.

[14] Y. Hsiao, C. Chnang, and C. Chien, "Ant Colony Optimization for Best Path Planning," vol. 2004, no. 1, 2004.

[15] Z. S. Abo-Hammour, O. M. Alsmadi, S. I. Bataineh, M. A. Al-Omari, and N. Affach, "Continuous genetic algorithms for collision-free Cartesian path planning of robot manipulators regular paper," *Int. J. Adv. Robot. Syst.*, vol. 8, no. 6, pp. 14–36, 2011.

[16] Q. C. Nguyen, Y. Kim, and H. D. Kwon, "Optimization of layout and path planning of surgical robotic system," *Int. J. Control. Autom. Syst.*, vol. 15, no. 1, pp. 375–384, 2017.

[17] M. Brandi, M. Masudai, N. Wehner, and X. Yu, "Ant Colony Optimization Algorithm for Robot Path Planning," 2010, vol. 3, no. 2010 International Conference On Computer Design And Appliations (ICCDA 2010), pp. 436–440.

[18] K. P. Hawkins, "Analytic Inverse Kinematics for the Universal Robots," 2013.

[19] Andersen RS. Kinematics of a UR5. Aalborg University. 2018 Mar