JST

Journal of Science and Technology

Journal homepage: http://penerbit.uthm.edu.my/ojs/index.php/jst
ISSN : 2229-8460 e-ISSN : 2600-7924

Distributed Path Planning Classification with Web-based 3D Visualization using Deep Neural Network for Internet of Robotic Things

Z. Iklima[1*], T M Kadarina[1]

[1]Department of Electrical Engineering,
Mercu Buana University, South Meruya No. 31, 11610, Jakarta, INDONESIA

*Corresponding Author

DOI: https://doi.org/10.30880/jst.2021.13.02.006
Received 13 July 2021; Accepted 26 November 2021; Available online 2 Disember 2021

Abstract: Internet of Robotic Things (IoRT) distributes heterogeneous intelligences among devices and platforms. A distributed control of a three-degree-of-freedom (3-DOF) robot manipulator is integrated with web-based 3D visualization. An asynchronous protocol was utilized to broadcast kinematic data of a 3-DOF robot manipulator between platforms. However, kinematic data computed using inverse kinematic equations directly cannot identify the singularity issue of robot manipulator. Singularity avoidance required to prevent robot component or joint from damage. Therefore, this study proposed a deep neural network approach as a classification-based of manipulator robot path planning to avoid singularity issues. Deep neural network (DNN) was trained in 12 minutes, 52 seconds in 500 iterations. Training accuracy measured with value 96,23 percent, validation accuracy measured with value 96,13 percent, and testing accuracy measured with value 96,48 percent Additionally, 3 DOF manipulator robot web-based 3D visualization was made using Web Graphics Library (WebGL). The distributed platform was tested successfully and can distribute and classify 2352 motions per second.

Keywords: IoRT, 3 DOF manipulator robot, 3D visualization, inverse kinematic, path planning, DNN

1. Introduction

The Internet of Robotic Things (IoRT) connects disparate intelligent devices using a distributed platform design. These systems can be deployed in the cloud or on-premises. This approach takes advantage of current IoT technology and robotic device convergence to improve robotic capabilities. Simply, it allows robots to be constructed on client-server architecture using HTTP protocol. These architectures were improved to provide bidirectional communication protocol as a distributed architecture. Hence, distributed control of the robot manipulator was implemented over TCP protocols which allows the web server to broadcast motion data among environments. For instance, a distributed control of a robot manipulator synchronized with 3D visualization. The web server has the ability to communicate bidirectionally into multiple robot manipulators and multiple 3D visualization clients. The web server transferred 122 Bytes of motion data with only 7,86 microseconds per request [1]. The data transferred was calculated using an inverse kinematic formula which computes the end-effector coordinate given by joint angles.

To achieve accurate trajectory planning, inverse kinematics is one of the most popular approaches to solve trajectory planning error [2]. A major problem occurs when computing an inverse kinematic solution namely kinematic singularity. Kinematic singularities can arise both within the internal singularities and at the external singularities. External singularities can be handled by shifting the robot's base, while internal singularities can be avoided by leveraging the additional degrees of freedom of the joints. Several manipulability measures, such as the manipulability index, the condition number, and the smallest singular value of the manipulator robot, have been proposed to characterize the proximity to kinematic singularities. Avoiding the kinematic singularities, which are associated with

*Corresponding author: zendi.iklima@mercubuana.ac.id

loss of or limited motion capabilities of the robot end-effector and infeasible joint velocity commands, is a major difficulty for robot manipulators. Particular approaches to solve kinematic singularities such as Jacobian matrix using filtered inverse [3], trajectory planning [4, 5], genetic algorithm [6], configuring redundancy resolution [7], singularity analysis [8], artificial potential fields [9], and so on.

Therefore, this paper proposed path planning classification using Deep Neural Network (DNN) which classifies path planning labels such as singular and non-singular. The model is able to retargeting end-effector coordinates while classified to singular label until reaching non-singular minimum end-effector coordinates fit to the set point direction. The term of a distributed system, reduces training load of conventional DNN training processes. This paper implements web-based 3D visualization which utilizes bidirectional communication protocol namely Socket.IO over TCP protocol [10].

2. Methods

Path planning classification was designed to prevent singularities issue of inverse kinematic solution. Fig. 1 shows the process of distributed path planning classification with web-based 3D visualization.

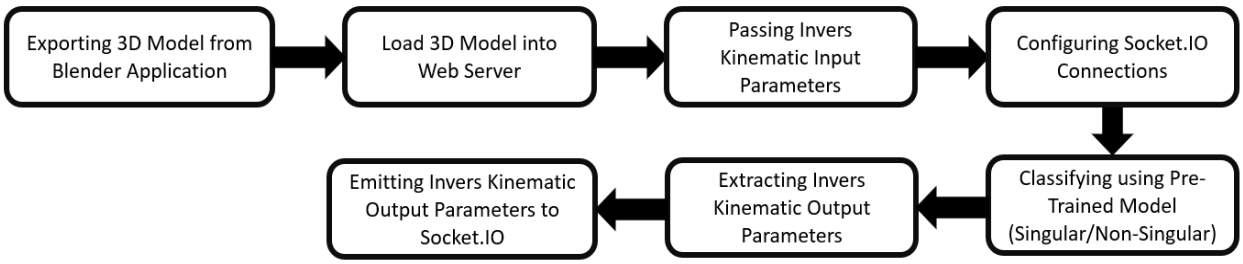


Fig. 1 - Path planning classification process with 3D model

Based on Fig.1, the 3D model of the robot manipulator was exported as a JSON file which allows the web server to render as a 3D mesh object. These exported 3D models contain mesh, materials, and bones. The bones utilized to control whole 3D materials move based on given end-effector coordinates which are calculated using inverse kinematic parameters. The 3D client (virtual manipulator robot) through Socket.IO emits the end-effector coordinates to the web-server. Furthermore, the inverse kinematic input parameters are classified using the DNN pre-trained model to determine selected paths of singular or non-singular. if the classifier output refers to singularity the model will recommend another end-effector coordinate in the same path or another path in the same end-effector. Non-singular inverse kinematic output, Socket.IO directly emitting among the environments (in this case virtual manipulator robot and manipulator robot).

2.1 System Architecture

Fig.2 shows the system architecture of distributed path planning classification for web-based 3D visualization constructed over ReactJS as back-end framework and AngularJS as front-end framework. Presentation layer represented in Fig. 2(a), handles front-end requirements using AngularJS which include WebGL engine. Additionally, presentation layer utilized as user control of manipulator robot 3D visualization. Application layer represented in Fig. 2(b), manages back-end function mainly data transmission over Socket.IO and singularity avoidance using path planning classifier. Data layer represented in Fig. 2(c) stores 3D objects such as bones, mesh and materials. Furthermore, the data layer stores a DNN pre-trained model which is utilized as a classifier model.

Based on Fig. 2, Socket.IO has an important role in supporting distributed systems that transmit data among environments. The environment consists of a web server (ReactJS), WebGL as a 3D visualization engine, and robot manipulator. Fig. 3 shows how Socket.IO manages the data transmitted over its events.

Socket.IO mainly has two events called '*onLinstening'* and '*onEmitting'*. '*onEmitting'* event transmits nonsingular inverse kinematic output which is classified by the classifier and distributed through its events. '*onListening'* event retrieves incoming inverse kinematic parameters. ReactJS web server listens to an event called '*end_eff_coor'* which retrieves incoming inverse kinematic input parameters such as end-effector coordinates. Vice versa, the clients emitting the end-effector coordinates to the web server. ReactJS web server emitting non-singular inverse kinematic output as angular values of manipulator robot joints. Vice versa, the clients retrieve and extract the angular values of manipulator robot joints as trajectory points.

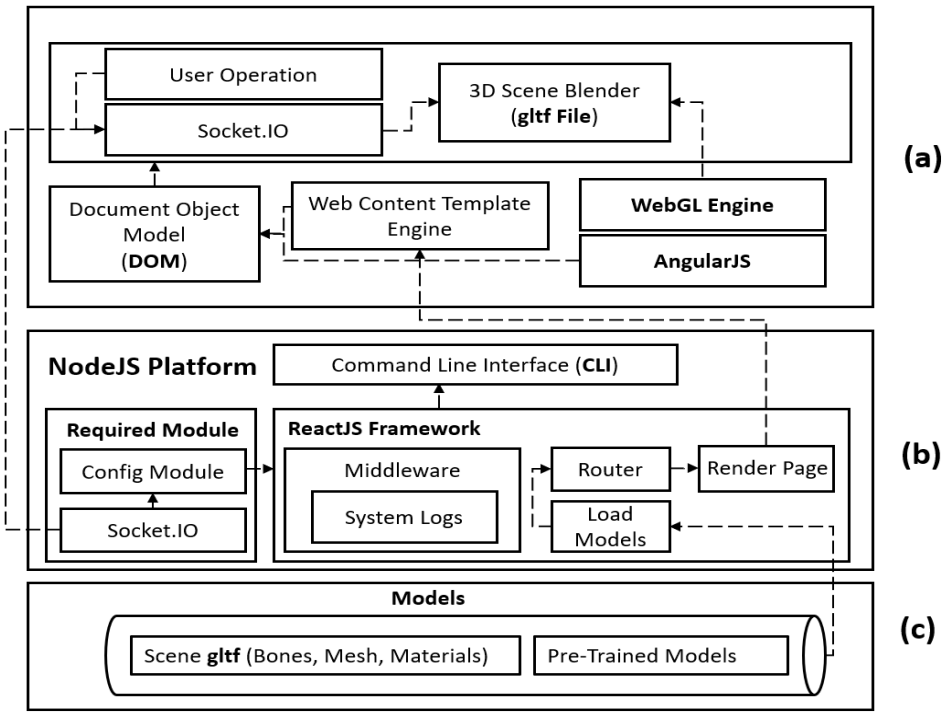


Fig. 2 - System architecture - (a) presentation layer; (b) application layer; (c) data layer

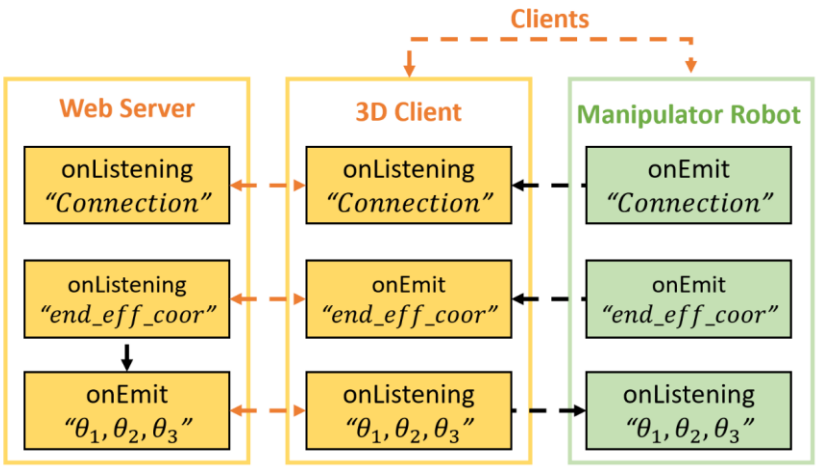


Fig. 3 - Socket.IO events

2.2 3 DOF Manipulator Robot

The design of a manipulator robot is a key component of creating equations to help with the robot's feasibility. It is defined as connecting the joint spatial geometry idea and end-effector coordinates by altering the geometrical equation. Forward kinematics is defined as the process of converting joint space $\theta_1, \theta_2, ...\theta_n$ to joint variable (end-effector x, y, z). Inverse kinematics converts joint variable (x, y, z) to joint space ($\theta_1, \theta_2, ...\theta_n$). Fig. 4 represents the local reference coordinate of 3 DOF manipulator robot [11].

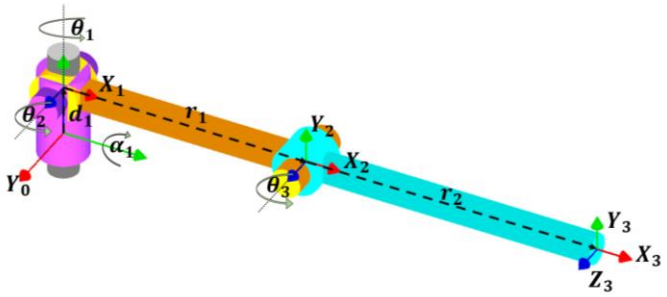


Fig. 4 - Local reference coordinate of 3 DOF manipulator robot

Fig. 4 clearly shows the local reference coordinate of the 3-DOF robot manipulator that can be parameterized using Denavit-Hartenberg parameters [12] in Table 1.

Table 1 - Denavit-Hartenberg parameters

Link-n	θ (rad)	d (mm)	α (rad)	l (mm)
1	θ_1	30	90	0
2	θ_2	0	0	120
3	θ_3	0	0	180

The join offset measured $d_1 = 30mm$, link length of join-2 measured $l_1 = 120mm$, link length of join-3 measured $l_2 = 180mm$, and joint angle measured $\alpha_1 = 90^o$. θ_1, θ_2, and θ_3 denotes as joint space that represents as an angle measured among x_i and x_{i+1} in z-axis direction.

2.3 Deep Neural Network

Deep Neural Network (DNN) approach requires constructing repetitive computation process to propose an inverse kinematic solution which determine joint variable (x, y, z) given by joint space ($\theta_1, \theta_2, ...\theta_n$). The path planning dataset was collected 25.906 motions and manually labelled such as singular or non-singular. The dataset normalization is required to be calculated into the data distribution. Fig. 5 shows the process of path planning classification using the DNN approach.

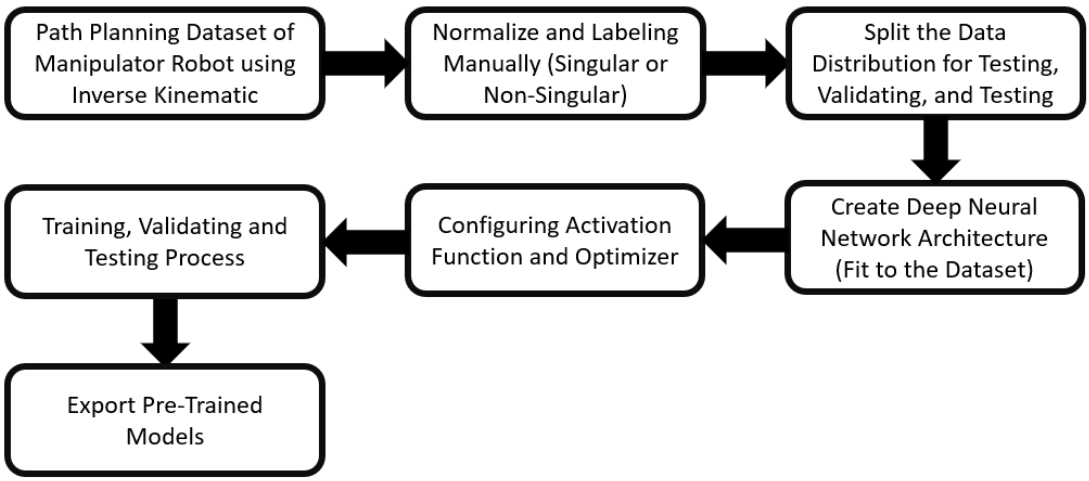


Fig. 5 - Path planning classification process using DNN.

Based on Fig. 5, the DNN architecture constructs three neurons in the input layer and two neurons in the output layer. The input layer fits to the total joint variable required. The output layer fits to the total classifier labels in this case singular label or non-singular label. The singular label totally counted has 495 motions data of 25.906 motions data. The path planning dataset is divided into three data distributions which are training, validation, and testing. DNN hyperparameter was initialized as particular DNN models analyzed to find an optimum model of the path planning data

distribution and avoid the model being overfitted. This paper configures 10 DNN models within the number of hidden layers (4, 8, 16, 32, and 64), the number of neurons in the hidden layer or hidden layer weights (8, 16, 32, and 64), the number of training iterations (100), LeakyReLu or Sigmoid as model activation function, RMSProp as model optimizer, and categorical cross-entropy as model loss function [13–15]. This paper proposed 12 DNN models which configures in Table 2.

Table 2 - DNN models performance

Model No.	Hidden Layer	Neuron in Hidden Layer	Activation Function
1	4	8	Sigmoid
2	4	16	Sigmoid
3	4	32	Sigmoid
4	4	64	Sigmoid
5	8	32	Sigmoid
6	16	32	Sigmoid
7	32	32	Sigmoid
8	64	32	Sigmoid
9	16	32	LeakyReLu
10	16	64	LeakyReLu
11	32	64	LeakyReLu
12	64	64	LeakyReLu

3. Results and Discussion

The DNN models were performed using Intel(R) Core i5-6300HQ CPU@2.30GHz, 16GB RAM, and Nvidia GeForce GTX 960M 4GB VRAM. Table 3 shows the DNN models performance comparison which captured training accuracy, validation accuracy, training accuracy, training loss, validation loss, recall, precision, and execution time.

Table 3 – DNN models performance

Model	Tr. Acc	Val. Acc	Ts. Acc	Loss	Val. Loss	Recall	Precision	F1-Score	Exec. Time
1	0,9617	0,9642	0,9639	0,1498	0,1442	0,9756	0,9719	0.9738	4 min, 4 sec
2	0,9615	0,9638	0,9641	0,1562	0,1345	0,9689	0,9697	0.9693	10 min, 16 sec
3	0,9615	0,9605	0,9643	0,1439	0,1359	0,9666	0,9681	0.9674	5 min, 3 sec
4	0,9617	0,9593	0,9645	0,1360	0,1393	0,9710	0,9712	0.9711	2 min, 35 sec
5	0,9615	0,9593	0,9638	0,1537	0,1470	0,9712	0,9713	0.9712	3 min, 16 sec
6	**0,9630**	**0,9613**	**0,9648**	**0,1372**	**0,1462**	**0,9837**	**0,9816**	**0.9827**	**4 min, 42 sec**
7	0,9617	0,9607	0,9642	0,1627	0,1682	0,9778	0,9709	0.9743	12 min, 57 sec
8	0,9615	0,9559	0,9636	0,1634	0,1667	0,9764	0,9724	0.9744	11 min, 32 sec
9	0,9616	0,9606	0,9642	0,1627	0,1682	0,9745	0,9722	0.9733	26 min, 27 sec
10	0,9616	0,9598	0,9644	0,1624	0,1680	0,9757	0,9711	0.9734	5 min, 23 sec
11	0,9617	0,9605	0,9639	0,1542	0,1578	0,9767	0,9721	0.9744	6 min, 57 sec
12	0,9616	0,9621	0,9642	0,1559	0,1582	0,9737	0,9745	0.9739	11 min, 25 sec

Based on Table 2, all models trained in 500 iterations and learning rate equal to 0,001. Model 6 structured 4×32 neuron weights and RMSProp optimizer. Model 6 configured as a proportional model to identify the singularity of robot manipulator path planning. Fig. 6 extend performance metrics shown in Table 2.

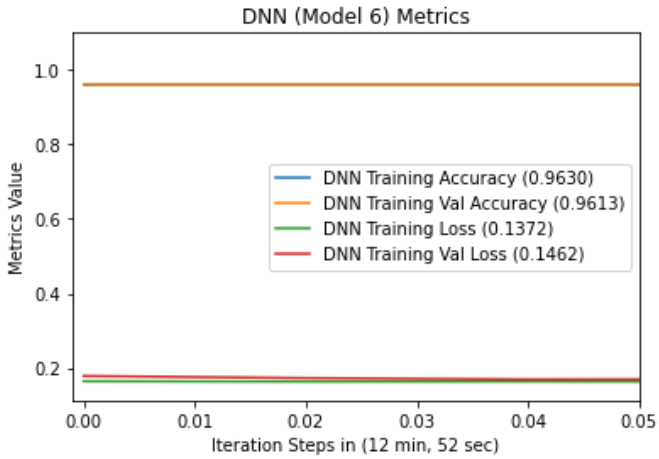
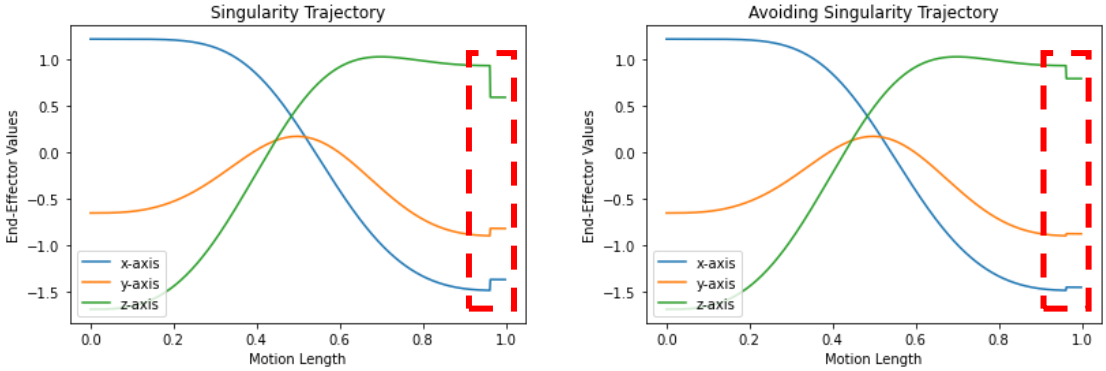


Fig. 6 - DNN (model 6) performance metrics

The pre-trained model (DNN Model 6) evaluates to classify singularity path planning of the robot manipulator. Fig.7 shows the pre-trained model (DNN Model 6) minimizing singularity issue with current path planning in the same end-effector direction.



(a) (b)

Fig. 7 - Singularity path planning correction– (a) before; (b) after

Based on Fig.7 (a), path planning of the robot manipulator has a singularity issue which interferes with the kinematic computation results. Therefore, the pre-trained model (DNN Model 6) classified the path planning of the robot manipulator to detect a singularity as shown in Fig. 7 (b). Additionally, the pre-trained model (DNN Model 6) recommends another path planning within the same end-effector direction or fixing the path planning to fit the end-effector set point. Distributed data transmission represents in Table 3, which emitting and listening joint variable (x, y, z) and joint space ($\theta_1, \theta_2, ...\theta_n$) among environments.

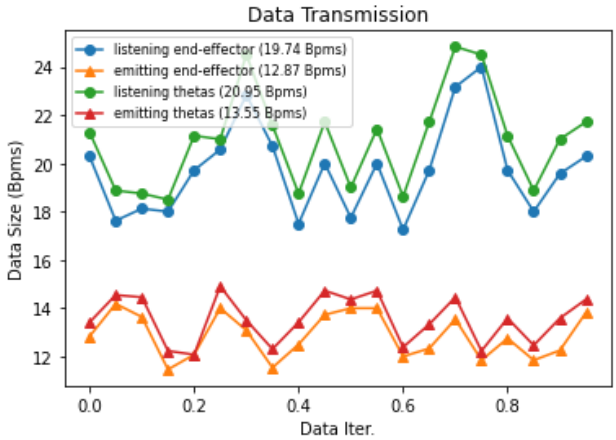


Fig. 8 - Data transmission

Based-on Fig. 3, Socket.IO was captured using Chrome/94.0.4606.61 that summarized in Fig. 8. Data distribution was captured over TCP protocols within 19.74 Bytes per milliseconds (for listening end-effector event), 12.87 Bytes per milliseconds (for emitting end-effector event), 20.94 Bytes per milliseconds (for listening theta event), and 13.55 Bytes per milliseconds (for emitting theta event). Moreover, Socket.IO was tested to distribute 2352 motions per second.

4. Conclusion

Distributed Path Planning Classification with Web-based 3D Visualization using DNN for IoRT was implemented using ReactJS Framework. Asynchronous data transmission allows the web server to broadcast inverse kinematic parameters to the environment. To analyze singularity path planning, the dataset preprocessing supports the DNN model to reach the best performance of inverse kinematic path planning solutions. To circumvent singularity difficulties, this study offered a deep neural network methodology as a classification-based manipulator robot path planning method. In 500 iterations, a deep neural network (DNN) was trained in 12 minutes, 52 seconds. Training accuracy was calculated at 96,23 percent, validation accuracy was calculated at 96,13 percent, and testing accuracy was calculated at 96,48 percent. Web Graphics Library was also used to create a 3 DOF manipulator robot web-based 3D visualization (WebGL). 2352 motions per second were disseminated and classified on the distributed platform.

Acknowledgement

The author expresses gratitude to the Electrical Engineering Department and Research Center of Mercu Buana University, Jakarta, Indonesia. Under their support, the authors have completed this research paper.

References

[1] I. Zendi, T. M. Kadarina, “Distributed Control with Web-based 3D Visualization using Kinematics Analysis for IoRT,” *J. Eng. Sci. Res.*, vol. 2, no. 2, pp. 107–114, 2020, doi: 10.23960/jesr.v2i2.62

[2] Y. Li, D. Shang, and Y. Liu, “Kinematic modeling and error analysis of Delta robot considering parallelism error,” *Int. J. Adv. Robot. Syst.*, vol. 16, no. 5, pp. 1–9, 2019, doi: 10.1177/1729881419878927

[3] L. V. Vargas, A. C. Leite, and R. R. Costa, “Overcoming kinematic singularities with the filtered inverse approach,” *IFAC Proc. Vol.*, vol. 19, pp. 8496–8502, 2014, doi: 10.3182/20140824-6-za-1003.01841

[4] J. Sverdrup-Thygeson, S. Moe, K. Y. Pettersen, and J. T. Gravdahl, “Kinematic singularity avoidance for robot manipulators using set-based manipulability tasks,” *1st Annu. IEEE Conf. Control Technol. Appl. CCTA 2017*, vol. 2017-Janua, pp. 142–149, 2017, doi: 10.1109/CCTA.2017.8062454

[5] W. Xu, D. Meng, H. Liu, X. Wang, and B. Liang, “Singularity-Free Trajectory Planning of Free-Floating Multiarm Space Robots for Keeping the Base Inertially Stabilized,” *IEEE Trans. Syst. Man, Cybern. Syst.*, vol. 49, no. 12, pp. 2464–2477, 2019, doi: 10.1109/TSMC.2017.2693232

[6] P. P. Rebouças Filho, S. P. Suane, V. N. Praxedes, J. Hemanth, and V. H. C. de Albuquerque, “Control of singularity trajectory tracking for robotic manipulator by genetic algorithms,” *J. Comput. Sci.*, vol. 30, pp. 55–64, 2019, doi: 10.1016/j.jocs.2018.11.006

[7] C. Faria, F. Ferreira, W. Erlhagen, S. Monteiro, and E. Bicho, “Position-based kinematics for 7-DoF serial manipulators with global configuration control, joint limit and singularity avoidance,” *Mech. Mach. Theory*, vol. 121, pp. 317–334, 2018, doi: 10.1016/j.mechmachtheory.2017.10.025

[8] S. Teshigawara and H. Harry Asada, “A Mobile Extendable Robot Arm: Singularity Analysis and Design,” *IEEE Int. Conf. Intell. Robot. Syst.*, pp. 5131–5138, 2019, doi: 10.1109/IROS40897.2019.8967768

[9] D. E. Kim, D. J. Park, J. H. Park, and J. M. Lee, *Collision and singularity avoidance path planning of 6-DOF dual-arm manipulator*, vol. 10985 LNAI. Springer International Publishing, 2018

[10] M. Wenzel, A. Klinger, and C. Meinel, “Tele-board prototyper - Distributed 3D modeling in a web-based real-time collaboration system,” *Proc. - 2016 Int. Conf. Collab. Technol. Syst. CTS 2016*, pp. 446–453, 2016, doi: 10.1109/CTS.2016.82

[11] I. Zendi, A. Adriansyah, and S. Hitimana, “Self-Collision Avoidance of Arm Robot Using Generative Adversarial Network and Particles Swarm Optimization (Gan-Pso),” *Sinergi*, vol. 25, no. 2, p. 141, 2021, doi: 10.22441/sinergi.2021.2.005

[12] L. Qingsheng and J. Andika, “Analysis of Kinematic for Legs of a Hexapod Using Denavit-Hartenberg Convention,” *Sinergi*, vol. 22, no. 2, p. 69, 2018, doi: 10.22441/sinergi.2018.2.001

[13] Khaire, U.M., Dhanalakshmi, R. "High-dimensional microarray dataset classification using an improved adam optimizer (iAdam)". J Ambient Intell Human Comput 11, 5187–5204 (2020)

[14] S. Bock, J. Goppold, and M. Weiß, “An improvement of the convergence proof of the ADAM-Optimizer,” pp. 1–5, 2018, [Online]. Available: http://arxiv.org/abs/1804.10587

[15] Z. Zhang, “Improved Adam Optimizer for Deep Neural Networks,” *2018 IEEE/ACM 26th Int. Symp. Qual. Serv. IWQoS 2018*, pp. 1–2, 2019, doi: 10.1109/IWQoS.2018.8624183