Inverse Kinematics Analysis of Cassie Robot using Radial Basis Function Networks

Mukund Mitra*
Centre for
Product Design and Manufacturing
Indian Institute of Science
Bangalore, India
mukundmitra@iisc.ac.in

Suman Raj*
Department of
Computational and Data Sciences
Indian Institute of Science
Bangalore, India
sumanraj@iisc.ac.in

Shishir Kolathaya
Department of
Computer Science and Automation
Indian Institute of Science
Bangalore, India
shishirk@iisc.ac.in

***Abstract*—Inverse Kinematics of bipedal humanoid robots remains a challenging problem in the domain of robotics and computation, due to high order non-linearity and computation involved in Inverse Kinematics solutions. Also, there are many constraints involved with the various joint parameters which makes their analysis even more complex.**

Through this paper, we attempt to solve the Inverse Kinematics problem of a bipedal humanoid robot, Cassie, using Radial Basis Function (RBF) Networks. Our method can also be applied to other higher degrees of freedom serial manipulators. Our simulation analyses the results based on size of datasets, data distribution and network parameters. We have considered datasets of size $\sim$300k and $\sim$1 million, single and multiple hidden layers, equal and random data distribution, different number of neurons in layers and different training functions. We achieve our target of limiting the Mean Squared Error (MSE) calculated using the trained model below 0.1° for each joint angle, which is under acceptable limits for practical implementation.

The configurations obtained from the RBF network are simulated and compared with the original input configuration. This is compared visually in MATLAB and the resulting pose of end-effector are similar for both the cases, complementing the performance that we get for the networks.

***Index Terms*—Cassie, Forward kinematics, Inverse kinematics, RBF Networks, Bipedal Robots**

I. Introduction

Robot manipulators have been the most crucial unit of automation industry and with research progressing towards *The Fourth Industry Revolution* i.e. Industry 4.0, the main focus has been to innovate entities, such as bipedal robots that can help cope up with advancements in modern smart technology. Bipedal robots are highly dynamic and have degree of freedom as high as 20 in the case of Cassie. Having a deep understanding of the kinematics and dynamics of these robots becomes critical as they interact with systems that have a direct impact on human beings. In order to evaluate the robot's sensitivity in the workspace, it becomes necessary to precisely evaluate its mobility in the long run.

Inverse kinematics for a bipedal robot is very essential to find all the corresponding joint angles for a particular pose (position and orientation) of the end-effector i.e. toe of the bipedal robot. This becomes crucial in precise control of the trajectory of the end-effector in multiple applications like pick and place of objects in warehouse, surgical operations, battlefield search and rescue, etc. Since, we need to add a lot of actuators to provide flexibility to the robot's movement so that it can achieve the same degree of freedom as a human, this makes it more complex to understand their configuration. In this paper, we consider a reduced order model [1] of Cassie which means that the entire upper body mass is concentrated at pelvis. Also, the right and left leg of Cassie are symmetric and independent of each other. Thus, kinematic analysis of one leg of Cassie is equivalent to a 7 DoF spatial serial robotic arm.

Solving inverse kinematics for higher DoF manipulators involves high order non-linearity and computation, along with constraints. Various methods such as algebraic methods, geometric methods, numerical iterative methods can be adopted to solve this problem. However, they are computationally intensive, time consuming and cannot cope up with changes in robot structure and dynamic environments. The Inverse Kinematic problem of a simple 6R planar manipulator consists of solving a 16 degree polynomial [2] which is highly non-linear. Additionally, there are many singular configurations of the end-effector inside the workspace where the solution is undefined. Hence, these singularities and non-linearity makes the analytical methods difficult to solve. In order to address this, recent research has explored methods such as using Artificial Neural Networks (ANN) and optimal algorithms. Although the method using ANN to solve the IK problem also belongs to the iterative method, it differs from the traditional iterative method and depends on the network structure [3].

Using learning techniques guarantees high performance, precise control and less computing time as compared to the analytical method. Moreover, the solution obtained can also be optimized by changing various network parameters. Also, in case of multiple solutions, we can easily find the best possible solution. Adding to this, we can incorporate as many number of constraints as required in the problem. Thus, we have used Artificial Neural Networks to solve our Inverse

*Both the authors have contributed equally.

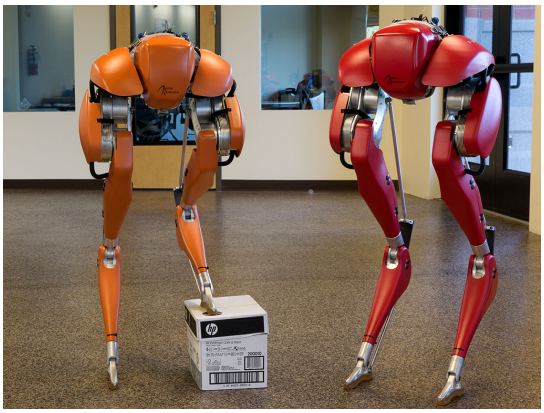

Fig. 1. Bipedal Robot - Cassie, from [5]

Kinematics problem of Cassie.

In this paper, our primary goal is to solve the IK problem for a suspended Cassie [4] using Radial Basis Function Networks and achieve a maximum error of 0.1° for joint angles. The rest of the paper is organized as follows: Section II discusses the related work, Section III details out the methodology that we adopt to solve forward and inverse kinematics of Cassie, Section IV explains the simulation setup and results and Section V concludes the paper and also discusses prospective future works.

II. Related work

Solving for an inverse kinematics problem of a bipedal robot is very important as it determines the amount of torque required to reach a particular position in task space. [6] and [7] have discussed the kinematic analysis of a 3 Degree of Freedom serial (DoF) robotic arm using algebraic methods. [8] proposes a novel design for a 6-DoF robotic arm and solves IK problem using coupled 3-DoF solutions using analytical method. In [9] IK solution is given for a mobile manipulator with strict non-holonomic constraints. In [10], analytical solution of IK problem of a redundant 7-DoF manipulator with link offsets is given. We see that analytical method for IK is highly non-linear and as the DoF increases, difficulty in obtaining a closed form solution increases. Therefore, we use learning techniques to analyse the IK problem of a higher DoF manipulator.

When it comes to bipedal robots, [11] examines the computational complexity and performance of different algorithms for solving IK for 30 DoF humanoid robot. The computationally most efficient method [12] can cause extraneous motion in null space which is rectified by another algorithm that incurs 3-fold increase in computational complexity. [13] studies the IK solutions for a general 6R manipulator using Matrix Polynomials where 75-80% of the time is spent in QR algorithms for computing eigen decomposition. The running time can be even further improved using better algorithms and implementations.

Our work, that overcomes the limitations posed by [11] and [13], is greatly inspired by [3] which constructs six

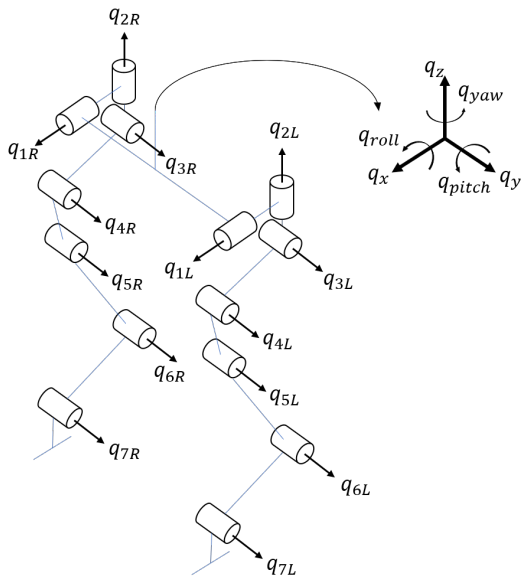

Fig. 2. Kinematic model of Cassie showing the robot's generalized coordinates in the body frame, where left and right leg are similar, from [19].

RBF networks for 6R manipulator. However, they do not specify the effect of change in network parameters on error in joint angles. [14] uses single RBF network and Genetic Algorithm for a 7-DoF manipulator, which results in an error in the range [0.93° 2.97°]. On the same lines, [15] achieves the best MSE as 11.52° with 1000 epochs. [16] uses six neural network for a 6-DoF redundant manipulator achieving a minimum error of 0.00706 radians i.e. 0.435° while [17] uses parallel Elman networks to achieve a minimum error of 1.49 radians with 1279 epochs. Studies have also been done on feeding back the current robot configuration in the design on ANN [18] in order to reduce the error, however adds to the complex computations. Our work aims to achieve a maximum error of 0.1° with an upper cap of 500 epochs and a simplified ANN design.

III. Bipedal Robot Kinematics

A visual image of Cassie is shown in Fig. 1. It is a highly dynamic bipedal robot with 20 degrees-of-freedom which is given by equation (4).

$$q\_pose = [q_x, q_y, q_z, q_{yaw}, q_{pitch}, q_{roll}] \quad (1)$$

$$q\_left\_leg = [q_{1L}, q_{2L}, q_{3L}, q_{4L}, q_{5L}, q_{6L}, q_{7L}] \quad (2)$$

$$q\_right\_leg = [q_{1R}, q_{2R}, q_{3R}, q_{4R}, q_{5R}, q_{6R}, q_{7R}] \quad (3)$$

$$q = [q\_pose, q\_left\_leg, q\_right\_leg]^T \quad (4)$$

where, (q_x, q_y, q_z) and $(q_{roll}, q_{pitch}, q_{yaw})$ are the Cartesian coordinates of the pelvis and the Euler Angles in Z-Y-X order, and $(q_1 ... q_{1L})$, $(q_1 ... q_{1R})$ are the generalized coordinates of the left and the right leg, respectively using which the kinematic modelling of Cassie is done. Figure 2 shows generalized coordinates of Cassie's pelvis, left and right leg in body frame, where left and right legs are similar.

TABLE I
TYPES OF LINKS AND JOINTS IN CASSIE

Link Number	Link Name	Joint Name	Joint Type	Parent Name
1	left_pelvis_abduction	fixed_left	fixed	pelvis
2	left_pelvis_rotation	hip_abduction_left	revolute	left_pelvis_abduction
3	left_hip	hip_rotation_left	revolute	left_pelvis_rotation
4	left_thigh	hip_flexion_left	revolute	left_hip
5	left_knee	knee_joint_left	revolute	left_thigh
6	left_shin	knee_to_shin_left	revolute	left_knee
7	left_tarsus	ankle_joint_left	revolute	left_shin
8	left_toe	toe_joint_left	revolute	left_tarsus
9	vectorNav	fixed_pelvis_to_vectorNav	fixed	pelvis
10	right_pelvis_abduction	fixed_right	fixed	pelvis
11	right_pelvis_rotation	hip_abduction_right	revolute	right_pelvis_abduction
12	right_hip	hip_rotation_right	revolute	right_pelvis_rotation
13	right_thigh	hip_flexion_right	revolute	right_hip
14	right_knee	knee_joint_right	revolute	right_thigh
15	right_shin	knee_to_shin_right	revolute	right_knee
16	right_tarsus	ankle_joint_right	revolute	right_shin
17	right_toe	toe_joint_right	revolute	right_tarsus

A. Kinematic Modelling of Cassie

Considering only one leg of the robot, the generalized coordinates of Cassie given by equation (2) and equation (3) can be written in matrix form as given by equation (5).

$$\begin{bmatrix} q_1 \\ q_2 \\ q_3 \\ q_4 \\ q_5 \\ q_6 \\ q_7 \end{bmatrix} = \begin{bmatrix} hip-roll \\ hip-yaw \\ hip-pitch \\ knee-pitch \\ shin-pitch \\ tarsus-pitch \\ toe-pitch \end{bmatrix} \tag{5}$$

Each leg of Cassie has seven DOF with five of them being actuated q_1, q_2, q_3, q_4, q_7 and the corresponding motor torques are u_1, u_2, u_3, u_4, u_5. The other two DOFs, q_5, q_6 are passive, corresponding to stiff springs.

Types of Links and Joints in Cassie: Each leg of Cassie has 8 joints, with one fixed and others revolute. The names of different links along with corresponding joint names, joint types and parent joint names are tabulated in Table I.

B. Forward Kinematics Methodology for Cassie

Using D-H conventions defined in [20], rotation matrix of each link with respect to the previous link is calculated and denoted by ${}^{i-1}_{i}[\mathrm{R}]$. Also, position vector of origin of each link with respect to the previous link ${}^{i-1}_{i}[\mathrm{O}]$ is calculated. Finally, the transformation matrix ${}^{i-1}_{i}[\mathrm{T}]$ of each link with respect to the previous link is calculated as given by equation (6).

$${}^{i-1}_{i}[\mathrm{T}] = \begin{pmatrix} {}^{i-1}_{i}[\mathrm{R}] & {}^{i-1}_{i}[\mathrm{O}] \\ & 1 \end{pmatrix} \tag{6}$$

Similarly, we obtain all the transformation matrices, i.e. ${}^{0}_{1}[\mathrm{T}]$, ${}^{1}_{2}[\mathrm{T}]$, ${}^{2}_{3}[\mathrm{T}]$, ${}^{3}_{4}[\mathrm{T}]$, ${}^{4}_{5}[\mathrm{T}]$, ${}^{5}_{6}[\mathrm{T}]$, ${}^{6}_{7}[\mathrm{T}]$, ${}^{7}_{8}[\mathrm{T}]$. Finally, the transformation matrix of the end-effector with respect to the inertial frame is obtained, which is given by ${}^{0}_{8}[\mathrm{T}]$.

In this paper, we focus on analysis of the left leg for which the end-effector is *left-toe*. The Euler angles ψ, θ, ϕ from ${}^{0}_{8}[\mathrm{T}]$ is obtained to get the roll, pitch and yaw angles of the end-effector with respect to the inertial frame. This gives us the orientation matrix $[\psi, \theta, \phi]$. Next, length vector of left toe given by ${}^{8}_{8}\vec{P} = \begin{bmatrix} P_x & P_y & P_z \end{bmatrix}^T$ is multiplied with the transformation matrix ${}^{0}_{8}[\mathrm{T}]$ as given in equation (7) to get the position matrix of the end-effector with respect to the inertial frame.

$${}^{0}_{8}\vec{P} = {}^{0}_{8}[\mathrm{T}]\,{}^{8}_{8}\vec{P} \tag{7}$$

The position and orientation matrix obtained above are concatenated to get the final Pose Matrix as given by equation (8) which contains position and orientation of the end-effector for a given configuration of joint angles.

$$\vec{Pose} = \begin{bmatrix} P_x & P_y & P_z & \psi & \theta & \phi \end{bmatrix}^T \tag{8}$$

All of the above mentioned process are done for every configuration.

C. Inverse Kinematics (IK) Methodology for Cassie

We perform IK using seven neural networks, where each network is trained for a specific joint. The data obtained from FK is separated into train and test data. Using train data, we train the network where the network input is pose of the end-effector and expected output is the corresponding joint angle. We thus calculate the Mean Square Error for each joint angles. Once the training is completed, the model is tested using the test data. For IK analysis of Cassie, we have chosen RBF network which have been explained as follows:

Radial Basis Function (RBF) Networks: The RBF network has certain advantages such as local approximation, higher convergence and ability to process non-linear mapping and has been used to solve the IK problem of the MOTOMAN manipulator [3].

In RBF network [21], the hidden layer, consists of a basis function, like Gaussian function as shown in equation (9) and output layer consists of a linear function. The basis function depends on the distance from center vector (centroid) and is

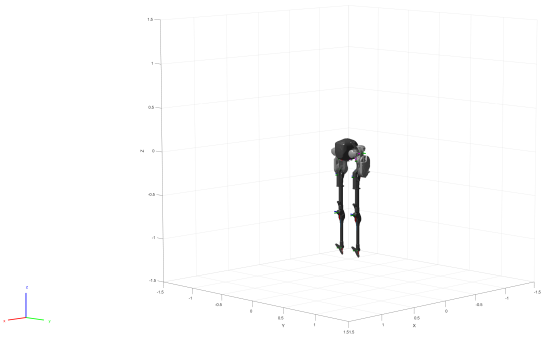

Fig. 3. Cassie model imported in MATLAB

TABLE II
GPU SPECIFICATIONS

Parameters	Values
MaxThreadsPerBlock	1024
MaxGridSize	[2.1475e+09 65535 65535]
MaxThreadBlockSize	[1024 1024 64]
SIMDWidth	32
MultiProcessorCount	5
ClockRateKHz	112400

radially symmetric about that vector. Due to local approximation, we get the strongest output when the input signals are near the centroid of the basis function.

$$R_i(x) = exp\left[\frac{-\|x - c_i\|^2}{2\sigma_i^2}\right]; i = 1, 2...m \tag{9}$$

where *x* is a vector of dimension *n*; c_i is the center of the i^{th} basis function of dimension *n*; σ_i is the standard deviation of i^{th} variable; $||x\text{-}c_i||$ is the norm of the vector x - c_i; m is the number of neurons; $R_i(x)$ gets the maximum at c_i, and with the increase of $||x\text{-}c_i||$, $R_i(x)$ decreases rapidly to zero. Given the input $x \epsilon R^n$ n, only a small section of inputs near *x* are activated. The nonlinear mapping $x \rightarrow R_i(x)$ is implemented in input layer whereas the linear mapping $R_i(x) \rightarrow y_k$ is implemented in output layer as given by equation (10),

$$y_k = \sum_{i=1}^{m} \omega_{ik} R_i(x); k = 1, 2...p \tag{10}$$

where p is the number of output neurons.

IV. SIMULATION

A. Setup

The simulation environment has been setup in MATLAB R2020b and uses the Robotics Toolbox, Parallel Computing Toolbox and Deep Learning Toolbox. The simulation ran on a PC with Intel(R) Core(TM) i5-7200U 2.50GHz CPU and 8GB RAM. The GPU used for training and inferencing the RBF network is Nvidia GeForce GTX 950M with specifications detailed in Table II.

The URDF model of Cassie [22] is imported in the MATLAB environment and the pose of end-effector (toe) is calculated using forward kinematics. According to our requirements, each joint angle is determined as given in equation (11) and equation (12).

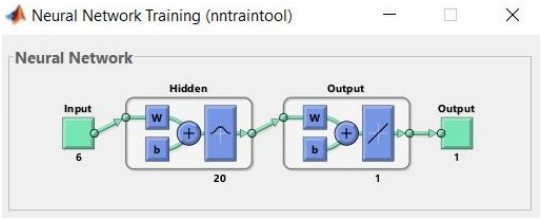

Fig. 4. Neural network setup in MATLAB

TABLE III
RBF NETWORK PARAMETERS AND TRAINING CONFIGURATION

Networks	No. of layers	Neurons in hidden layer 1	Neurons in hidden layer 2
Type 1	1	10	0
Type 2	1	30	0
Type 3	2	10	10
Type 4	2	30	10
Type 5	2	30	30

$$-5^\circ \leq q_1, q_7 \leq 10^\circ, 8^\circ \leq q_2 \leq 15^\circ; \tag{11}$$

$$10^\circ \leq q_3, q_4 \leq 20^\circ, -5^\circ \leq q_5, q_6 \leq 20^\circ \tag{12}$$

B. Dataset Generation

For our experiments, we have generated 2 datasets of size 279,936 (6^7) and 823,543 (7^7), the latter being close to 1 million. We have adopted two techniques for generating the joint angle values, equal interval data (linspace) and random interval data (randi). Given the number of points *n* from the given range *R* = [q_1, q_2], the former returns a row vector of *n* evenly spaced data points from *R* whereas the latter picks up *n* random data points from *R*.

C. Training using RBF

In order to perform inverse kinematics, we train 7 independent RBF networks for each of the joint angles. The training is done on GPU, specified in Table II with training function *Scaled Conjugate Gradient (SCG)* and the results are compared with training on CPU using training function *Bayesian Regularisation (BR)*. We use the Neural Network Training Tool (nntraintool) provided by MATLAB that opens the neural network training GUI.

Figure 4 shows that the RBF network consists of one input layer, hidden layer and output layer each. The input layer consists of 6 neurons which takes values from the pose of the end-effector, 20 neurons in the hidden layer with activation function as Gaussian function, and the output layer has one neuron with linear activation. There are 7 such networks for each joint of Cassie with network configurations as specified in Table III.

The training stops when either of the following conditions is met.

TABLE IV
MEAN SQUARED ERROR VARIATION WITH DATASETS FOR DIFFERENT RBF NETWORKS

Dataset	Training Function	Configuration	q_1 (°)	q_2 (°)	q_3 (°)	q_4 (°)	q_5 (°)	q_6 (°)	q_7 (°)
8R	SCG	Type 1	0.0128	0.0128	0.1718	0.1718	0.211	0.194	0.024
		Type 2	0.0129	0.0124	0.166	0.166	0.2119	0.183	7.33e-03
		Type 3	0.0129	0.0125	0.1718	0.166	0.2	0.166	0.024
		Type 4	0.0129	0.0123	0.1718	0.173	0.194	0.1604	0.0222
		Type 5	0.013	0.0125	0.167	0.166	0.200	0.1615	0.021
	BR	Type 1	4.41e-08	9.24e-08	0.068	0.0132	0.131	0.063	0.0186
		Type 2	1.86e-08	2.48e-08	0.063	0.0119	0.126	0.057	0.017
		Type 3	2.189e-07	1.12e-07	0.063	0.011	0.12	0.052	0.015
8E	SCG	Type 1	4e-05	5.38e-05	0.223	0.103	0.418	0.1718	0.0475
		Type 2	1.27 e-05	3.35 e-05	0.223	0.097	0.4125	0.166	0.047
		Type 3	2.69e-05	6.26e-05	0.229	0.097	0.418	0.166	0.043
		Type 4	1.62e-05	4.69e-05	0.223	0.0922	0.418	0.16	0.0428
		Type 5	1.24e-05	3.9e-05	0.223	0.0916	0.406	0.16	0.043
	BR	Type 1	2.81e-07	6.03e-08	0.223	0.0916	0.395	0.16	0.042
		Type 2	3.87e-08	2.81e-08	0.2177	0.085	0.383	0.16	0.042
		Type 3	3.48e-08	7.74e-08	0.2177	0.085	0.383	0.16	0.041
3R	SCG	Type 2	4.2e-06	1.04e-05	0.0426	0.05	0.223	0.052	0.012
	BR	Type 2	4e-08	3.08e-08	0.027	0.033	0.16	0.037	8.95e-03

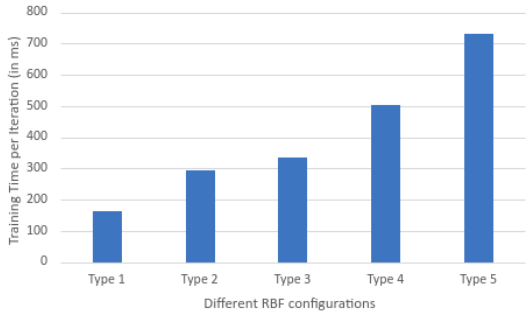

Fig. 5. Training Time per iteration for ∼1 million dataset for the networks discussed in Table III on GPU

- Maximum number of iterations is achieved.
- Minimum gradient is reached.
- Validation check is done.

We have set the maximum number of iterations to 500 to avoid over-training of data.

D. Results and Analysis

Table IV summarizes the performance (mean squared error) achieved by the networks for seven joint angles. We have generated three datasets for our analysis, *3R* refers to the dataset of size 279,936 configurations with random interval distribution, *8E* and *8R* refers to the dataset of size 823,543 configurations with equal and random interval distribution respectively.

1) Performance variation with dataset size: The results obtained by taking 279,936 configuration values (3R type 2) is more promising when compared to 823,543 configuration values (8R type 2) for all the seven networks i.e. joint angles. This could be derived from the fact that huge datasets that do not have many features can add to the noise. The network ends up memorizing the data and causes overfitting which results in an increase in the error.

2) Performance variation with data distribution: Taking random and even data distribution do not show any trend in error for a specific training function. In case of SCG, for some joint angles, taking randomly distributed dataset gives more accurate result as can be seen in the case of q_3, q_5 and q_7 whereas for joint angles q_1, q_2, q_4 and q_6, taking evenly distributed dataset gives accurate results.

3) Performance variation with network parameters: In case of *8R*, increasing the number of neurons in hidden layers does not show significant error reduction, rather increases computational time as can be visualised in Fig. 5. In contrast, error decreases by a small amount on increasing the number of neurons in each layer from *10* to *30* for 8E. Increasing number of layers from Type 1 to Type 3 and Type 2 to Type 5 gives same error for each joint angles for *8R* whereas a slight decrease in error for *8E*.

4) Performance variation with training function: Considering datasets with random data distribution i.e. *8R* and *3R*, BR performs better than SCG. BR is a robust algorithm that converts nonlinear regression into a statistical problem and reduces the need for lengthy cross-validations. On the contrary, SCG is based on conjugate directions, but this algorithm does not perform a line search at each iteration unlike other conjugate gradient algorithms which require a line search at each iteration which making the system computationally expensive. A comparative study of these algorithms has been done in [23].

The configurations achieved using the predicted joint angles from RBF networks is visually compared with the configurations achieved using the dataset joint angles in Fig. 6, where left and right half shows the predicted and actual configuration respectively. The configurations look similar in both the cases, which complements our results in Table IV. The trained RBF networks suffices our requirement for the position of end-effector in the range [0.30387 0.778]m, [0.32079 0.35989]m and [-1.0315 -0.63184]m for *x, y* and *z*

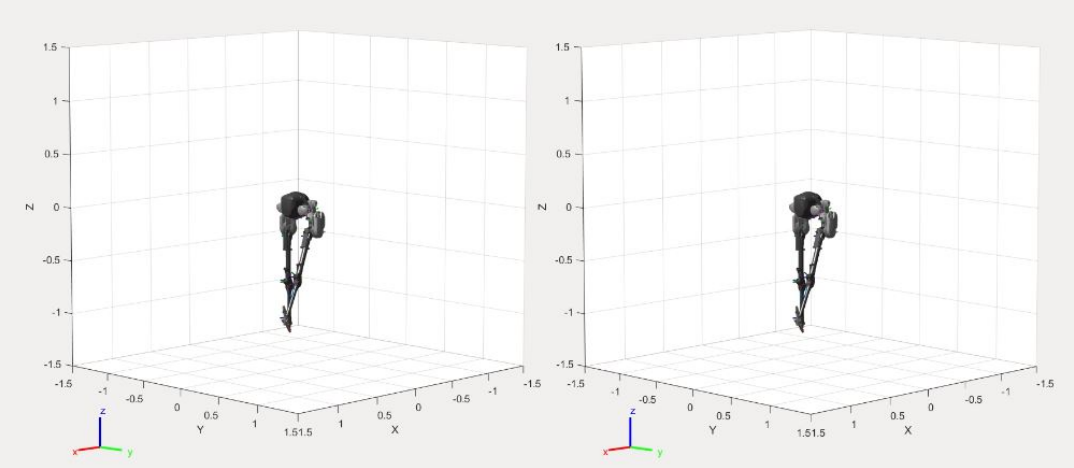

Fig. 6. Visual Comparison of Cassie Configuration with RBF networks outputs

co-ordinates respectively.

V. Conclusions

In this article, we have discussed the forward kinematics methodology and provided a neural network based approach to solve the inverse kinematics problem for Cassie robot. RBF network is selected because of its ability to process non-linear mapping. Going with Gaussian basis function for each neuron provides a better precision and faster convergence. We have trained the model with different number of hidden layers, training functions, number of neurons in hidden layers, dataset size and type of data distribution to analyse the effect of such parameters on our neural network by calculating MSE for every joint angle and configuration in IV-D. Our future work will build on the results that we have developed in this paper, to compare with other neural networks, enhance the dataset and calculate the corresponding joint torques to be used as a feedback controller.

References

[1] Avinash Siravuru. *Geometric Control and Learning for Dynamic Legged Robots*. PhD thesis, Carnegie Mellon University, 2020.

[2] HY Lee, C Woernle, and M Hiller. A complete solution for the inverse kinematic problem of the general 6r robot manipulator. 1991.

[3] Pei-Yan Zhang, Tian-Sheng Lü, and Li-Bo Song. Rbf networks-based inverse kinematics of 6r manipulator. *The International Journal of Advanced Manufacturing Technology*, 26(1-2):144–147, 2005.

[4] Cassie: Agility's first product 2017-2019.

[5] IEEE Cassie Photos taken as Agility Robotics/Oregon State University.

[6] Fei Liu, Guanbin Gao, Lei Shi, and Yongfeng Lv. Kinematic analysis and simulation of a 3-dof robotic manipulator. In *2017 3rd International Conference on Computational Intelligence & Communication Technology (CICT)*, pages 1–5. IEEE, 2017.

[7] Syed Baqar Hussain, Farah Kanwal, et al. Design of a 3 dof robotic arm. In *2016 Sixth International Conference on Innovative Computing Technology (INTECH)*, pages 145–149. IEEE, 2016.

[8] Austin Gregg-Smith and Walterio W Mayol-Cuevas. Inverse kinematics and design of a novel 6-dof handheld robot arm. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2102–2109. IEEE, 2016.

[9] KangKyu Lee, Jaesung Oh, Okkee Sim, Hyoin Bae, and Jun-Ho Oh. Inverse kinematics with strict nonholonomic constraints on mobile manipulator. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2469–2474. IEEE, 2017.

[10] Giresh K Singh and Jonathan Claassens. An analytical solution for the inverse kinematics of a redundant 7dof manipulator with link offsets. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2976–2982. IEEE, 2010.

[11] Gaurav Tevatia and Stefan Schaal. Inverse kinematics for humanoid robots. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 1, pages 294–299. IEEE, 2000.

[12] Alain Liegeois et al. Automatic supervisory control of the configuration and behavior of multibody mechanisms. *IEEE transactions on systems, man, and cybernetics*, 7(12):868–871, 1977.

[13] Dinesh Manocha and John F Canny. Efficient inverse kinematics for general 6r manipulators. *IEEE transactions on robotics and automation*, 10(5):648–657, 1994.

[14] Yugui Yang, Guangzheng Peng, Yifeng Wang, and Hongli Zhang. A new solution for inverse kinematics of 7-dof manipulator based on neural network. In *2007 IEEE International Conference on Automation and Logistics*, pages 1958–1962. IEEE, 2007.

[15] Morteza Alebooyeh and R Jill Urbanic. Neural network model for identifying workspace, forward and inverse kinematics of the 7-dof yumi 14000 abb collaborative robot. *IFAC-PapersOnLine*, 52(10):176–181, 2019.

[16] Raşit Köker, Tarık Çakar, and Yavuz Sari. A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators. *Engineering with Computers*, 30(4):641–649, 2014.

[17] RaşIt KöKer. A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization. *Information Sciences*, 222:528–543, 2013.

[18] Ahmed RJ Almusawi, L Canan Dülger, and Sadettin Kapucu. A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242). *Computational intelligence and neuroscience*, 2016, 2016.

[19] Yukai Gong, Ross Hartley, Xingye Da, Ayonga Hereid, Omar Harib, Jiunn-Kai Huang, and Jessy Grizzle. Feedback control of a cassie bipedal robot: Walking, standing, and riding a segway. In *2019 American Control Conference (ACC)*, pages 4559–4566. IEEE, 2019.

[20] F Merat. Introduction to robotics: Mechanics and control. *IEEE Journal on Robotics and Automation*, 3(2):166–166, 1987.

[21] X Wen, L Zhou, and DL Wang. Matlab neural networks design and application. *Science, Beijing*, 2000.

[22] Bruce Huang Ross Hartley, Ayonga Hereid. URDF model of CASSIE robot, Dec 2018.

[23] Ali Al Bataineh and Devinder Kaur. A comparative study of different curve fitting algorithms in artificial neural network using housing dataset. In *NAECON 2018 - IEEE National Aerospace and Electronics Conference*, pages 174–178, 2018.