A research on inverse kinematics solution of 6-DOF robot with offset-wrist based on Adaboost Neural Network

Qun Shi and Jiajun Xie
School of Mechatronic Engineering and Automation
Shanghai University
Shanghai - 201900, China
Email: jiajun.xie.sh@gmail.com

***Abstract*—Adaboost has been verified as an effective algorithm that improves the performance of weak learning algorithms which are slightly better than random guessing. In this paper, we use Adaboost based feed forward neural network to solve the inverse kinematics of 6 Degrees-of-Freedom (DOF) robot with offset-wrist. The algorithm aims to overcome the dilemma that most robots with offset wrists can only be solved by numerical methods in low efficiency. Experiments indicate the algorithm has a good performance in both accuracy and stability by reducing more than 70% of the average error and over 80% of the variance, and the generalization ability of the strategy is quite effective by testing through severe cases.**

I. Introduction

In recent years, Neural Network based solutions [3], [4], [5] have attracted considerable attention and could overcome the inverse kinematics in robots with offset-wrists. Neural Networks have very strong ability of learning, optimization and mapping, they have been successfully used in many areas, such as image processing, signal processing and nonlinear optimization,etc. The application of NN to inverse kinematics of robot is to break through the limitations of some traditional methods. Applying Neural Networks to the inverse kinematics problem has many advantages. The trained network will be in high efficiency regardless of the kinematics structure and it could avoid the singularity in robotics. Its shortcoming is also quite obvious that the training of the network could be very hard for achieving high precision. The reason is that the results extremely dependent on the data sampling which results in the limitation of its generalization ability. These problems make it hard to take advantage of neural networks.

Researchers have made efforts to apply NNs to the solution of inverse kinematics. Most of the research focus on the application of feed forward NN, which has the advantage of clear structure and strong mapping ability, but the disadvantage is its slow convergence speed, local minimum and the difficulty in deciding the number of hidden layers and nodes which is always decided by personal experience. Many researchers have carried on the experimental analysis to different structure of such vanilla NNs in inverse kinematics[6], [7], [8], [9], but they did not pay attention to the data sets which led to the lack of accuracy and generalization ability. Structure of other networks such as RBF NN[10], Elman NN[11] have been proposed to solve inverse kinematics problems, but they all have their own disadvantages. Though RBF NN is fast in training speed and good in mapping ability, it has much worse performance with large training set, and couldn't be used in a wide range of robot motion space fitting task. The lacking of data will lead to unacceptable error and fluctuation when using Elman NN. On the other hand, poor generalization ability made Elman NN difficult to meet the practical application requirements. Many new methods such as using fusion approach have been announced, which makes use of RBF NN for prediction[12], taking advantage of back propagation(BP) algorithm to solve the IK problem[13], these methods have obtained certain breakthrough but they're similar to iterative approaches, so they sacrificed the NN's efficiency. Some methods change the training inputs, such as[14] using the current angles as inputs, this method has high accuracy, but it can only be used in trajectory tracking and talked nothing about its generalization ability. Some scholars take advantage of heuristic algorithm, such as genetic and simulated annealing algorithm is introduced[4], [15], [16], but these methods always need a long training time and not always reliable.

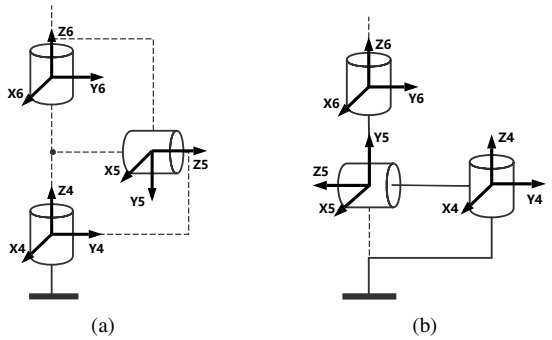


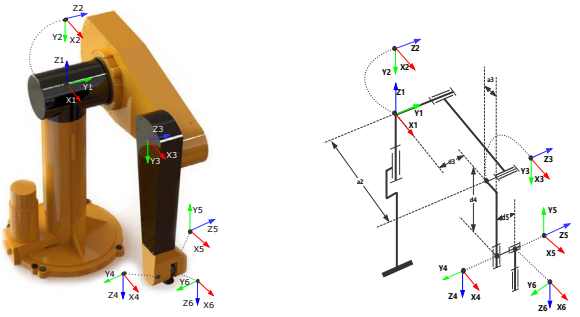
Fig. 1. Euler Wrist(a) & Cross Wrist(b)

The above shows that using neural networks in inverse kinematics problems has been attempted for several years, and have achieved a certain progress, but it has not been able to get a wide range of application, and the network's stability and generalization ability should be further considered. If the method wants to keep a good generalization ability, the

978-1-5386-3135-5/17/$31.00 ©2017 IEEE

TABLE I
MDH PARAMETERS OF PUMA560 WITH OFFSET-WRIST

$Joint_i$	MDH Parameters			
	$\theta_i(rad)$	$d_i(m)$	$a_{i-1}(m)$	$\alpha_{i-1}(rad)$
i=1	θ_1	0	0	0
i=2	θ_2	0	0	-pi/2
i=3	θ_3	d_3	a_2	0
i=4	θ_4	d_4	a_3	-pi/2
i=5	θ_5	d_5	0	pi/2
i=6	θ_6	0	0	-pi/2



(a) (b)

Fig. 2. Coordinate Distribution

training set should be larger, so the training is time-consuming and the convergence will be hard.

In this paper, experiments are based on 6 DOF serial robot with offset-wrist. In robot's trajectory tracking experiment, we analysize the accuracy and generation ability. What's more, the Adaboost Neural Network(ANN) is introduced in the inverse kinematic problem, experiments indicate that this method greatly increased the accuracy and stability of the prediction without increasing the size of sampling data, and the boosted network has a good ability in generalization by testing in different sampling sets. So the solution greatly promotes the NNs' application in the field of inverse kinematics, and could overcome the inverse kinematics problem of 6 DOF robots with offset-wrist. All the experiments introduced in this paper are based on Matlab Robotic toolbox[17], which is stable and ensures the reproducibility of the experiments.

II. KINEMATIC ANALYSIS OF ROBOTIC MANIPULATORS

A manipulator is composed of serial links that are connected to each other with revolute or prismatic joints from the base to the end-effector. In this paper, modified Denavit-Hartenberg model[18] is used to describe the robot's kinematics structure, Fig.2(b) shows the coordinate distribution of puma560 with offset-wrist. The modified D-H parameters are shown in table.I, some parameters inside it are $a_2 = 0.4318, a_3 = 0.0203, d_3 = 0.15, d_4 = 0.4318, d_5 = 0.15$ and could be found in Fig.2(b), the same coordinate distributions are shown in Fig.2(a) which is a 3-D model.

With MDH model, the general transformation matrix for a single link can be obtained as (1), "R" represents rotation and "D" represents translation, the offset represent the axis to be operated.

$$ {}^{i-1}_{i}T = R_X(\alpha_{i-1})D_X(a_{i-1})R_Z(\theta_i)D_X(d_i) \quad (1) $$

TABLE II
JOINT RANGES OF PUMA560 WITH OFFSET-WRIST

	Joint Angles (rad)					
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
min	-2.792	-3.926	-0.785	-1.919	-1.745	-4.642
max	2.792	0.785	3.926	2.967	1.745	4.642

The transformation matrix can be further calculated as(2). Inside the equation, "s" means "sin" and "c" means "cos".

$$ {}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & \alpha_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2) $$

Then the end-effector's position and orientation can be represented in the world coordinate as shown in(3)

$$ \begin{aligned} T_{end-effector} &= {}^0_6T = {}^0_1T{}^1_2T{}^2_3T{}^3_4T{}^4_5T{}^5_6T \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} & p_x \\ r_{21} & r_{22} & r_{23} & p_y \\ r_{31} & r_{32} & r_{33} & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \end{aligned} \quad (3) $$

From equation(3), we get the posture of the end-effector, p_x, p_y, p_z represent the position and $r_{11} - r_{33}$ represent the orientation, we can use RPY(roll-pitch-yaw) to replace these nine parameters, the RPY makes the orientation be more intuitive. What's more, this method reduces two-thirds the amount of data, thus greatly reduced training times of neural networks. A specific group of angles could be chosen from table.II to calculate the posture of the end effector in the Cartesian coordinate system.

Then forward kinematics solutions can be described as (4).

$$ F_{fk}(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) = (X, Y, Z, \phi_x, \phi_y, \phi_z) \quad (4) $$

And the inverse kinematics can be clearly described as (5).

$$ F_{ik}(X, Y, Z, \phi_x, \phi_y, \phi_z) = (\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6) \quad (5) $$

III. PROPOSED SOLUTION SYSTEM

Adaboost, short for "Adaptive Boosting", is a machine learning meta-algorithm formulated by Yoav Freund and Robert Schapire. It can be used in conjunction with many other types of learning algorithms to improve their performance. Individual learners can be weak, but the final model can be proven to converge to a strong learner[19]. Adaboost has been widely used in classification problems, but it will be used in prediction in this paper.

The main idea of Adaboost used in prediction is that several weak predictors merged into a strong predictor to provide a more efficient performance. The main steps are shown as follows: Firstly, decide the model of weak predictors and the sample space. Secondly, select m groups of training data from the sample space and set all the weights of training data to

$1/m$. Then train n weak predictors, and update the weights of the training data depending on each predictor's performance. After several times of iteration, a sequence of weights of the predictors $w_1, w_2, w_3, ..., w_n$ can be calculated from the weights of the training data. The detailed algorithm is shown in algorithm.1

Algorithm 1 ANN training Algorithm

Obtain $\{(x_1, y_1), ..., (x_m, y_m)\}$ *from samples*
Normalize x and y,then
each x_i has an associated $y_i \in \{-1, 1\}$
set (λ) *and* (ε)
Initial data weights $q(k,j)$ to$\frac{1}{m}, k = 1, 2, ..., N; j = 1, 2, ..., m$
while $k < N$ **do**
$wp(k) = weakpredictor(k) \Rightarrow train$
$p(k)$ = $wp(k) \Rightarrow predict$
for $j = 1$ *to* m **do**
$error(k,j)$=$\|p(k,j) - y(k,j)\|$
if $error(k,j) > \varepsilon$ **then**
$\alpha(k)$=$\alpha(k)$+$q(k,j)$
$q(k+1,j)$=$q(k,j) * \lambda$
else
$q(k+1,j)$=$q(k,j)$
end if
end for
$w(k)$=$0.5/exp(abs(\alpha(k)))$
$q(k+1) = q(k+1)/\sum_{j=1}^{m} q(k+1,j)$
end while
$w = w/\sum_{k=1}^{N} w(k)$
end
⁎ N is the number of weak predictors, λ means compensation and ε means threshold⁎

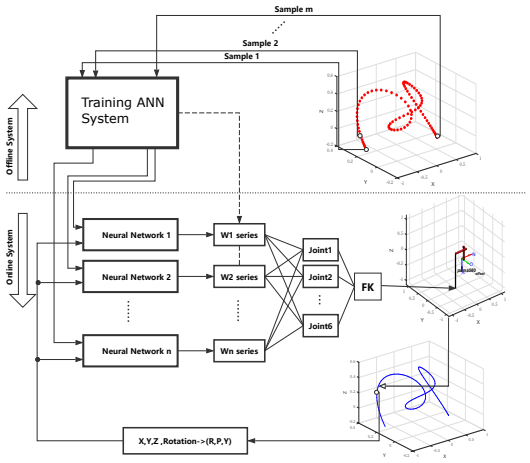


Fig. 3. ANN System

Fig.3 shows the ANN learning system in the experiment of trajectory tracking. The system contains two parts, the offline system and the online system. In the offline system, the main task is to achieve sparse samples from data, and use algorithm.1 to train the system. When finished, the algorithm will provide the weights of weak neural networks and the boost predictor. The online system of ANN shown in Fig.3 is used to solve the inverse kinematics problem. The trained feed forward NN works as weak predictors provide basic values to the desired position and posture of the end effector, and the weights multiply these basic values form the final angles of joints. For the first joint, the output would be $ANN_1(x, y, z, r, p, y) = w_1 p_1 + w_2 p_2 + ... + w_n p_n$. The last experiment will test the execution speed of the algorithm, analysis the end effector error and make comparison with the results of similar researches in recent years.

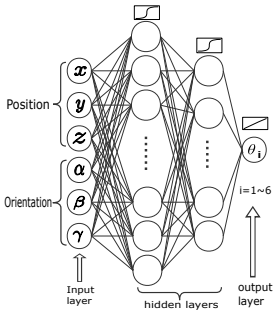


Fig. 4. Structure of weak predictors

TABLE III
START & END POSITION OF JOINTS (RAD)

	Joint Angles (rad)					
	θ_1	θ_2	θ_3	θ_4	θ_5	θ_6
start	-2.79	-3.9	-0.78	-1.4	-1.5	-1.6
end	2.79	0.78	3.9	1.6	1.6	2.0

IV. Experimental Analysis

A. Trajectory tracking experiment with ANN

Quintic polynomial trajectory planning (6) has been used for the experimental study. It is used with default zero boundary conditions for velocity and acceleration, so the $q_s, q_f, \dot{q}_s, \dot{q}_f, \ddot{q}_s, \ddot{q}_f$ are already known (the subscript "s" means "start" and "f" means "final").

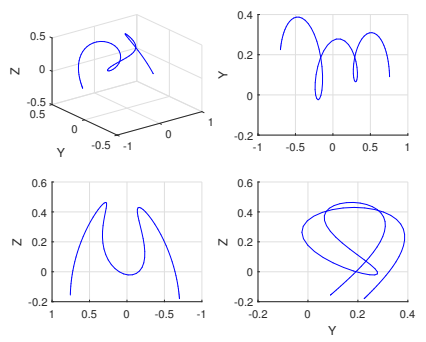
$$q(t) = a_0 + a_1 t + a_2 t^2 + a_3 t^3 + a_4 t^4 + a_5 t^5 \tag{6}$$

By a series of functions shown in (7), the coefficients in (6) could be solved. The start and the end of the joint angles are given in table.III

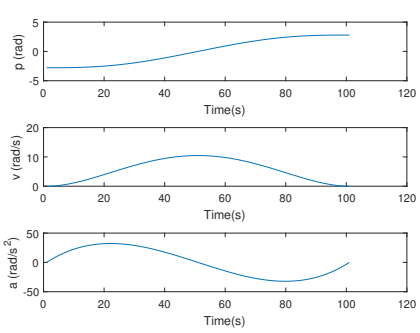
$$\begin{cases} q_s = a_0 + a_1 t_s + a_2 t_s^2 + a_3 t_s^3 + a_4 t_s^4 + a_5 t_s^5 \\ q_f = a_0 + a_1 t_f + a_2 t_f^2 + a_3 t_f^3 + a_4 t_f^4 + a_5 t_f^5 \\ \dot{q}_s = a_1 + 2a_2 t_s + 3a_3 t_s^2 + 4a_4 t_s^3 + 5a_5 t_s^4 \\ \dot{q}_f = a_1 + 2a_2 t_f + 3a_3 t_f^2 + 4a_4 t_f^3 + 5a_5 t_f^4 \\ \ddot{q}_s = 2a_2 + 6a_3 t_s + 12a_4 t_s^2 + 20a_5 t_s^3 \\ \ddot{q}_f = 2a_2 + 6a_3 t_f + 12a_4 t_f^2 + 20a_5 t_f^3 \end{cases} \tag{7}$$

Fig.5(a) shows different views of the trajectory. Fig.5(b) shows the angle, velocity and acceleration of the first joint at different times.

We want to get a kind of neural network to keep the ability of stable and high accuracy in solving the inverse kinematics



(a)



(b)

Fig. 5. Views Of The Trajectory & Angles Velocity Acceration(b)

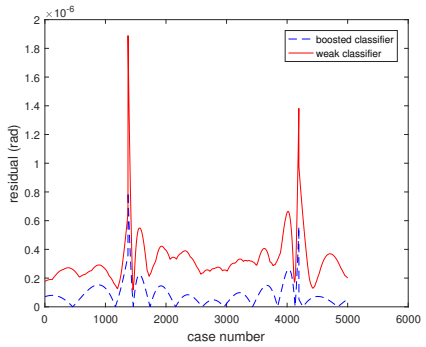
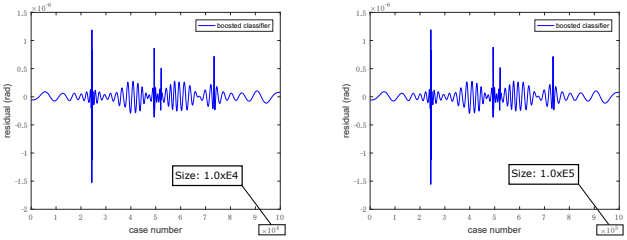


Fig. 6. Absolute residual analysis

problem, and it should have a good generalization ability. The ANN introduced in this paper is capable of conquering this problem.

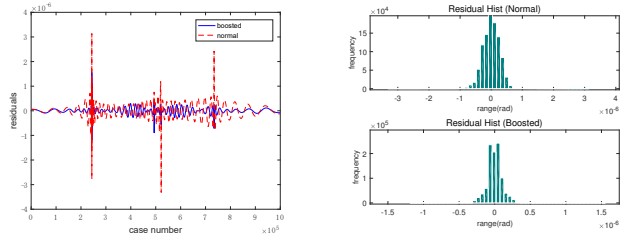
There are two experiments in this section. The first experiment is to track the trajectory with ANN and compare the prediction given by the strong predictor with the average prediction of the weak predictors. The second experiment is a further analysis to check the generalization ability of the network and make a comparison between ANN and normal BP network.

a) Experiment 1: In this experiment, ANN contains 10 weak predictors, each weak predictor is in the structure of "$10|8|$", the inputs of 6 parameters are $\{X, Y, Z, R, P, Y\}$, the output is the first joint. The trajectory is the same as Figure.5(a). The amount of collected samples is 10,000, the amount of training data are 7,000, the test and verification data



(a) (b)

Fig. 7. Residual analysis with 100,000 Samples(a) & Residual Analysis With 1,000,000 Samples(b) by ANN



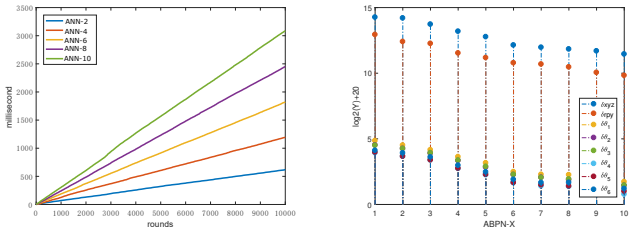
(a) (b)

Fig. 8. The Analysis(a) & The Histogram Comparison(b) Between ANN And Vanilla NN

are both 1,500. The training standard of each weak predictor sets MSE(mean square error) to 1.0e-14.

After prediction, the absolute residuals are shown in Fig.6. The solid red curve is the average absolute residual of ten weak predictors, and the blue dash curve is the absolute residual of the boosted predictor. The result indicates that ANN has a great improvement in both accuracy and stability compared with the average value of the weak predictions, and it is very effective in inhibition of mutations.

b) Experiment 2: Using the same ANN to predict the testing set which is 10 times the size of training set and more than 100 times of the real training data can been seen



(a) (b)

Fig. 9. Speed measurement(a) & End effector error analysis(b)

TABLE IV
SPEED MEASUREMENT OF ANN

Size	hidden layers and neurons		
	10	10|8	10|8|6
ANN-2	0.020844ms	0.041539ms	0.061935ms
ANN-4	0.039128ms	0.078719ms	0.119423ms
ANN-6	0.056742ms	0.118104ms	0.182168ms
ANN-8	0.075269ms	0.157575ms	0.245102ms
ANN-10	0.096252ms	0.195990ms	0.308681ms

TABLE V
END EFFECTOR ERROR ANALYSIS

	$\delta_{\theta_1}(rad)$	$\delta_{\theta_2}(rad)$	$\delta_{\theta_3}(rad)$	$\delta_{\theta_4}(rad)$	$\delta_{\theta_5}(rad)$	$\delta_{\theta_6}(rad)$	$\delta_{XYZ}(mm)$	$\delta_{RPY}(rad)$	promotion
ANN-1	2.750e-05	2.203e-05	2.157e-05	1.447e-05	1.497e-05	1.649e-05	1.872e-02	7.473e-03	—
ANN-2	2.173e-05	1.823e-05	1.821e-05	1.195e-05	1.212e-05	1.437e-05	1.789e-02	5.162e-03	+4.47%
ANN-3	1.695e-05	1.695e-05	1.448e-05	0.987e-05	0.998e-05	1.155e-05	1.286e-02	4.670e-03	+31.34%
ANN-4	1.182e-05	0.964e-05	0.984e-05	0.633e-05	0.641e-05	0.749e-05	0.889e-02	2.821e-03	+52.56%
ANN-5	0.854e-05	0.679e-05	0.708e-05	0.453e-05	0.464e-05	0.529e-05	0.666e-02	2.214e-03	+64.42%
ANN-6	0.534e-05	0.459e-05	0.462e-05	0.298e-05	0.302e-05	0.356e-05	0.429e-02	1.675e-03	+77.08%
ANN-7	0.464e-05	0.398e-05	0.394e-05	0.251e-05	0.268e-05	0.302e-05	0.381e-02	1.570e-03	+79.65%
ANN-8	0.455e-05	0.352e-05	0.363e-05	0.244e-05	0.251e-05	0.304e-05	0.348e-02	1.354e-03	+81.41%
ANN-9	0.434e-05	0.329e-05	0.348e-05	0.228e-05	0.216e-05	0.287e-05	0.315e-02	1.008e-03	+83.18%
ANN-10	0.315e-05	0.260e-05	0.262e-05	0.163e-05	0.191e-05	0.219e-05	0.267e-02	0.865e-03	+85.76%

TABLE VI
SYSTEM PERFORMANCE COMPARISON

Research	Platform	DOF	Method	Errors(%)	End effector errors(mm)	Execution speed(ms)
Proposed in this paper	PUMA560-Offset	6	ANN	-	0.00267	0.3
J.Raglend[20] 2016	Scorbot-ER Vu Plus	5	NARX	-	0.067	-
A. R. Almusawi[14] 2016	Denso	6	NN	x=0.17 y=0.36 z=0.12	-	-
Köker et al.[8] 2014	Hitachi M6100	6	NNCM	-	≈ 0.5	≈ 0.9
Luv et al.[21] 2014	PUMA560	6	NN	x=4.93 y=7.29 z=3.74	-	-
Köker[4] 2013	Stanford	6	Elman+GA	-	0.003	94
Hasan et al.[22] 2010	FANUC M-710i	6	NN	x=3.34 y=6.72 z=0.35	-	-

in Fig.7(a) and Fig.7(b), which indicates ANN has good generalization ability, and can be trained with little percentage of the data set to predict the testing set in much higher sampling frequency.

Fig.8(a) is a comparison between ANN and vanilla neural network, they are in the same MSE level. Through the comparison of data, ANN decreased 78.6% of the residual and 86.1% of the variance in this comparison, this is a great improvement. Fig.8(b) shows the frequency distribution histogram of normal multilayer neural network and ANN. Through the comparison between these histograms, it is clear that ANN has better average accuracy and variance, meantime they are in the same MSE and training in the same volume of data. This indicates that ANN can achieve better performance in inverse kinematics under the same conditions.

B. Performance testing and comparison

a) Execution speed testing: The execution speed is very important in robot control system. We test ANN algorithm with weak predictors in different size of hidden layers, and the quantity of weak predictors are 2,4,6,8,10 separately. We test each type of ANN for 10,000 rounds to get more accurate measurements. Fig.9(a) is the process of the test that the hidden layers are $10|8|6|$. All the mean execution speed with different structures are shown in Tab.IV. The table indicates that the execution speed decreased quickly with the increasing of hidden layers, and this tendency is nonlinear, for example, adding neurons to the third layer is more influential than adding neurons to the second layer. The algorithm is coded in Matlab and runs on the computer with Intel Core i7-6700HQ @2.60GHz. We can see that the execution speed of the algorithm is really fast, even we use ten weak predictors in the ANN system, the longest execution time is only 0.3 ms, which is acceptable for real time applications.

b) End effector error analysis: In the previous experiments, we put emphasis on the discussion of joint errors, and make it clear that ANN greatly increase the joints' accuracy and stability when solving inverse kinematics problems. With these joint angles, the position of the robot's end effector can be easily calculated through direct kinematics equations, and the position error can also be obtained by using a three-dimensional distance equation between the robot's end-effector and the target point, the Euclidean distance formula can be used in the distance computation as (8), the calculation method of δ_{RPY} is familiar to δ_{XYZ}.

$$\delta_{XYZ} = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2} \quad (8)$$

The experiment of testing the end effector error is based on motion space fitting, and the weak predictors' hidden layers used here are $10|8|6|$. Be careful that the output of each weak network should be one angle, if you try to directly output six angles, then the training process could be very tough. The performances of algorithm with different size of weak

predictors are shown in tab.V, the data is average level of 1000 tests for each kind of ANN system. Though the data is in very small digit, the last column of the table indicates that the enhancement is actually very large by comparing to ANN-1. What's more, the weak predictors in this algorithm could be replaced by any effective method and it will still have enhancement as tab.V shows, the related figure can been seen in fig.9(b), the x axis of this figure is the type of ANN system and the y axis is about joint errors and end effector errors, be careful that the y axis has been processed by log to give a more intuitive feeling of the data. Both tab.V and fig.9(b) indicate that if you want to keep both high accuracy and execution speed, ANN-6 may be a better choice than ANN-10, because ANN-6's execution speed is 0.18ms(shown in tab.IV) which improves the speed performance by 40% compared to ANN-10 and their enhancements are very close.

c) System performance comparison: Tab.VI shows the performance comparison among this research and some other related studies in recent years. Many researchers have made great efforts to the improvement of end effector's accuracy and execution time, but it is really hard to collect all the experiment data from various researches to make an absolutely fair comparison. Tab.VI shows that the algorithm demonstrated in this paper has achieved great progress comparing to other methods, the end-effector's error and execution speed are both acceptable.

V. Conclusion

The experiment of using ANN improves the stability and generation ability of Neural Networks' performance in solving inverse kinematics problems on the condition of limited training data volume. The comparison among different ANN structures and related researches in recent years indicates that the application of ANN in solving inverse kinematics problems have made great success.

The method introduced in this paper could effectively solve the inverse kinematics of 6 DOF robot with offset-wrist. The algorithm does not specify a certain type of robot, and the weak predictors used in ANN system does not limited to feed forward neural networks too. So the algorithm can still be promoted by enhancing the weak predictors and the code can run in parallel to get a much better performance.

References

[1] M. Raghavan and B. Roth, "Inverse kinematics of the general 6r manipulator and related linkages," *Journal of Mechanical Design*, vol. 115, no. 3, pp. 502–508, 1993.

[2] S. Kucuk and Z. Bingul, "Inverse kinematics solutions for industrial robot manipulators with offset wrists," *Applied Mathematical Modelling*, vol. 38, no. 78, pp. 1983 – 1999, 2014.

[3] A. T. Hasan, A. M. S. Hamouda, N. Ismail, and H. M. A. A. Al-Assadi, "An adaptive-learning algorithm to solve the inverse kinematics problem of a 6 dof serial robot manipulator," *Advances in Engineering Software*, vol. 37, no. 7, pp. 432–438, 2006.

[4] R. Köker, "A genetic algorithm approach to a neural-network-based inverse kinematics solution of robotic manipulators based on error minimization," *Information Sciences*, vol. 222, pp. 528 – 543, 2013.

[5] H. Toshani and M. Farrokhi, "Real-time inverse kinematics of redundant manipulators using neural networks and quadratic programming: A lyapunov-based approach," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 766–781, 2014.

[6] B. Karlik and S. Aydin, "An improved approach to the solution of inverse kinematics problems for robot manipulators," *Engineering Applications of Artificial Intelligence*, vol. 13, no. 2, pp. 159–164, 2000.

[7] R. Köker, C. öz, T. akar, and H. Ekiz, "A study of neural network based inverse kinematics solution for a three-joint robot," *Robotics and Autonomous Systems*, vol. 49, no. 3-4, pp. 227–234, 2004.

[8] R. Köker, T. akar, and Y. Sari, "A neural-network committee machine approach to the inverse kinematics problem solution of robotic manipulators," *Engineering with Computers*, vol. 30, no. 4, pp. 641–649, 2014.

[9] A. V. Duka, "Neural network based inverse kinematics solution for trajectory tracking of a robotic arm ," *Procedia Technology*, vol. 12, no. 1, pp. 20–27, 2014.

[10] P. Y. Zhang, T. S. L, and L. B. Song, "Rbf networks-based inverse kinematics of 6r manipulator," *The International Journal of Advanced Manufacturing Technology*, vol. 26, no. 1, pp. 144–147, 2005.

[11] R. Köker, "Reliability-based approach to the inverse kinematics solution of robots using elman's networks," *Engineering Applications of Artificial Intelligence*, vol. 18, no. 6, pp. 685–693, 2005.

[12] S. S. Chiddarwar and N. Ramesh Babu, "Comparison of rbf and mlp neural networks to solve inverse kinematic problem for 6r serial robot by a fusion approach," *Engineering Applications of Artificial Intelligence*, vol. 23, no. 7, pp. 1083–1092, 2010.

[13] P. Yuan, F. Su, Z. Shi, T. Wang, and D. Chen, "Autonomous path planning solution for industrial robot manipulator using backpropagation algorithm," *Advances in Mechanical Engineering*, vol. 7, no. 12, pp. 1–9, 2015.

[14] A. R. Almusawi, L. C. Dulger, and S. Kapucu, "A new artificial neural network approach in solving inverse kinematics of robotic arm (denso vp6242)," *Computational Intelligence and Neuroscience*, vol. 2016, pp. 1–10, 2016.

[15] R. Köker, "A neuro-simulated annealing approach to the inverse kinematics solution of redundant robotic manipulators," *Engineering with Computers*, vol. 29, no. 4, pp. 507–515, 2013.

[16] R. Köker and T. Çakar, "A neuro-genetic-simulated annealing approach to the inverse kinematics solution of robots: a simulation based study," *Engineering with Computers*, vol. 32, no. 4, pp. 1–13, 2016.

[17] P. I. Corke, *Robotics, Vision & Control: Fundamental Algorithms in Matlab*. Springer, 2011.

[18] B. Siciliano and O. Khatib, *Springer Handbook of Robotics*. Springer Berlin Heidelberg, 2009.

[19] Y. Freund, R. Schapire, and N. Abe, "A short introduction to boosting," *Journal of Japanese Society for Artificial Intelligence*, vol. 14, pp. 771–80, Sept. 1999.

[20] G. G. D. I.Jacob Raglend, M.Dev Anand and D. M. M. S. R. Prabha, "Inverse kinematics solution of a five joint robot using narx algorithm," *Journal of Chemical and Pharmaceutical Sciences*, vol. 9, no. 4, pp. 2677–2687, 2016.

[21] L. Aggarwal, K. Aggarwal, and R. J. Urbanic, "Use of artificial neural networks for the development of an inverse kinematic solution and visual identification of singularity zone(s) ," *Procedia Cirp*, vol. 17, no. 2, pp. 812–817, 2014.

[22] A. T. Hasan, N. Ismail, A. M. S. Hamouda, I. Aris, M. H. Marhaban, and H. M. A. A. Al-Assadi, "Artificial neural network-based kinematics jacobian solution for serial manipulator passing through singular configurations," *Advances in Engineering Software*, vol. 41, no. 2, pp. 359–367, 2010.