Trajectory Optimization of Spherical Parallel Robots using Artificial Neural Network

R. Alibakhshi*
Department of Mechanical Engineering,
Babol University of Technology, Iran
E-mail: sh_a_bakhshi@yahoo.com
*Corresponding author

H. R. Mohammadi Daniali
Department of Mechanical Engineering,
Babol University of Technology, Iran
E-mail: mohammadi@nit.ac.ir

Received: 24 January 2013, Revised: 14 May 2013, Accepted: 20 July 2013

Abstract: This article addresses an efficient and novel method for singularity-free path planning and obstacle avoidance of parallel manipulator based on neural networks. A modified 4-5-6-7 interpolating polynomial is used to plan a trajectory for a spherical parallel manipulator. The polynomial function which is smooth and continuous in displacement, velocity, acceleration and jerk is used to find a path avoiding obstacles and singularities. The polynomial is further modified to plan a trajectory with minimum passing length through the obstacle and singularity, and the best kinematics conditioning index, as well. An artificial neural network is implemented to solve forward kinematics of the manipulator to estimate the distance between gripper and singularity or obstacle in Euler coordinate. Moreover, the simulation results prove the efficiency of the proposed algorithm.

Keywords: Collision Avoidances, Neural Networks, Parallel Robots, Singularity

Reference: Alibakhshi, R., Mohammadi Daniali, H. R., "Trajectory Optimization of Spherical Parallel Robots using Artificial Neural Network", Int J of Advanced Design and Manufacturing Technology, Vol. 7/ No. 1, 2014, pp. 91-98.

Biographical notes: R. Alibakhshi has MSc degree in Mechanical Engineering from Babol Noshirvani University of Technology. His field of research is on parallel manipulators. **H. M. Daniali** is currently an Associate Professor in the Department of Mechanical Engineering, at Babol University of Technology. Dr. Daniali graduated as a Mechanical Engineer from Ferdosi University in 1965. He received his MSc from Tehran University in 1968 and his PhD degree in Mechanical Engineering from McGill University, Canada, in 1995. His research interests focus on theoretical kinematics and parallel manipulators.

1 INTRODUCTION

A parallel manipulator is a closed-loop mechanism in which a moving platform is connected to the base by several legs. This manipulator is faster, stiffer and more accurate than its serial counterpart. However, the closed-loop nature of a parallel manipulator limits its workspace and creates complex kinematics singularities within the workspace. Because of limited workspace coupled with singularities and obstacles, it is essential to have an algorithm that can plan a path free of singularities and obstacles inside the workspace.

Dasgupta and Mruthyunjaya have proposed an algorithm which finds safe via points and plans a continuous path (free-of-singularity) connecting two points [1]. Dash et al., presented a singularity-free path generation method upon finding out the reachable workspace of parallel manipulator [2]. Bazaz and Tondu studied minimum time on-line joint trajectory generator based on low order spline method for industrial manipulators [3]. Saramago and Junior optimized trajectory of a 3-DOF parallel manipulator in presence of moving obstacle [4-5]. They could minimize the distance between moving obstacle and gripper.

An artificial neural network is a massively parallel-distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. Boudreau et al., studied Parallel manipulator kinematics learning using neural network models [6]. Kim et al., analyzed an efficient type of neural network namely, cascade-correlation feed-forward [7]. Seyyedi parsa et al., optimized trajectory of 3-RRR planar parallel manipulator planned by 3-4-5 polynomial [8]. They added a new term to the predefined polynomial in order to plan a continuous and smooth path avoiding singularities and obstacles, but they did not consider continuity of the jerk.

This paper presents two algorithms, namely, solving forward kinematics problem (FKP) of parallel manipulators based on the data of its inverse kinematics (IK) and planning a smooth path to avoid singularities and obstacles. It is noteworthy that FKP is generally more complicated than inverse kinematics problem (IKP) in case of parallel manipulators. A trajectory based on 4-5-6-7 interpolating polynomial is planned, where a new term is added which has continuity in displacement, velocity, acceleration and jerk to avoid the possible singularity or obstacle.

Therefore, the motion undergone by desired trajectory is as smooth as possible; i.e., abrupt changes in position, velocity, acceleration and jerk are avoided. In order to estimate the distance between the gripper and any obstacle or singular point, a minimization algorithm is applied. It is possible to verify if collisions occur by solving forward kinematics (FK) for joint angles in every step of the trajectory and comparing the answer with singular points or obstacle positions.

2 SPHERICAL PARALLEL ROBOTS

As depicted in Fig. 1, spherical parallel manipulators (SPRs), which can provide three degree of freedom of rotation, have been applied for most applications such as camera-orienting. The Agile Eye is a 3-RRR spherical parallel robot in which the axes of all pairs of adjacent joints are orthogonal. This robot is also a high-performance mechanism capable of orienting a camera within a workspace larger than that of a human eye [9]. Its mobile platform is a moving equilateral triangle linked to a fixed equilateral triangle. The axes of the first base joint, the second intermediate joint, and the third platform joint have a common point in the centre of a sphere. A base reference frame, $O_{u_1u_2u_3}$, is selected in such a way that its u_1 axis is along the axis of the first base joint, its u_2 axis is along the axis of the second base joint, and its u_3 axis is along the axis of the third base joint, as shown in Fig. 1.

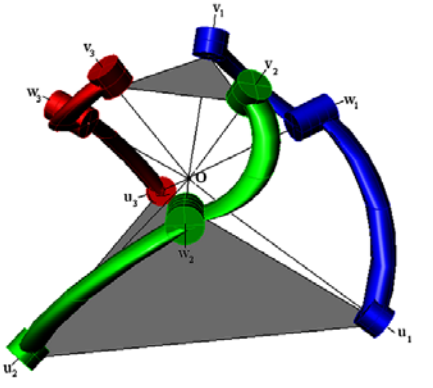

Fig. 1 Spherical parallel robot in general configuration

A mobile reference frame, $O_{v_1v_2v_3}$, is fixed at the mobile platform. The position of moving platform is defined by $\boldsymbol{v}_1, \boldsymbol{v}_2$ and $\boldsymbol{v}_3$ unit vectors. Gosselin and Gagne suggested three simple methods to solve forward kinematics of the spherical parallel manipulator and claimed that this manipulator admits at most eight solutions [10]. Bonev et al., derived the jacobian matrices of the Agile-Eye and studied its singularities [11]. The rotation matrix $\boldsymbol{R}$ describes the orientation of the mobile frame with respect to the base frame. The *ZYX* Euler-angle convention is used here because it simplifies the kinematics analysis. In the base frame, the axes of the platform joints are defined as:

$$\boldsymbol{v}_i^{Fixed-frame} = \boldsymbol{R}\boldsymbol{v}_i^{Moving-frame} \quad (1)$$

Where $\boldsymbol{v}_i^{Fixed-frame}$ and $\boldsymbol{v}_i^{Moving-frame}$ are unit vectors at the fixed and mobile frames, respectively. The IK is to determine the actuator joint angles $(\theta_1, \theta_2, \theta_3)$ given the pose of the end-effector (φ, θ, ψ) in the base frame. Here, the notation used by Bonev et al. is adopted [11]. For a given orientation of the mobile platform, each leg admits maximum two solutions for θ_i in $(-\pi, \pi]$ as:

$$\theta_1 = arctan\left(\frac{cos(\theta)sin(\psi)}{cos(\varphi)cos(\psi)+sin(\varphi)sin(\theta)sin(\psi)}\right)$$
$$\theta_2 = arctan\left(\frac{sin(\phi)sin(\psi)+cos(\phi)sin(\theta)cos(\psi)}{cos(\theta)cos(\psi)}\right) \quad (2)$$
$$\theta_3 = arctan(tan(\phi))$$

3 ARTIFICIAL NEURAL NETWORKS

Artificial neural networks (ANN) are composed of simple elements operating in parallel. They are most commonly used for prediction, pattern recognition, and nonlinear function fitting. Neural network can be trained to perform a specific function by adjusting the values of the weights between inputs and outputs. Commonly ANN can store the sample with distributed coding, thus forming a trainable nonlinear system. In this work, it is used the Cascade-feed forward algorithm for training process because of its advantages over Feed-forward back propagation algorithm [7]. Like Feed-forward networks, Cascade-forward networks uses back-propagation algorithm for updating of weights and biases but the main symptoms of this network is that each layer neurons connected to all previous layer neurons. It is proved that the cascade-forward neural network with back-propagation learning provides the best performance in terms of convergence time, optimum network structure and recognition performance. The structure of this network is shown in Fig. 2.

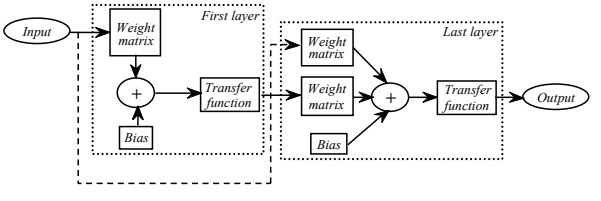

Fig. 2 Cascade-forward network general structure

4 FORWARD KINEMATICS SOLUTION VIA NEURAL NETWORKS

In this part, FKP of Agile-Eye is solved by ANN. The FKP deals with motion of end-effector of the robot according to the world coordinate system. For parallel manipulators, in general, this is a system of nonlinear algebraic equations. Here, ANN is implemented to solve the FKP of the Agile-Eye. First, some known points of the Euler-Angels are taken. By solving IKP, joint angels $(\theta_1, \theta_2, \theta_3)$ related to different (φ, θ, ψ) are computed. These values are recorded in a file to form the learning sets of neural network. With regard to the rule which claim that the IKP of the 3-RRR spherical manipulator admits eight sets of solutions, eight sets of training files are constructed [10]. Approximately 3,000 randomly selected positions of the gripper in Euler angles are used. About 75% of these data sets were used in training of the network, and the rest were used in testing. Eight multi-layer perceptron neural networks were designed separately and each one includes one hidden layer with 7 neurons and *Tansig* functions as transfer function. The learning rate set was 0.4 and momentum constant was 0.95 which are experimentally chosen.

The number of neurons for input and output layer is 20 and 3, respectively. The transfer function for input layer set *Tansig* and for output layer set *Purelin*. The back propagation algorithm was implemented, and the error at the end of learning process was bounded to 9.97717×10^{-5} for the training sets. Every set of IK answers was applied as a training set individually. So, eight neural networks are available for solving the FKP. In order to solve FKP, first the input data (motors' angles) must be labelled to a branch of IKP. Then, motors' angles will be presented to the neural network which was trained with that set of IKP data.

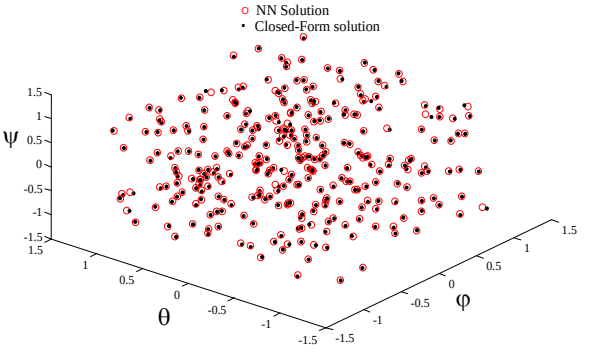

Fig. 3 Comparison between exact and ANN solutions in 3D (φ, θ, ψ)

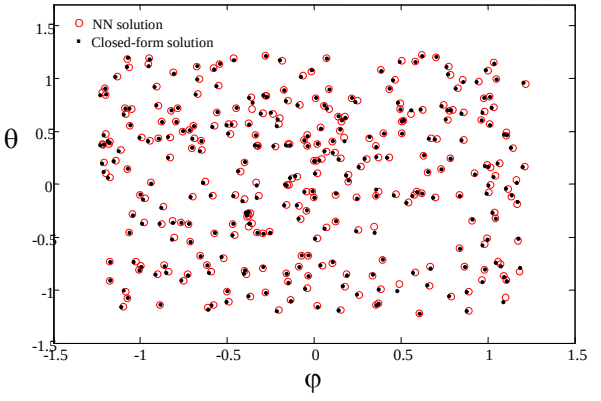

Fig. 4 Comparison between 2D exact and ANN solutions (φ,θ)

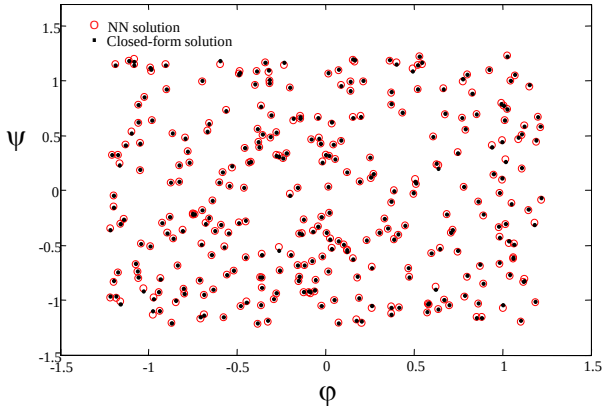

Fig. 5 Comparison between 2D exact and ANN solutions (φ,ψ)

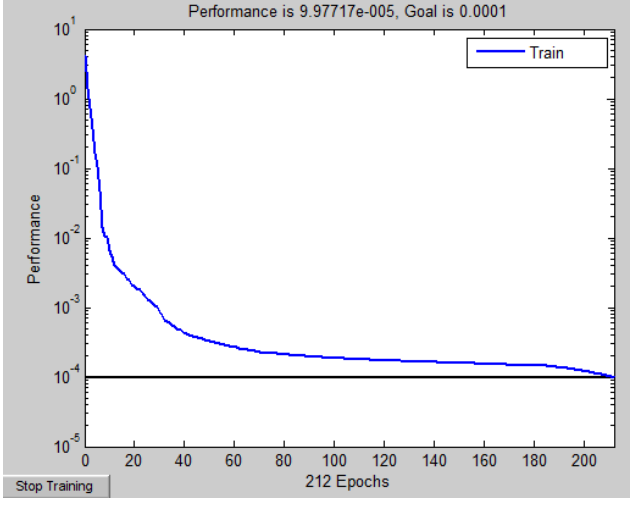

Fig. 6 The trained neural network performance

Finally, the trained networks may be tested by choosing some data which have not been used in the training process. After presenting the data to the completed neural network, the answers of the network are compared with those from the IKP as depicted in Fig. 3. Moreover, the projections of the data on the $(\varphi-\theta)$ and $(\varphi-\psi)$ planes as depicted in Figs. 4 and 5 show the accuracy of the trained network. Furthermore, the performance of the trained network is shown in Fig. 6 which shows its fast convergences.

5 SINGULARITY ANALYSIS

Algebraically, singularity deals with the rank deficiency of the associated Jacobian matrices. While geometrically, it is observed that the manipulator gains at least one additional uncontrollable degree of freedom or loses one or a few degrees of freedom in singular points. By considering the angle ϑ_1 between $\boldsymbol{v}_i$ and $\boldsymbol{w}_i$, it is suggested to obtain the Jacobian by differentiating the enclosure equation:

$$\boldsymbol{w_i}.\boldsymbol{v_i} = cos(\vartheta_1) \tag{3}$$

Which leads to:

$$\dot{\boldsymbol{w}}_{\boldsymbol{i}}.\boldsymbol{v_i} + \boldsymbol{w_i}.\dot{\boldsymbol{v}}_{\boldsymbol{i}} = 0 \tag{4}$$

Where:

$$\dot{\boldsymbol{v}}_{\boldsymbol{i}} = \boldsymbol{\omega}\times\boldsymbol{v_i}\ , \qquad \dot{\boldsymbol{w}}_{\boldsymbol{i}} = \dot{\boldsymbol{\theta}}_{\boldsymbol{i}}\cdot\boldsymbol{u_i}\times\boldsymbol{v_i} \tag{5}$$

Substituting the Eq. (5) into Eq. (4) yields:

$$\boldsymbol{J\omega} + \boldsymbol{K}\dot{\boldsymbol{\theta}} = 0 \tag{6}$$

Where, $\boldsymbol{\omega}$ and $\dot{\boldsymbol{\theta}}$ are the angular velocity of the mobile platform and the active-joint rates respectively. Although very simple, the jacobian matrices are obtained by:

$$\boldsymbol{J} = \begin{bmatrix} (\boldsymbol{w_1}\times\boldsymbol{v_1})^T \\ (\boldsymbol{w_2}\times\boldsymbol{v_2})^T \\ (\boldsymbol{w_3}\times\boldsymbol{v_3})^T \end{bmatrix},$$

$$\boldsymbol{K} = \begin{bmatrix} (\boldsymbol{w_1}\times\boldsymbol{v_1})^T\,\boldsymbol{u_1} & 0 & 0 \\ 0 & (\boldsymbol{w_2}\times\boldsymbol{v_2})^T\,\boldsymbol{u_2} & 0 \\ 0 & 0 & (\boldsymbol{w_3}\times\boldsymbol{v_3})^T\,\boldsymbol{u_3} \end{bmatrix} \tag{7}$$

In which $\boldsymbol{u}_i, \boldsymbol{v}_i$ and $\boldsymbol{w}_i$ are unit vectors in the fixed-frame. The singularities encountered in closed loop kinematics chain can be divided into three main groups [12]. The first type of singularity occurs whenever K is singular which leads to fully extended or folded leg. Moreover, this type of singularity occurs at the boundary of workspace. Type 2 singularity is specified by studying matrix J and occurs whenever the three vectors $(\boldsymbol{w}_i \times \boldsymbol{v}_i)$ for i=1, 2, 3, are coplanar or collinear. The third type of singularity occurs in the case that both types of the foregoing singularities occur simultaneously.

For the Agile-Eye, these vectors can not be collinear. So, when these three vectors are coplanar, the moving platform can rotate about the axis passing through the centre O and normal to the plane of these vectors, even if all the actuators are locked. Moreover, kinematic conditioning index (KCI) is defined as the inverse of the condition number. KCI is bounded between zero and one. It may be inferred that a higher KCI makes a matrix closer to the isotropic condition and a lower KCI makes it closer to singularity. If it is possible to avoid singularity, an optimal path may be found with high value of kinematic conditioning index. For manipulator at hand, all of its singular points are found by this method as shown in Fig. 7.

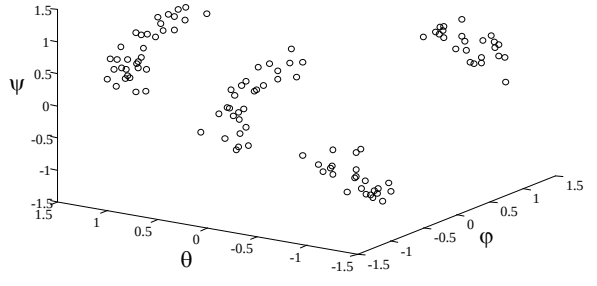

Fig. 7 Type II of singular points

6 TRAJECTORY OPTIMIZATION PROCEDURE

In this study, a path is planed between two given points in Eulerian coordinate. A 4-5-6-7 polynomial interpolation based method is applied to plan a trajectory in the joint space. It can be three-times differentiated and thus, ensures the smoothness of the trajectory. A smooth motion is defined using eight conditions between the initial and the final poses. Therefore, a seventh-order polynomial $S(\tau)$ has been chosen to study the trajectory planning as:

Where t, is the time counted from the initial pose, i.e., corresponding to t=0 and T is the implementation time. A normal polynomial that represents each of the joint variables $\boldsymbol{\theta}_j$ throughout its range of motion is defined by:

$$\{\boldsymbol{\theta}_j\} = \{\boldsymbol{\theta}_j\}^{Initial} + S(\tau)\left(\{\boldsymbol{\theta}_j\}^{Final} - \{\boldsymbol{\theta}_j\}^{Initial}\right) \tag{9}$$

Where $\{\boldsymbol{\theta}_j\}^{Initial}$ and $\{\boldsymbol{\theta}_j\}^{Final}$ are the given initial and final values of the j^{th} joint variable, respectively. It is thus possible to determine the evolution of each joint variable if both its end values and the time required to accomplish the motion are known. In the absence of singularities, then, the conditions of zero velocity, acceleration and jerk imply zero joint velocity, acceleration and jerk, respectively. With respect to the initial and final conditions, the normal polynomial found is:

$$S(\tau) = -20\tau^7 + 70\tau^6 - 84\tau^5 + 35\tau^4 \tag{10}$$

Equation (9) is used for trajectory planning without considering the singularities and obstacles. The polynomial trajectories discussed above only met the Eulerian trajectories prescribed at the initial and final instants. If collisions and singularities occur, the trajectory has to be changed so as to eliminate them, while keeping the trajectory as smooth as before. This proposed modification is done by adding the value of smooth and continuous displacement functions to the polynomial of Eq. (9). As mentioned earlier, Eq. (9) guarantees the continuity of displacement, velocity, acceleration and jerk. So, the new term should not disobey this rule. This term is considered as:

$$S^*(\tau^*) = L\begin{bmatrix} A\sin(\frac{\pi}{2}\tau^*) + B\cos(\frac{\pi}{2}\tau^*) + C\sin(\pi\tau^*) \\ + D\cos(\pi\tau^*) + E\sin(\frac{3\pi}{2}\tau^*) \\ + F\cos(\frac{3\pi}{2}\tau^*) + G\sin(2\pi\tau^*) \\ + H\cos(2\pi\tau^*) + I\tau^* \end{bmatrix} \tag{11}$$
$$\tau^* = t^*/T^*$$

Where, L is the amplitude of the displacement function, T^* is the total time for collision avoidance process, and t^* is the time step while obstacle avoidance is taking place. Moreover, L and T^* are constants depending on the sizes of the obstacles or singularities. The boundary conditions are also defined by:

$$S^*(0)=S^*(1)=0, S^*(\frac{1}{2})=\frac{1}{2}, (S^*)'(0)=(S^*)'(1)= \\ (S^*)''(0)=(S^*)''(1)=(S^*)'''(0)=(S^*)'''(1)=0 \quad (12)$$

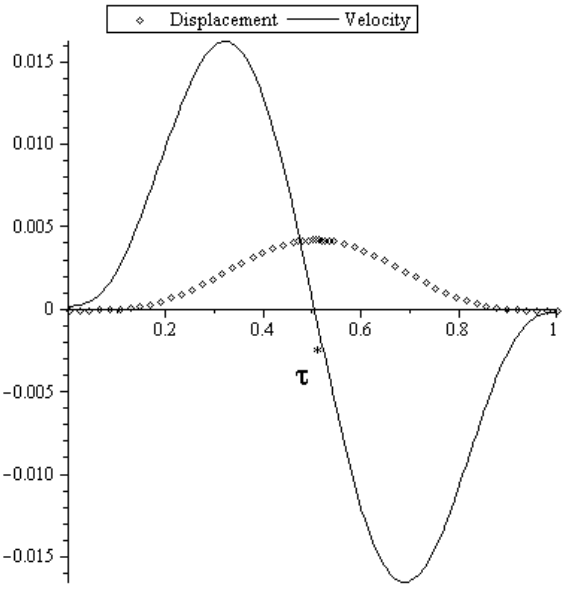

Fig. 8 Smooth displacement and velocity

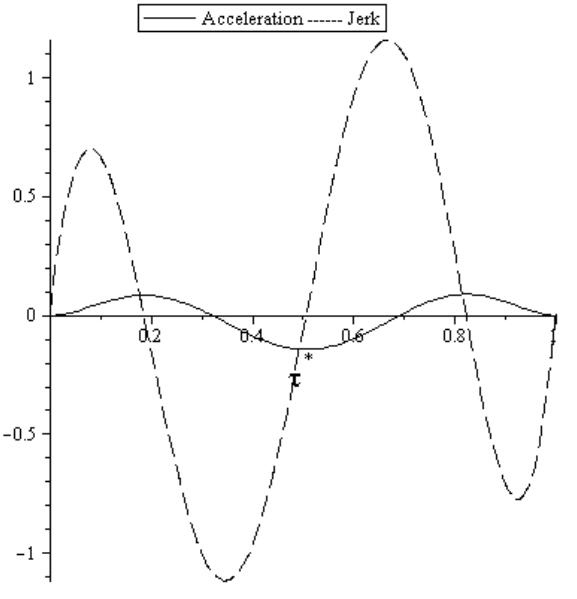

Fig. 9 Smooth acceleration and jerk

Figures 8 and 9 show the continuity of displacement, velocity, acceleration and jerk for the new proposed term. In order to avoid each obstacle, the goal here is to minimize the sum of the distances between the new trajectory and three steps as depicted in Fig. 10, i.e.,

$$\text{Minimize } F=|L1|+|L2|+|L3| \quad (13)$$

Where $|\bullet|$ denotes 2-norm.

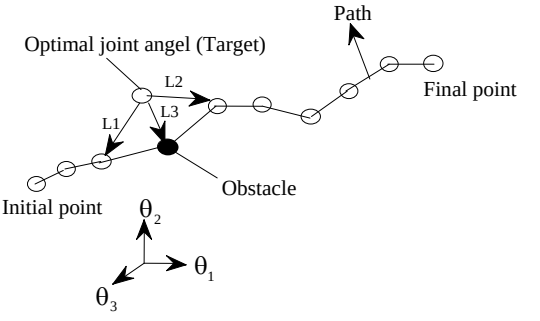

Fig. 10 Obstacle avoidance in planned trajectory

7 MINIMIZATION ALGORITHM

It is obvious that every minimization algorithm requires objective function and at least one constraint which may be linear or nonlinear equality or inequality. To specify this constraint, the definition of the plate is used in such away that its normal is along with the robot motion (i.e. tangent to the path) which consists the obstacle. In other words, the plate equations will be updated in each collision step and can be written as follows:

$$P: a_1(\theta_{1new}-\theta_{1obs})+b_1(\theta_{2new}-\theta_{2obs})+c_1(\theta_{3new}-\theta_{3obs})=0. \quad (14)$$

Where $\{a_1, b_1, c_1\}$ is the normal of the plate in obstacle coordinate and $\{\theta_{1new}, \theta_{2new}, \theta_{3new}\}$ is the optimal joint angel which is located in the plate that can be obtained by Eq. (14). First, a preliminary trajectory is planned from the initial to the final poses in the joint space, disregarding the obstacles and singularities. Then, it is verified if collisions or singularities occur. If this is the case, $S^*(\tau^*)$ is added to $S(\tau)$ as:

$$\{\boldsymbol{\theta_j}\}=\{\boldsymbol{\theta_j}\}^{Initial}+\left[S(\tau)+S^*(\tau^*)\right]\left(\{\boldsymbol{\theta_j}\}^{Final}-\{\boldsymbol{\theta_j}\}^{Initial}\right) \quad (15)$$

On the other hand, in general, it is possible to change every joint variable separately as the following:

$$\{\boldsymbol{\theta_j}\}=\{\boldsymbol{\theta_j}\}^{Initial}+\left[\begin{Bmatrix}1\\1\\1\end{Bmatrix}S(\tau)+\begin{Bmatrix}\alpha\\\beta\\\gamma\end{Bmatrix}S^*(\tau^*)\right]\left(\{\boldsymbol{\theta_j}\}^{Final}-\{\boldsymbol{\theta_j}\}^{Initial}\right) \quad (16)$$

Where α,β and γ are the parameters. *F* and *P* are both considered, which are composed of three variables α,β and γ, defined as an objective function and constraint equation, respectively. Here, minimization procedure is used which implements various numerical optimization Routines, including sequential quadratic programming algorithm to solve for constrained optima. This algorithm attempts to find a constrained minimum of a scalar function of several variables starting at an initial estimate. If any singular point or obstacle lies on or near the path, this path is restructured by an algorithm as demonstrated in Fig. 11.

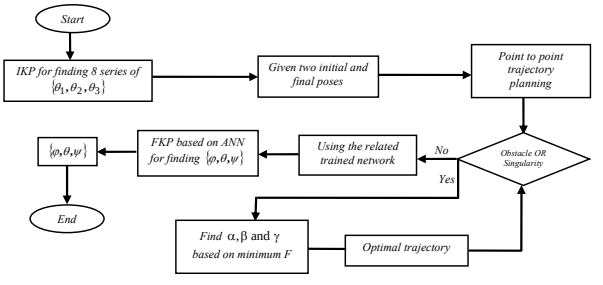

Fig. 11 Trajectory planning and machine learning procedure

As mentioned earlier, the proposed algorithm may find a path with minimum passing length and the best average of the KCI. If the time step is assumed as 'm', the summation of path length passing through the obstacle or singular point is defined as:

$$\text{Distance} = L_1 + L_2 + ... + L_m \tag{17}$$

Where:

$$L_i = \left\| \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix}_{i+1} - \begin{Bmatrix} \theta_1 \\ \theta_2 \\ \theta_3 \end{Bmatrix}_i \right\| \tag{18}$$

8 SIMULATION RESULTS

To demonstrate the efficiency of the proposed algorithm, here an example with the following data is included:

$\theta_1^I = 1.3206,\ \theta_2^I = -0.0337,\ \theta_3^I = -1.1890,\ \theta_1^F = 1.3461,$
$\theta_2^F = -0.1918,\ \theta_3^F = -0.3439$ all in radians,
$T = 4\text{sec}\ ;\ T^* = 1\text{sec}\ ;\ L=0;\ \vartheta_1 = \pi/2\ ;$

The angle between $\boldsymbol{v}_i$ and $\boldsymbol{w}_i$, $\vartheta_1 = \pi/2$
The angle between $\boldsymbol{u}_i$ and $\boldsymbol{w}_i$, $\vartheta_2 = \pi/2.$

This algorithm starts with L=0 and if collision occurs, the amplitude will be increased automatically. The gripper will be asked to find an optimal path between the initial and final poses in the presence of obstacles and singularity in the 3-D motion (Fig. 12). Subsequently, a path planning algorithm is used to determine an optimal path avoiding these obstacles with maximum average of KCI. The algorithm leads to $\alpha = -0.05,\ \beta = \gamma = 0,\ L = 2000$, average KCI=0.2986 and $\alpha = 0,\ \beta = 0.05,\ \gamma = 0.001,\ L = 500$, average KCI=0.3607 in order to avoid singular point No. 1 and No. 2 respectively. Then the optimal trajectories in the joint space yields:

$$\{\boldsymbol{\theta_j}\} = \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} + \left[\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} S(\tau) + \begin{Bmatrix} -0.05 \\ 0 \\ 0 \end{Bmatrix} S^*(\tau^*) \right] \left(\{\boldsymbol{\theta_j}\}^{\boldsymbol{Final}} - \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} \right) \tag{19}$$

$$\{\boldsymbol{\theta_j}\} = \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} + \left[\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} S(\tau) + \begin{Bmatrix} 0 \\ 0.05 \\ 0.001 \end{Bmatrix} S^*(\tau^*) \right] \left(\{\boldsymbol{\theta_j}\}^{\boldsymbol{Final}} - \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} \right) \tag{20}$$

Moreover, the algorithm yields to the followings in order to avoid singular point No. 3, obstacles No. 1 and No. 2 with $\alpha = \beta = 0.05, \gamma = 0,\ L = 500$, KCI=0.3607; i.e.:

$$\{\boldsymbol{\theta_j}\} = \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} + \left[\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} S(\tau) + \begin{Bmatrix} 0.05 \\ 0.05 \\ 0 \end{Bmatrix} S^*(\tau^*) \right] \left(\{\boldsymbol{\theta_j}\}^{\boldsymbol{Final}} - \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} \right) \tag{21}$$

The proposed algorithm yields to $\alpha = 0.03, \beta = 0.02,\ \gamma = 0.001, L = 210$, KCI=0.4201 for obstacles No. 3 and No. 4 avoidances as below:

$$\{\boldsymbol{\theta_j}\} = \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} + \left[\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} S(\tau) + \begin{Bmatrix} 0.03 \\ 0.02 \\ 0.001 \end{Bmatrix} S^*(\tau^*) \right] \left(\{\boldsymbol{\theta_j}\}^{\boldsymbol{Final}} - \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} \right) \tag{22}$$

The optimal trajectories are also obtained in order to avoid singular point No. 4 and obstacle No. 5 with $\alpha = 0.04, \beta = 0,\ \gamma = 0.002, L = 190$, KCI=0.3921 and obstacle No. 6 with $\alpha = 0, \beta = 0.06,\ \gamma = 0.004,$ $L = 300$, KCI=0.4112 as:

$$\{\boldsymbol{\theta_j}\} = \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} + \left[\begin{Bmatrix} 1 \\ 1 \\ 1 \end{Bmatrix} S(\tau) + \begin{Bmatrix} 0.04 \\ 0 \\ 0.002 \end{Bmatrix} S^*(\tau^*) \right] \left(\{\boldsymbol{\theta_j}\}^{\boldsymbol{Final}} - \{\boldsymbol{\theta_j}\}^{\boldsymbol{Initial}} \right) \tag{23}$$

$$\{\theta_j\} = \{\theta_j\}^{Initial} + \left[\begin{Bmatrix}1\\1\\1\end{Bmatrix} S(\tau) + \begin{Bmatrix}0\\0.06\\0.004\end{Bmatrix} S^*(\tau^*)\right] \left(\{\theta_j\}^{Final} - \{\theta_j\}^{Initial}\right) \quad (24)$$

The optimal trajectory is depicted in Fig. 12, which shows the efficiency of the algorithm.

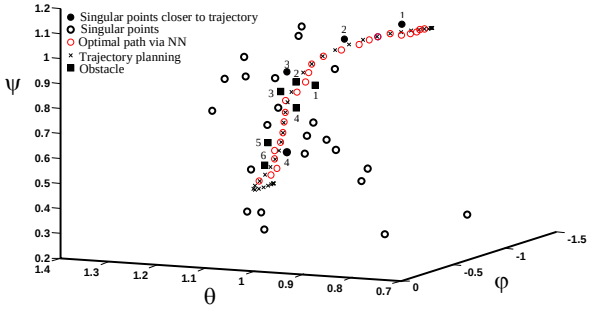

Fig. 12 Optimal path planning based on ANN

9 CONCLUDING REMARKS

In this paper, a simple and novel method for singularity-free path planning and obstacle avoidance of parallel manipulator based on neural networks was presented. A modified 4-5-6-7 interpolating polynomial was used to plan a trajectory between two poses. The polynomial was modified to plan a trajectory with the best kinematics conditioning index while avoiding singularities and obstacles. The simulation results for the Agile-Eye which leaded to a smooth trajectory avoiding singularities and obstacles demonstrated the efficiency of the algorithm. Moreover, an artificial neural network was implemented to solve forward kinematics of the manipulator to estimate the distance between the gripper and singularity or obstacle. The algorithm may be easily adopted to solve forward kinematics of any parallel manipulator and its trajectory planning.

REFERENCES

[1] Dasgupta, D., Mruthyunjaya, T. S., "Singularity free path planning for the stewart platform manipulators", Journal of Mechanism and machine theory, Vol. 33, No. 6, 1998, pp. 965-974.

[2] Kumar Dash, A., Chen, I., Yeo, SH., and Yang, G., "Workspace generation and planning singularity free path for parallel manipulators", Journal of Mechanism and machine theory, Vol. 40, No. 7,2005, pp. 776-805.

[3] Ahmed Bazaz S., Tondu, B., "Minimum time on-line joint trajectory generator based on low order spline method for industrial manipulators", Journal of Robotics and Autonomous Systems, Vol. 29, No. 4, 1999, pp. 257-268.

[4] Saramago, S. F. P., Steffen Junior, V., "Obstacle avoidance and robot trajectory planning optimization", Proceeding of forth world congress on computational Mechanics (IV WCCM), Buenos Aires, Argentina, 1998.

[5] Saramago, S. F. P., Steffen Junior, V., "Optimal trajectory planning of robot manipulators in the presence of moving obstacles", Journal of Mechanism and Machine Theory, Vol. 35, No. 8, 2000, pp. 1079-1094.

[6] Boudreau, R., Levesque, G., and Darenfed, S., "Parallel manipulator kinematics learning using holographic neural network models", Journal of Robotic and Computer Integrated Manufacturing, Vol. 14, No. 1, 1998, pp. 37-44.

[7] Kim, J., Mowat, A., Poole, P., and Kasabov, N., "Linear and non-linear pattern recognition models for classification of fruit from visible-near infrared spectra", Chemometrics and Intelligent Laboratory Systems, Vol. 51, No. 2, 2000, pp. 201-216.

[8] Soheil, S. P., Daniali H. M., and Ghaderi, R., "Optimization of parallel manipulator trajectory for obstacle and singularity avoidances based on neural network", International Journal of Advanced Manufacturing Technology, Vol. 51, No. 5, 2010, pp. 811-816.

[9] Gosselin, C. M., St. Pierre E., and Gagne, M., "On the development of the agile eye", IEEE Robotics & Automation Magazine, 1996.

[10] Gosselin, C. M., Gange, M., "A closed-form solution for the direct kinematics of a special class of spherical three-degree-of-freedom parallel manipulators", Workshop on computational kinematics, 1995, pp. 231-240.

[11] Bonev, I., Chablat, D., and Wenger, P., "Working and assembly modes of the agile eye" IEEE International Conference on Robotics and Automation, 2006, pp. 2317-2322.

[12] Mohammadi Daniali, H. R., Zsombor-Murray, P. J., and Angeles, J., "The isotropic design of two general classes of plannar parallel manipulator", Journal of Robotic System, Vol. 12, No. 12, 1995, pp. 795-805.