International Journal of Soft Computing 13 (1): 6-17, 2018
ISSN: 1816-9503
© Medwell Journals, 2018

3D Face Recognition Based on Neural Network in Quaternionic Domain

Sushil Kumar and B.K. Tripathi
Harcourt Butler Technical University, Kanpur, 208002 Uttar Pradesh, India

Abstract: There are various high dimensional engineering and scientific applications in communication, control, robotics, computer vision, biometrics, etc. Where researchers are facing problem to design the intelligent and robust system through high dimensional neural network. Although, some of the researcher used different structures of conventional neural classifier to solve the problem associated with high dimensional parameters. These network structures possess complex network, hence, very time consuming and weak to noise. These networks are also not able to learn magnitude and direction of each component simultaneously at 3D space. The quaternion is the number which possesses the magnitude and phase information in all four directions. This study presents the quaternionic domain neural network and its generalized learning algorithm that can finely process magnitude and phase information in respective directions. Its learning and generalization capability presented through different simulations demonstrate its applicability in 3D imaging.

Key words: Quaternion, quaternionic algebra, quaternionic domain neural network, interpretation of 3D motion, 3D face recognition, direction

INTRODUCTION

The high dimensional information processing through neural network is emerging a fascinating but challenging field of research in second generation of neurocomputing. The recent researches in high dimensional neural networks have established their superiority (Tripathi and Kalra, 2011a, b; Nitta, 1997, 2000; Muezzinoglu *et al*., 2003) over real-valued or first generation neural networks. Although, high dimensional data can be processed through real-valued neurons but the corresponding network needs to employ too many neurons, huge structure and slow learning. The Complex-Valued Neural Networks (CVNN) have demons trated their outperformance over conventional or Real-Valued Neural Network (RVNN) for two dimensional as well as single dimension problems (Nitta, 1995, 1997; Muezzinoglu *et al*., 2003; Tripathi *et al*., 2011). The complex-valued neural networks promptly consider two dimensional information as a single number which has drastically reduced the complexity of network along with far better performance. But neural network for three dimensional information still need an exhaustive investigation. The applications with three dimensional information are popular in computer vision, robotics, biometrics, bioinformatics, etc. The few researches attempted machine learning for three dimensional information considering it as a vector (Tripathi and Kalra, 2011a; Nitta and Garis, 1992) and developed Vector-Valued Neural Network (VVNN). The corresponding learning algorithms have restriction on weight matrix and a vector does not provide freedom like a complex number as in CVNN (Tripathi and Kalra, 2012). Thus, it is very demanding to have neural network which may promptly process different high dimensional parameters as numbers. The machine learning with conventional neural network has been widely accepted for wide spectrum of problems because of simplicity of learning algorithm due to employment of only real numbers. Similarly, CVNN came out with popularity having parameters as numbers and is successfully incorporated in neural network for various applications of intelligent machine design (Tripathi and Kalra, 2011b; Nitta, 1997, 2000; Muezzinoglu *et al*., 2003). In interpretation of 3D motion, 3D transformations of objects and 3D face matching, the 3D Vector-Valued Neural Network (3D-VVNN) is designed whose learning and generalization ability is investigated by Tripathi and Kalra (2011a, 2012), Nitta and Garis (1992). In this network, three dimensional parameters are considered in vectors as single body because there is no number system in three dimensions (Tripathi, 2014). In the enhancement of higher order number systems the complex numbers (2D), quaternions (4D), octaves (8D), sedenions (16D) were developed by mathematicians in past.

The neurocomputing with high dimensional number systems will definitely lead to lower complexity and shall overcome from learning and generalization problems of

Corresponding Author: Sushil Kumar, Harcourt Butler Technical University, Kanpur, 208002 Uttar Pradesh, India

huge conventional neural network. In this study, quaternionic number system has been considered in high dimension neural network as four dimensional parameters. It is one of the hyper-complex number system initially introduced by Iris Mathematician Hamilton. This number system has been extensively concerned in fields of quantum mathematics, physics, computer graphics, signal processing and control of satellites (Kuipers, 1998; Hoggar, 1992; Ujang *et al*., 2011; Wang *et al*., 2011). This number system has recently popped up in neural network through quaternionic neurons as in complex or real-valued neurons to develop efficient machine learning in high dimensions. Few attempts have been made in this direction (Isokawa *et al*., 2012, 2013; Ujang *et al*., 2011; Wang *et al*., 2011; Nitta, 2004) but it is still necessary to present a generalized learning algorithm for a sufficiently general structure of Quaternionic Domain Neural Network (QDNN) along with a wide spectrum of applications, like motion interpretation and recognition in space. The orthogonal decision boundary of single quaternionic neuron has been introduced to solve 4-bit parity problem by Nitta (2004) and compared the performance through time and space complexity over real and complex domain neural networks. Isokawa *et al*. (2003) and Nitta (2004) presented quaternionic-valued neural network with sigmoidal activation function, hence, it is requires to establish learning algorithm for more generalized structure of neural network which can be applied in various applications directly without any modifications. This need motivated us to build the broad learning algorithm in QDNN. Isokawa *et al*. (2012) proposed quaternionic MLPs which would face the problem of the existence of singularities which need to be handled similarly as in the case of complex-valued MLPs for the removal or avoidance of such singularities. The researcher left the universality with handling singularities as future research. Ujang *et al*. (2011) and Wang *et al*. (2011) presented quaternion-valued algorithms for adaptive filtering for many cases. But in this study, we present a simple, straightforward but potential machine learning algorithm for general structure of QDNN which has efficiently solved the problems like motion interpretation and recognition in high dimensions.

This study presents the general structure of Quaternionic Domain Neural Network (QDNN) with learning algorithm where all synaptic weights, biases, inputs-outputs and internal potentials are quternions and represented as quaternionic matrix in multilayer neural network. Although, Hamilton (1853) proposed quaternionic number ($q = q_0+q_1i+q_2+q_3k$) for 4D number system but it can also be used to represent any 3D information in the space after equating its real part zero. Thus, neural network in quternionic domain can solve any typical class of problems in 3 and 4D efficiently. An efficient error back-propagation algorithm is presented in this study and tested by benchmark problems as well as 3D face matching as a real life application. The study of suitable activation function is one of the complicated issues in the network dealing with quaternion to obtain the output of a neuron. The analytic (Tripathi and Kalra, 2011a, 2012) and split type (Tripathi and Kalra, 2011a) activation functions have been chosen in two or high dimensional neural networks. Although, these type of activation functions have their own issues concerning boundedness and analyticity. The split type function is not appropriate when analyticity is concerned, similarly the analytic function sometimes not suitable when singularity arises. The presented QDNN with split activation function performs with fewer computations, lesser number of neurons and comparatively faster learning which is not possible with conventional neural network. QDNN Model has an ability to learn and generalize 3D motion of objects whereas CVNN and RVNN cannot because it has ability to capture phase information during learning.

Machine learning in quaternionic domain

Quaternionic matrix in four dimension: Quaternionic number system is the straightforward extension of complex number system with four components, incorporated in single variable where one acts as real and other three as imaginary components with unit vectors (i, j, k). These imaginary components can be drown as three axes in three dimensional spaces (Kuipers, 1998; Hamilton, 1853). There are huge range of applications such as in computer vision, robotics, orbital mechanics, molecular dynamics, etc. where this quaternionic number system can be directly embedded for computing four as well as three dimensional information.

A quaternionic variable ($q = q_0+q_1i+q_2+q_3k$) consists of a real component (q_0) and three imaginary components (q_1, q_2, q_3). Its bases (i, j, k) are orthogonal special vectors. Thus, they follow the properties as and cross product properties as $I\times j = -(j\times i) = k$, $j\times k = -(k\times j) = I$, $k\times i = -(I\times k) = j$. In a prominent representation a quaternion (q) can be expressed in the form of matrix, i.e., quaternionic matrix: (bold type letter denotes quaternionic variable or quaternionic matrix):

$$\mathbf{q} = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \quad (1)$$

The conjugate of quaternionic variable ($q^* = q_0-q_1 i-q_2 j-q_3 k$) is similar to complex conjugate and the conjugate of quaternionic matrix denotes the transpose of the quaternionic matrix, defined as follows:

$$q^* = q^T = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix}^T = \begin{bmatrix} q_0 & -q_1 & -q_2 & -q_3 \\ -q_1 & q_0 & q_3 & -q_2 \\ q_2 & -q_3 & q_0 & q_1 \\ q_3 & q_2 & -q_1 & q_0 \end{bmatrix} \quad (2)$$

The addition and subtraction of two quaternionic matrices q and r are calculated same as matrices addition and subtraction. The multiplication of two different quaternionic matrices q and r does not follow the commutative property ($qr \neq rq$). The inner product of two quaternionic matrices q and r is expressed by:

$$q \odot r = \begin{bmatrix} q_0 & q_1 & q_2 & q_3 \\ -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix} \odot \begin{bmatrix} r_0 & -r_1 & -r_2 & -r_3 \\ -r_1 & r_0 & -r_3 & r_2 \\ -r_2 & r_3 & r_0 & -r_1 \\ -r_3 & -r_2 & r_1 & r_0 \end{bmatrix} = \begin{bmatrix} q_0r_0 & q_1r_1 & q_2r_2 & q_3r_3 \\ q_1r_1 & q_0r_0 & q_3r_3 & q_2r_2 \\ q_2r_2 & q_3r_3 & q_0r_0 & q_1r_1 \\ q_3r_3 & q_2r_2 & q_1r_1 & q_0r_0 \end{bmatrix} \quad (3)$$

The norm of quaternionic matrix is expressed as:

$$\|q\| = \frac{1}{2}\sqrt{\sum \text{diag}(qq^T)} = \sqrt{q_0^2+q_1^2+q_2^2+q_3^2} \quad (4)$$

Learning in quaternionic domain neural network: Let a three layer (L-M-N) QDNN Model employs quaternionic neurons at input (L), a hidden (M) and output (N) layers of the network, respectively. In the network, all inputs, outputs, weights and biases signals are considered as quaternionic matrices. Quaternionic matrix is the representation of quaternionic variable as presented in Eq. 1. In order to develop a QDNN based learning machine, we have designed the back propagation learning algorithm. The derivation of optimization technique incorporates the basic operations of quaternionic algebra such as quaternionic matrix addition, subtraction, multiplication and conjugate (Hamilton, 1853; Kuipers, 1998). These operations present the compact and generalized Derivation of Back Propagation algorithm of three-layer network (QDBP) as follows (bold letters denote the quternionic matrix or matrix containing quaternionic matrices as elements).

Let us consider I^r_1, I^x_1, I^y_1, I^z_1 be the 4D quaternionic input of lth (l = 1, ..., L) neuron at input layer of the network. The quaternionic input has expressed as a quaternionic matrix (I_l) as follows:

$$I_1 = \begin{bmatrix} I_1^r & I_1^x & I_1^y & I_1^z \\ -I_1^x & I_1^r & -I_1^z & I_1^y \\ -I_1^y & I_1^z & I_1^r & -I_1^x \\ -I_1^z & -I_1^y & I_1^x & I_1^r \end{bmatrix} \quad (5)$$

The Input matrix (I) at input layer of the network is defined by:

$$I = [I_1\ I_2\ I_3, ..., I_L] \quad (6)$$

The initialization of synaptic connection weights w_{ml} and s_{nm} are defined for lth input to mth (m = 1, ..., M) hidden neuron pair and for mth hidden to nth (n = 1, ..., N) output neuron pair of network, respectively. These weights are presented in quaternionic matrices containing a real and other three imaginary components as follows:

$$w_{ml} = \begin{bmatrix} w_{ml}^r & w_{ml}^x & w_{ml}^y & w_{ml}^z \\ -w_{ml}^x & w_{ml}^r & -w_{ml}^z & w_{ml}^y \\ -w_{ml}^y & w_{ml}^z & w_{ml}^r & -w_{ml}^x \\ -w_{ml}^z & -w_{ml}^y & w_{ml}^x & w_{ml}^r \end{bmatrix} \quad (7)$$

$$S_{nm} = \begin{bmatrix} S_{nm}^r & S_{nm}^x & S_{nm}^y & S_{nm}^z \\ -S_{nm}^x & S_{nm}^r & -S_{nm}^z & S_{nm}^y \\ -S_{nm}^y & S_{nm}^z & S_{nm}^r & -S_{nm}^x \\ -S_{nm}^z & -S_{nm}^y & S_{nm}^x & S_{nm}^r \end{bmatrix} \quad (8)$$

Similarly, the initialization of biases α_m and β_n are defined for mth hidden and nth output neuron of the network, respectively as:

$$\alpha_m = \begin{bmatrix} \alpha_m^r & \alpha_m^x & \alpha_m^y & \alpha_m^z \\ -\alpha_m^x & \alpha_m^r & -\alpha_m^z & \alpha_m^y \\ -\alpha_m^y & \alpha_m^z & \alpha_m^r & -\alpha_m^x \\ -\alpha_m^z & -\alpha_m^y & \alpha_m^x & \alpha_m^r \end{bmatrix} \quad (9)$$

$$\beta_n = \begin{bmatrix} \beta_n^r & \beta_n^x & \beta_n^y & \beta_n^z \\ -\beta_n^x & \beta_n^r & -\beta_n^z & \beta_n^y \\ -\beta_n^y & \beta_n^z & \beta_n^r & -\beta_n^x \\ -\beta_n^z & -\beta_n^y & \beta_n^x & \beta_n^r \end{bmatrix} \quad (10)$$

The internal potential matrix U at hidden layer of the network is defined as:

$$U = WI+\alpha \quad (11)$$

$$\begin{bmatrix} U_1 \\ U_2 \\ U_3 \\ \vdots \\ U_M \end{bmatrix} = \begin{bmatrix} w_{11} & w_{12} & w_{13}, \ldots, & w_{1L} \\ w_{21} & w_{22} & w_{23}, \ldots, & w_{2L} \\ w_{31} & w_{32} & w_{33}, \ldots, & w_{3L} \\ \vdots & \vdots & \vdots & \vdots \\ w_{M1} & w_{M2} & w_{M3}, \ldots, & w_{ML} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_L \end{bmatrix} + \begin{bmatrix} \alpha_1 \\ \alpha_2 \\ \alpha_3 \\ \vdots \\ \alpha_M \end{bmatrix} \quad (12)$$

Where:

W = The Weight matrix which contains interconnection synaptic quaternionic weights between input and hidden neurons as elements correspondingly

α = The bias matrix which also contains all quaternionic biases of hidden neurons as elements in column

Let, f be an activation function and f be its derivative. The Output matrix (O) is obtained by applying split-type activation function over internal potential matrix (U) at hidden layer, defined as follows:

$$O = f(U) \quad (13)$$

$$[O_1\ O_2\ O_3, \ldots, O_M]^T = [f(U_1) f(U_2) f(U_3), \ldots, f(U_M)]^T \quad (14)$$

Where:

$$O_m = f(U_m) \begin{bmatrix} f(U_m^r)\ f(U_m^x)\ f(U_m^y)\ f(U_m^z) \\ f(-U_m^x)\ f(U_m^r)\ f(-U_m^z)\ f(U_m^y) \\ f(-U_m^y)\ f(U_m^z)\ f(U_m^r)\ f(-U_m^x) \\ f(-U_m^z)\ f(-U_m^y)\ f(U_m^x)\ f(U_m^r) \end{bmatrix} \quad (15)$$

The internal potential matrix V at output layer of the network is defined as:

$$V = SO + \beta \quad (16)$$

$$\begin{bmatrix} V_1 \\ V_2 \\ V_3 \\ \vdots \\ V_N \end{bmatrix} = \begin{bmatrix} S_{11} & S_{12} & S_{13}, \ldots, & S_{1M} \\ S_{21} & S_{22} & S_{23}, \ldots, & S_{2M} \\ S_{31} & S_{32} & S_{33}, \ldots, & S_{3M} \\ \vdots & \vdots & \vdots & \vdots \\ S_{N1} & S_{N2} & S_{N3}, \ldots, & S_{NM} \end{bmatrix} \begin{bmatrix} O_1 \\ O_2 \\ O_3 \\ \vdots \\ O_M \end{bmatrix} + \begin{bmatrix} \beta_1 \\ \beta_2 \\ \beta_3 \\ \vdots \\ \beta_N \end{bmatrix} \quad (17)$$

Where:

S = The weight matrix which contains interconnection synaptic quaternionic weights between hidden and output neurons as elements correspondingly

β = The bias matrix which also contains all quaternionic biases of output neurons as elements in column

The output matrix (Y) is obtained by applying split-type activation function (f) over internal potential matrix (V) at output layer, defined as follows:

$$Y = f(V) \quad (18)$$

$$[Y_1\ Y_2\ Y_3, \ldots, Y_N]^T = [f(V_1)\ f(V_2)\ f(V_3), \ldots, f(V_N)]^T \quad (19)$$

Where:

$$Y_n = f(V_n) = \begin{bmatrix} f(V_n^r)\ f(V_n^x)\ f(V_n^y)\ f(V_n^z) \\ f(-V_n^x)\ f(V_n^r)\ f(-V_n^z)\ f(V_n^y) \\ f(-V_n^y)\ f(V_n^z)\ f(V_n^r)\ f(-V_n^x) \\ f(-V_n^z)\ f(-V_n^y)\ f(V_n^x)\ f(V_n^r) \end{bmatrix} \quad (20)$$

The error back-propagation algorithm is applied to minimize the mean square Error (E) of the network as follows:

$$E = \frac{1}{8N} \sum_{n=1}^{4N} \text{diag} \left(\begin{bmatrix} e_1 & & & & 0 \\ & e_1 & & & \\ & & e_1 & & \\ & & & \ddots & \\ 0 & & & & e_N \end{bmatrix} \right) \left(\begin{bmatrix} e_1^* & & & & 0 \\ & e_2^* & & & \\ & & e_3^* & & \\ & & & \ddots & \\ 0 & & & & e_N^* \end{bmatrix} \right) \quad (21)$$

where, * denotes quaternionic conjugate as presented in Eq. 2 and the output error matrix (e) presents the difference between actual (Y) and desired (Y^D) output matrix at output layer of the network, defined as:

$$e = Y - Y^D \quad (22)$$

$$\begin{bmatrix} e_1 \\ e_2 \\ e_3 \\ \vdots \\ e_N \end{bmatrix} = \begin{bmatrix} Y_1 \\ Y_2 \\ Y_3 \\ \vdots \\ Y_N \end{bmatrix} - \begin{bmatrix} Y_1^D \\ Y_2^D \\ Y_3^D \\ \vdots \\ Y_N^D \end{bmatrix} = \begin{bmatrix} Y_1 - Y_1^D \\ Y_2 - Y_2^D \\ Y_3 - Y_3^D \\ \vdots \\ Y_N - Y_N^D \end{bmatrix} \quad (23)$$

The update equations of weight and bias matrices are obtained by gradient decent optimization approach on mean Square error (ΔS). The weight updating matrix between hidden-output layers and Bias updating matrix ($\Delta\beta$) at output layer of the network are presented as follows:

$$\Delta\beta = \begin{bmatrix} \Delta\beta_1 \\ \Delta\beta_2 \\ \Delta\beta_3 \\ \vdots \\ \Delta\beta_N \end{bmatrix} = \frac{\eta}{N} \begin{bmatrix} e_1 \odot f'(V_1) \\ e_2 \odot f'(V_2) \\ e_3 \odot f'(V_3) \\ \vdots \\ e_N \odot f'(V_N) \end{bmatrix} \tag{24}$$

$$\Delta S = \begin{bmatrix} \Delta S_{11}\ \Delta S_{12}\ \Delta S_{13}, \ldots, \Delta S_{1M} \\ \Delta S_{21}\ \Delta S_{22}\ \Delta S_{23}, \ldots, \Delta S_{2M} \\ \Delta S_{31}\ \Delta S_{32}\ \Delta S_{33}, \ldots, \Delta S_{3M} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \Delta S_{N1}\ \Delta S_{N2}\ \Delta S_{N3}, \ldots, \Delta S_{NM} \end{bmatrix} = \frac{\eta}{N} \begin{bmatrix} e_1 \odot f'(V_1) \\ e_2 \odot f'(V_2) \\ e_3 \odot f'(V_3) \\ \vdots \\ e_N \odot f'(V_N) \end{bmatrix} \begin{bmatrix} 0_1^* \\ 0_2^* \\ 0_3^* \\ \vdots \\ 0_N^* \end{bmatrix} \tag{25}$$

Where:

$n \in \mathbb{R}^+$ = A learning rate and

$\odot$ = Element-wise multiplication of two quaternionic matrices as defined in Eq. 3

Similarly, Weight updating matrix (ΔW) between input-hidden layers and bias updating matrix ($\Delta\alpha$) at hidden layer of the network are presented as follows:

$$\Delta\alpha = \begin{bmatrix} \Delta\alpha_1 \\ \Delta\alpha_2 \\ \Delta\alpha_3 \\ \vdots \\ \Delta\alpha_N \end{bmatrix} = \frac{\eta}{N} \left(\begin{bmatrix} \Delta S_{11}\ \Delta S_{12}\ \Delta S_{13}, \ldots, \Delta S_{1M} \\ \Delta S_{21}\ \Delta S_{22}\ \Delta S_{23}, \ldots, \Delta S_{2M} \\ \Delta S_{31}\ \Delta S_{32}\ \Delta S_{33}, \ldots, \Delta S_{3M} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \Delta S_{N1}\ \Delta S_{N2}\ \Delta S_{N3}, \ldots, \Delta S_{NM} \end{bmatrix}^T \begin{bmatrix} e_1 \odot f'(V_1) \\ e_2 \odot f'(V_2) \\ e_3 \odot f'(V_3) \\ \vdots \\ e_N \odot f'(V_N) \end{bmatrix} \right) \odot \begin{bmatrix} f'(U_1) \\ f'(U_2) \\ f'(U_3) \\ \vdots \\ f'(U_M) \end{bmatrix} \tag{26}$$

$$\Delta W = \begin{bmatrix} \Delta w_{11}\ \Delta w_{12}\ \Delta w_{13}, \ldots, \Delta w_{1L} \\ \Delta w_{21}\ \Delta w_{22}\ \Delta w_{23}, \ldots, \Delta w_{2L} \\ \Delta w_{31}\ \Delta w_{32}\ \Delta w_{33}, \ldots, \Delta w_{3L} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \Delta w_{M1}\ \Delta w_{M2}\ \Delta w_{M3}, \ldots, \Delta w_{ML} \end{bmatrix}$$

$$= \frac{\eta}{N} \left(\left(\begin{bmatrix} \Delta S_{11}\ \Delta S_{12}\ \Delta S_{13}, \ldots, \Delta S_{1M} \\ \Delta S_{21}\ \Delta S_{22}\ \Delta S_{23}, \ldots, \Delta S_{2M} \\ \Delta S_{31}\ \Delta S_{32}\ \Delta S_{33}, \ldots, \Delta S_{3M} \\ \vdots \quad \vdots \quad \vdots \quad \vdots \\ \Delta S_{N1}\ \Delta S_{N2}\ \Delta S_{N3}, \ldots, \Delta S_{NM} \end{bmatrix}^T \begin{bmatrix} e_1 \odot f'(V_1) \\ e_2 \odot f'(V_2) \\ e_3 \odot f'(V_3) \\ \vdots \\ e_N \odot f'(V_N) \end{bmatrix} \right) \odot \begin{bmatrix} f'(U_1) \\ f'(U_2) \\ f'(U_3) \\ \vdots \\ f'(U_M) \end{bmatrix} \right) \begin{bmatrix} I_1^* \\ I_2^* \\ I_3^* \\ \vdots \\ I_L^* \end{bmatrix} \tag{27}$$

Learning algorithm in quaternionic domain for 3D imaging: In the field of 3D imaging, QDNN can be employed straightforwardly. For this application, we have considered points cloud data of 3D objects or human faces. A uniformly distributed countable points are considered to build the 3D surface. In the point cloud data, we have embedded near to zero value in real component to devise purely imaginary quaternionic varaible ($0+x_i i+y_i+z_i k$). These points act as inputs and transformation of these inputs act as desired outputs of proposed network.

The pseudo code for training of QDNN, we have developed an algorithm QDNN_TRANING() which contains QDNN_INITIALIZATION(), QDNN_FORWARD() and QDNN_BP() procedures. This optimization algorithm facilitates to minimize the mean square error at output layer and yields the generalization ability of neural structure containing L-M-N neurons at input, hidden and output layer, respectively. The procedure QDNN_INITIAIZATION() randomly initializes the weight and bias matrices at hidden and output layer. It calls the RANDOM_QUATERNIONIC_MATRIX(a, b) procedure which randomly generates the quaternionic matrix of each interconnection weight and bias of neuron in the range from a-b. The QDNN_FORWARD() procedure is intended to implement forward pass of QDNN, hence, generate internal potentials (U, V) and (O, Y) outputs matrices at respective layers. The ACTIVATION_FUNCTION() limits the output of corresponding neuron of the network. For updating weight and bias matrices, backward pass QDNN_BP( ) is developed in quaternionic domain neural network. All required procedures are presented in pseudo code for 3D imaging as follows (Algorithm 1):

Algorithm 1; procedure in pseudo code:

```
procedure QDNN_TRAINING (I, YD, η, e)
begin
    QDNN_ININTIALIZATION(L, M, N)
    while E_r>∈ do
        for I ← 1 until S = length (I) do
            U, O, V, Y←QDNN_FORWORD(W, α, S, β, I)
            e←Y-Y^D

                E_i ← (1/8N) Σ_{n=1}^{4N} diag(diagonalmatrix(e)diagonalmatrix(e*))

            QDNN_BP(W, α, U, O, S, β, V, Y, η, I)

                E_r ← (1/S) Σ_{i=1}^{S} E_i

end
procedure QDNN_INITIALIZATION(L, M, N)
begin
    for m←1 until M do
        for I←1 until L do
```

```
        W_ml←RANDOM_QUATERNION_MATRIX(a, b)
    α_m←RANDOM_QUATERNION_MATRIX(a, b)
for n←1 until N do
      S_nm←RANDOM_QUATERNION_MATRIX(a, b)
β_n←RANDOM_QUATERNION_MATRIX(a, b)
end
procedure QDNN_FORWORD(W, α, S, β, I)
begin
    U←WI+α
    O←ACTIVATION_FUNCTION(U)
    V←SO+β
    Y←ACTIVATION_FUNCTION(V)
end
procedure QDNN_BP(W, α, U, O, S, β, V, Y, η, e)
begin
    Δβ←(η/N)e∘DER_ACTIVATION(V)
    ΔS←(η/N)(e∘DER_ACTIVATION(V))O*T
    Δα←(η/N)[(S]Te∘DER_ACTIVATION(V)))∘DER_ACTIVATION(U)
ΔW←(η/N)[S]T e∘DER_ACTIVATION(V)))∘DER_ACTIVATION(U))I*T
    β←β+Δβ
    S←S+ΔS
    α←α+Δα
    W←W+ΔW
end
procedure RANDOM)_QUATERNION_MATRIX(a, b)
begin
    q0←[a+(b-a)]RAND(1)
    q1←[a+(b-a)]RAND(1)
    q2←[a+(b-a)]RAND(1)
    q3←[a+(b-a)]RAND(1)
    q4←[a+(b-a)]RAND(1)

  q ← [ q0  q1  q2  q3
        q1  q0  q3  q2
        q2  q3  q0  q1
        q3  q2  q1  q0 ]
end
procedure QDNN_ACTIVATION_FUNCTION(q)
begin
    Q = f(q)
end
```

Mapping in 3D through quaternionic matrix: Quaternionic domain back propagation learning algorithm is capable to learn linear transformations (translation, rotation and scaling) of 3D objects or human faces. Each quaternionic variable $q_i = 0+x_i i+y_i j+z_i k$ under goes a Transformation function (T) through a trained network and correspondingly yields a transformed quaternionic variable $q'_i = 0+x'_i i+y'_i j+z'_i k$ represented in quaternionic matrix as follows:

$$q'_i = T(q_i) = aq_i + b \quad (i = 1, 2, 3, \ldots, n_p) \tag{28}$$

$$\begin{bmatrix} 0 & x'_i & y'_i & z'_i \\ -x'_i & 0 & -z'_i & y'_i \\ -y'_i & z'_i & 0 & -x'_i \\ -z'_i & -y'_i & x'_i & 0 \end{bmatrix} = \begin{bmatrix} 0 & a_x & a_y & a_z \\ -a_x & 0 & -a_z & a_y \\ -a_y & a_z & 0 & -a_x \\ -a_z & -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} 0 & x_i & y_i & z_i \\ x_i & 0 & -z_i & y_i \\ -y_i & z_i & 0 & -x_i \\ -z_i & -y_i & x_i & 0 \end{bmatrix} + \begin{bmatrix} 0 & b_x & b_y & b_z \\ -b_x & 0 & -b_z & b_y \\ -b_y & b_z & 0 & -b_x \\ -b_z & -b_y & b_x & 0 \end{bmatrix} \tag{29}$$

where, n_p denotes the number of points that lies on the surface of 3D object and a and b are quaternions such that norm of a:

$$\|a\| = \sqrt{0^2 + a_1^2 + a_2^2 + a_3^2}$$

i.e., denotes the scaling factor. Argument of a yields rotation in q while b performs translation of 3D object with the distance ($\|b\|$).

Performance evaluation of QDNN through benchmark problems: In order to evaluate the performance of QDNN, we have considered a 2-M-2 three layer neural structure. As benchmark problem, this study presents the learning of linear transformations (rotation, scaling and translation and their combinations) and its generalization ability of QDNN is performed tested over the complicated 3D objects. These combined transformations are defined by quaternionic matrix form as represented in Eq. 28. The combinations facilitate the viewing of 3D objects from different orientations, interpretation of their motion etc. For training of proposed QDNN, we have considered six and two quaternionic neurons at hidden and output layer respectively. In learning process, we have considered a straight line containing few input data points (21 points) and reference point (mid point 0, 0, 0) in 3D space for all experiments. First input receives set of point (x, y, z) that lies on straight line and second input passes the reference point (x_r, y_r, z_r). The incorporation of reference point of object provides more information to learning system which yields better accuracy. Similarly, the first and second output neurons of output layer obtain the transformed point (x', y', z') on line and transformed reference point (x'_r, y'_r, z'_r), respectively. The transformed line and its reference points are achieved through QDBP learning of QDNN with suitable learning rate. The trained QDNN is able to generalize over huge number of points cloud data of complicated geometrical structure like sphere, cylinder and torus. The generalization ability of network presents the 3D motion interpretation of objects.

The learning of QDNN (2-6-2 Model) is performed for similarity transformation with mapping over straight line containing 21 points; the reference is and scaling factor 1/2 as shown in Fig. 1a. Convergence of mean square error Fig. 1b shows the smart learning capability of proposed network. The training of QDNN with 0.00005 learning rate converges to 1.005567e-05 after 20000 iterations. The trained network is able to generalize over many complicated standard geometric structures like sphere 4141 data points, cylinder 2929 data points and torus 10201 data points as shown in Fig. 2a-c, respectively.

Similarly, the learning of QDNN is performed for combination of scaling (scaling factor 1/2) and translation

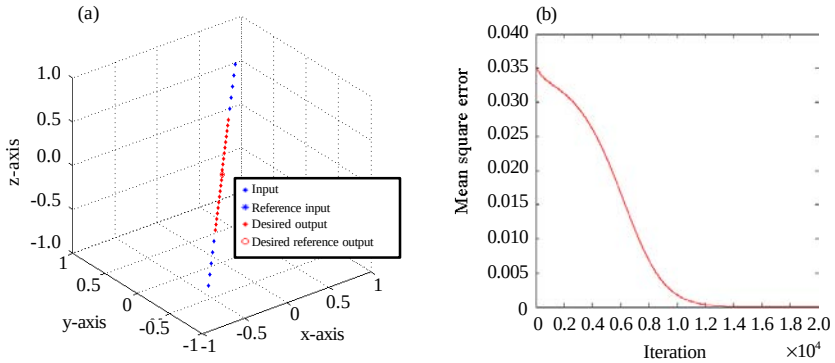

Fig. 1: a) Training input-output mapping for scaling with scaling factor 1/2 and b) convergence of mean square error

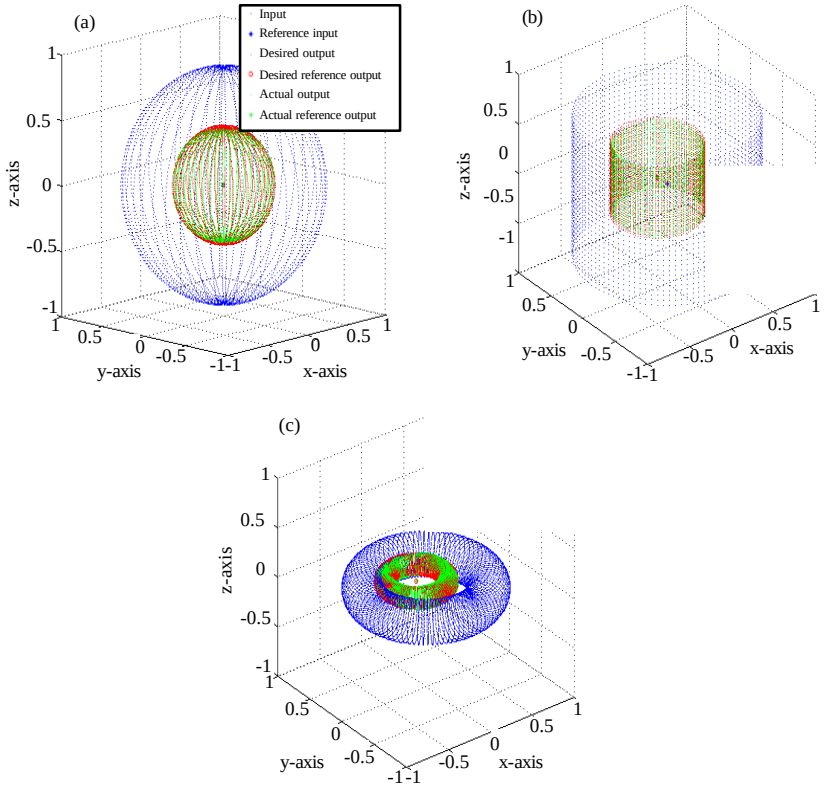

Fig. 2: Testing results for similarity transformation (scaling factor 1/2) through sphere, cylinder and torus

(0.3 unit in positive y-direction) with mapping over straight line and containing reference at as shown in Fig. 3a. The 2-6-2 QDNN Model is used for training of input-output mapping of these transformations through 21 data points on straight line as shown in Fig. 3a. Convergence of QDNN with 0.00005 learning rate, up to 2.58514e -05 square error shows the smart learning capability of proposed network after 20000 iterations as shown in Fig. 3b. The trained network is also able to generalize over many complicated standard geometric structures like sphere 4141 data points, cylinder 2929 data points and torus 10201 data points as shown in Fig. 4a-c, respectively.

The learning of QDNN for general linear transformation (scaling factor 1/2, counterclockwise rotation about x-axis by $\pi/2$ radian and translation by (0, 0, 0.3) is performed for through input-output mapping over straight line and reference as shown in Fig. 5a. The same QDNN (2-6-2) Model is used for training of these transformations through 21 data points on straight line.

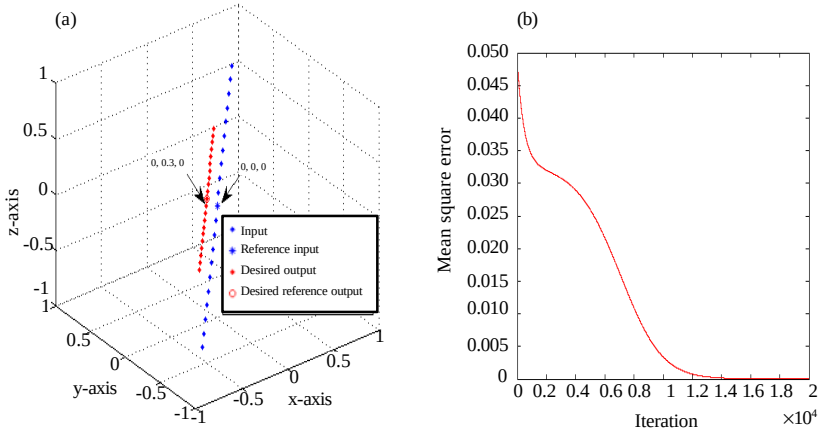

Fig. 3: a) Training patterns; input-output mapping shows scaling with scaling factor 1/2, followed by translation with 0.3 units in positive y-direction and b) Convergence of mean square error

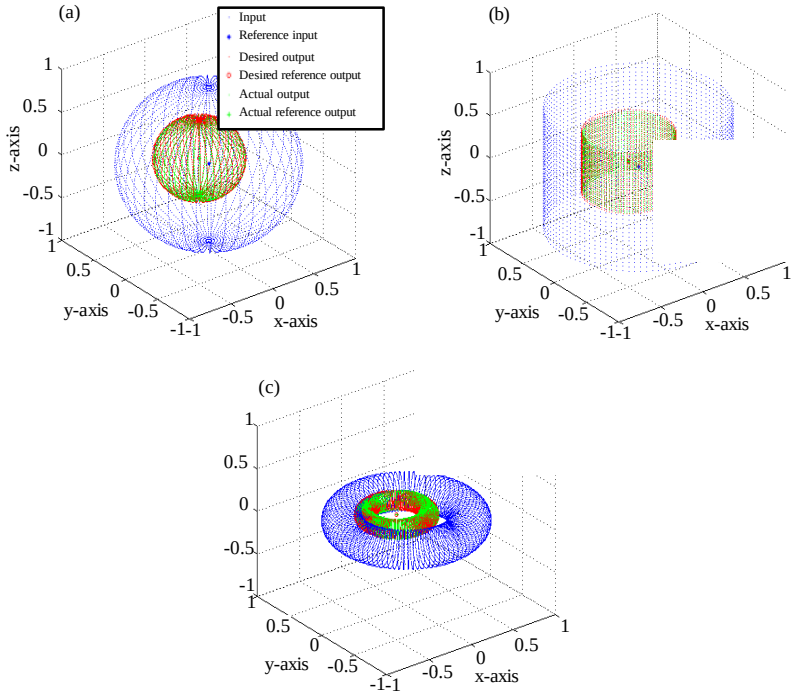

Fig. 4: Testing results for transformation (scaling factor 1/2, followed by translation with 0.3 units in positive y-direction) through sphere, cylinder and torus

Convergence of mean square error 1.0e-04 after 20000 iterations is achieved with 0.00005 learning rate as shown in Fig. 5b. The trained network is also able to generalize over many complicated standard geometric structures like sphere (4141 data points), cylinder (4141 data points) and torus (10201 data points) as shown in Fig. 6a-c, respectively.

All experiments show the intelligent behavior QDNN for motion interpretation of 3D objects. Further, this technique provides an approach to generalize the motion in intelligent system for variety of operations.

3D face recognition as biometrics application through proposed QDNN: This study presents applicability of QDNN for 3D face recognition as biometrics application. The method has a great deal to perform successful recognition of faces in variable head position, orientation and facial expressions. Two experiments are conducted

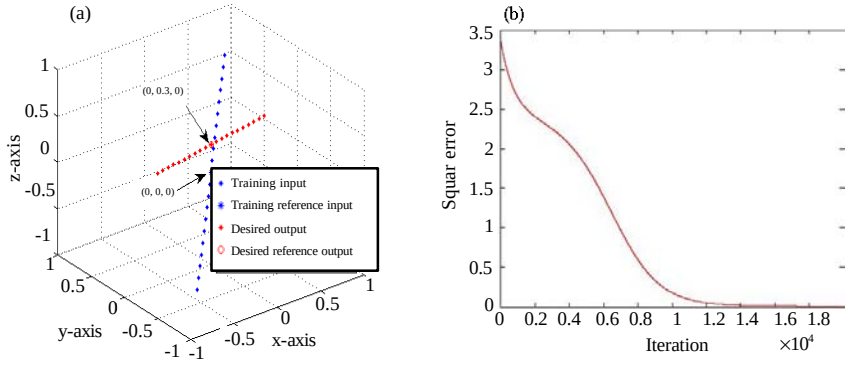

Fig. 5: a) Training patterns through straight line (scaling factor 1/2, counterclockwise rotated about x-axis by π/2 radian and translated by 0, 0, 0, 3) and b) Square error during training of straight line pattern

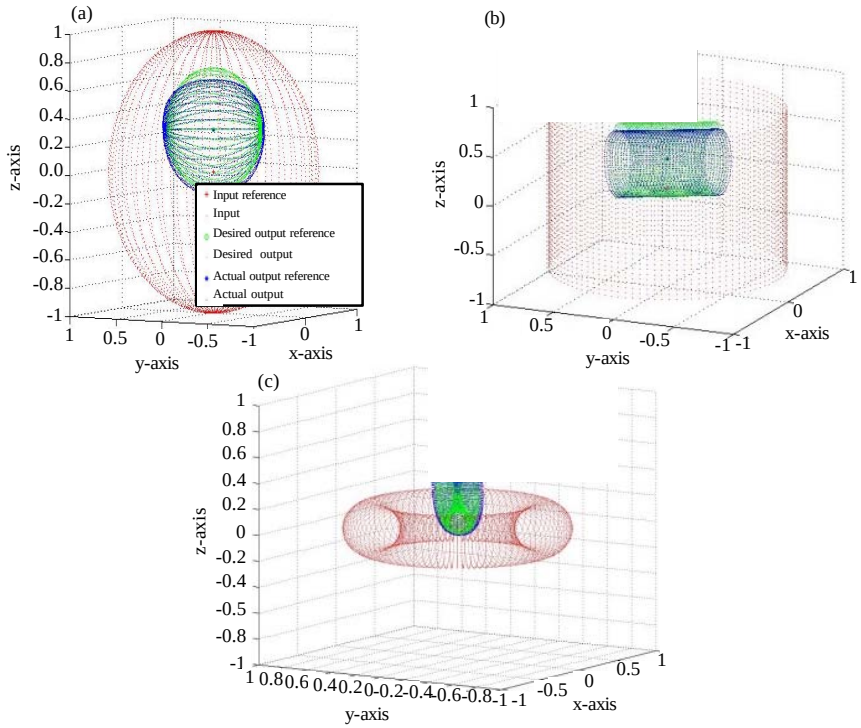

Fig. 6: General linear transformation (scaling factor 1/2, counterclockwise rotated about x-axis by π/2 radian and translated by 0, 0, 0, 3); a) Testing through sphere; b) Testing through cylinder and c) Testing through torus

conducted here to learn and classify point cloud data of 3D faces using proposed Quaternionic Domain Back-Propagation (QDBP) learning algorithm. A simple structure of QDNN containing one, two and one quaternionic neurons at input, hidden and output layer, respectively is able to perform following two experiments. First experiment contains a dataset of five faces of same person and another dataset contains a set of five faces of different persons with different orientation and poses (Tripathi and Kalra, 2011a, 2012). After learning of network estimates the learning parameters (weights and biases) and that will be used for testing of rest of the faces. The first experiment is performed on dataset containing 05 faces of same person with different orientation and poses; the learning of QDNN is done with one face Fig. 7a and testing over all faces. Each 3D face consists of 4654 points cloud data. Table 1 presents the testing MSE (Mean Square Error) of each face. This table shows that, the testing error of all five faces comparable to each other which demonstrate that they are faces of same person irrespective of minor variations in face orientation and poses. These results infer the straightforward learning and generalization capability of a simple quaternionic domain neural network structure.

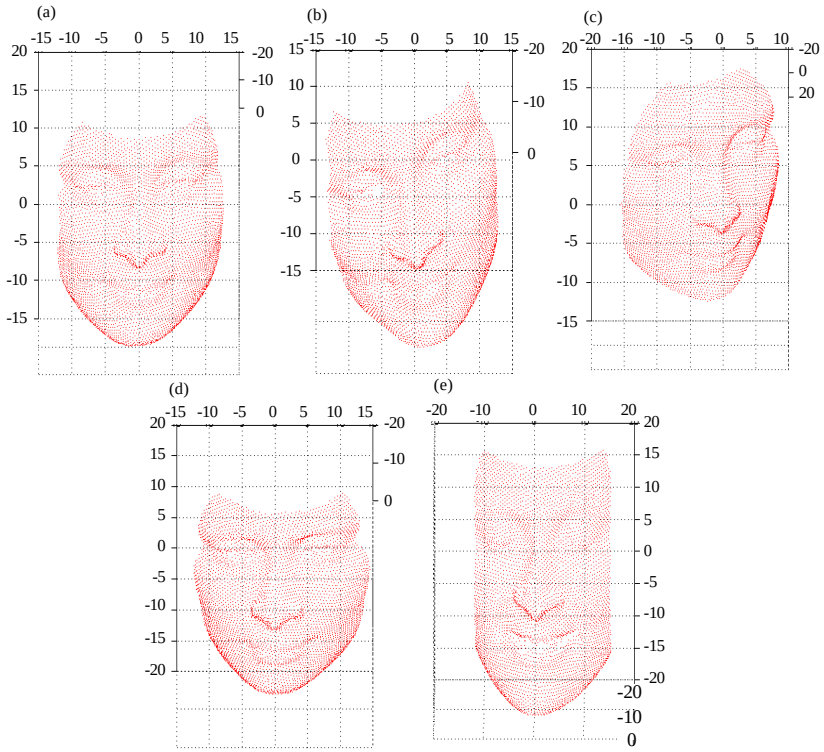

Fig. 7: Five 3D faces of same person with different orientation and poses

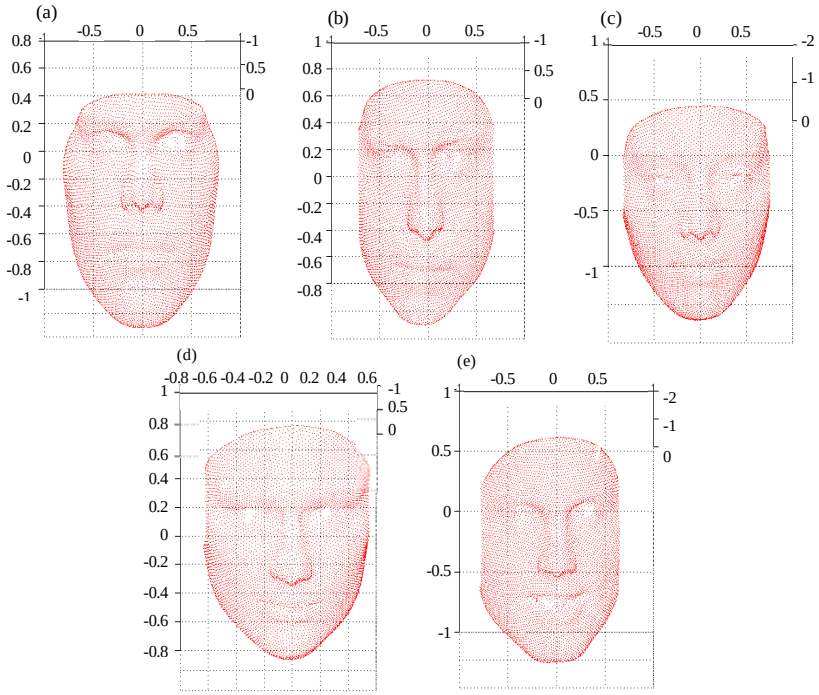

Fig. 8: a-e) Five 3D faces of different persons

Similarly, the second experiment is performed on dataset containing 05 faces of different person; the learning of QDNN is done with one face Fig. 8a and testing over all rest faces. Each 3D face consists of

Table 1: Comparison of testing error of each face of same person with different orientation and poses

MSE training (Target error) = 0.00011

Face (Figures)	Test error
7a	2.4842e-04
7b	3.5431e-03
7c	5.1153e-03
7d	4.5212e-04
7e	3.9148e-04

Table 2: Comparison of testing error of each face of different person

MSE training (Target error) = 0.00011

Face (Figures)	Test error
8a	1.8214e-04
8b	8.1344e-01
8c	3.5709e-00
8d	6.2814e-02
8e	3.1738e-01

6397 points cloud data. Table 2 presents the testing MSE of each face computed from trained network. This table shows that, the testing error of all five faces; the MSE of other four faces are much higher in comparison to the face Fig. 8a which is used in training. This demonstrates that the network correctly classifies the faces of same or different person with different orientation and poses. These results also again infer the learning and generalization capability of a simple quaternionic domain neural network structure (Cui *et al*., 2014).

CONCLUSION

In this study, we present an efficient and generalized learning algorithm for 3D imaging and evaluate it with the operations like generalization of motion and object recognition in space. A simple structure of QDNN learns the compositions of three basic transformations (translation, scaling and rotation) through input-output mapping over a small set of points on line. The trained system has proved its generalization ability over complex non-linear three dimension geometrical structure such as sphere, cylinder and torus. The simplicity of network structure arises due to typical computational power of quaternionic neurons which is not possible through complex or real-valued neuron. Thus, it does not improve the classification power in 3D imaging but also results in faster convergence. The network designed by complex or real-valued neuron is not able to learn 3D affine transformation because these neurons are not able to learn magnitude and direction of each component simultaneously at 3D space. The quaternion is the number which possesses the magnitude of intended components and phase information of each component is embedded in it.

RECOMMENDATIONS

Further, a simple QDNN has demonstrated its learning and generalization ability for the recognition operations over 3D face images. As biometrics application, a simple three layer 1-2-1 QDNN is employed to classify the faces of same or different persons with different orientation and poses. This study presents the possible directions of quaternionic neuron in 3D motion interpretation, computer vision, acoustic signal processing, etc.

REFERENCES

Cui, Y., K. Takahashi and M. Hashimoto, 2014. Remarks on quaternion neural network based controller with application to an inverted pendulum. Proceedings of the 2014 Annual Conference on SICE (SICE'14), September 9-12, 2014, IEEE, Sapporo, Japan, ISBN: 978-1-4799-6548-9, pp: 137-142.

Hamilton, W.R., 1853. Lectures on Quaternions. Hodges and Smith, Dublin, Ireland, Pages: 745.

Hoggar, S.G., 1992. Mathematics for Computer Graphics. Cambridge University Press, Cambridge, Massachusetts, USA.,.

Isokawa, T., H. Nishimura and N. Matsui, 2012. Quaternionic multilayer perceptron with local analyticity. Inf., 3: 756-770.

Isokawa, T., T. Kusakabe, N. Matsui and F. Peper, 2003. Quaternion Neural Network and its Application. In: Knowledge-Based Intelligent Information and Engineering Systems, Palade, V., R.J. Howlett and L. Jain (Eds.). Springer, Berlin, Germany, ISBN:978-3-540-40804-8, pp: 318-324.

Kuipers, J.B., 1998. Quaternions and Rotation Sequences: A Primer with Applications to Orbits, Aerospace and Virtual Reality. Princeton University Press, Princeton, New Jersey, USA.,.

Muezzinoglu, M.K., C. Guzelis and J.M. Zurada, 2003. A new design method for the complex-valued multistate Hopfield associative memory. IEEE. Trans. Neural Netw., 14: 891-899.

Nitta, T. and H.D. Garis, 1992. A 3D vector version of the back-propagation algorithm. Proceedings of the International Joint Conference on Neural Networks (IJCNN'92), November 3-6, 1992, The International Neural Network Society (INNS), Bandera County, Texas, pp: 511-516.

Nitta, T., 1995. A quaternary version of the back-propagation algorithm. Proceedings of the 1995 IEEE International Conference on Neural Networks Vol. 5, November 27-December 1, 1995, IEEE, Perth, Western Australia, ISBN:0-7803-2768-3, pp: 2753-2756.

Nitta, T., 1997. An extension of the back-propagation algorithm to complex numbers. Neural Netw., 10: 1391-1415.

Nitta, T., 2000. An analysis of the fundamental structure of complex-valued neurons. Neural Process. Lett., 12: 239-246.

Nitta, T., 2004. A solution to the 4-bit parity problem with a single quaternary neuron. Neural Inf. Process. Lett. Rev., 5: 33-39.

Tripathi, B.K. and P.K. Kalra, 2011b. On efficient learning machine with root-power mean neuron in complex domain. IEEE. Trans. Neural Netw., 22: 727-738.

Tripathi, B.K. and P.K. Kalra, 2011a. On the learning machine for three dimensional mapping. Neural Comput. Appl., 20: 105-111.

Tripathi, B.K. and P.K. Kalra, 2012. On the Intelligent Machine Learning in Three Dimensional Space and Applications. In: Engineering Applications of Neural Networks, Jayne, C., S. Yue and L. Iliadis (Eds.). Springer, Berlin, Germany, ISBN:978-3-642-32908-1, pp: 375-384.

Tripathi, B.K., 2014. High Dimensional Neurocomputing: Growth, Appraisal and Applications. Springer, London, UK., ISBN:978-81-322-2073-2, Pages: 164.

Tripathi, B.K., B. Chandra, M. Singh and P.K. Kalra, 2011. Complex generalized-mean neuron model and its applications. Applied Soft Comput., 11: 768-777.

Ujang, B.C., C.C. Took and D.P. Mandic, 2011. Quaternion-valued nonlinear adaptive filtering. IEEE. Trans. Neural Netw., 22: 1193-1206.

Wang, M., C.C. Took and D.P. Mandic, 2011. A class of fast quaternion valued variable stepsize stochastic gradient learning algorithms for vector sensor processes. Proceedings of the 2011 International Joint Conference on Neural Networks (IJCNN'11), July 31-August 5, 2011, IEEE, San Jose, California, ISBN:978-1-4244-9635-8, pp: 2783-2786.