Volume 5, Issue 9, September 2015 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Gesture Control Interface Using Machine Learning Algorithms

Harjatin Singh Baweja*
Computer Science, VIT University
India

Tanvir Parhar
Electronics, VIT University
India

Srikanth Malla
Electronics, VIT University
India

Abstract— Human robot interaction plays a crucial role in many robotic applications like industrial and surgical robots. To accurately classify the gestures can be challenging task in real time scenarios. This paper presents a novel approach for gesture interface between Kinect sensor and Robots. Comparative study has been done on different machine learning(ML) algorithms to find the best approach for gesture control among various ML algorithms.

Keywords— Feed-Forward Neural-Networks, Kinect, Red Green Blue Depth(RGBD) camera, Extreme Learning Machine(ELM), Recurrent Neural Network(RNN), Support Vector Machine(SVM)

I. INTRODUCTION

Hand gestures are an intimate part of human communication. They are used to convey a message in a short, precise and natural manner. Hand gestures are a more intuitive and natural way of communicating short messages. With the increase in the computational power and technology, robots are being used in a closer vicinity to humans than ever before. Hence, various methods of human-robot interaction have been developed for a more natural and fluent interaction between humans and machines.

The interaction techniques vary from voice recognition to gesture detection and recognition. Hand gestures may be used in environments where it is not feasible to communicate normally or through speech. For example in a virtual reality environment or in industrial environment.

Image processing techniques like HAAR cascading [1] and HOG cascading [2] are available and have been successfully implemented. But they have drawbacks like low accuracy and computationally expensive. From image processing to ultrasonic ranging[3], there are various techniques available for gesture recognition. But none is perfect. While gesture recognition using an array of ultrasonic sensors might not be accurate for intricate gestures, techniques like image processing prove to be computationally very intensive.

Hence gesture recognition using Kinect gives an optimum solution, which is cost-effective, computationally less intensive and accurate. Kinect is a cheap RGB plus Depth camera. It generates the 3-D disparity map of its surroundings in the form of colored point clouds. Each frame consists of as many as 300,000 points. Thus the 3-D map, produced by Kinect can be used to detect the position of human body parts in the 3-D space, in real time. The position of the body part which is being used to make gestures is used to generate the data-sets for various gestures. These data-sets are used to train Machine Learning algorithms. Once trained, the 3-D position data is given to the algorithm, as inputs for gesture recognition

II. WORKING PRINCIPLE OF KINECT AND SKELETON TRACKING

The Kinect is a RGB plus Depth camera. It gives information of the surroundings in the form of 3-D point clouds. The depth camera provides the position information of the point cloud whereas Red Green Blue (RGB) camera gives the color to the point cloud.

A. DEPTH CALCULATION

The IR emitter on-board the Kinect projects the environment with a mesh of IR markers. Each individual IR marker is a unique collection of IR speckles shown in Fig.1. The IR receiver in the Kinect then differentiates each speckle pattern as an individual marker.

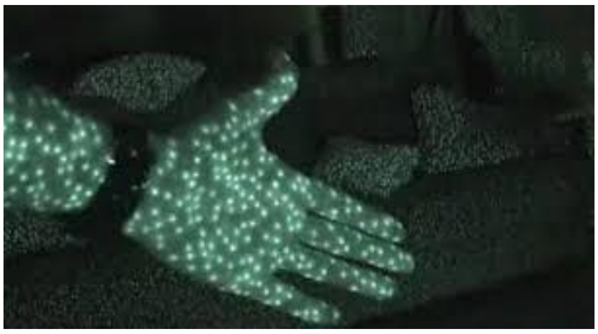
Fig. 1. The room being illuminated with IR markers

The measurement of depth in Kinect is more or less a triangulation process. The Kinect consists of two layers, the reference layer and the object layer. Reference layer is a known pattern of IR markers that has been calibrated at a fixed distance. The Object layer is the layer that consists of markers, lying on the object in front of the camera. The markers in the object layer might be farther away or nearer to the camera as compared to the respective markers in the reference layer. These positional changes for each marker are measured, to form a disparity image.

B. DEPTH CALCULATION OF A SINGLE POINT

For depth calculation at a single point, let us assume a point k in the object plane. Whenever the distance of the speckle 2 form the camera changes, its position in the focal plane will change in x direction. This change in distance, d is called the disparity of that particular marker. This change in disparity is used to find the depth of the point.

Here, C is the camera that detects the markers, L is the laser source, D is the 3-D disparity, Z0 is distance of the camera from the Reference plane, Zk is the distance of the point k form the camera. Fig. 2 depicts the distance calculation for a single marker, with triangulation.

$$\frac{D}{b} = \frac{Z_0 - Z_k}{Z_0} \qquad (1)$$

$$\frac{d}{f} = \frac{D}{Z_k} \qquad (2)$$

By solving (1) and (2)

$$Z_k = \frac{Z_0}{1 + \frac{Z_0}{f} d} \qquad (3)$$

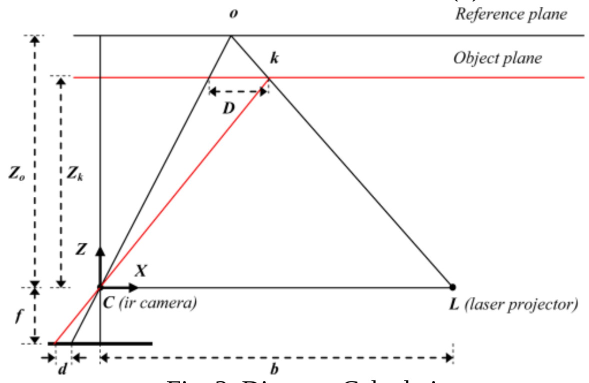

Fig. 2. Distance Calculation

The same process is repeated for all the markers. Once, the distances of all the markers are triangulated, then a disparity map, shown below is produced[Fig. 3].

Once the depth cloud is attained, an open-source software called NITE is used to interpret the point-clouds, that look similar to human figure, into human skeletons. Hence, the skeletal information is attained. The NITE toolbox publishes 3-D position information about various body parts, like hands, elbows, knees, etc. The tracked skeleton looks similar to the image below. In our case, we take the 3-D positional data of the right hand and train the algorithms[Fig. 4].

III. DATA PRE-PROCESSING AND TRAINING

A. FEATURE NORMALIZATION

To cope with unaccounted variances in the data set some standardization procedure needs to be implemented. Thus normalization is performed on the input data set, so that each feature contributes approximately, proportionately to the

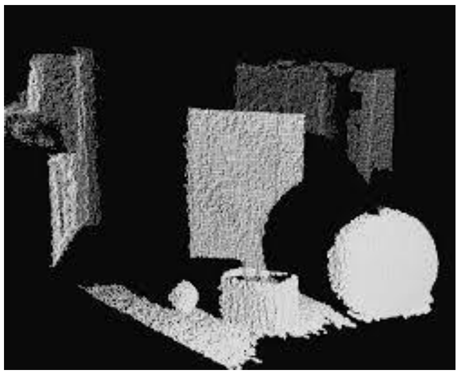

Fig. 3. The Disparity Map

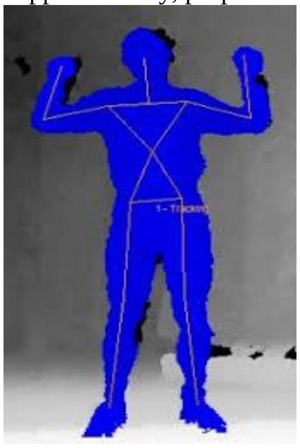

Fig. 4. Tracked Human Skeleton output of the classifier.

$$x^1 = (x - \mu)/\sigma \qquad (4)$$

x : feature
x^1 : normalized feature
μ : mean of feature x
σ : standard deviation of feature x

B. BACK PROPAGATION

The 3D vectors are computed from the time series 3D points, generated by Kinect. The x, y, and z components of the vectors are fed individually to the input neurons. To account for high non-linearity, the network consists of 2 hidden layers with 3 30 and 10 neurons respectively[Fig. 5]. The back-propagation algorithm aligns the weights such that the error minimizes with each iteration.[4] It uses gradient descent algorithm iteratively, till a minima is attained. The drawback is, it sometimes get stuck in local minima.

$$w_{t+1} = w_t - n(dE/dw_t) \qquad (5)$$

Where n is the learning rate, t is time

$$E \text{ is the output error} = 0.5 * (y - t)^2 \qquad (6)$$

Figure 6 depicts the decrement in error for both test and training data, with recursive training iterations for Back Propagation Algorithm

C. RECURRENT NEURAL NETWORK

The drawback with the feed-forward network is, all the data points have to be fed to the network at once. Hence dataset needs to be collected for a span of time, which leads to discontinuity in gesture recognition. Whereas in a recurrent neural network, the time series data can be fed continuously [5]. The network keeps an instance of data, in dynamic memory. This leads to faster and more accurate response. Fig.7 denotes the general architecture of a recurrent Neural-Network. The x, y and z coordinates of the gesture are fed to the three different input neurons, as a time series. The recurrence takes place in the second layer.

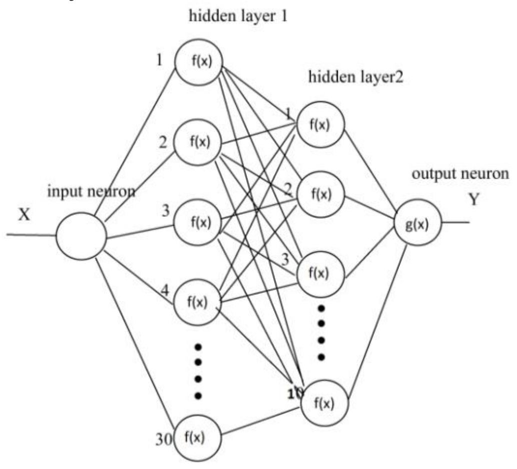

Fig. 5. single hidden layer feed forward neural network structure used to train gesture control

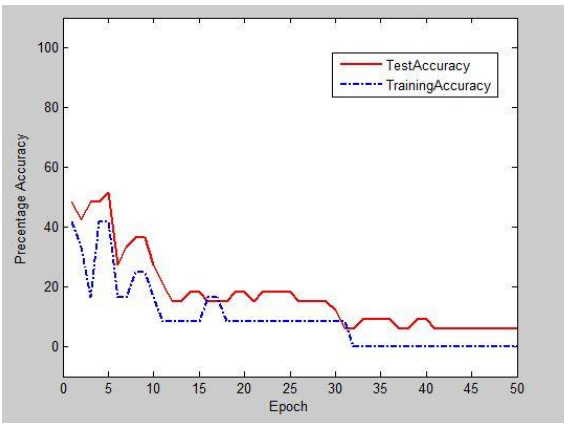

Fig. 6. Error vs Epoch in Feed-Forward Neural-Network

$$\tau \dot{y}_i = -y_i + \sigma\left(\sum_{j=1}^{n} W_{ij}\, y_i - \theta_j\right) + I_i(t) \qquad (7)$$

Where,
τi : Time constant of postsynaptic node
yi : Activation of postsynaptic node
y˙i: Rate of change of activation of postsynaptic node
wji : W eight of connection from pre to postsynaptic node
σ(x) : Sigmoid of x, i.e., σ(x) = 1/(1 + e^{-x})
yj : Activation of presynaptic node
θj : Bias of presynaptic node
Ii(t) : Input (if any) to node

Fig.8 depicts the Error vs Epoch graph for the training, test and validation data respectively. It can be seen that more epochs are required to train the Recurrent Neural-Network, as compared to Feed-Forward Network.

D. ELM TRAINING

It's a single step training algorithm. Taking input as pose matrix x and feeding it through the input neuron. The hidden outputs, lets say matrix H and the output of the neural network be y and the target be t, The hidden weights are assigned randomly and the output weights are found with the following least squares[7] technique as shown below

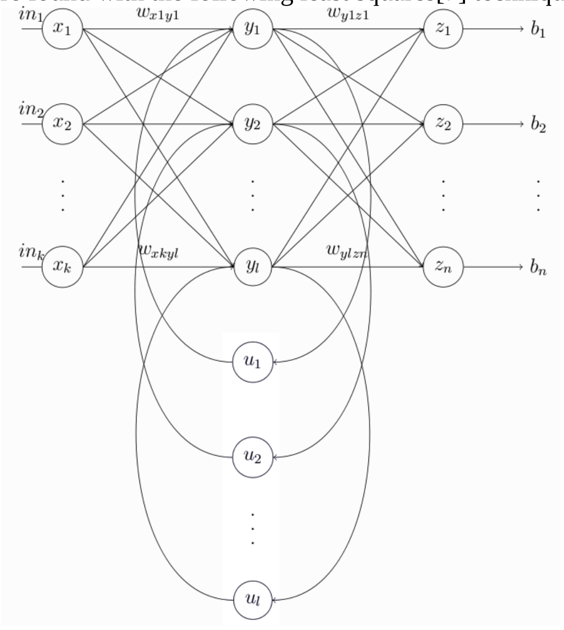

Fig. 7. The Recurrent Neural Network Architecture

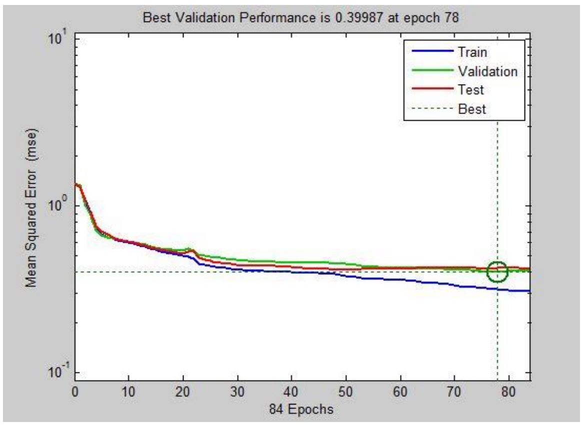

Fig. 8. Error vs Epoch in Recurrent Neural-Network

$$H = x * w_{hidden}^{T} + b_{Hidden} \quad (8)$$

$$w_{out}^{t} = H^{+} * (y - b_{out}) \quad (9)$$

E. SUPPORT VECTOR MACHINE

The drawback of neural networks is the input data travels through the whole network for generation of output. This is time consuming and computationally expensive. Support vector machines, on the other hand construct a hyperplane for data classification. Three classifiers are made and trained to differentiate between three gestures. Since the relation is highly non-linear, so Gaussian radial basis function is used as a kernel for the SVM[7].

$$k(x_i, x_j) = \exp(-\gamma||x_i - x_j||^2) \quad (10)$$

Where, γ is normalization factor.

Three such Support Vector Machines are made for differentiating between three different gestures. Fig. 9 denotes the contour plot for the first gesture. The inputs that are marked as green are classified as positive by the SVM and the red ones are classified as negative.

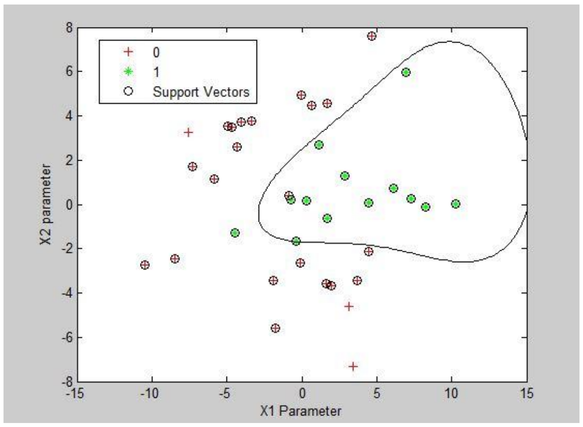

Fig. 9. Contour Graph for First gesture

IV. CONCLUSION

Hence gesture recognition is implemented and the results are compared amongst various machine learning algorithms. The Feed-Forward neural network has the highest accuracy of 93.3%. This is because the Neural-Network gets the input vectors of the complete gesture at once. This induces a lag, equivalent to the time taken to record a gesture. Akin to Feed-Forward neural network SVM also gets the input vectors of the complete gesture at once. The inferior accuracy of SVM, around 83.3%, can be accounted to the fact that it uses a predefined kernel. However the computational speed of an SVM is faster than that of Feed-Forward Neural Network. Recurrent Neural-Network takes the time series data. Hence Gesture recognition is fast and real time. It has a dynamic memory, to keep into account the previous data of the series. Since, all the data is not given to the network at once, the accuracy, plummeting down to 72.3% is lower as compared to it's Feed-Forward counterpart. ELM has a lower accuracy, as it trains only the final layer. It has a tendency of getting stuck in a local minima or get over fitted. But the computational time is very less, as compared to other techniques. It is a single iteration training algorithm. The accuracy is the least, around 66.7%.

Table I. Table showing accuracies and average time taken

Training algorithm	Accuracy (in %)	Time Taken(sec)
ELM	**66.67**	**3.0003**
SVM	**83.33**	**3.00011**
RNN	**72.64**	**0.0063**
Back Propagation	**93.93**	**3.00061**

REFERENCES

[1] Kar, Abhishek. "Skeletal tracking using microsoft kinect." Methodology 1 (2010): 1-11.
[2] Zhu, Qiang, et al. "Fast human detection using a cascade of histograms of oriented gradients." Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on. Vol. 2. IEEE, 2006.
[3] Zimmerman, Thomas G., et al. "A hand gesture interface device." ACM SIGCHI Bulletin. Vol. 18. No. 4. ACM, 1987.
[4] Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams.Learning internal representations by error propagation. No. ICS-8506. CALIFORNIA UNIV SAN DIEGO LA JOLLA INST FOR COGNITIVE SCIENCE, 1985.
[5] Funahashi, Ken-ichi, and Yuichi Nakamura. "Approximation of dynamical systems by continuous time recurrent neural networks." Neural networks 6.6 (1993): 801-806.
[6] Huang, Guang-Bin, Qin-Yu Zhu, and Chee-Kheong Siew. "Extreme learning machine: theory and applications." Neurocomputing 70.1 (2006): 489-501.
[7] Cortes, Corinna, and Vladimir Vapnik. "Support-vector networks." Machine learning 20.3 (1995): 273-297.

BIBLIOGRAPHY

All authors have contributed equally in this research paper.