3D RECOGNITION USING NEURAL NETWORKS

M. Elhachloufi, A. El Oirrak, D. Aboutajdine & M.N. Kaddioui
Faculté des sciences Semlalia, Dept Informatique, Marrakech, Maroc
Faculté des sciences, LEESA-GSCM, BP 1014, RABAT
(E-mail: elhachloufi@yahoo.fr1, oirrek@yahoo.fr2, aboutaj@fsr.ac.ma3, kaddioui@yahoo.fr4)

Abstract: With the advent of the Internet, exchanges and the acquisition of information, description and recognition of 3D objects have been as extensive and have become very important in several domains, which require the establishment of methods to develop description and recognition techniques to access intelligently to the contents of these objects.

This paper deals for 3D models recognition. Thus under general affine transform we propose an approach based on neural network.

The recognition is done by measuring the similarity between a sample of object and its transformed obtained by parameters extracted from neural networks using the euclidean distance.

Keywords: 3D Object, recognition, neural networks, affine transformed.

1. INTRODUCTION

Databases of 3D objects that are used in different fields (e-commerce, gaming, medical, etc.) become bigger, which mandates the establishment of effective methods for users to find the corresponding 3D objects to the request object.

In this context, several research systems by the content search engines and 3D models are available on the web [1] [2] [3]. Many methods have been developed in this direction.

The shape spectrum descriptor proposed by Zaharia et al based on surface geometry, is recommended by MPEG-7 [4].

Filali et al. proposed the descriptor based on 2D views named Adaptive Views Clustering (AVC). It is a probabilistic Bayesian method that selects the most interesting views from several views of a 3D model [1].

Based on the statistics, Osada et al. proposed the descriptor named shape distribution (D2) [5]. Paquet et al. proposed the method of cord histograms [6].

The ratio area/volume is used as a feature vector to describe the 3D models by Zhang and Chen [7]. Even if this descriptor computes the feature vectors easily and quickly, it needs a high quality of mesh.

Vranic and Saupe [8] proposed the method named Ray based descriptor. It uses the extents obtained from the center of mass of the model to intersection with its surface in directions which are constructed by an icosahedron. This approach is not robust to noise and needs a high dimension of feature vectors, this is why the authors construct the feature vectors from the complex function on a sphere, composed with Ray based feature vectors and shading based feature vectors, presented in frequency domain by applying the spherical harmonics.

2. REPRESENTATION OF 3D OBJECTS

3D object is represented by a set of points denoted \( M = \{P_i\}_{i=1...n} \) where \( P_i = (x_i, y_i, z_i) \in \mathbb{R}^3 \), arranged in a matrix \( X \). Under the action of an affine transformation, the coordinates \((x, y, z)\) are transformed into other coordinates \((\tilde{x}, \tilde{y}, \tilde{z})\) by the following equation:

\[
\begin{align*}
  f: \mathbb{R}^3 & \rightarrow \mathbb{R}^3 \\
  (x(t), y(t), z(t)) & \rightarrow f(X) = Y(\tilde{x}(t)\tilde{y}(t), \tilde{z}(t)) \\
  Y & = A \times X (x(t), y(t), z(t)) + B
\end{align*}
\]

With \( A \) invertible matrix and \( B \) is a vector translation.
3. THE ARTIFICIAL NEURAL NETWORKS

The artificial neural networks (ANN) are mathematical models inspired by the structure and behavior of biological neurons [9]. They are composed of interconnected units called artificial neurons capable of performing specific and precise functions [10].

ANN can approximate nonlinear relationships of varying degrees of complexity and significant to the recognition and classification of data. Figure 1 illustrates this situation.

![Black Box of Artificial Neural Networks](image1)

**Figure 1:** Black Box of Artificial Neural Networks.

3.1. Architecture of Artificial Neural Networks

For an artificial neural network, each neuron is interconnected with other neurons to form layers in order to solve a specific problem concerning the input data on the network [11, 12].

The input layer is responsible for entering data for the network. The role of neurons in this layer is to transmit the data to be processed on the network.

The output layer can present the results calculated by the network on the input vector supplied to the network. Between network input and output, intermediate layers may occur; they are called hidden layers. The role of these layers is to transform input data to extract its features which will subsequently be more easily classified by the output layer.

In these networks, information is propagated from layer to layer, sometimes even within a layer via weighted connections.

A neural network operates in two consecutive phases: a design phase and use phase. The first step is to choose the network architecture and its parameters: the number of hidden layers and number of neurons in each layer. Once these choices are fixed, we can train the network. During this phase, the weights of network connections and the threshold of each neuron are modified to adapt to different conditions of input. Once the training on this network is completed, it goes into use phase to perform the work for which it was designed.

3.2. Multilayer Perceptron

For a multilayer network, the number of neurons in the input layer and output layer is determined by the problem to be solved [13], [11], [12]. The architecture of this type of network is illustrated in Figure 2.

According to R. LEPAGE and B. Solaiman [13], the neural network has a single layer with a hidden number of neurons approximately equal to: 

\[ J = 1 + \sqrt{N(M + 2)} \]

where:

- \( N \) : number of input parameters.
- \( M \) : the number of neurons in the output layer.

![Architecture of a Multilayer Perceptron Network](image2)

**Figure 2:** Architecture of a Multilayer Perceptron Network.

3.3. Learning

The learning algorithm used is the gradient back propagation algorithm [14]. This algorithm is used in the feed forward type networks, which are networks of neurons in layers with an input layer, an output layer, and at least one hidden layer. There is no recursion in the connections and no connections between neurons in the same layer.

The principle of backpropagation is to present the network a vector of inputs to the network, perform the calculation of the output by propagating through the layers, from the input layer to the output layer through hidden layers.

This output is compared to the desired output, an error is then obtained. From this error, is calculated the gradient of the error which in turn is propagated from the output layer to the input layer, hence the term back propagation.

This allows modification of the weights of the network and the refire learning. The operation is repeated for each input vector and that until the stop criterion is verified [15].
3.4. Learning Algorithm

The objective of this algorithm is to minimize the maximum possible error between the outputs of the network (or calculated results) and the desired results.

We spread the signal forward in the layers of the neural network: \( x^{(n-1)} \rightarrow x^{(n)} \). The spread forward is calculated using the activation function \( g \), the aggregation function \( h \) (often a scalar product between the weights and the inputs of the neuron) and synaptic weight \( w_{jk} \) between the neuron \( x_k^{(n-1)} \) and the neuron \( x_j^{(n)} \) as follows:

\[
x_j^{(n)} = g^{(n)}(h_j^{(n)}) = g^{(n)} \left( \sum_k w_{jk} x_k^{(n-1)} \right)
\]

When the forward propagation is complete, we get the output result \( y \).

It then calculates the error between the output \( y \) given by the network and the desired vector \( s \).

For each neuron \( i \) in output layer is calculated:

\[
e_i = g'(h_i^{(n)}) [s - y]
\]

It propagates the error backward \( e^{(n)} \rightarrow e^{(n-1)} \) through the following formula:

\[
e_j^{(n-1)} = g'(h_j^{(n-1)}) \sum_i w_{ij} e_i^{(n)}
\]

It updates the weights in all layers:

\[
\Delta w_{ij}^{(n)} = \lambda e_i^{(n)} X_j^{(n-1)}
\]

where \( \lambda \) is the learning rate (of low magnitude and less than 1.0).

4. RECOGNITION FOR 3D VOLUME USING NEURAL NETWORKS APPLIED ON 2D SLICE

Typical scalar volume data is composed of a 3-D array of data and three coordinate arrays of the same dimensions. The coordinate arrays specify the \( x \), \( y \), and \( z \)-coordinates for each data point. The units of the coordinates depend on the type of data. For example, flow data might have coordinate units of inches and data units of psi. Slice planes provide a way to explore the distribution of data values within the volume by mapping values to colors.

We can orient slice planes at arbitrary angles, as well as use nonplanar slices. In this case each volume \( V \) is characterized by a set of slices, each slice is a 2D image having coordinates \( (x, y) \) and color information.

In this experiment only slices 1, 8 and 27 are considered. For each one six normalized color coefficients are extracted.

This technique suppose that \( V \) and \( \tilde{V} \) are stocked with the same number of slices. We can use another technique based on isosurface [18]. An isosurface define the 3D volume bounded by a particular isovalue. The volume inside contains values greater (or less) than the isovalue. The volume outside contains values less (or greater) than the isovalue. The isovalue can be an interval \([\text{min}, \text{max}]\) in this case the isosurface is a list of all cells for which the isovalue is contained in the interval \([\text{min}, \text{max}]\). Also the choice of isovalue on \( V \) and \( \tilde{V} \) is another problem. Fig. (4) shows some subvolumes obtained by this technique.

![Figure 3: Subvolumes Obtained by Isosurface Technique.](image)

Divided 3D volume \( V \) and it’s transformed \( \tilde{V} \) into five slices \( (S_1, ..., S_5) \) respectively \( (\tilde{S}_1, ..., \tilde{S}_5) \) as shown in figure 5 and figure 6. Those volumes figure 1 are initially stocked into 24 slices.

![Figure 4: 3D Volume and Its Transformed. (a) MRI Volume. (b) Transformed Volume.](image)

![Figure 5: Five 2D Slices of 3D Volume.](image)
Figure 6: Five 2D Slices of Transformed 3D Volume.

we take as a first step two points:

\[ p_i = (x_i, y_i) \in S \quad \text{and} \quad \tilde{p}_i = (\tilde{x}_i, \tilde{y}_i) \in \tilde{S} \]

After we study the relation between \( p_i \) and \( \tilde{p}_i \). To do this we extract the parameters \( a_0 \) and \( b_0 \) that can transmit \( p_i \) and \( \tilde{p}_i \) as follows:

\[ \tilde{p}_i = a_0 p_i + b_0 \tag{5} \]

using neural networks as shown in the figure below:

\[ p_o \xrightarrow{RN} \tilde{p}_i \]

Figure 7: Parameters Extraction and \( a_0 \) and \( b_0 \).

\( p_o \): Represents the vector of input data.
\( \tilde{p}_i \): Represents the vector of output data.

In the second step, we calculates the points \( \tilde{p}_j \) using the previously extracted parameters \( a_0 \) and \( b_0 \) by the formula (1) as follows:

\[ \tilde{p}_j = a_0 p_j + b_0 \tag{6} \]

then we proceed to calculate the errors \( \text{err}_j \) defined as follows:

\[ \text{err}_j = \tilde{p}_j - \tilde{p}_j \tag{7} \]

corresponding to the pairs of samples \((p_j, \tilde{p}_j)\) with \( j = 1, ..., 5 \).

Table 1

<table>
<thead>
<tr>
<th>(( p_j, \tilde{p}_j ))</th>
<th>( \text{err}_j )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( (p_1, \tilde{p}_1) )</td>
<td>0.0008 ( (10^{-4}) )</td>
</tr>
<tr>
<td>( (p_2, \tilde{p}_2) )</td>
<td>0.0007 ( (10^{-4}) )</td>
</tr>
<tr>
<td>( (p_3, \tilde{p}_3) )</td>
<td>0.00013 ( (10^{-4}) )</td>
</tr>
<tr>
<td>( (p_4, \tilde{p}_4) )</td>
<td>0.00024 ( (10^{-4}) )</td>
</tr>
<tr>
<td>( (p_5, \tilde{p}_5) )</td>
<td>0.00037 ( (10^{-4}) )</td>
</tr>
</tbody>
</table>

According to the table above we concluded that all points of \( S(i = 1, ..., 5) \) are converted into points of \( \tilde{S}(i = 1, ..., 5) \) by the same parameters \( a_0 \) and \( b_0 \) \( \Rightarrow \) \( \tilde{S} \) is an affine transformation \( S \Rightarrow \tilde{V} \) is an affine transformation \( V \).

6. COMPARISON

According to the figures below we can see that the \( \text{err}_i \) \((i = 1, ..., 5)\) - corresponds to the difference between the \( S(i = 1, ..., 5) \) of the slice of 3D volume and its transformation \( \tilde{S}(i = 1, ..., 5) \) by an affine transformation- calculated by our method (table 1) is smaller than those obtained by the methods of Fourier transform and Moments (figures 8 and figure 9).

In addition, the computing time of our method is less than the method of Fourier transform and Moments. This leads us to conclude that our method applied to these objects is more efficient than Fourier transform and Moments.

Figure 8: Error\(^4\) Representation by Fourier Transform.

Figure 9: Error\(^4\) Representation by Method Moments.

7. CONCLUSION

In this work, we have developed an approach for recognizing 3D objects based on neural networks. The principal advantages of proposed approaches is the possibility to handle affine transform and 3D volume.

The simulation results were presented and an evaluation of the designed system has been made. They were generally satisfactory and show the validity of the proposed approach.
In future the 3D search engine can be extended for medical databases (MRI magnetic resonance imaging), this data typically contains a number of slice planes taken through a volume, such as the human body.

REFERENCES