2016 23rd International Conference on Pattern Recognition (ICPR)
Cancún Center, Cancún, México, December 4-8, 2016

Face Verification with Three-Dimensional Point Cloud by Using Deep Belief Networks

Dong-Han Jhuang, Daw-Tung Lin*, and Chi-Hung Tsai
Department of Computer Science and Information Engineering
National Taipei University
New Taipei City, Taiwan
*Corresponding Author: dalton@mail.ntpu.edu.tw

***Abstract*—Developing reliable and robust face verification systems has been a tough challenge in computer vision, for several decades. The variation in illumination and head pose may seriously inhibit the accuracy of two-dimensional face recognition. With the invention of a depth map sensor, more three-dimensional volume data can be processed to mitigate the problem associated with face verification. This paper presents a three-dimensional face verification approach that includes three phases. First, point cloud library is applied to estimate features such as normal vectors and principal curvatures of every point on a human face point cloud acquired from three-dimensional depth sensor. Next, we adopt deep belief networks to train the identification model using extracted features. Finally, face verification is accomplished by using the pre-trained deep belief networks to justify if new incoming face point cloud feature is the one we specified. The experimental results demonstrate that the proposed system performs exceptionally well with about 96.43% verification accuracy.**

***Keywords*—face verification, 3D point cloud, feature extraction, principal curvature estimation, deep belief networks.**

I. Introduction

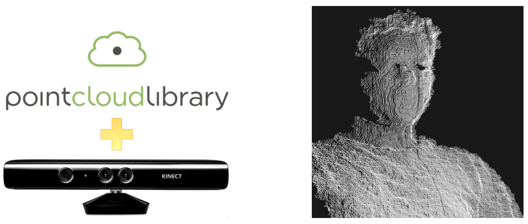
Face recognition plays a critical role in the field of computer vision, particularly in security surveillance applications. Recently, deep learning in neural networks has been extensively studied. Fu *et al.* provided brief introductions for determining the type of mechanism to use in development [1]. Typically, deep neural networks include three parts: deep generative architectures, convolutional neural networks, and auto-encoders. In this study, the deep generative architectures type has been applied because it can learn low-level features without overfitting. Furthermore, according to Bengio *et al.* [2], the deep neural network model can be trained in a more generative way without excessive time consumption. With the generative features learned in hidden layers, the deep neural networks can not only be used to classify different things, but be used to restore the data by using learned low-level features [3].

Several strategies were provided to address problems associated with three-dimensional (3D) face recognition [4]. Mohammadzade and Hatzinakos asserted that 3D information has a feature for face recognition exhibits high performance and the accuracy of correctness is extremely high [5]. Thakare and Thakare adopted the depth image features train the hybrid fuzzy neural network and verify the new incoming image [6]. From their experiments, neural networks have the advantage to recognize different things in an efficient way. Schimbinschi also demonstrated that 3D shape descriptors have exceptional ability for object classification [7]. Nair and Hinton introduced a third-order Restricted Boltzmann Machine Deep Belief Net combining gradients for generative and discriminative models to perform 3D object recognition tasks. [8]. In addition, further studies on face recognition without depth information have demonstrated that neural network model has exceptional ability to predict an unknown input feature vector [9], [10], [11]. Furthermore, Hinton mathematically described that a learning method with multiple-layered representation has exceptional ability to manage high-dimensional data [12] . Patil *et al.* used the frequency domain features in face recognition and exhibited high performance [13]. Apart from the problems associated with face recognition, Lv *et al.* solved the various poses problem by using various types of features to train deep belief networks (DBNs)[14]. Zhu *et al.* proposed a novel multi-view perceptron (MVP) deep neural network, which can emulate the behavior of human brain in encoding 3D face model given a single 2D face image and infer a full spectrum of multi-view images [15]. Taigman *et al.* improved face model alignment using generic 3D affine transformation and employed a nine-layer deep neural network to better represent the 3D face, and the evaluation performance reaches an accuracy of 97.35% on a large scale dataset LFW [16]. Li *et al.* used multiple-combined features including angles, triangle area, graph distance, and mean curvature as descriptors to do the recognition. Due to the variance of facial expression, low-level features were applied and the associated experimental results demonstrated a high recognition rate [17]. Berretti *et al.* proposed iso-geodesic stripes features to train the 3D face recognition model. The pre-processing methods were applied before the experiments which consist of hole interpolation and nose tip to crop the face region [18].

The rest of the paper is organized as follows. Section II describes the proposed method and the system architecture. Section III demonstrates the experimental results. Finally, Section IV offers the conclusion.

II. Methodology

With the development of a depth sensor and point cloud library (PCL), as shown in Figure 1a, the information of geometric coordinates of real-world objects can be easily acquired [19]. A point cloud is a data structure used to



(a) PCL and Kinect. (b) Example of 3D coordinate point cloud data.

Fig. 1. 3D point cloud data acquisition tools and a sample scan.

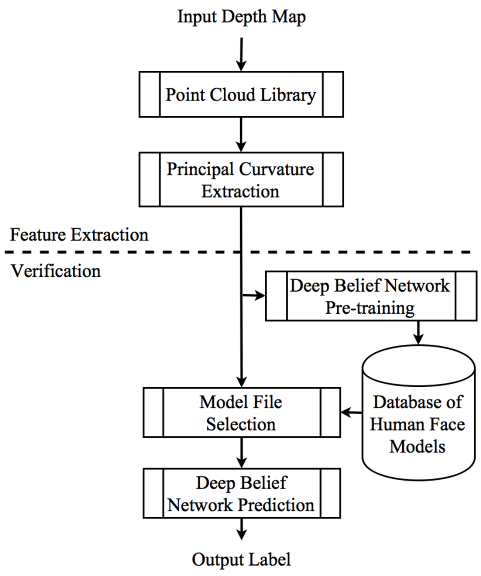

Input Depth Map
Point Cloud Library
Principal Curvature Extraction
Feature Extraction
Verification
Deep Belief Network Pre-training
Database of Human Face Models
Model File Selection
Deep Belief Network Prediction
Output Label

Fig. 2. System flowchart.

represent a set of multidimensional points and is typically used to represent 3D data. Figure 1b is a 3D point cloud data comprising coordinate information of each point which was obtained from the depth sensor. To solve the problem of object verification, we split our system architecture into three phases, namely feature extraction phase, training phase, and verification phase, respectively. The system flowchart is shown in Figure 2.

A. Feature Extraction Phase

The pre-process and feature extraction procedures for depth raw data are necessary so that the data can be analysed and transformed into useful information efficiently. Figure 3 presents the flowchart of this feature extraction phase.

The feature extraction procedure consists of five steps: range filtering, down sampling, normal vector estimation, principal curvatures estimation, and feature normalization. The details of each processing step is illustrated as follows.

1) Range Filtering and Down Sampling: Under the Point Cloud Library (PCL) developing environment, every point in the point cloud virtual space has its own coordinates which can be estimated and analysed. The region of interest (ROI) can be arbitrarily assigned depending on the applications.

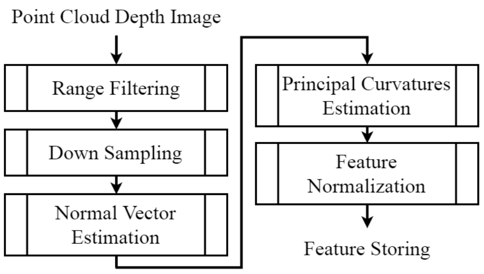


Fig. 3. Feature extraction phase flowchart.

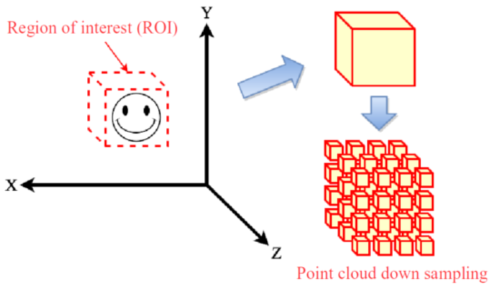


Fig. 4. Conceptual diagram of range filtering and down sampling.

Figure 4 shows the conceptual diagram of range filtering and down sampling. Briefly speaking, after specifying a fixed region by slicing ROI in the input space, we can filter out the unwanted portion and get the interested region of point cloud data. Figure 5a presents the original point cloud image and Figure 5b shows the corresponding output after applying range filter. Indeed, a frontal face detector such as Adaboost Haar detector can play the role of range filtering to localize face positions. For the sake of computation complexity, range filter is adopted in this study.

Based on our experiments, the parameters of coordinate x value is in the range from -0.2 to 0.2, coordinate y value is in the range from -0.2 to 0.2, and coordinate z value is in the range from 0.0 to 0.6. The physical volume of the ROI is around 24cm×24cm×36cm. Due to the constraint of computation time, down sampling procedure is necessary before the primary estimation of so many cloud points. After the range filter having been applied, the number of points in one input video frame is around 6000 to 8000. To make the computation

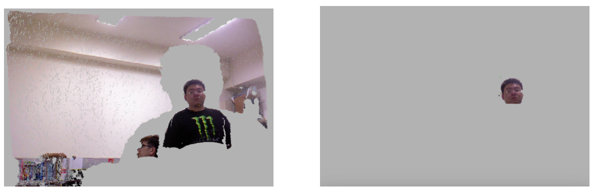

(a) Original point cloud image. (b) Image after range filtering.

Fig. 5. Example images of range filter.

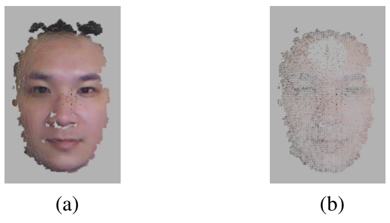


Fig. 6. Example images of down sampling: (a) Image before down sampling; (b) Image after down sampling.

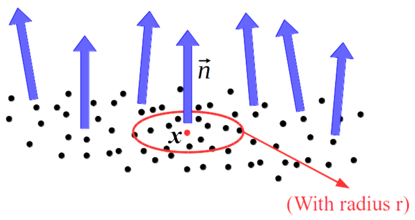


Fig. 7. Radius searching in PCL.

more efficient, the processed frame will be scanned by using a rectangular voxel window to decide whether each of the points should be retained or discarded. The right side in Figure 4 presents the concept of down sampling, Figure 6a displays the data before down sampling, and Figure 6b depicts the data after down sampling.

2) Normal Vector Estimation: Feature selection greatly influences the accuracy of verification [20]. We must first determine how to choose a discriminating feature. According to [21], least-square plane fitting estimation was used to determine the normal vector to a point on the surface of the object. Lots of methods can be applied to calculate the normal vectors. To ensure correctness and computational efficiency, the method of radius searching in neighbours of a specific point is chosen.

Figure 7 illustrates the aforementioned concept. Here, we use dot and arrow to represent a point $\boldsymbol{x}$ and its normal vector $\vec{\boldsymbol{n}}$, respectively. The distance function from a point $p_i \in \mathcal{P}^k$ to a plane is defined as $d_i = (p_i - \boldsymbol{x}) \cdot \vec{\boldsymbol{n}}$ where $\mathcal{P}^k$ represents all points within radius r in a point cloud having k points. According to the least-square method, we let $d_i = 0$ by using $\boldsymbol{x} = \bar{p} = \frac{1}{k} \cdot \sum_{i=1}^{k} p_i$ to estimate the centroid of $\mathcal{P}^k$.

3) Principal Curvatures Estimation: The vector value of $\vec{\boldsymbol{n}}$ can be deduced by analysing the eigenvalue of a covariance matrix $\mathcal{C} \in \mathcal{R}^{3\times 3}$:

$$\mathcal{C} = \frac{1}{k}\sum_{i=1}^{k} \xi_i \cdot (p_i - \bar{p}) \cdot (p_i - \bar{p})^T, \qquad \mathcal{C} \cdot \vec{v_j} = \lambda_j \cdot \vec{v_j},\ j \in \{0,1,2\} \tag{1}$$

where ξ_i represents the weight of p_i. According to principal component analysis, eigenvector $\vec{v_0}$ generally has the lowest eigenvalue λ_0, where $0 \le \lambda_0 \le \lambda_1 \le \lambda_2$. If $\lambda_0 = \min(\lambda_j)$, the surface curvature around point $\boldsymbol{p}$ can be inferred using eigen-analysis:

$$\sigma_p = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{2}$$

The value σ_p approximates the change of curvature at point $\boldsymbol{p} \in \mathcal{P}^k$ with a searching radius k around point $\boldsymbol{p}$. In order to fit the best tangent plane on point $\boldsymbol{p}$, the weight value ξ_i can be defined as follows:

$$\xi_i = \begin{cases} \exp(-\frac{d_i^2}{u^2}) & ,\boldsymbol{p}_i^k \text{ outlier} \\ 1 & ,\boldsymbol{p}_i^k \text{ inlier} \end{cases} \tag{3}$$

The value u is the mean distance from the query point $\boldsymbol{p}_x$ to all its neighbours in $\mathcal{P}^k$ and d_i is the distance from $\boldsymbol{p}$ to its $\boldsymbol{p}_i^k$ neighbours. We use estimated eigenvalue of each point to train the network model.

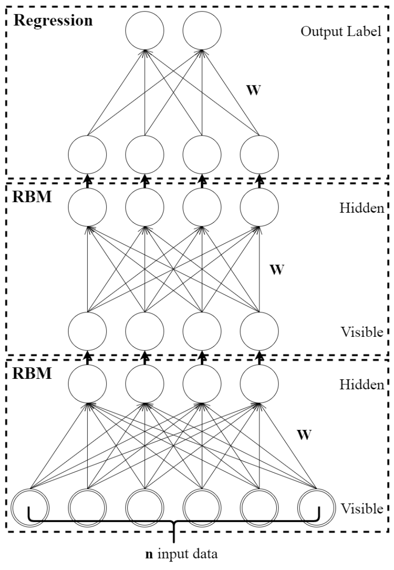


Fig. 8. Deep belief networks architecture.

B. Training and Verification Phases

The fundamental concept of the Boltzmann machine and energy function was proposed in [22]. According to [23], multiple layers of representation demonstrates high performance in model learning. Based on the improved training procedure reported in [24] and [25], we constructed a stacked restricted Boltzmann machine to train our model, which is described as follows:

1) Network Architecture: In this study, we designed a deep belief network (DBN) with one visible layer comprising n neurons, two hidden layers having $n/2$ neurons, and one regression layer comprising two output labels.

As shown in Figure 8, the number of neurons and therefore the training time depend on the size of the input vector.

2) Restricted Boltzmann Machine: The energy function is defined as:

$$E(v,h) = -\sum_{i \in visible} a_i v_i - \sum_{j \in hidden} b_j h_j - \sum_{i,j} v_i h_j w_{ij}, \tag{4}$$

where v_i and h_j denote the binary states of the visible unit i and hidden unit j, respectively; a_i and b_j are the biases of the visible and hidden layers; w_{ij} represents the weight value between each layer. This network assigns a probability to each pair of visible and hidden layers by using the aforementioned energy function:

$$p(v,h) = \frac{1}{Z}\, e^{-E(v,h)}, \tag{5}$$

where the partition function $Z = \sum_{v,h} e^{-E(v,h)}$.

Thus, the probability that will be assigned to each visible vector is determined by summing all possible hidden vectors:

$$p(v) = \frac{1}{Z}\sum_{h} e^{-E(v,h)} \tag{6}$$

Because there is no direct connection between the hidden units, the binary state h_j of each hidden unit is set at 1 with probability $p(h_j = 1 \mid v) = \sigma(b_j + \sum_i v_i w_{ij})$, where $\sigma(.)$ represents a logistic sigmoid function. As with hidden layer, the binary state v_i of each visible unit is set at 1 with probability $p(v_i = 1 \mid h) = \sigma(a_i + \sum_j h_j w_{ij})$.

We then adopt the derivative of the log probability of a training vector:

$$\frac{\partial \log p(v)}{\partial w_{ij}} = \langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model} \tag{7}$$

The angular brackets represent the expectations of the distribution. The learning rate ϵ is then easily deduced using stochastic steepest ascent as follows:

$$\Delta w_{ij} = \epsilon(\langle v_i h_j \rangle_{data} - \langle v_i h_j \rangle_{model}) \tag{8}$$

To approximate the model more closely to the gradient, the next step is to perform contrastive divergence.

3) Contrastive Divergence: According to [26], we should assume all the visible and hidden units are in a binary state. The learning procedure in this study created an excellent generative model of the set of training vectors. Therefore, we updated the hidden states by the following equation:

$$p(h_j = 1) = \sigma(b_j + \sum_i v_i w_{ij}) \tag{9}$$

The visible states are updated by using equation:

$$p(v_i = 1) = \sigma(a_i + \sum_j h_j w_{ij}) \tag{10}$$

4) Fine-Tune: To fine-tune our network [27], we use the logistic sigmoid function $p = \sigma(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+e^0}$. With a binary unit, the aforementioned probability determines whether each neuron is in an ***on*** state or not. The function can be generalized to *K* alternative states: $p_j = \frac{e^{x_j}}{\sum_{i=1}^{K} e^{x_i}}$. The result of the fine-tune procedure here is often called a "softmax" unit and can be considered a method for mutually constraining each state so that only one of the *K* states has a value of 1 and the remaining states have a value of 0.

TABLE I
CONVERGENCE PERFORMANCE WITH DIFFERENT NUMBER OF LAYERS.

Visible neurons	Hidden neurons	Layer(s)	Accuracy
1024	512	1	71.3%
1024	512	2	95.58%
1024	512	3	92.1%

III. EXPERIMENTAL RESULTS

This section comprises three subsections that demonstrate our implementation results. Subsection III-A briefly lists the features we used to train our DBNs, subsection III-B describes the setting of the parameters used to train the model and its procedure, and subsection III-C demonstrates the experimental results and discussion.

A. Training Data

To evaluate the proposed face verification system, a dataset with 3D face scans of 25 Asian individuals (15 males and 10 females) was collected. The dataset includes one teenager, 22 adults and two elders. One thousand scans of 3D frontal facial point cloud with neutral expression were taken for each individual. Figure 9 shows four of the data samples that we have used to train the network. At the training stage, 500 point cloud scans were used as positive training samples and 500 samples were used as negative training examples for each individual. At the feature extraction step, the normal vector and principal curvature estimation were performed for all n points of each scan. In this study, each point cloud scan was resized and normalized into 1024 points for network model 1 and 2048 points for network model 2. Then the eigenvalues obtained from principal curvature estimation of all the n points were given as input to the Deep Belief network in the zig-zag order from left to right and from top to bottom, for each 3D facial scan point cloud.

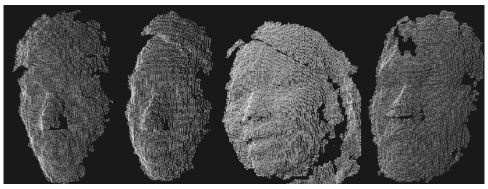
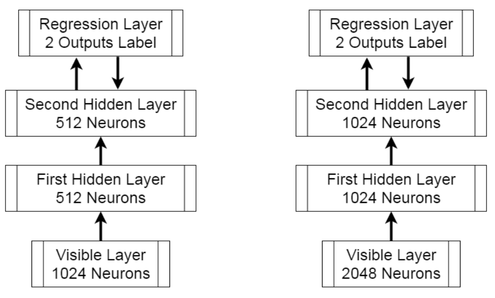


Fig. 9. Training samples of four people.

B. Parameter Setting

According to the experimental results shown in Table I, if there is only one hidden layer, the verification accuracy will decrease due to under-fitting learning. With the increase of layers, e.g. three hidden layers network, the verification accuracy will also decrease due to the problem of over-fitting learning.

Therefore, in the training phase of network model 1, 1024 neurons were presented in the visible layer, two hidden layers comprised 512 neurons for each, and the regression layer comprised two output neurons. The epochs of our training



(a) Network model 1. (b) Network model 2.

Fig. 10. Experimental setting with different network models.

```
1.  function DBNs construct ($parameters$)
2.      $visible\ layer\ number \leftarrow sizeof(input)$
3.      $hidden\ layer\ size \leftarrow 2$
4.      $hidden\ layer\ number \leftarrow sizeof(input)/2$
5.      $learning\ rate \leftarrow 0.2$
6.      $training\ epochs \leftarrow 800$
7.      $finetune\ epochs \leftarrow 400$
8.  function DBNs pre-train ($inputs$)
9.  for $i = 1$ to $epochs$ do
10.     for $file = 1$ to $N$ do
11.         Gibbs Sampling
12.     Contrastive Divergence
13. function DBNs fine-tune ($inputs,\ labels$)
14. for $i = 1$ to $epochs$ do
15.     for $file = 1$ to $N$, $label = 1$ to $N$ do
16.         Gibbs Sampling
17.     Logistic Regression
```

Algorithm 1. DBNs Training

and fine-tuning procedures are 800 and 400, respectively. In the training phase of network model 2, 2048 neurons were presented in the visible layer, two hidden layers comprised 1024 neurons for each, and the setting of regression layer was the same as network model 1. Figure 10a shows a diagram of parameter setting of network model 1, Figure 10b shows the parameter setting of network model 2, Algorithm 1 describes the training procedure, and Algorithm 2 describes the verification procedure.

C. System Performance Evaluation

After the verification procedure, the VTK GUI interface, a built-in point cloud library tool for displaying 3D data, is applied to demonstrate estimated results. The output label depends on the pre-trained input network model.

```
1. function DBNs construct ($model\ file$)
2.     $parameters \leftarrow model\ file\ retrieving$
3. function DBNs verification ($input$)
4. for all neurons do
5.     Forward propagation
6. Logistic Regression
7. return label
```

Algorithm 2. DBNs Verification

TABLE II
EXPERIMENTAL RESULTS OF 3D FACE VERIFICATION.

Network model	Input nodes	Correct	False	Accuracy
1	1024	3271	151	95.58%
2	2048	3300	122	96.43%

TABLE III
EXPERIMENTAL RESULTS OF 3D FACE VERIFICATION WITHOUT FEATURE EXTRACTION PRE-PROCESS.

Network model	Input nodes	Correct	False	Accuracy
1	1024	1486	1936	43.40%
2	2048	1514	1908	44.24%

The system assessments were performed with 1024 input nodes (network model 1) and 2048 input nodes (network model 2) using 3422 3D facial test scans for 25 persons. Table II presents the experimental results and demonstrates that the proposed system performs up to 95.58% verification accuracy with parameter setting of network model 1. By applying network model2, with two times the number of visible neurons, the verification accuracy increases to 96.43% in average for 3422 tests for 25 individuals. The real strength of deep learning architectures comes from their exceptional capability of learning features that best describe the domain. However, in this study, we found the verification performance was significantly decreased as shown in Table III if normal vector estimation and principal curvatures estimation feature extraction steps were omitted. We also applied SVM to perform the verification tasks using the same input data set. Figure 3 gives ROC curves of DBNs method and traditional SVM method. Thus the proposed DBNs method outperforms those of SVM methods.

To further evaluate the proposed DBNs face verification method, the FRGC 2.0 database were adopted to compare the verification rates of our algorithm with other state-of-the-art methods of Li et al. [17] and Berretti et al. [18] as illustrated in Section I. In our experiments, 460 negative samples were randomly chosen from the neutral expression faces in the FRFC dataset to train the proposed DBNs model using 2048 input nodes and then the identification error was calculated by means of 10-fold cross-validation for each person. The performance comparison results are tabulated in Table IV. The verification performance of the proposed method is in general compatible to those of Li's and Berretti's methods. Furthermore, the verification process of our method can be executed in real-time. It is important to note that, this study compares the recognition performance based on the experimental results of the above mentioned algorithms presented in their respective literature. Hence, the evaluation and performance comparison could also be biased, as it is not always feasible to know the exact test set used with the various algorithms under test. Our method may be further improved by applying the pre-processing techniques and post-processing model-fitting approaches.

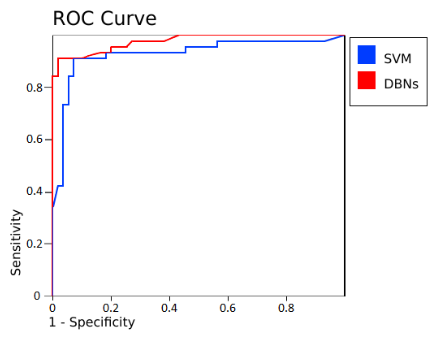


Fig. 3. Comparison of DBNs method and SVM method.

TABLE IV
COMPARISON OF RECOGNITION ACCURACY USING FRGC 2.0 DATABASE.

Methods	Features	Accuracy	Execution Time
Our method	Principal curvatures	96.43%	1.3 sec.
Li et al.[17]	EID low-level features	96.67%	N/A
Berretti et al.[18]	Iso-Geodesic stripes	97.7%	N/A

IV. CONCLUSION

This paper was motivated by the demand for a security surveillance system with high accuracy. The contribution of this paper is that we use the features with depth information to train a generative model. According to the experimental results, using more representative features to train the DBNs indeed increases the verification accuracy. Although the time complexity related to pre-training causes computational inefficiency because of the considerable number of training samples, by learning inherited-style features layer by layer, the prediction result had significantly high accuracy. To further justify the DBN verification phase, we will compare the performance of the proposed system with different classifiers, such as multi-class softmax, kNN, and other deep learning methods with self-learning features.

ACKNOWLEDGMENT

This work was supported in part by the Ministry of Science and Technology, Taiwan, under Grant MOST 103-2221-E-305-008-MY2.

REFERENCES

[1] Z.-P. Fu, Y.-N. Zhang, and H.-Y. Hou, "Survey of deep learning in face recognition," in *IEEE International Conference on Orange Technologies (ICOT)*, 2014, pp. 5–8.

[2] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," *Advances in neural information processing systems*, vol. 19, p. 153, 2007.

[3] M. Ranzato, J. Susskind, V. Mnih, and G. E. Hinton, "On deep generative models with applications to recognition," in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2011, pp. 2857–2864.

[4] J. Choi, A. Sharma, and G. Medioni, "Comparing strategies for 3D face recognition from a 3D sensor," in *IEEE RO-MAN*, 2013, pp. 19–24.

[5] H. Mohammadzade and D. Hatzinakos, "Iterative closest normal point for 3D face recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 35, no. 2, pp. 381–397, 2013.

[6] N. M. Thakare and V. M. Thakare, "An innovative hybrid approach to construct fuzzy-neural network for 3D face recognition system," in *Eleventh International Conference on Hybrid Intelligent Systems (HIS)*, 2011, pp. 463–467.

[7] F. Schimbinschi, "Machine learning for 3D face recognition using off-the-shelf sensors," Ph.D. dissertation, Department of Artificial Intelligence, University of Groningen, 2013.

[8] V. Nair and G. E. Hinton, "3D object recognition with deep belief nets," in *Advances in Neural Information Processing Systems*, 2009, pp. 1339–1347.

[9] M. Lin and X. Fan, "Low resolution face recognition with pose variations using deep belief networks," in *Fourth International Congress on Image and Signal Processing (CISP)*, vol. 3, 2011, pp. 1522–1526.

[10] D. Li, H. Zhou, and K. M. Lam, "High-resolution face verification using pore-scale facial features," *IEEE Transactions on Image Processing*, vol. 24, no. 8, pp. 2317–2327, 2015.

[11] D. R. Anggraini, "Face recognition using principal component analysis and self organizing maps," in *Third ICT International Student Project Conference (ICT-ISPC)*, 2014, pp. 91–94.

[12] I. Sutskever and G. E. Hinton, "Learning multilevel distributed representations for high-dimensional sequences," in *International Conference on Artificial Intelligence and Statistics*, 2007, pp. 548–555.

[13] N. K. Patil, S. Vasudha, and L. R. Boregowda, "Performance improvement of face recognition system by decomposition of local features using discrete wavelet transforms," in *International Symposium on Electronic System Design (ISED)*, 2013, pp. 172–176.

[14] Y. Lv, Z. Feng, and C. Xu, "Facial expression recognition via deep learning," in *International Conference on Smart Computing (SMARTCOMP)*, 2014, pp. 303–308.

[15] Z. Zhu, P. Luo, X. Wang, and X. Tang, "Deep learning multi-view representation for face recognition," *arXiv preprint arXiv:1406.6947*, 2014.

[16] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708.

[17] X. Li, T. Jia, and H. Zhang, "Expression-insensitive 3D face recognition using sparse representation," in *2009 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009, pp. 2575–2582.

[18] S. Berretti, A. D. Bimbo, and P. Pala, "3D face recognition using isogeodesic stripes," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 12, pp. 2162–2177, 2010.

[19] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, 2011, pp. 1–4.

[20] G. Gunlu and H. S. Bilge, "Feature extraction and discriminating feature selection for 3D face recognition," in *24th International Symposium on Computer and Information Sciences (ISCIS)*, 2009, pp. 44–49.

[21] R. B. Rusu, "Semantic 3D object maps for everyday manipulation in human living environments," *KI-Künstliche Intelligenz*, vol. 24, no. 4, pp. 345–348, 2010.

[22] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for boltzmann machines," *Cognitive Science*, no. 9, pp. 147–169, 1985.

[23] G. E. Hinton, "Learning multiple layers of representation," *Trends in Cognitive Sciences*, vol. 11, no. 10, pp. 428–434, 2007.

[24] G. E. Hinton, S. Osindero, and Y. Teh, "A fast learning algorithm for deep belief nets," *Neural Computation*, no. 18, pp. 1527–1554, 2006.

[25] R. Salakhutdinov and G. E. Hinton, "Deep boltzmann machines," in *International Conference on Artificial Intelligence and Statistics*, 2009, pp. 448–455.

[26] G. E. Hinton and M. A. Carreira-Perpignan, "On contrastive divergence learning," in *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, 2005, pp. 33–40.

[27] M. Mohri, A. Rostamizadeh, and A. Talwalkar, *Foundations of Machine Learning*. The MIT Press, 2012.