# Reduced-dimension Representations of Human Performance Data for Human-to-Robot Skill Transfer

Christopher Lee*† Yangsheng Xu†‡
†Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA
‡Department of Mechanical and Automation Engineering
The Chinese University of Hong Kong, Hong Kong

## Abstract

*Despite the large amount of research currently directed toward programming robots by demonstration, a significant problem with this method of human-to-robot skill transfer has not yet been addressed: developing representations of human performances which isolate the intrinsic dimensions of the performances (and thus the skills which guide them) within high-dimensional, raw human performance data. In this paper we propose the use of three methods for representing high-dimensional human performance data within lower-dimensional spaces: principal-component analysis (PCA), nonlinear principal component analysis (NLPCA), and sequential nonlinear principal component analysis (SNLPCA). We compare the appropriateness of these methods for modeling a simple human grasping operation.*

## 1 Introduction

This paper presents linear and nonlinear principal component analysis as methods for improving the representation of human performance data for aiding the process of human to robot skill transfer. The term *skill* refers to an ability to use knowledge effectively for the performance of a given task, and we represent a skill as a mapping from the performer's sensory inputs to her or his actions.

The process of human to robot skill transfer typically involves recording many individual human task performances, including task-state or sensory inputs available to the performer, and the performer's actions. Performance data is generally recorded by sampling at regular time intervals using signals from various input devices (e.g. Cyberglove, joystick, mouse, haptic interface) or measuring devices (e.g. visual trackers, instrumented cockpits, etc...). Although there may be a large number of dimensions in the resulting representations of recorded performances, the actual *intrinsic* dimensionality of the underlying human skill is often much lower. Linear or nonlinear dependencies between dimensions of the recorded performance data can effectively lower the number of independent dimensions required to adequately describe the performance. For example, using an instrumented glove to measure a performer's actions during a grasping task will typically yield 16-20 channels of data, each measuring a joint angle in the performer's hand. However, it is conceivable that most of the variation in hand configuration may be explained by a single 'feature-score' indicating roughly how closed the fingers are, and most of the remaining variation by a general measure such as how spread-out the fingers are from one another.

Not only may one or two feature-scores explain most of the variation in the recorded state information of a human performance description; they may also be a more meaningful description than the raw data set. A reduced-dimension, feature-based representation may be simpler to interpret and understand than raw performance data, and the functional mapping between the features and the raw data values may capture important information about the skill being performed. If some aspect of a performer's actions or configurations during the performance of a task can be represented with a reduced-dimension description, we know that the actions or configurations of the performer form a manifold within the space of possible actions or configurations. These manifolds may be useful for comparing one performer's task-skill to another's, for evaluating task performances, and for subtask recognition during performances. Moreover, machine-learning algorithms may learn skills more easily from human teachers when they are trained on a more focused, lower-dimension representation.

In Section 2 we present an example human performance data set, and in the following sections we discuss the analysis of this data set using traditional principal component analysis, nonlinear principal component analysis, and sequential nonlinear principal component analysis.

*Supported by a DOE Fellowship in Integrated Manufacturing

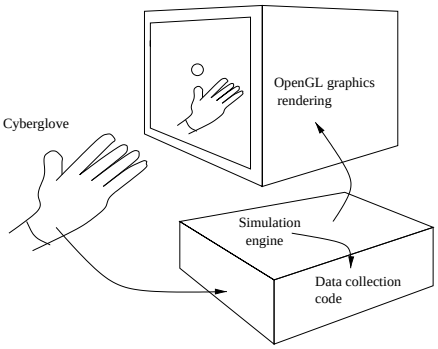


Figure 1: Skill demonstration system

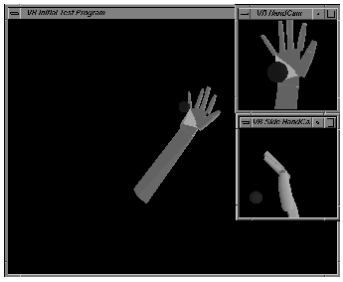

Figure 2: Graphical feedback

## 2 Experiment and data set

In this section, we present an example set of recorded human task performances which may be appropriately represented by a reduced-dimension description: data collected from the grasping phase of a ball-catching task which was demonstrated in a simple virtual environment.

A human subject performed a number of catches of a virtual ball using the system outlined in Figure 1. The input device was a Virtual Technologies Cyberglove which measures 18 joint angles in the fingers and wrist of its wearer, with a Polhemus sensor which returns the overall position and orientation of the wearer's hand. The feedback to the user was a rendering of the hand and ball from several perspectives, on the graphical display monitor of an SGI workstation (as shown in Figure 2). The recorded data was interpolated and resampled evenly at 10Hz, and stored in a performance database for later retrieval and analysis.

We manually segmented the performances into approach and grasp phases, and subjectively graded each recorded grasp on a scale from 0-9. We selected the vectors representing the joint angles in the fingers and wrist during the grasp phase of all catches for which the grasp satisfied a minimum grade. This generated a data set of 461 vectors, each representing a hand configuration of 18 joint-angles during grasping motions. These vectors were randomly allocated into a training set of 155 points, a cross-validation set of 153 points, and a test set of 153 points.

This data set was then analyzed using the PCA, NLPCA, and SNLPCA techniques presented in the following sections.

## 3 PCA

Principal component analysis (PCA) or the Karhunen-Loève transform [1] is a well-understood and commonly used method which, when given a set of multidimensional vectors, finds a linear mapping between these vectors and each lower-dimensional space such that when the vectors are mapped to a lower-dimensional space and then mapped back to the original space, the sum-of-squared errors of the reconstructed vectors are minimized. In the compressed representation of a vector, we can consider each dimension a separate *feature*, and the value of that dimension of the feature-vector a *feature-score*.

To perform principal component analysis of a set of $m$ $N$-dimensional zero-normed vectors $X_{N\times m} = [x_1|x_2|\ldots|x_m]$, we find the eigenvalues $\lambda_i$ and eigenvectors $v_i$ of the symmetric matrix $XX^T$. To generate a $f$-dimensional feature-vector representation of a $N$-dimensional vector $y$ (where $f < N$), we form a $N \times f$ matrix $V_f = [v_1|\ldots|v_f]$ where $v_1 \ldots v_f$ are the eigenvectors corresponding to the $f$ largest eigenvalues. Then the *feature-vector* $y^*$ is

$$y^* = V_f^T y, \tag{1}$$

and the reconstructed approximation $\tilde{y}$ of the original vector $y$ from feature-vector $y^*$ is

$$\tilde{y} = V_f y^* = V_f V_f^T y. \tag{2}$$

The value of the $i^{\text{th}}$ element of the feature vector $y^*$ is the magnitude of the projection of $y$ upon the corresponding eigenvector $v_i$, and the relative significance of each eigenvector in explaining the variations in the set of vectors $X$ is indicated by the magnitude of the corresponding eigenvalue $\lambda_i$.

Performing this analysis on the data set from Section 2, gives us a set of eigenvectors which attempt to explain the data set in terms of linear dependencies between its dimensions. The ability of the first 5 eigenvalues to represent the data set is summarized in Table 1. The numbers plotted in the PCA column are the relative magnitude of the residuals,

$$\%\text{-Err}_f = \|Y - V_f V_f^T Y\|_{\text{frob}} / \|Y\|_{\text{frob}}, \tag{3}$$

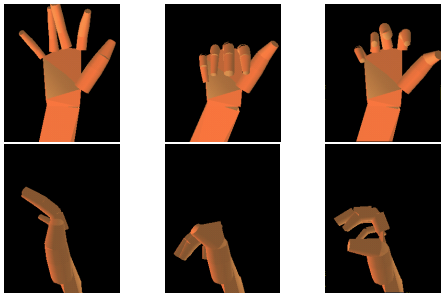


Figure 3: PCA analysis of grasp gesture. Initial position, and first and second principal "eigenhands," front and side views.

where $Y$ is the set of testing vectors from the experiment (independent from the set $X$ used to generate the eigenvectors), and $\|\cdot\|_{\text{frob}}$ is the Frobenius ($L_2$) norm.

Figure 3 provides a graphical illustration of the effects of the first and second linear principal components, which we might call "eigenhands."[1] These hand configurations are generated by varying the components $y_1^*$ and $y_2^*$ of the feature vector, and then mapping these features to $\tilde{y}$. We can see that the main effect of the first principal component is to bring the fingers closer together and to bend the fingers at the knuckles (mcp), while the effect of the second principal component is to curl the fingers at the second (pip) and third (dip) joints. Moreover, we see that although the grasp positions of the hand generated from the first and second eigenvectors look plausible, the best attempt to create an adequate initial (open) position for the grasp using only the first principal component looks less plausible. Closer inspection of this configuration reveals that the pinky and index fingers overlap in space, and that it is thus physically impossible. The problem is the linear nature of the mapping. The first principal component reduces the average sum-of-squares error for all joint-angles during the grasps by fitting a straight line in the space of training configurations, but if the general trend of the performances in state space is not linear, then this principal component will fit the trend of the performances poorly at some stages. Nevertheless, we see from Table 1 that PCA allows much of the 19-dimensional data set to be explained by only a few linear feature values.

## 4 NLPCA

Nonlinear principal component analysis (NLPCA) also attempts to find mappings between a multidimensional data set to and from a lower-dimensional feature-space while minimizing reconstruction error, but allows the mappings to be nonlinear. In contrast to linear mappings (1) and (2), the nonlinear mappings are of the general form

$$y^* = G(y) \tag{4}$$

$$\tilde{y} = H(y^*) = H(G(y)) = N(y). \tag{5}$$

If the lower-intrinsic dimensionality of a data set arises from a nonlinear relationship between the different dimensions of the data set, a nonlinear principal component analysis is capable of better representing the original data set with a reduced-dimension representation than would a linear principal component analysis. Several methods proposed for performing NLPCA include the use of autoassociative neural networks, as described by Kramer [2]; principal curves analysis, as described by Hastie and Stuetzle [3], and Dong and McAvoy [4]; adaptive principal surfaces, as described by LeBlanc and Tibshirani [5]; and optimizing neural network inputs, as presented by Tan and Mavrovouniotis [6]. In this section we will focus on Kramer's method for NLPCA, and in Section 5 we look at a modification of that method called SNLPCA.

Kramer's method for NLPCA involves training a neural network with three hidden layers, such as the one shown in Figure 4. These neural networks are autoassociative, meaning they are trained to map a set of input vectors $X$ to an identical set of output vectors. If the second hidden layer, or "bottleneck" layer, has a lower dimension than the input and output layers, then training the network creates a lower-dimensional representation $X^*$ of the vectors presented to the networks inputs in the form of the activations of the units in the bottleneck layer. The mapping from the input vectors to these activations in the bottleneck layer is the "compression" transform $G$, and the mapping

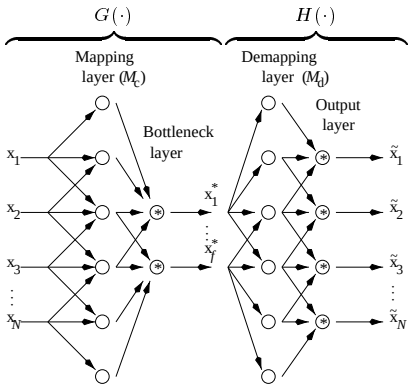


Figure 4: Neural network architecture for NLPCA. ◯ indicates a sigmoidal unit, and ⊛ indicates a unit which may be either linear or sigmoidal.

[1] The thumb positions in these diagrams are slightly erroneous due to a sensor which was not working during the experiments.

from the bottleneck activations to the activations of the output units is the "decompression" transform $H$. Using sigmoidal units of the form

$$\sigma(x) = (1 + e^{-x})^{-1} - \tfrac{1}{2} \quad (6)$$

in the first and third hidden layers allows the mapping function $G$ and de-mapping function $H$ to take the forms

$$G_k(X) = \tilde{\sigma}\left(\sum_{j=0}^{M} w_{kj}^{(2)} \sigma\left(\sum_{i=0}^{N} w_{ij}^{(1)} x_i\right)\right) k \in 1 \ldots f \quad (7)$$

$$H_k(X^*) = \tilde{\sigma}\left(\sum_{j=0}^{M} w_{kj}^{(4)} \sigma\left(\sum_{i=0}^{f} w_{ij}^{(3)} x_i^*\right)\right) k \in 1 \ldots N, \quad (8)$$

where $\tilde{\sigma}(\cdot)$ may either be the sigmoidal function (6) or the identity function $\tilde{\sigma}(x) = x$ depending on whether sigmoidal or linear units are used in the bottleneck and output layers. Given enough mapping units, these functional forms may approximate any bounded, continuous multidimensional nonlinear function $v = f(u)$ with arbitrary precision [7]. Just as PCA defines a linear mapping to and from a reduced-dimension representation which minimizes the sum of squared reconstruction error $\|X - V^T V X\|^2$ for a given set of vectors, training the weights $W$ of the autoassociative neural network to minimize the sum of squared error

$$E_f^2(X) = \|X - N_f(X)\|_{\text{frob}}^2 \quad (9)$$

of mapping vectors $x_i$ to themselves through the bottleneck layer effectively performs a nonlinear principal component analysis of the vectors.

The principal advantage of NLPCA over PCA is its ability to represent and learn more general transformations, which is necessary in cases when one wishes to eliminate correlations between dimensions in a set of data which cannot be adequately approximated by a linear dependency. However, NLPCA also has important disadvantages compared to PCA.

The trade-off for the extra-representational power of the nonlinear mapping functions $G$ and $H$ is that they cannot be as easily interpreted as the eigenvector-based mappings returned by PCA. In addition, NLPCA tends to require several orders more computation time than linear PCA, and because training the neural network is a high-dimensional nonlinear optimization problem over the weights of the neural network, we can guarantee only a locally optimal solution, unlike the globally optimal solution returned by PCA. In the version of NLPCA presented to this point the relative importance of each output dimension of the compression mapping cannot be determined by the training process, and there is no guarantee that any one of the output dimensions corresponds to a primary nonlinear factor of the training data. However, if an explicitly prioritized factorization is desired, Kramer's sequential NLPCA algorithm (SNLPCA), discussed in Section 5, may be used.

We used Kramer's NLPCA method to analyze the data set from Section 2. The neural networks were trained using the L-BFGS-B implementation of Byrd et. al [8]. We used a network architecture with linear units for the bottleneck and output layers, and without direct interconnections between the input and bottleneck layers, nor between the bottleneck and output layers. Choosing linear output units for the output layer allowed the network to be trained on data which was not rescaled to fit within the output range of the sigmoidal function (although the data was zero-normed). This gave better results than rescaling the data and using sigmoidal units.

The parameter $M$, the number of mapping units in the first and third hidden layers of each network, was chosen by a heuristic search over the range $f \ldots \min(\frac{m}{4}, p-1)$ where $m$ is the number of training vectors, and $p$ is the smallest possible value for $M$ for which the number of weights in the network will exceed the available number of values in the training matrix, $Nm$. $M$ must be at least as great as $f$ if there are no interconnections across the mapping layers; otherwise, the mapping layers would be the real bottlenecks. The selection criteria for the value of $M$ was not cross-validation error (although it is used for early-stopping of the weight-optimization algorithm), but rather the information theoretic criterion described in [2]:

$$\text{AIC} = \ln(e) + 2N_w/N_d, \quad (10)$$

where $N_w$ is the number of weights in the network, $N_d$ is the number of training vectors times the dimension of the training vectors, and $e = E/(2N)$ is the average sum of squares error. This criterion penalizes network complexity, and thus tends to reduce over-fitting of the training data. At least two units are used in the mapping layers, even when $f = 1$, because a network architecture with only a single unit in each of the three hidden layers is degenerate and unable to learn significantly nonlinear principal components.

In Table 1 we show the results of this NLPCA analysis. The effect of NLPCA on residual error for each number of principal nonlinear factors is calculated as

$$\%\text{-Err}_f = \|Y - N_f(Y)\|_{\text{frob}} / \|Y\|_{\text{frob}}. \quad (11)$$

The most interesting aspect of the analysis is that although NLPCA explains significantly more of the data set than PCA when $f = 1$, the two methods perform similarly when $f > 1$. This is due to the fact that the grasping motions analyzed are relatively simple and open-loop in

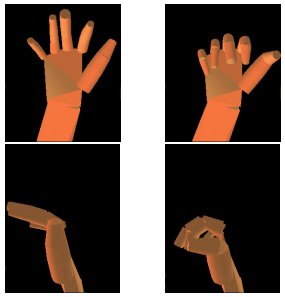


Figure 5: NLPCA analysis of grasp gesture. Initial position, and effect of first nonlinear principal component, front and side views

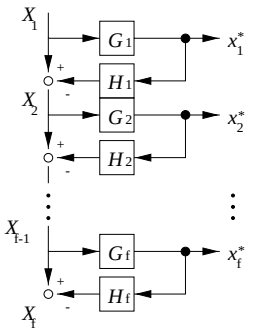


Figure 6: SNLPCA computation

nature, due to a lack of haptic feedback in the virtual environment in which they were demonstrated. We can thus think of each performance as following a nominal grasping trajectory in configuration-space from an open hand to a closed hand, with some stochastic variation. The first nonlinear principal component follows the nominal grasping-trajectory, and since this trajectory is curved, it explains the data in the testing data set better than the first linear principal component. However, since the basic non-stochastic structure of the grasp is adequately explained by the first nonlinear principal component, it is plausible that when a higher-dimensional feature-space is used NLPCA simply optimizes the mutual orthogonality of the resulting features to most efficiently explain the more stochastic nature of residual performance data, and the result is thus similar to PCA.

Figure 5 illustrates the first nonlinear principal component from our analysis. These configurations are formed by varying a scalar value $y_1^*$ and mapping it to a hand configuration using function $H_1$. The grasping configuration of the hand looks similar to that generated by the first linear principal component in Figure 3, but the initial position generated from the nonlinear principal component is more plausible. Moreover, the grasping motion resulting from smoothly varying the feature-value looks more natural than the motion generated by the linear principal-feature.

It should be noted that in this example we started by building a representation for individual hand configurations, but this resulted in a mapping which we could use to animate a nominal grasping performance by smoothly and monotonically varying the nonlinear principal component $y_1^*$ from an initial to a final value. This was possible because the grasping motion is a smooth directed path in configuration space, and because the forms of the mapping functions (7) and (8) are well suited to learning such smooth mappings.

## 5 SNLPCA

Kramer's sequential NLPCA algorithm (SNLPCA) [2], is a modification to the NLPCA method which produces a nonlinear factorization, and where the training process prioritizes each resulting feature as to its relative power in explaining the variations of the training set. SNLPCA performs a series of NLPCA operations, each training a neural network with a bottleneck layer consisting of a single unit. Training such a neural network on the raw performance data set explains as much of the set's variation as can be represented by a single variable, and thus trains the compression part of the network to perform the primary nonlinear factorization of the data set. The next nonlinear factorization should explain the variation in the data set which is not accounted for by the first factorization, and thus a second network is trained on the residuals formed by subtracting each vector of the original data set by its estimate as calculated by the first network. This process, summarized in Figure 6, continues until the desired number of iterations has been reached or until enough of the original data set's variation has been explained.

The algorithm is

- $X_0 \leftarrow X, f \leftarrow 0$
- Loop for $f$ in $0 \ldots (F-1)$ or until $\|X_f\| < \epsilon$
  - Train $W_{f+1}$ to minimize $\|X_f - N_{f+1}(X_f)\|^2$, where $N_{f+1}(\cdot) \equiv N(W_{f+1}, \cdot)$
  - $X_{f+1} \leftarrow X_f - N_{f+1}(X_f)$
  - Split $N_{f+1} : \Re^N \rightarrow \Re^N$ into $G_{f+1} : \Re^N \rightarrow \Re^1$ and $H_{f+1} : \Re^1 \rightarrow \Re^N$ such that $H_{f+1}(G_{f+1}(\cdot)) \equiv N_{f+1}(\cdot)$
  - $X^*_{(f,\cdot)} \leftarrow G_{f+1}(X_f)$ (i.e sets row $f$ of $X^*$).

This algorithm generates a reduced-dimension representation $X^*$ of $X$, but does not automatically generate the mapping and de-mapping functions $G$ and $H$. These may be formed in two ways. The first is to cascade the networks $G_f$, which were trained in the SNLPCA algorithm, to form a large single network for computing $G$ in

| $f$ | PCA | NLPCA | | SNLPCA | | |
|---|---|---|---|---|---|---|
| | %-Err | %-Err | $M$ | %-Err | $M_c$ | $M_d$ |
| 1 | 72.0 | 58.5 | 12 | 59.9 | 14 | 3 |
| 2 | 41.4 | 40.9 | 3 | 47.4 | 14 | 6 |
| 3 | 33.4 | 30.1 | 8 | 43.1 | 12 | 5 |
| 4 | 26.8 | 25.7 | 5 | 42.0 | 15 | 9 |
| 5 | 23.1 | 20.2 | 10 | 36.8 | 19 | 8 |

Table 1: Experimental results. Percent of test data set $Y$ unexplained by $f$ factors for each method, calculated by (3) for PCA and by (11) for NLPCA and SNLPCA. $M$ is the size of the mapping layers in NLPCA, and $M_c$ and $M_d$ are the sizes of the mapping layers in the compression and decompression networks trained by SNLPCA.

a manner corresponding to the computation shown in Figure 6. In a similar fashion, networks $H_f$ can be cascaded to form a network for computing $H$. This method requires no additional training and will perform exactly the mappings calculated by the SNLPCA algorithm. The second method is to train separate neural networks to perform the mapping from $X$ to $X^*$ and $X^*$ to $X$. This method results in smaller, more computationally efficient networks, but may not necessarily converge to acceptable approximations of the desired mappings.

Table 1 shows the results of SNLPCA on the data set from Section 2. For this data set, we see that it is roughly equivalent to NLPCA for $f = 1$, and performs worse than both NLPCA and PCA for $f > 1$. This is due to the fact that the first iteration has eliminated almost all of the underlying structure of the grasping skill, thus leaving the residual vectors $X_2$ dominated by stochastic variations in the individual human performances. Since the nonlinear factors are trained sequentially, it is difficult for the SNLPCA algorithm not to over-fit the remaining noisy residuals at each iteration, rather than learn to represent this noise by building an orthonormal basis of linear features in the manner of PCA. In addition, as the number of features used increases, the compression networks have increasing difficulty learning the generated compression mappings.

## 6 Conclusion and discussion

Based on its ability to represent the nominal trajectory of the grasping skill in configuration space, we conclude that the one-dimensional NLPCA mapping (equivalent to the one-dimensional SNLPCA mapping) is the best representation of the grasping skill of the models we generated. For faithful reconstruction of a given hand configuration from a reduced-dimensional representation, PCA analysis is the simplest and most effective method. Higher-dimensional NLPCA and SNLPCA models might potentially be more appropriate for more complex skills, particularly those involving more sensory feedback.

Malthouse [9] discusses Kramer's NLPCA method and indicates that it has several important limitations, including an inability to model curves and surfaces that intersect themselves, and an inability to parameterize curves with parameterizations involving discontinuous jumps. These limitations are due to the fact that the mapping and demapping projections (7) and (8) are continuous functions. For our example data set, and for many typical human skills, continuous mappings to and from the feature representation are not particularly restrictive because the skills can be smoothly parameterized. However, Malthouse indicates that Kramer's NLPCA method also tends to result in suboptimal projections, and indicates that methods based on the principal curves methods [3, 4, 5] tend to result in better parameterizations. We plan in our future work to compare these methods to the ones presented in this paper in terms of their efficiency, accuracy, and feasibility for modeling human performance data.

## References


[1] I. T. Jolliffe, *Principal component analysis*. New York: Springer-Verlag, 1986.

[2] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChe Jounal*, vol. 37, pp. 233–43, February 1991.

[3] T. Hastie and W. Steutzle, "Principal curves," *Journal of the American Statistical Association*, vol. 84, pp. 502–16, June 1989.

[4] D. Dong and T. McAvoy, "Nonlinear principal component analysis-based on principal curves and neural networks," *Computers & Chemical Engineering*, vol. 20, pp. 65–78, January 1996.

[5] M. LeBlanc and R. Tibshirani, "Adaptive principal surfaces," *Journal of the American Statistical Society*, vol. 89, pp. 53–64, March 1994.

[6] S. Tan and M. L. Mavrovouniotis, "Reducing data dimensionality through optimizing neural network inputs," *AIChe Jounal*, vol. 41, pp. 1471–1480, June 1995.

[7] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Contr. Signals Syst.*, vol. 2, pp. 303–314, 1989.

[8] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal of Scientific Computing*, vol. 16, no. 5, pp. 1190–1208, 1995.

[9] E. C. Malthouse, "Limitations of nonlinear PCA as performed with generic neural neworks," *IEEE Transactions on Neural Networks*, vol. 9, pp. 165–73, January 1998.