PHYSCON 2009, Catania, Italy, September, 1–September, 4 2009

LEARNING SEQUENCES WITH NEURAL GAS FOR ROBOT MOTION PLANNING

Ignazio Aleo
DIEES
University of Catania
Italy
ignazio.aleo@diees.unict.it

Paolo Arena
DIEES
University of Catania
Italy
paolo.arena@diees.unict.it

Luca Patané
DIEES
University of Catania
Italy
luca.patane@diees.unict.it

Abstract

The aim of this paper is to investigate novel solutions for motion sequence learning based on an extension of the Neural Gas with local Principal Component Analysis (NGPCA) algorithm. As an abstract Recurrent Neural Network (RNN), this model is able to complete a partially given pattern. Under this point of view it is possible to generalize the model as a dynamical system in which for a given actual configuration and a particular task the desired state variables are retrieved as outputs converging to a particular state iteratively. The developed architecture has been tested in the control of a redundant manipulator in simple forward and inverse kinematic problem solving and in motion sequence reproduction.

Key words

Robotics, sequence learning, RNN, PCA, Neural Gas

1 Introduction

The development of efficient low level control algorithms for both kinematic and dynamic problem solving, even for highly-redundant mechanical structure, introduces the importance of higher level control strategies for more complex problem solving and for Human-Robot Interaction.

The learning of motion sequences is probably one the first step in this direction. Our idea is to investigate novel solutions based on an abstract Recurrent Neural Network (RNN) model named NGPCA [Hoffmann, 2003].

With the local Principal Components it is possible to represent sensorimotor distributions locally constrained to subspaces with fewer dimensions than the space of the training data. Under this point of view, as described in [Hoffmann, 2003], PCA is able to model distributions in which directions with almost zero variance exist.

The model architecture can be easily divided in a learning phase, in which the data distribution is approximated and in a recall phase in which an uncomplete pattern is presented to the network in order to retrieve the output (pattern completition). As already discussed (see [Cruse, 1998], [Steinkuhler, 1998]), similarly to RNNs, this model is able to cope with multiple solution tasks providing one of the possible solutions and it is also possible to choose the role of input and output neurons, after training, simply modifying the recall phase. The sequence generation extension of the model, with the state variables input pattern portion, gives the motion planning possibility for the control of the considered serial manipulator [Arena, 2008].

2 MODEL DESCRIPTION

2.1 NG with local PCA

The Neural Gas algorithm is a variant of the soft-clustering vector quantization with the addiction of annealing. For a given pattern space $P \subseteq \Re^d$, the algorithm starts choosing *m* points, in this hyperspace, from the *N* code-book vectors. During each step a random pattern, $\mathbf{x}$, is chosen from the training set. Then each j-point, $\mathbf{c_j}$ is updated relating to its rank, r_j, function of the distance from the selected pattern trough the learning rate ε and the neighborhood range ϱ as follows:

$$\mathbf{c_j}(t+1) = \mathbf{c_j}(t) + \alpha_j \cdot [\mathbf{x} - \mathbf{c_j}(t)], \tag{1}$$

where α_j is defined by $\alpha_j = \varepsilon \cdot e^{-r_j/\varrho}$. The local PCA extension of the NG considers hyper-ellipsoid units, with q principal components, instead of simple points and therefore the ranking of the units cannot depend on an Euclidean distance. One of the possible distance measure is the normalized Mahalanobis distance, an elliptical distance that can be computed, for each j particle, as:

$$\begin{aligned} E(x) = \xi^T W \Lambda^{-1} W^T \xi + \tfrac{1}{\sigma^2}(\xi^T \xi - \xi^T W W^T \xi) + \\ + ln(det\Lambda) + (d-q) ln\sigma^2, \end{aligned} \tag{2}$$

where $\xi = \mathbf{x} - \mathbf{c}$ is the deviation of vector x from the center unit, W is the eigenvector matrix, Λ is a diagonal matrix containing the eigenvalues. The second term of equation (2) is the reconstruction error divided by σ^2 that depends on the total residual variance v_{res}, among all $d - q$ minor dimensions, as in equation (3).

$$\sigma^2 = \frac{v_{res}}{d-q}. \tag{3}$$

The total residual variance is updated according to

$$\begin{aligned} v_{res}(t+1) &= v_{res}(t) + \\ &+ \alpha \cdot (\xi^T W \Lambda^{-1} W^T \xi - \xi^T W W^T \xi - v_{res}(t)). \end{aligned} \tag{4}$$

To modify principal components of existing ellipsoids, one step of the Robust Recursive Least Square Algorithm (RRLSA), described in [Ouyang, 2000] is performed.

$$\begin{aligned} \{W(t+1), \Lambda(t+1)\} &= \\ &\mathbf{PCA}\{W(t), \Lambda(t), \xi(t), \alpha(t)\} \end{aligned} \tag{5}$$

Since the orthogonality of W is not preserved after each step, the Gram-Schmidt orthogonalization method has been introduced. The algorithm overall block diagram is shown in Figure 1.

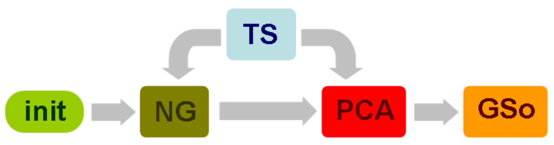

Figure 1. Algorithm block diagram for learning phase of the Neural Gas with local Principal Component Analysis (NGPCA). The NG block performs center updating for a given training vector, extracted by Training Selector (TS), the PCA block executes one step of local principal components algorithm and GSo (Graham-Schmidt orthonormalization) is able to normalize eigenvectors.

Relevant parameters used in the model are summarized in table 1.

Table 1. Summary table

m	number of particles
N	data set dimension
d	pattern space dimension
q	number of principal components
$\mathbf{c}_j$	center of particle j
r_j	ranking of particle j
v_{res}	total residual variance
Λ_j	eigenvalues matrix of particle j
W_j	eigenvectors matrix of particle j

2.2 Particle volume-normalized distance

In order to reduce the dependence of the error in equation (2) from the volume of the considered ellipsoid and to avoid useless points (with weight α almost zero) in the pattern space it is possible, as in [Hoffmann, 2003], to modify the distance measure as follows:

$$\begin{aligned} \widetilde{E}(x) &= (\xi^T W \Lambda^{-1} W^T \xi + \\ &+ \tfrac{1}{\sigma^2}(\xi^T \xi - \xi^T W W^T \xi)) V^{2/d} \end{aligned} \tag{6}$$

where V is the volume of the considered ellipsoid unit and can be computed according to

$$V = \sigma^{d-q} \sqrt{det \Lambda}. \tag{7}$$

2.3 Recall

After the learning phase, the data distribution is represented by m hyper-ellipsoids with center $\mathbf{c}_j$ (with $j = 1, 2, ..., m$), semi-axes lengths $\sqrt{\lambda_j^k}$ (with $k = 1, 2, ..., q$), $\mathbf{w_j}^k$ principal component eigenvectors and a residual variance σ_j^2. In order to perform the recall, a potential function in the constrained subspace $E_j(\mathbf{z})$ can be computed for each ellipsoid as:

$$\begin{aligned} E_j(\mathbf{z}) &= \mathbf{y}_j^T \Lambda_j^{-1} \mathbf{y}_j + \tfrac{1}{\sigma_j^2}(\xi_j^T \xi_j - \mathbf{y}_j^T \mathbf{y}_j) + \\ &+ ln(det \Lambda_j) + (d-q) ln \sigma_j^2, \end{aligned} \tag{8}$$

where ξ_j is the displacement from the center $\xi_j = \mathbf{z} - \mathbf{c}_j$ and $\mathbf{y}_j = W_j^T \xi_j$ is the representation in the local coordinate system of the ellipsoid. The input to the network is given in form of an offset $\mathbf{p}$ in the constrained space $\mathbf{z}(\eta) \subseteq \Re^d$ as follows:

$$\mathbf{z}(\eta) = \mathbf{M}\eta + \mathbf{p}, \tag{9}$$

where M matrix aligns the constrained space to a particular parameters space while $\eta \in \Re^s$ is a vector of

free parameters. For each unit j, the point of constrained space with smallest potential $\widehat{z}_j$ is determined, according to equation (8), and then the unit j^* that has the minimal potential among all $E_j(\widehat{\mathbf{z}}_j)$ is chosen as complete pattern.
As shown in [Hoffmann, 2003], the function $E(\eta)$ is convex and it is possible to determine analytically the only minimum $\widehat{\eta}_j$ computing:

$$\widehat{\eta}_j = \mathbf{A}_j(\mathbf{p} - \mathbf{c}_j), \tag{10}$$

with

$$\left.\begin{array}{l} \mathbf{A}_j = -(\mathbf{M}^T D_j \mathbf{M})^{-1}\mathbf{M}^T\mathbf{D}_j, \\ \mathbf{D}_j = \mathbf{W}_j\mathbf{\Lambda}_j^{-1}\mathbf{W}_j^T + \frac{1}{\sigma_j^2}(\mathbf{I} - \mathbf{W}_j\mathbf{W}_j^T). \end{array}\right\} \tag{11}$$

3 NGPCA for Sequences generation

One of the possible extensions of this vector reconstruction strategy is to introduce the actual state $\mathbf{x_i}$ and next state $\mathbf{x_{i+1}}$ of the considered system as part of the input vector

$$\mathbf{p}_i = (\mathbf{x}_i, \mathbf{x}_{i+1}), \qquad i = 1, 2, ..., N-1. \tag{12}$$

In this way it is possible to learn not only a static complex law but even a particular sequence. Under this point of view it is possible to make both short and long term prediction and then analyze model prediction with real sensor information for updating the internal model.
The so generalized model, shown in Figure 2, has been applied to control a custom built seven degrees of freedom redundant manipulator [Arena, 2008] not only in forward and inverse kinematic problem solving (see [Hoffmann, 2003]) but also in operating space motion planning.
The introduced Pattern Constructor block (PC) processes inputs for the abstract network (NGPCA) in order to determine the constrained space for the current iteration, reading the joint variables θ, and therefore leading the system toward the given reference P_d (*e.g.* solving forward kinematics), as in

$$\mathbf{x_i} = f(\theta_\mathbf{i}, \mathbf{P_d}). \tag{13}$$

The miniARM block is the serial manipulator itself and has been implemented both in simulation and in real hardware setup. It takes an input vector reference and gives a feedback vector (*e.g.* it can be implemented in the operating space solving the inverse kinematics and reading joint angular position) (see [Cruse, 1993],[Arena, 2007]).

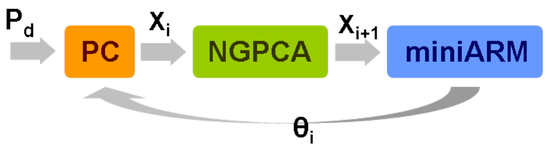

Figure 2. Algorithm block diagram for sequence learning with Neural Gas with local Principal Component Analysis (NGPCA). The Pattern Constructor block (PC) performs control modifying the input pattern with the offset P_d. The miniARM block is the serial redundant manipulator (controlled in the operating space).

The same overall model architecture could also be used to determine an iterative converging recall algorithm, not discussed in this paper, modifying the one described in the previous section, in order to control the system with constrains toward a desired trajectory in the free-parameters subspace.

4 Experimental setup

As introduced, the algorithm has been tested with a real manipulator, the MiniARM [Arena, 2008]. The MiniARM, shown in Figure 3, is a serial manipulator with rotational joints, completely custom built, developed in order to analyze and compare algorithm performances with low implementation effort. The overall control is made up of a high level sequence control in the computer and a low level hardware layer control (custom built) with a 32 bit microcontroller unit. Architecture functional diagram is shown in Figure 4.

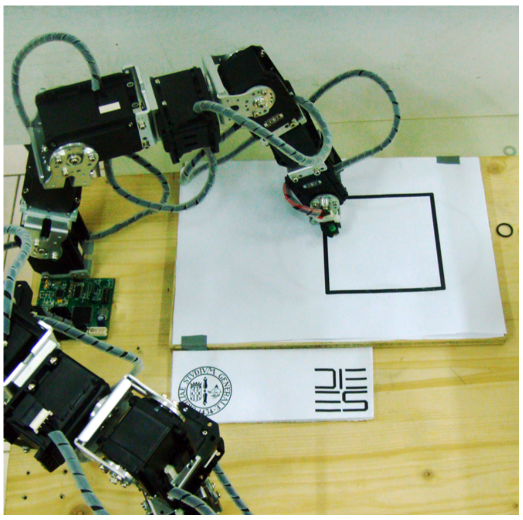

Figure 3. Picture of MiniARM first prototype realized in our laboratories. Experimental setup for square sequence learning in a plane.

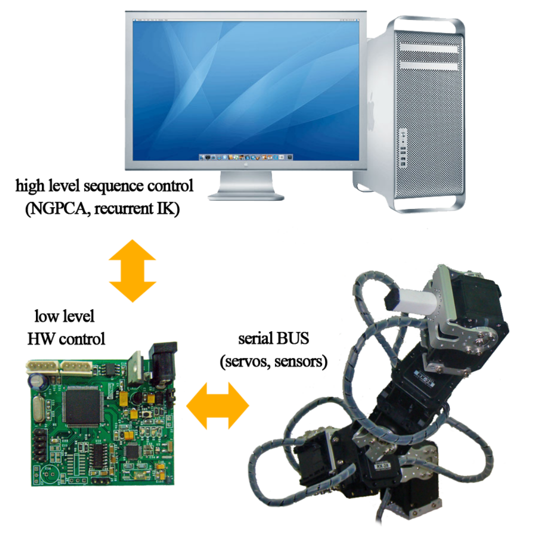

Figure 4. Functional diagram of the hardware layer (main host computer, low level control board and actuators/sensors) of the implemented control algorithm.

4.1 Training set

The training set have been acquired from the real robot reading all the encoder positions through the direct kinematic of the manipulator. Each data set is made up of N=1000 three-dimensional points acquired every 0.08s inside the operating space during a user guided real-time trajectory following.

$$\begin{aligned} \mathbf{x_i} &\triangleq (x_i, y_i, z_i) \quad i = 1, 2, ..., N-1, \\ \mathbf{p}_i &= (x_i, y_i, z_i, x_{i+1}, y_{i+1}, z_{i+1}). \end{aligned} \tag{14}$$

Though desired trajectories are planar and repetitive, the acquisition method is very noisy and three dimensional by definition, as shown in Figure 5.

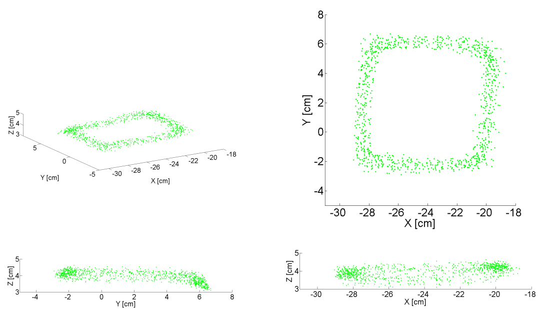

Figure 5. Three dimensional plot of an example data set acquired with $N = 1000$ (up-left), XY plane plot (up-right), YZ (down-left), XZ (down-right).

Learning patterns have been built up using two consecutive points from acquired data set.

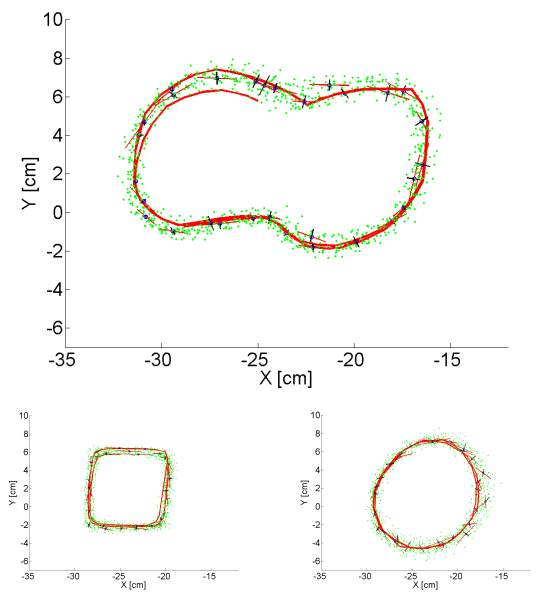

Figure 6. Examples of sequences learned with the NGPCA.

5 Results

Performance of the algorithm have been tested under different operating conditions. All presented results have been obtained with the experimental setup shown in Figure 3.

5.1 Normal operations

As described in the following relationship, the algorithm provides just the next point of the sequence in the operating space given the actual point as part of the reconstructed pattern $\widehat{\mathbf{z}}^*$.

$$\widehat{\mathbf{z}}^* = \mathbf{M}\widehat{\eta}_{\mathbf{j}^*} + \mathbf{p}, \tag{15}$$

where the offset vector $\mathbf{p}$ is able to define the constrained subspace as in equation (16) iteration after iteration.

$$\mathbf{p}_i = (x_i, y_i, z_i, 0, 0, 0) \qquad i = 1, 2, ..., N. \tag{16}$$

The $\mathbf{M}$ matrix aligns free and constrained subspaces in separate regions of the whole space.

$$\mathbf{M} = \begin{bmatrix} \mathbf{0} \\ \mathbf{I} \end{bmatrix}, \quad \mathbf{0} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \quad \mathbf{I} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{17}$$

The joint space trajectories are generated solving iteratively the inverse kinematic problem (see [Cruse, 1993],[Arena, 2007]). Robot reproduced trajectories are shown in Figure 6. Three different trajectories have been reproduced in order to analyze the generalization capabilities. All these have been generated with human-guided points acquisition method.

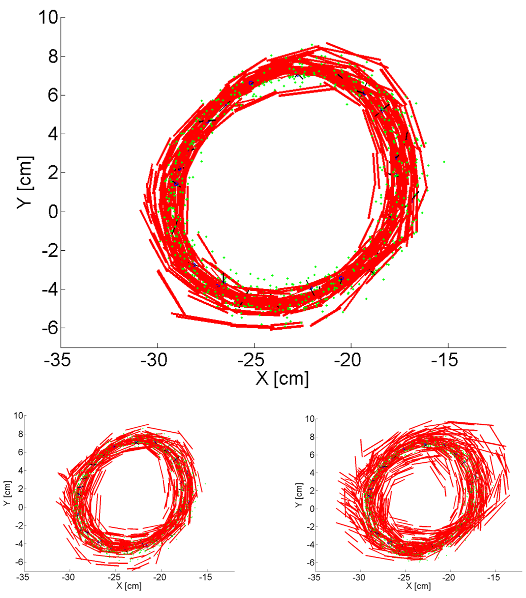
Figure 7. Examples of NGPCA sequences reproduction in presence of simulated feedback errors: $e = 0.05$ (5% of random component in θ) (on the top), $e = 0.1$ of random component (down-left) and $e = 0.15$ (down-right).

5.2 Numeric robustness

One of the most important features of the algorithm is that both learning phase and recall phase show a very high numeric robustness. As depicted in Figure 5, learning data set defines a really noisy trajectory. The same trajectories have been reproduced adding noise in the feedback variable θ as in equation (18).

$$\theta = \widehat{\theta}(1 + e(rand - 0.5)), \tag{18}$$

where $\widehat{\theta}$ is the measured feedback variable vector while $rand \subseteq [0, 1]$ with uniform probability density distribution and $e \subseteq [0, 1]$ is the maximum error amplitude. The circle trajectory has been chosen in order to test numeric robustness of sequence reproduction as shown in Figure 7 with $e \in \{0.05, 0.1, 0.15\}$.

5.3 Performance outside learned space

The performance outside the operating space have been tested using multiple distances from the center of the trajectory and measuring the number of steps needed to go through the sequence (*i.e.* the Euclidean distance E_i under a chosen threshold $E_{th} = 0.1\ cm$ as in inequality 19). Though particular values strongly depend on the shape of the path and on the learning phase, for a given trajectory and a defined training phase values can be compared through all different distances. In order to normalize these distances with the effective dimension of the path, an adimensional *Distance over Dimension Ratio* (DDR) is defined as in equation (20).

$$E_i < E_{th}, \tag{19}$$

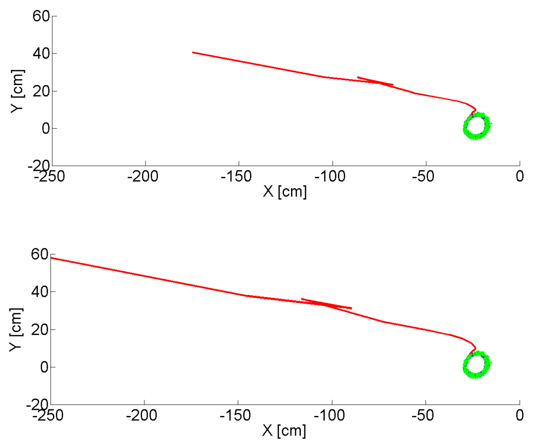
Figure 8. Trajectories reproduced from points far away from the trained operating space. DDR=7 on the top and DDR=10 on the bottom.

Table 2. Algorithm performance, outside operating space

DDR	2	10	100	1000
n	13	18	24	36
n_{10}	16	20	24	36

$$DDR = \frac{d}{m_d}, \tag{20}$$

where d is the distance from the center of the trajectory and m_d is the maximum dimension of it. Examples of reproduced trajectories starting from a point outside from the learned space are shown in Figure 8. Table 2 and Figure 9 summarize the algorithm performance: n is the number of steps noiseless and n_{10} is the same quantity when an additional 10% of random component is added in feedback variable θ.

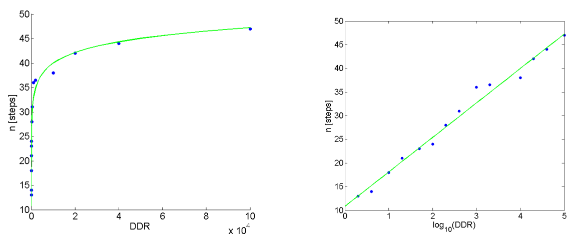
Figure 9. Number of steps needed to go into the sequence with espect to the DDR on the left (dots are real data while line is a logarithmic interpolation), same plot with DDR on logarithmic scale (abscissa) on the right.

5.4 Reversed operations

As in common RNNs, in the NGPCA the pattern reconstruction is possible careless on which part of the pattern is lacking. However, a distinct feature if this architecture with respect to the other RNNs is that the choice of which part of the pattern has to be reconstructed does not affect the learning phase. Therefore it is possible to use the same learning not only for one way sequences reproduction but also for reversed sequences. The same performance are so obtained using the same learned architecture and employing the following pattern for reverse reconstruction purposes.

$$\mathbf{p}_i = (0, 0, 0, x_i, y_i, z_i), \quad i = 1, 2, ..., N, \quad \mathbf{M} = \begin{bmatrix} \mathbf{I} \\ \mathbf{0} \end{bmatrix}. \tag{21}$$

6 Conclusions

The Neural Gas algorithm with local Principal Component Analysis implementation has been tested and extended for the control of motion sequences for a redundant serial manipulator. Algorithm robustness, in terms of noise and performance outside the learned space, has been tested. Though the training phase needs a complete training set and a computational effort, the recalling phase is very fast and possible to implement also in a common microcontroller-based platform. The one-to-many mappings, the prediction capability and the inputs/outputs role independence in the training phase shows the possibility of generalization over a wide range of control applications.

Acknowledgements

The authors acknowledge the support of the European Commission under the project ICT 216227 - SPARK II Spatial-temporal patterns for action-oriented perception in roving robots.

References

Arena, P., Cruse, H., Fortuna, L. and Patané, L. (2007) An Obstacle avoidance method for a redundant manipulator controlled through a recurrent neural network. In *Proc. of Microtechnologies for the New Millennium (SPIE 07)* Gran Canaria (SPAIN), May.

Arena, P. (2008) EU Project SPARK II, website online at *www.spark2.diees.unict.it/MiniArm.html.*

Cruse, H. and Steinkuhler, U. (1993) Solution of the direct and inverse kinematics problems by a common algorithm based on the mean of multiple computations. *Biol. Cybernetics*, vol. 69 (2), pp. 345-351, 1993.

Cruse, H., Steinkuhler, U. and Burkamp, Ch. (1998) MMC - A recurrent neural network which can be used as manipulable body model. *From animals to animats*, vol. 5, R. Pfeifer, B. Blumberg, J.-A. Meyer, S.W. Wilson (eds.) MIT Press, pp. 381-389.

Hoffmann, H., Möller, R. (2003) Unsupervised learning of a kinematic arm model. *Artificial Neural Networks and Neural Information Processing*, vol. 2714, (ICANN/ICONIP), LNCS, Kaynak O, Alpaydin E, Oja E, Xu L,Springer, Berlin, pp. 463-470.

Ouyang, S., Bao, Z., Liao, G.S. (2000) Robust recursive least squares learning algorithm for principal component analysis. *IEEE Transactions on Neural Networks*, vol. 11(1), pp. 215-221.

Steinkuhler, U., Cruse, H. (1998) A holistic model for an internal representation to control the movement of a manipulator with redundant degrees of freedom. *Biol. Cybernetics*, vol. 79, pp. 457-466.

Tipping, M. E., Bishop, C. M. (1999) Mixtures of probabilistic principal component analyzers. *Neural Computation*, vol. 11, pp: 443-482, 1999.