An Introduction to the Special Issue on Advances on Interactive Multimedia Systems

Z. Bojkovic [1], F. Neri [2]
[1] University of Belgrade, Belgrade, SERBIA,
[2] Editor in Chief of WSEAS Transactions on Systems,
Department of Computer Science, University of Naples "Federico II", Naples, ITALY
z.bojkovic@yahoo.com, nerifil@gmail.com

This Special Issue intents to collect high-quality papers providing theoretical and/or practical matters dealing with advances on interactive multimedia systems (IMS). The perspective of today's information society calls for a multiplicity of devices including IP-enabled home appliances, personal computers, sensors, etc., all of which are globally connected. IMS have become important for information exchange and the core communication environment for business relations as well as for social interactions. Every day millions of people all over the world use interactive systems for a plethora of daily activities including searching, information access and exchange, multimedia communication enjoyment, buying and selling goods, keeping in touch with friends, etc. Current multimedia communication systems and architectural concepts must evolve to cope with complex connecting requirements. This is vital to the development of present day multimedia technologies as a natural step to all progression in general research about IMS. The fact that interactive multimedia is currently one of the fastest growing sectors in broadband networking, quality of service (QoS) standards and audio/video processing techniques must be taken into account. An IMS require real-time processing of data and media streams with the support of user interactions at any time. In this respect, the aim of the Special Issue is to highlight technological challenges in order to offer high-quality and low-delay multimedia services, together with designing a good user interface for smart interactive systems and the incorporation of automatic perception of human activities like presence, speech, interaction.

The papers included have given a view point of advances on IMS by balancing the article selection between research and industrial community in order to include the high quality papers from both arenas. The contributions cover hot topics in the field and constitute one step among many others to recent progress in IMS.

We start this Special Issue with the article entitled "Recognition of Assamese Spoken Words using a Hybrid Neural Framework and Clustering Aided Apriori Knowledge". Artificial Neural Network based model is proposed for recognition of discrete Assamese speech, using a self Organizing Map-based phoneme count determination technique. It acts as a self sustaining, fully automated mechanism for spoken word recognition with 3, 4 and 15-phoneme variation.

The second paper "Multimedia Signal Session in a Web Environment" includes the fields of health, IT, design, IMS and communications. The development provides new opportunities for interaction in telemedicine, particularly in the area of training and obtaining a surgical opinion in a Web environment. The presented system is technically feasible with high potential of medical staff training, which will result in a better patient care requiring surgery.

The paper "New online RKPCA-RN Kernel method Applied to Tenessee Eastman Process" proposes a new method for online identification of a nonlinear system using a linear combination of Kernel functions applied to the used training set observations. Through several experiments the accuracy and good scaling properties of the proposed method have been demonstrated. The proposed method may be helpful in designing an adaptive control strategy of nonlinear systems.

A unified approach in analysis of designs and reference software implementation of three generation for MPEG/ITU video coding standards is demonstrated in the paper "MPEG Video deployment in interactive multimedia systems: HEVC vs. AVC codec performance study". Bit reduction and coding gain based on peak-signal-to-noise ratio measure and complexity as well as encoding/decoding time of high efficient video coding (HEVC) vs. advance video coding (AVC) high profile (HP) are tested using the low-delay encoding constraints.

Next, in the paper "Methods and Tools for Structural Information Visualization", it is emphasized that Higres, Visual Graph and ALVIS systems are aimed at supporting of structural information visualization on the base hierarchical

graph modes. These systems are suited for visual processing and can be used in many areas where the visualization of structural information is needed.

An algorithm framework for cross-layer QoS adaptation in multiservice heterogeneous environments is proposed based on Service-Oriented Architecture (SOA) principles in the paper "A Novel Algorithm for SOA-based Cross-Layer QoS Support in Multiservice Heterogeneous Environments". The proposed algorithm takes into account the diversity of user/application requirements networking technologies, terminal and provider capabilities. A proposal of framework for cross-layer QoS adaptation in order to support characteristics of various heterogeneous networking technologies, and to discover and select the proper network services that support QoS requirements for different applications, is presented, too.

Finally, in the paper "Fuzzy Authentication Algorithm with Applications to Error Localization and Correction of Images", the proposed image authentication methods have the ability to authenticate images even in the presence of small noise. They are in the position to localize errors as well as to correct them using the corresponding error correcting codes.

Before we leave you to enjoy this Special Issue, we would like to thank all the authors, who invested a lot of work and effort providing their really valuable contributions as the results of their research work. These papers make a significant contribution to the problem of interactive multimedia systems. Of course, much work remains to achieve high quality technologies at low cost. Special thanks go to reviewers who dedicated their precious time in providing numerous comments and suggestions, criticism and constant and enthusiastic support. In closing, we hope that the contributions included should provide not only knowledge, but inspiration to researchers and developers of interactive multimedia systems.

New online RKPCA-RN Kernel method Applied to Tennessee Eastman Process

NADIA SOUILEM, ILYES ELAISSI, OKBA TAOUALI & HASSANI MESSEOUAD
Unité de Recherche d'Automatique, Traitement de Signal et Image (ATSI),
Ecole Nationale d'Ingénieur Monastir
Rue Ibn ELJazzar 5019 Monastir ; Tel : +(216) 73 500 511, Fax : + (216) 73 500 514,
TUNISIA

nadiasuilem@yahoo.fr; ilyes_elaissi@yahoo.fr; taoualiokba@yahoo.fr;
hassani.messaoud@enim.rnu.tn

Abstract: -This paper proposes a new method for online identification of a nonlinear system using RKHS models. The RKHS model is a linear combination of kernel functions applied to the used training set observations. For large datasets, this kernel based to severs computational problems and makes identification techniques unsuitable to the online case. For instance, in the KPCA scheme the Gram matrix order grows with the number of training observations and its eigen decomposition. The proposed method is based on Reduced Kernel Principal Component Analysis technique (RKPCA), to extract the principal component will be time consuming.

Key-Words: - *RKHS, RN, SLT, Kernel method, RKPCA, Online RKPCA-RN, Tennessee.*

1 Introduction

The last few years have registered the birth of a new modelling method of nonlinear systems using Reproducing Kernel Hilbert Space (RKHS) [1], [7], [15], [16], [17], [21], [22] called kernel methods. These methods have been applied to a large class of problems, such as face recognition [12], [24], time series prediction [4], identification of nonlinear system [11], [14], [23], …. The proposed method proceeds in two steps, first in an off line phase we select a set of kernels functions using the RKPCA technique, then in the online phase a RKHS model is constructed and successively updated.

An eigen decomposition of Gram matrix can simply become too time-consuming to extract the principal components and therefore the system parameter identification becomes a tough task. To overcome this burden recently a theoretical foundation for online learning algorithm with kernel method in reproducing kernel Hilbert spaces was proposed [8], [10]. When the system to be identified is time varying, the online kernel algorithm is more useful, because these algorithms can automatically track changes of system Behaviour with time-varying and time lagging characteristic.

In this paper we propose a new method for online identification of a non linear system parameters modeled on Reproducing Kernel Hilbert Space (RKHS). This method uses the Reduced Kernel Principal Component Analysis (RKPCA) that selects the observations data to approach the Principal Components Analysis [11]. The selected observations are used to build an RKHS model with a reduced parameter number. The numbers of reduced observations are fixed and we actualise the parameters on minimizing a criterion. The proposed technique may be very helpful to design an adaptive control strategy of non linear systems.

The paper is organized as follows. In section 2, we remind the Reproducing Kernel Hilbert Space (RKHS). In section 3, we remind the Reduced Kernel Principal Component Analysis RKPCA method. In section 4, we propose the new online RKPCA-RN method. The proposed algorithm has been tested to identify the nonlinear system and a Tennessee Eastman process.

2 Reproducing Kernel Hilbert Space

Let $X \subset \mathbb{R}^d$ an input space and $L^2(X)$ the Hilbert space of square integrable functions defined on X. Let $k: X^2 \to \mathbb{R}$ be a continuous positive definite kernel. It is proved [18] that it exists a sequence of an orthonormal eigen functions $(\psi_1, \psi_2, ..., \psi_l)$ in $L^2(X)$ and a sequence of corresponding real positive eigenvalues $(\sigma_1, \sigma_2, ..., \sigma_l)$ (where l can be infinite) so that:

$$k(x, t) = \sum_{j=1}^{l} \sigma_j \psi_j(x) \psi_j(t) \quad ; \quad x, t \in X \tag{1}$$

Let $H \subset L^2(X)$ be a Hilbert space associated to the kernel k and defined by:

$$H = \left\{ f \in L^2(X) / f = \sum_{i=1}^{l} w_i \varphi_i \ \text{ and } \ \sum_{j=1}^{l} \frac{w_j^2}{\sigma_j} < +\infty \right\} \tag{2}$$

Where $\varphi_i = \sqrt{\sigma_i}\, \psi_i \ \ i = 1, ..., l$. The scalar product in the space H is given by:

$$< f, g >_H = \ < \sum_{i=1}^{l} w_i \varphi_i, \sum_{j=1}^{l} z_j \varphi_j >_H \ = \sum_{i=1}^{l} w_i z_i \tag{3}$$

K is a Reproduisant Kernel for the space H if

* $\forall x \in X$, the function k_x such as

$$\begin{aligned} k_x &: X \to \mathbb{R} \\ t &\mapsto k_x(t) = k(x, t) \end{aligned} \tag{4}$$

is a function of the space H.

* $\forall x \in X \ ; \ \forall f \in H \ \ \langle f, k_x \rangle_H = f(x)$ (5)

H is a Reproducing Kernel Hilbert Space (RKHS) for the kernel k.

The relation (5) describes the property of reproducibility of the function for the space H [1]. In other words the scalar product between each function $f \in H$ and the function k_x enables to determine $f(x)$.

Let's define the application Φ :

$$\begin{aligned} \Phi &: X \to IR^l \\ x &\mapsto \Phi(x) = \begin{pmatrix} \varphi_1(x) \\ \vdots \\ \varphi_l(x) \end{pmatrix} \end{aligned} \tag{6}$$

Where φ_i are given in (2).

The kernel trick [1] is so that:

$$k(x, x') = \langle \Phi(x), \Phi(x') \rangle \quad x, x' \in X \tag{7}$$

Vapnik [18] proposes to adopt the (Structural Risk Minimisation: SRM)

$$D(f) = \frac{1}{N} \sum_{i=1}^{N} V(y_i, f(x_i)) + \lambda \| f \|_H^2 \tag{8}$$

Based on the representer theorem [19] the optimal function f_{opt} which minimizes $D(f)$ can be written as:

$$f_{opt}(x) = \sum_{i=1}^{N} a_i \ k(x_i, x) \tag{9}$$

Where $a_i, \ i = 1, ..., N$ are the model parameters.

3 RKPCA method

Let a nonlinear system with an input $u \in \mathbb{R}$ and an output $y \in \mathbb{R}$ from which we extract a set of observations $\{u_i, y_i\}_{i=1, ..., N}$. Let H an RKHS space with kernel k. To build the input vector x_i of the RKHS model we use the NARX (Nonlinear auto regressive with eXogeneous input) structure as:

$$x_i = \left\{ u_i, ..., u_{i-m_u}, y_{i-1}, ..., y_{i-m_y} \right\}^T \ ; \ m_u, m_y \in \mathbb{N} \tag{10}$$

The set of observations becomes $D = \{x_i, y_i\}_{i=1, ..., N}$ where $x_i \in \mathbb{R}^{m_u + m_y + 1}$ and $y_i \in \mathbb{R}$.

and the RKHS model of this system based on (9) can be written as:

$$\tilde{y}_j = \sum_{i=1}^{N} a_i \, k(x_i, x_j) \tag{11}$$

Let the application Φ :

$$\Phi : X \to \mathbb{R}^l$$

$$x \mapsto \Phi(x) = \begin{pmatrix} \varphi_1(x) \\ . \\ . \\ . \\ \varphi_l(x) \end{pmatrix} \quad (12)$$

Where φ_i are given in (13).

The Gram matrix K associated to the kernel k is an N - dimensional square matrix, so that:

$$k_{i,j} = k(x_i, x_j) \text{ for } i, j = 1, ..., N \quad (13)$$

The kernel trick [1] is so that:

$$< \Phi(x), \Phi(x') > = k(x, x') \; x, x' \in X \quad (14)$$

We assume that the transformed data $\{\Phi(x_i)\}_{i=1, ..., N} \in \mathbb{R}^l$ are centered [11]. The empirical covariance matrix of the transformed data is symmetrical and l - dimensional. It is written as following:

$$C_\phi = \frac{1}{M} \sum_{i=1}^{M} \Phi(x_i) \, \Phi(x_i)^T, \; C_\phi \in \mathbb{R}^{l \times l} \quad (15)$$

Let l' the number of the eigenvectors $\{V_j\}_{j=1, ..., l'}$ of the C_ϕ matrix that corresponding to the non zeros positive eigenvalues $\{\lambda_j\}_{j=1, ..., l'}$. It is proved in [11] that the number l' is less or equal to N.

Due to the large size l of C_ϕ, the calculus of $\{V_j\}_{j=1, ..., l'}$ can be difficult. The KPCA method shows that these $\{V_j\}_{j=1, ..., l'}$ are related to the eigenvectors $\{\beta_j\}_{j=1, ..., l'}$ of the gram matrix K according to [11]:

$$V_j = \sum_{i=1}^{N} \beta_{j,i} \; \Phi(x_i) \;\; , \;\; j = 1, ..., l' \quad (16)$$

Where $(\beta_{j,i})_{j=1, ..., P}$ are the components of $\{\beta_j\}_{j=1,...,l'}$ associated to their nonzero eigenvalues $\mu_1 > ... > \mu_{l'}$.

The principle of the KPCA method consists in organizing the eigenvectors $\{\beta_j\}_{j=1, ..., l'}$ in the decreasing order of their corresponding eigenvalues $\{\mu_j\}_{j=1, ..., l'}$. The principal components are the P first vectors $\{V_j\}_{j=1, ..., P}$ associated to the highest eigenvalues and are often sufficient to describe the structure of the data [11]. The number P satisfies the Inertia Percentage criterion IPC given by:

$$P^* = \arg(IPC \geq 99) \quad (17)$$

Where

$$IPC = \frac{\sum_{i=1}^{P} \mu_i}{\sum_{i=1}^{M} \mu_i} * 100 \quad (18)$$

The RKHS model provided by the KPCA method is [1].

$$\tilde{y}_{new} = \sum_{q=1}^{P} w_q \; \sum_{i=1}^{N} \beta_{q,i} \; k(x_i, x_{new}) \quad (19)$$

Since the principal components are a linear combination of the transformed input data $\{\Phi(x_i)\}_{i=1, ..., N}$ [12], the Reduced KPCA approaches each vector $\{V_j\}_{j=1,...,P}$ by a transformed input data $\Phi(x_j^b) \in \{\Phi(x_i)\}_{i=1, ..., N}$ having a high projection value in the direction of V_j [5].

The projection of the $\Phi(x_i)$ on the V_j called $\tilde{\Phi}(x_i)_j \in \mathbb{R}$ and can be written as:

$$\tilde{\Phi}(x_i)_j = \langle V_j, \Phi(x_i) \rangle; \quad j = 1, ..., P \quad (20)$$

According to (18) and (16), the relation (22) is written:

$$\tilde{\Phi}(x_i)_j = \sum_{m=1}^{N} \beta_{j,m} \; k(x_m, x_i); \quad j = 1, ..., P \quad (21)$$

To select the vectors $\{\Phi(x_i^b)\}$, we project all the $\{\Phi(x_i)\}_{i=1,\ldots,N}$ vectors on each principal component $\{V_j\}_{j=1,\ldots,P}$ and we retained $x_j^b \in \{x_i\}_{i=1,\ldots,N}$ that satisfies

$$\begin{cases} \Phi(x_j^b)_j = \underset{i=1,\ldots,N}{Max}\ \tilde{\Phi}(x_i)_j \\ and \\ \Phi(x_j^b)_{i\neq j} < \zeta \end{cases} \quad (22)$$

Where ζ is a given threshold.

Once the $\{x_j^b\}_{j=1,\ldots,P}$ corresponding to the P principal component $\{V_j\}_{j=1,\ldots,P}$ are determined, we transform the vector $\Phi(x)\in\mathbb{R}^l$ to the $\hat{\Phi}(x)\in\mathbb{R}^P$ vector that belongs to the space generated by $\{\Phi(x_j^b)\}_{j=1,\ldots,P}$ and the proposed reduced model is:

$$\tilde{y}_{new} = \sum_{j=1}^{P} \hat{a}_j\ \hat{\Phi}(x_{new})_j \quad (23)$$

Where :

$$\hat{\Phi}(x_{new})_j = \langle \Phi(x_j^b), \Phi(x_{new}) \rangle; \qquad j = 1, \ldots, P \quad (24)$$

And according to the kernel trick (7), the model (23) is:

$$\tilde{y}_{new} = \sum_{j=1}^{P} \hat{a}_j\ k_j(x_{new}) \quad (25)$$

Where:

$$k_j(x) = k(x_j^b, x) \quad for\ j = 1, \ldots, P \quad (26)$$

The model (23) is less complicate than that provided by the KPCA. The identification problem can be formulated as a minimization of the regularized least square written as:

$$J_r(\hat{a}) = \frac{1}{2}\sum_{i=1}^{N}\left(y_i - \sum_{j=1}^{P} \hat{a}_j\ k_j(x_i)\right)^2 + \frac{\rho}{2}\|\hat{a}\|^2 \quad (27)$$

Where: ρ is a regularization parameter and $\hat{a} = (\hat{a}_1, \ldots, \hat{a}_P)^T$ is the parameter estimate vector
The solution of the problem (27) is:

$$\hat{a}^* = (F + \rho I_P)^{-1} G \quad (28)$$

With :

$$G = \begin{pmatrix} \sum_{i=1}^{N} k_1(x_i)\, y_i \\ \vdots \\ \sum_{i=1}^{N} k_b(x_i)\, y_i \end{pmatrix} \in \mathbb{R}^P$$

and

$$F = \begin{pmatrix} \sum_{i=1}^{N} k_1(x_i)k_1(x_i) & \cdots & \sum_{i=1}^{N} k_1(x_i)k_b(x_i) \\ \vdots & & \\ \sum_{i=1}^{N} k_b(x_i)k_1(x_i) & \cdots & \sum_{i=1}^{N} k_b(x_i)k_b(x_i) \end{pmatrix} \in \mathbb{R}^{P\times P}$$

And $I_P \in \mathbb{R}^{P\times P}$ is the P identity matrix

The RKPCA algorithm is summarised by the five following steps:

1. Determine the nonzero eigenvalues $\{\mu_j\}_{j=1,\ldots,l'}$ and the eigenvectors $\{\beta_j\}_{j=1,\ldots,l'}$ of Gram matrix K .
2. Order the $\{\beta_j\}_{j=1,\ldots,l'}$ on the decreasing way with respect to the corresponding eigenvalues.
3. For the P retained principal components, choose the $\{(x_j^b)\}_{j=1,\ldots,P}$ that satisfy (22).
4. Solving (29) to determine $\hat{a}^* \in \mathbb{R}^P$
5. The reduced RKHS model is given by (2)

4 Online RKPCA-RN method

Prior to the online identification, we start with offline identification on a set of observations $D=\{(x_1, y_1), \ldots, (x_n, y_n)\}$.

In this offline phase we apply the RKPCA technique to reduce the number of the parameters of the RKHS model. The model provided by the RKPCA is given by:

$$\tilde{y}(x)=\sum_{j=1}^{P} a_j\ k\left(x_j^b, x\right) \tag{29}$$

The on line identification phase begins from the instant $n+1$. At this instant a new couple of observations is available $\{x_{n+1}, y_{n+1}\}$.

From the equation (29), we calculate the output RKHS model given by:

$$\tilde{y}_{n+1}=\sum_{j=1}^{P} a_j\ k\left(x_j^b, x_{n+1}\right) \tag{30}$$

The error between the estimated output and the measured on actual one is:

$$e_{n+1}=\left(\tilde{y}_{n+1}-y_{n+1}\right) \tag{31}$$

If

$$\left|e_{n+1}\right|<\varepsilon_1 \tag{32}$$

where ε_1 is a given threshold, we can say the model approaches sufficiently the system behavior. If not, we update the parameters $\{a_j\}$ by minimizing the criterion $J_{r,n+1}$

$$\begin{aligned} J_{r,n+1}(A_{n+1})=&\frac{1}{2}\sum_{i=1}^{P}\left(y_i^b-\sum_{j=1}^{P} a_j\ k\left(x_j^b, x_n\right)\right)^2+\\ &\frac{1}{2}\left(y_{n+1}-\sum_{j=1}^{P} a_j\ k\left(x_j^b, x_n\right)\right)^2+\frac{\rho}{2}\left\|A_{n+1}\right\|^2 \end{aligned} \tag{33}$$

Where

$$A_{n+1}=\begin{pmatrix} a_1 \\ \vdots \\ a_P \end{pmatrix}\in IR^P$$

We can write

$$J_{r,n+1}\left(A_{n+1}\right)=\frac{1}{2}\left(\left\|K_{(n+1)P}\ A_{n+1}-Y_{n+1}\right\|^2+\rho\left\|A_{n+1}\right\|^2\right)$$

Where

$$K_{(n+1)P}=\begin{pmatrix} k\left(x_1^b, x_1\right) & \ldots & k\left(x_P^b, x_1\right) \\ \vdots & \ldots & \vdots \\ k\left(x_1^b, x_P\right) & \ldots & k\left(x_P^b, x_P\right) \\ k\left(x_1^b, x_{n+1}\right) & \ldots & k\left(x_P^b, x_{n+1}\right) \end{pmatrix}\in IR^{(P+1)\times P} \quad \text{and}$$

$$Y_{n+1}=\begin{pmatrix} y_1^b \\ \vdots \\ y_P^b \\ y_{n+1} \end{pmatrix}\in IR^{P+1} \tag{34}$$

The criterion (33) can be written as the following :

$$\min_{A_{n+1}\in IR^P} J_{r,n+1}(A_{n+1})=$$

$$\begin{aligned} &\min_{A_{n+1}\in IR^P}\frac{1}{2}\left(\left\|K_{(n+1)P}\ A_{n+1}-Y_{n+1}\right\|^2+\rho\left\|A_{n+1}\right\|^2\right)\\ &=\min_{A_{n+1}\in IR^P}\left(\frac{1}{2}\ \left(K_{(n+1)P}\ A_{n+1}-Y_{n+1}\right)^T\left(K_{(n+1)P}\ A_{n+1}-Y_{n+1}\right)+\frac{\rho}{2}{A_{n+1}}^T A_{n+1}\right) \end{aligned} \tag{35}$$

The minimum of $J_{r,n+1}$ is reached for :

$$\frac{\partial J_{r,n+1}}{\partial A_{n+1}}=0$$

$$\frac{\partial J_{r,n+1}}{\partial A_{n+1}}=$$

$$\begin{aligned} &\frac{\partial}{\partial A}\left[\frac{1}{2}\left({A_{n+1}}^T K_{(n+1)P}^T-{Y_{n+1}}^T\right)\left(K_{(n+1)P}.A_{n+1}-Y_{n+1}\right)+\frac{\rho}{2}{A_{n+1}}^T A_{n+1}\right]\\ &=\frac{1}{2}\frac{\partial}{\partial A_{n+1}}\left[{A_{n+1}}^T K_{(n+1)P}^T.K_{(n+1)P}.A_{n+1}-{A_{n+1}}^T K_{(n+1)P}^T.Y_{n+1}\right.\\ &\left.-{Y_{n+1}}^T.K_{(n+1)P}.A_{n+1}-{Y_{n+1}}^T Y_{n+1}+\rho{A_{n+1}}^T A_{n+1}\right] \end{aligned} \tag{36}$$

As ${A_{n+1}}^T K_{(n+1)P}.Y_{n+1}$ et ${Y_{n+1}}^T.K_{(n+1)P}.A_{n+1}$ are scalar and transposed, then

$$\frac{\partial J_{r,n+1}}{\partial A_{n+1}} = \frac{\partial}{\partial A_{n+1}}\Big[{A_{n+1}}^T K_{(n+1)P}^T K_{(n+1)P}.A_{n+1} - 2.{Y_{n+1}}^T.K_{(n+1)P}.A_{n+1} - {Y_{n+1}}^T Y_{n+1} + \rho {A_{n+1}}^T A_{n+1}\Big] \quad (37)$$

Therefore

$$\frac{\partial J_{r,n+1}}{\partial A_{n+1}} = K_{(n+1)P}^T K_{(n+1)P} A_{n+1} - K_{(n+1)P}^T Y_{n+1} + \rho A_{n+1}$$

$$= \left(K_{(n+1)P}^T K_{(n+1)P} + \rho I_P\right) A_{n+1} - K_{(n+1)P}^T Y_{n+1}$$

Then

$$A_{n+1} = \left(K_{(n+1)P}^T K_{(n+1)P} + \rho I_P\right)^{-1} K_{(n+1)P}^T Y_{n+1} \quad (38)$$

To every new observation, we apdate the parameters of the model (30) using the relation (38).

Online RKPCA- NR algorithm

Offline phase:

According to (17) and (18) we determine the P retained principal components resulting from the processing of an N measurement set.

1- Then we determine the $I = \{x_j^b\}_{j=1,\ldots,P}$ set according to (22), used during the online phase.

2- Write RKHS model obtained by RKPCA

$$\tilde{y}_n = \sum_{j=1}^{P} a_j \; k\left(x_j^b, x_n\right)$$

Online phase:

For a new couple of observation $\{x_{n+1}, y_{n+1}\}$

1 - Calculate the output of the model RKHS

$$\tilde{y}_{n+1} = \sum_{j=1}^{P} a_j \; k\left(x_j^b, x_{n+1}\right)$$

2- Calculate the value of e_{n+1}.

3- If the condition (32) is satisfied $\Rightarrow$ we comes back to 1 for a new observation $\{x_{n+2}, y_{n+2}\}$

Otherwise:

- Estimate the parameters $\{a_j\}$ $j = 1, \ldots, P$ using the relation (38).

5 Simulations

The proposed method has been tested for modelling a nonlinear system and a Tennessee Eastman process.

5. 1 System nonlinear

We consider the nonlinear system

$$y(i) = \log\left(\left|u(i-1)^2 - u(i-2) + 0.6\; y(i-2)\right| + 0.4\; y(i-1) + 1\right) + e(i) \quad (39)$$

Where $e(i)$ is a gaussian noise. The input vector of RKHS model has the structure

$$x(i) = [\, u(i-1), u(i-2),\; y(i-1)]^T$$

$u(i)$ is the process input chosen as gaussian signal. To build the RKHS model we use the ERBF Kernel (Exponential Radial Basis Function)

$$k(x, x') = \exp\left(-\sqrt{\frac{\|x - x'\|}{\mu}}\right) \quad (40)$$

With $\mu = 5$ and $\|\ \|$ is the euclidean norm. The term of regularisation $\lambda = 10^{-6}$

The chosen threshold is: $\varepsilon_1 = 0.04$

We performed the online identification using the online RKPCA-RN algorithm developed in section 4.

The number of observations in identification offline phase is 150 and the number of principal component analysis obtained by RKPCA method is equal to 5.

The number of observations in online identification phase is 300.

The minimal normalized mean square error between real output and estimated one (NMSE) is defined.

$$NMSE = \frac{\sum_{i=1}^{N} (\tilde{y}_i - y_i)^2}{\sum_{i=1}^{N} (y_i)^2} \quad (41)$$

Where $\tilde{y}_i$ and y_i design the model RKHS output and the system output respectively.

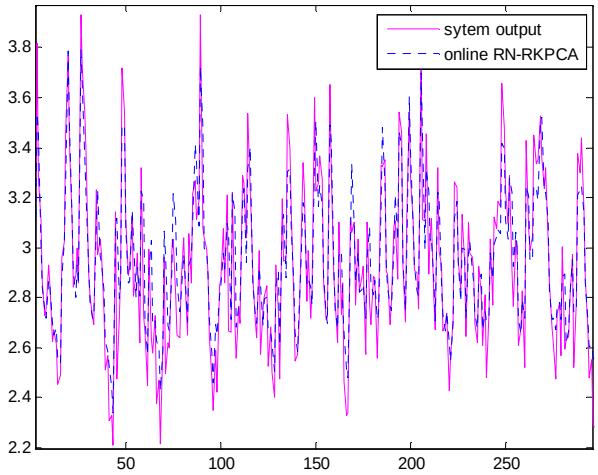

Figure 1: System and model outputs during the online Identification

In Figure 1, we represent the online RKPCA-RN output as well as the system output. We remark that the model output is in concordance with the system output, indeed the Normalized mean Square Error is equal to $0,002$. This shows the good performances of the proposed online identification method.

To evaluate the performance of the proposed method we plot in Figure 2, the evolution of the NMSE.

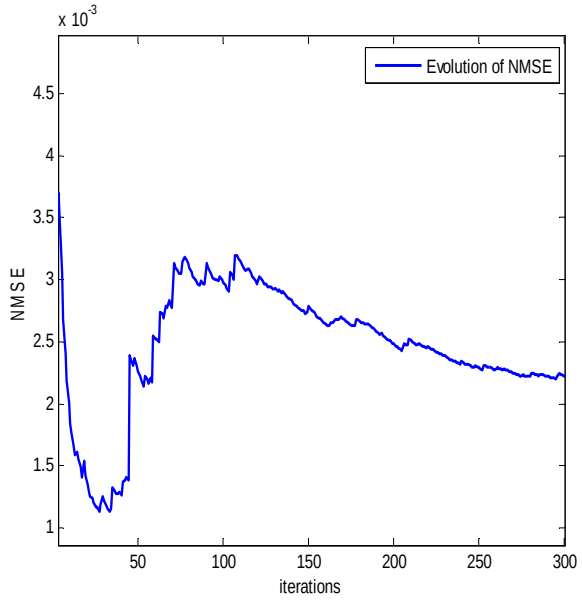

Figure 2: Evolution of NMSE

5. 2 Tennessee Process

To illustrate the efficiency of the proposed model we proceed to its validation on Tennessee Eastman process.

5. 2. 1 Process description

The Tennessee Eastman (TE) process [9] is a highly non linear, non-minimum phase, and open-loop unstable chemical process consisting of a reactor/separator/recycle arrangement. This process produces two products G and H from four reactants A, C, D and E. Also a byproduct F is present in the process. The simultaneous, irreversible and exothermic gas-liquid reactions are:

$$A(g)+C(g)+D(g)\rightarrow G(\text{lig}),\text{Product 1}$$
$$A(g)+C(g)+E(g)\rightarrow G(\text{lig}),\text{Product 2}$$
$$A(g)+E(g)\rightarrow F(\text{lig}),\text{byroduct}$$
$$3D(g)\rightarrow 2F(\text{lig}),\text{byroduct}$$

The process has 12 valves available for manipulation and 41 measurements available for monitoring or control. The detailed description of these variables, process disturbances and base case operating conditions, is given in [3]. The process flowsheet is presented in Figure 3

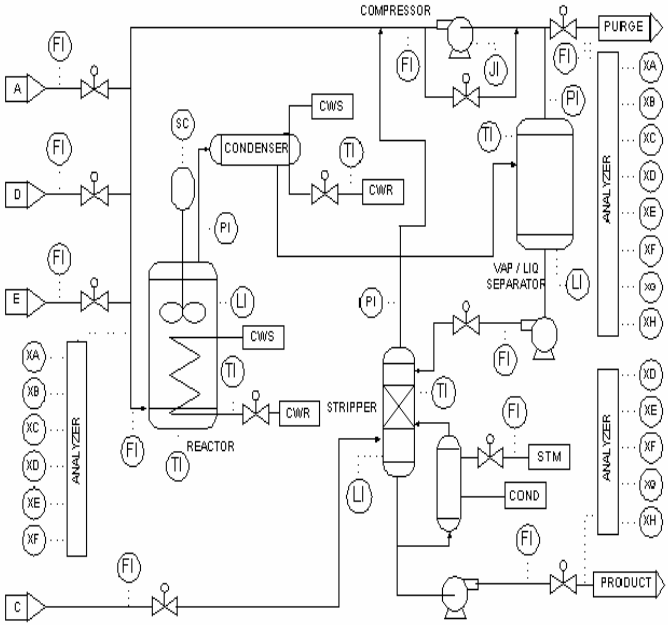

Figure 3: Tennessee Process

The modelling and the identification of the Tennessee Eastman process represent a challenge for the control community. It has been the subject of several studies [2] but most of them have tackled the process control without giving importance to modeling step. [13] have used input/ output process data to identify an autoregressive (AR) model parameters.

In our paper we intend to identify the parameters of the corresponding RKPCA- RN method of this process using the same technique of [20] for generating the data.

5. 2. 2 Data extraction

The input/output data used to built the model were generated from the model of Tennessee implemented for the program Matlab © in the toolbox Simulink [9]. The process has 12 inputs and 41 outputs. According to the work of the process was divided into two fields. The first with a PID controller to maintain the process stability. The second field is devoted to the identification where only four inputs (reactor pressure, reactor level, D feed flow and E feed flow) are tuned and the others are maintained as suggested by mode 3 of Simulink model. Assuming the reactor outputs, we select the separator temperature product.

5. 2. 3 Knowledge model of Tennessee Process

In this section, we consider for the knowledge model of the Tennessee process that suggested by [9] as shown by Figure 4.

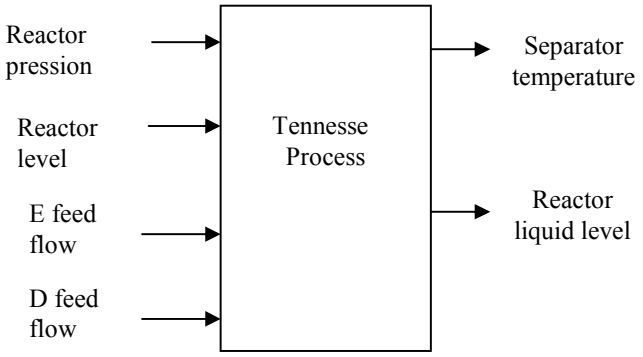

Figure 4: Sheme input / output of Tennessee process

* Output equations:

The first output, separator temperature is given by:

$$T_S = \left(T_{CW,S,out} - T_{CW,S,in}\right) \quad (42)$$

Where:

$T_{CW,S,out}$ Cooling water outlet temperature in the separator

$T_{CW,S,\text{in}}$ Cooling water inlet temperature in the separator

The temperature T_S is linked to the energy Q_S removed for the separator by the differential equation:

$$\dot{Q}_S = m_{CW,S} c_{p,CW} \left(T_{CW,S,out} - T_{CW,S,in}\right) \quad (43)$$

Where:

$C_{p,CW}$ Specific heat capacity cooling water, kJ kg^{-1} K^{-1}

$m_{CW,S}$ Cooling water flow rate separator, kg h^{-1}

For the second output, the reactor liquid level is given by:

$$V_{Lr} = \sum \frac{N_{i,r}}{\rho_i} \; ; i = D, E, F, G, H \quad (44)$$

Where: ρ_i Molar density of component i, mol m^{-3} , $N_{i,r}$ is the total molar holdup of the component i in the reactor.

5. 2. 4 Modelling in RKHS

To generate the data from simulink, the simulation step size was 0.0005 s and the data were collected every 0.02 s.

To build the RKHS model we use the Kernel RBF (Radial Basis Function)

$$K(x, x') = \exp\left(-\frac{\|x - x'\|}{\mu}\right) \quad (45)$$

Where $\mu = 120$, and $\| \; \|$ is the euclidean norm. The term of regularisation $\lambda = 10^{-6}$

The chosen threshold are:
$\varepsilon_1 = 0.01$

The number of observations in phase of identification offline is 300 and the number of principal component analysis obtained by RKPCA method is equal to 10.

The number of observations in online identification phase is 5000.

In Figure 5, we represent the online RKPCA-RN output as well as the Tennessee Eastman output. We remark that the model output is in concordance with the system output, indeed the Normalized mean Square Error is equal to $5,6310^{-6}$. This shows the good performances of the proposed online identification method.

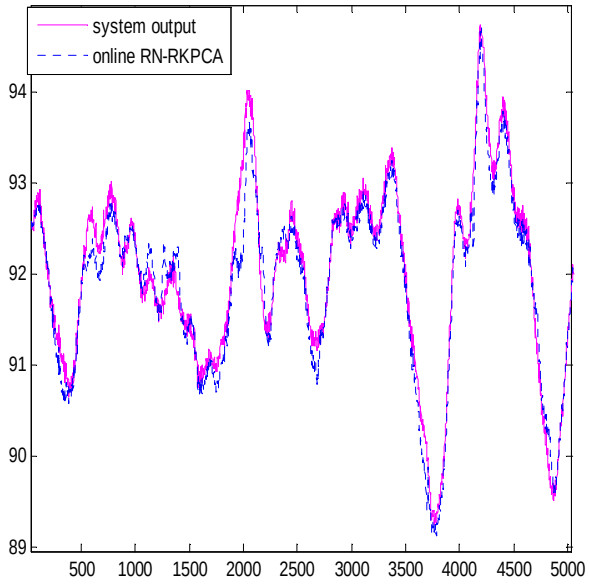

Figure 5: Validation phase

To evaluate the performance of the proposed method we plot in Figure 6, the evolution of the NMSE.

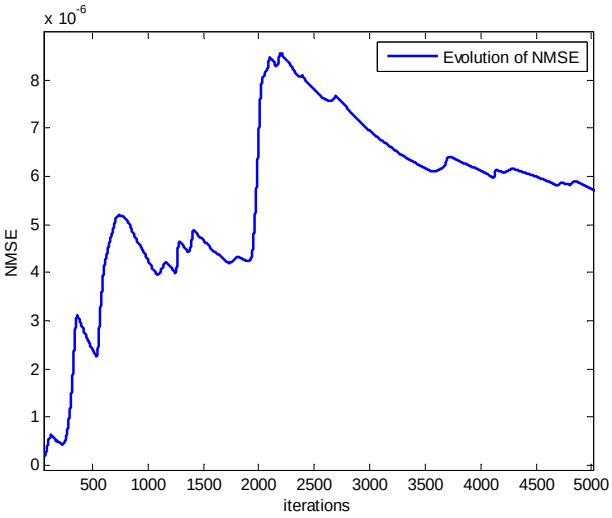

Figure 6: Evolution of NMSE

6 Conclusion

In this paper, we have proposed an online reduced kernel principal component analysis method for nonlinear system parameter identification. Through several experiments, we showed the accuracy and good scaling properties of the proposed method. This algorithm has been tested for identifying a nonlinear system and a Tennessee Eastman process and the results were satisfactying. The proposed technique may be very helpful to design an adaptive control strategy of non linear systems.

References:

[1] Aronszajn N. Theory of reproducing kernels. *Transaction American Mathematical Society* 68, 3,1950, pp. 337- 404,.

[2] Banerjee A and Arkun Y. Control configuration design applied to the Tennessee Eastman plant-wide control problem. *Computers Chemical Engineering*, 19, 1995, pp. 453-480.

[3] Downs, J.J and Vogel, E, A plant-wide industrial process control problem. *Computers and Chemical Engineering,* 17, 1993, pp245–255.

[4] Fernandez R. Predicting Time Series with a Local Support Vecto r Regression Machine. *In ACAI conference*, 1999.

[5] Elaissi I, *Modélisation, Identification et Commande Prédictive des systèmes non linéaires par utilisations des espaces RKHS*, thèse de doctorat 2009, ENIT Tunisie.

[6] Elaissi I., Jaffel I., Taouali O. and Hassani M. Online prediction model based on the SVD–KPCA method. *ISA Transactions, 2013.*

[7] Fabien Lauer, *Machines à Vecteurs de Support et Identification de Systèmes Hybrides*, thèse de doctorat de l'Université Henri Poincaré – Nancy 1, octobre 2008, Nancy, France.

[8] Kuzmin D, Warmuth M.K, Online Kernel PCA with Entropic Matrix Updates, *Proceedings of the 24 the International Conference on Machine Learning*, 2007.

[9] Mauricio Sales Cruz, A.. *Tennessee Eastman Plant –wide Industrial Process*. Technical report, CAPEC, Technical University of Denmark,2004.

[10] Cedric Richard, Senior, José Carlos M. Bermudez, Paul Honeine, *Online prediction of time series data with kernels*, *IEEE TRANSACTIONS ON SIGNAL PROCESSING*, 57, 2009, pp1058-1067.

[11] Scholkopf B. and Smola A. et Muller K-R, Nonlinear Component analysis as Kernel eigenval problem, *Neural computation*. 10, 1998,pp 1299-1319.

[12] Scholkopf B. and Smola A., *Learning with kernels*. The MIT press, 2002.

[13] Sriniwas R. and Arkun. Y. Control of the Tennessee Eastman process using input–output models. *Journal of Process Control*, 7, 1997,pp. 387-400.

[14] Saidi N, Taouali O. and Messaoud H. Modelling of a SISO and MIMO nonlinear communicatio channe using two modelling techniques,*WSEAS TANSACTION ON CIRCUITS AND SYSTEMS*, Issue 6, Vol 8, 2009, pp 518-528.

[15] Taouali O. , Elaissi I. and Hassani M, Online identification of nonlinear system using reduced kernel principal component analysis. *Neural Comput & Application*, vol 21, 2012, pp161-169.

[16] Taouali O, Elaissi I., Garna T. and Messaoud H, A new approach for identification of MIMO nonlinear system with RKHS model *WSEAS TRANSACTIONS on INFORMATION SCIENCE and APPLICATIONS*, issue 7, Vol 7,2010.

[17]Taouali O,Saidi N.and Hassani M, Identification of Nonlinear MISO Process using RKHS and Volterra models, *WSEAS TRANSACTIONS ON SYSTEMS,* issue 6, Vol 8, 2009, pp 723-732**.**

[18] V.N. Vapnik, *Statistical Learning Theory*, Wiley New York., 1998.

[19] G. Wahba, *An introduction to model building with Reproducing Kernel Hilbert Spaces*. Technical report No 1020,Department of Statistics, University of Wisconsin-Madison, 2000.

[20] T. Ylöstalo and H. Hyötyniemi. Maps of Process Dynamics. *Proceedings of EUROSIM'98 Simulation Congress*, K. Juslin, Finland: Federation of European Simulation Societies, 2,1998, pp 225-229.

[21] Lingli Yu, Min Wu, Zixing Cai, Yu Cao. A particle filter and SVM integration framework for fault-proneness prediction in robot dead reckoning system. *WSEAS TRANSACTIONS on SYSTEMS* Vol. 10, No11, 2011, pp 363-375

[22] Neri F. Cooperative evolutive concept learning: an empirical study". *WSEAS Transaction on Information Science and Applications,* WSEAS Press (Wisconsin, USA),Vol. 2, issue 5, 2005,pp. 559-563.

[23] Zeghbib, A.; Palis, F.; Tesenov. G.; Shoylev, N.; Mladenov, V. Performance of surface EMG signals identification using intelligent computational methods. *WSEAS transactions on systems,* 2005,Vol. 4. Issue: 7, pp: 1118-1125.

[24] Sofianita binti Mutalib, marina binti yusoff, Shuzlina binti Abdul rahman, Baseline extraction algorithm for online signature recognition, *WSEAS Transactions on Systems*, 2009.

Methods and Tools for Structural Information Visualization

V. N. KASYANOV

Laboratory for Program Construction and Optimization
Institute of Informatics Systems
Lavrentiev pr. 6, Novosibirsk, 630090
RUSSIA
kvn@iis.nsk.su http://pco.iis.nsk.su/~kvn

Abstract: - In the paper, we consider a practical and general graph formalism called hierarchical graphs and graph models. It is suited for visual processing and can be used in many areas where the strong structuring of information is needed. We present also the Higres, Visual Graph and ALVIS systems that are aimed at supporting of structural information visualization on the base hierarchical graph modes.

Key-Words: - Information visualization, Hierarchical graphs, Hierarchical graph models, Graph algorithm animation, Graph editor, Graph drawing, Graph visualization, Visualization system

1 Introduction

Visualization is a process of transformation of large and complex abstract forms of information into visual form, strengthening user's cognitive abilities and allowing them to take the most optimal decisions.

Graphs are the most common abstract structure encountered in computer science and are widely used for structural information representation [11, 21]. Many graph visualization systems, graph editors and libraries of graph algorithms have been developed in recent years. Examples of these tools include VCG [30], daVinci [7], Graphlet [13], GLT&GET [27], yEd [34] and aiSee [1].

In some application areas the organization of information is too complex to be modeled by a classical graph. To represent a hierarchical kind of diagramming objects, some more powerful graph formalisms have been introduced, e.g. higraphs [9] and compound digraphs [31]. The higraphs are an extension of hypergraphs and can represent complex relations, using multilevel "blobs" that can enclose or intersect each other. The compound digraphs are an extension of directed graphs and allow both inclusion relations and adjacency relations between vertices, but they are less general then the higraph formalism. One of the recent non-classical graph formalisms is the clustered graphs [6]. A clustered graph consists of an undirected graph and its recursive partitioning into subgraphs. It is a relatively general graph formalism that can handle many applications with hierarchical information, and is amenable to graph drawing.

Hence, there is a need for tools capable of visualization of such structures. Although some general-purpose graph visualization systems provide recursive folding of subgraphs, this feature is used only to hide a part of information and cannot help us to visualize hierarchically structural information. Another weak point is that usual graph editors do not have a support for attributed graphs. Though the GML file format, used by Graphlet, can store an arbitrary number of labels associated with graph elements, it is impossible to edit and visualize these labels in the Graphlet graph editor. The standard situation for graph editors is to have one text label for each vertex and, optionally, for each edge.

The size of the graph model to view is a key issue in graph visualization [11]. Large graphs pose several difficult problems. If the number of graph elements is large it can compromise performance or even reach the limits of the viewing platform. Even if it is possible to layout and display all the graph elements, the issue of viewability or usability arises, because it will become impossible to discern between nodes and edges of graph model. It is well known that comprehension and detailed analysis of data in graph structures is easiest when the size of the displayed graph is small. Since none of the static layouts can overcome the problems caused by large graphs, hierarchical presentation, interaction and navigation are essential complements in information visualization.

An algorithm animation visualizes the behavior of an algorithm by producing an abstraction of both the data and the operations of the algorithm [26]. Initially it maps the current state of the algorithm

into an image, which then is animated based on the operations between two succeeding states in the algorithm execution. Animating an algorithm allows for better understanding of the inner workings of the algorithm, furthermore it makes apparent its deficiencies and advantages thus allowing for further optimization.

Applications of graph algorithm visualization can be divided into two types according to the method they implement: interesting events and the data-driven method [4]. Methods of the first type are based on selection of events that occur during execution of an algorithm, for example, comparing the vertex attribute value or removing an edge. Methods of this type create some visual effects for each interesting event. Methods of the second type are based on data changing. During an operation, the memory status is changed, for example, the values of variables. Further these changes are visualized in some understandable way. In the simplest case such changes can be displayed in a form of a table of variable values. This approach is used in debuggers of integrated development environments.

The existing algorithm visualizers have several disadvantages. One of the major drawbacks is that if there is a need to build visualization of an algorithm arbitrarily close to the original algorithm, then it is necessary to build a new visualizer. As a rule, visualizers also do not show the correspondence between the algorithm instructions and the generated visual effects or do not allow reassignment of visual effects to the corresponding events.

In the paper, we consider a practical and general graph formalism called hierarchical graphs and graph models [14]. It is suited for visual processing and can be used in many areas where the strong structuring of information is needed [3, 15, 16, 17, 20, 21, 22, 23, 24, 25, 28, 29]. We present also the Higres, Visual Graph and ALVIS systems that are aimed at supporting of information visualization on the base hierarchical graph modes. The Higres system is a visualization tool and an editor for attributed hierarchical graphs and a platform for execution and animation of graph algorithms [12]. The Visual Graph system was developed to visualize and explore large hierarchical graphs that present the internal structured information typically found in compilers [32]. The ALVIS system was developed to build the algorithm visualization with the help of a flexible system of visual effects and using a visualized graph algorithm as an input parameter.

2 Hierarchical graphs and graph models

2.1 Hierarchical graphs

Let G be a graph of some type, e.g. G can be an undirected graph, a digraph or a hypergraph. A graph C is called a *fragment* of G, denoted by $C \subseteq G$, if C includes only elements (vertices and edges) of G. A set of fragments F is called *a hierarchy of nested fragments* of the graph G, if $G \in F$ and $C_1 \subseteq C_2$, $C_2 \subseteq C_1$ or $C_1 \cap C_2 = \varnothing$ for any $C_1, C_2 \in F$.

A *hierarchical graph* $H = (G,T)$ consists of a graph G and a rooted tree T that represents an immediate inclusion relation between fragments of a hierarchy F of nested fragments of G. G is called the *underlying graph* of H. T is called the *inclusion tree* of H.

A hierarchical graph H is called a *connected* one, if each fragment of H is connected graph, and a *simple* one, if all fragments of H are induced subgraphs of G.

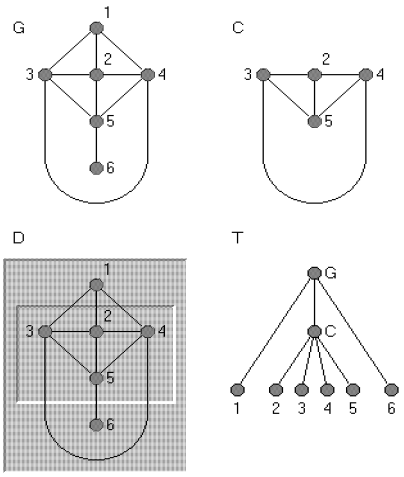

Fig. 1. A simple hierarchical graph $H=(G, T)$ and its drawing D

It should be noted that any clustered graph H can be considered as a simple hierarchical graph $H=(G, T)$, such that G is an undirected graph and the leaves of T are exactly the trivial subgraphs of G (See Fig. 1)

2.2 Hierarchical graph drawing

A *drawing* (or *layout*) D of a hierarchical graph $H = (G,T)$ is a representation of H in the plane such that the following properties hold (See Fig. 1).

- Each vertex of *G* is represented either by a point or by a simple closed region. The region is defined by its *boundary* - a simple closed curve in the plane.
- Each fragment of *G* is drawn as a simple closed region which include all vertices and subfragments of the fragment.
- Each edge of *G* is represented by a simple curve between the drawings of its endpoints.

D is a *structural* drawing of *H* if all edges of any fragment of *H* are located inside the region of the fragment. A hierarchical graph is called a *planar* one if it has such a structural drawing that there are no crossing between distinct edges and the boundaries of distinct fragments. The following properties hold.

Theorem 1. There are nonplanar hierarchical graphs *H=(G,T)* with planar underlying graphs *G*.

Theorem 2. There are nonplanar hierarchical graphs *H=(G,T)* having nonstructural planar drawing.

Theorem 3. A simple connected hierarchical graph *H=(G,T)* is a planar graph if and only if there is such a planar drawing *D* of *G* that for any vertex *p* of *T* all vertices and edges of *G-G(p)* are in the outer face of the drawing of *G(p).*

2.3 Hierarchical graph models

Let *V* be a set of objects called simple *labels* (e.g. *V* can include some numbers, strings, terms and graphs). Let *W* be a set of *label types* of graph elements and let a label set $V(w)= V_1 \times V_2 \times \ldots \times V_s$, where $s \geq 1$ and for any i, $1 \leq i \leq s$, $V_i \subseteq V$, be associated with each $w \in W$.

A labelled hierarchical graph is a triple *(H,M,L),* where *H* is a hierarchical graph, *M* is *a type function* which assigns to each element (vertex, edge and fragment) *h* of *H* its type $M(h) \in W$, and *L* is a *label function,* which assigns to each element *h* of *H* its label $L(h) \in V(M(h))$.

The semantics of a hierarchical graph model is provided by an equivalence relation which can be specified in different ways, e.g. it can be defined via *invariants* (i.e. properties being inherent in equivalent labelled graphs) or by means of so-called *equivalent* transformations that preserve the invariants.

2.4 Hierarchical control flow graphs

Many problems in program optimization have been solved by applying a technique called *interval analysis* to the control flow graph of the program [10, 18]. A control flow graph which is susceptible to this type of analysis is called *reducible*.

Let *F* be a minimal set which includes *G* and is closed under the following property: if $C \in F$ and *p* is such an entry vertix of *C* that subgraph {*p*} does not belong to *F* then *F* contains all maximum strongly connected subgraphs of graph which is obtained from *C* by removing of all edges which are ingoing in *p*. Let $H_F=(G,T)$ be such a simple hierarchical graph that *T* represents an immediate inclusion relation between fragments of the hierarchy *F*.

The following properties hold.

Theorem 4. A control flow graph *G* is reducible if and only if for the simple hierarchical graph $H_F=(G,T)$ the set of all fragments corresponding vertices $p \in T$ is a hierarchy of nested single-entry strongly connected regions.

Theorem 5. A control flow graph *G* is reducible if and only if that for any $p \in T$ of the simple hierarchical graph $H_F=(G,T)$ the fragment which is obtained from fragment corresponding to *p* by reducing all its inner fragments from *F* into their entry vertices is an interval.

3 The Higres system

3.1 Graph models in Higres

A hierarchical graph supported by the Higres consists of vertices, fragments and edges which we call objects (See Fig. 2). Vertices and edges form an underlying graph. This graph can be directed or undirected. Multiple edges and loops are also allowed.

The semantics of a hierarchical graph is represented in Higres by means of object types and external modules (see below). Each object in the graph belongs to an object type with a defined set of labels. Each label has its data type, name and several other parameters. A set of values is associated with each object according to the set of labels defined for the object type to which this object belongs. These values, along with partitioning of objects to types, represent the semantics of the graph. New object types and labels can be created by the user.

3.2 Visualization

In the Higres system each fragment is represented by a rectangle. All vertices of this fragment and all subfragments are located inside this rectangle.

Fragments, as well as vertices, never overlap each other. Each fragment can be closed or open (See Fig. 2). When a fragment is open, its content is visible; when it is closed, it is drawn as an empty rectangle with only label text inside it. A separate window can be opened to observe each fragment. Only content of this fragment is shown in this window, though it is possible to see this content inside windows of parent fragments if the fragment is open.

Most part of visual attributes of an object is defined by its type. This means that semantically relative objects have similar visual representation. The Higres system uses a flexible technique to visualize object labels. The user specifies a text template for each object type. This template is used to create the label text of objects of the given type by inserting labels' values of an object.

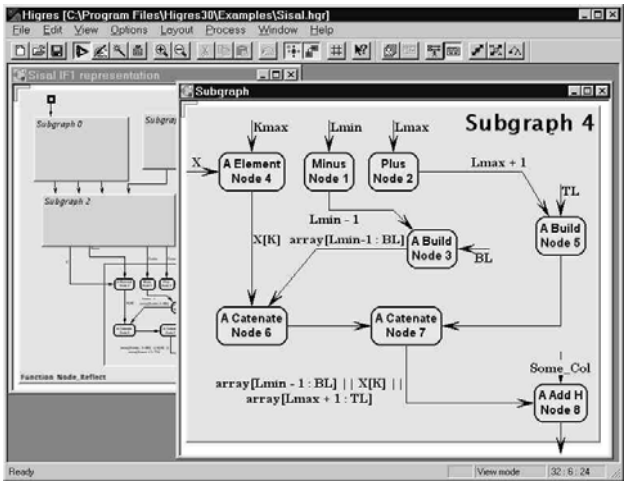

Fig. 2. The Higres system

Other visualization features include the following:

- various shapes and styles for vertices;
- polyline and smooth curved edges;
- various styles for edge lines and arrows;
- the possibility to scale graph image to an arbitrary size;
- edge text movable along the edge line; colour selection for all graph components;
- external vertex text movable around the vertex; font selection for labels text;
- two graphical output formats;
- a number of options to control the graph visualization.

Now Higres uses three graph drawing algorithms for automatic graph allocation. The first one is a force method, which is very close to original algorithm from [5]. The second one is our improvement of the first. The third one allocates rooted trees on layers.

3.3 The user interface

The comfortable and intuitive user interface was one of our main objectives in developing Higres. The system's main window contains a toolbar that provides a quick access to frequently used menu commands and object type selection for creation of new objects. The status bar displays menu and toolbar hints and other useful information on current edit operation.

The system uses two basic modes: view and edit. In the view mode it is possible only to open/close fragments and fragment windows, but the scrolling operations are extended with mouse scrolling. In the edit mode the left mouse button is used to select objects and the right mouse button displays the popup menu, in which the user can choose the operation he/she wants to perform. It is also possible to create new objects by selecting commands in this menu. The left mouse button can be also used to move vertices, fragments, labels texts and edge bends, and resize vertices and fragments. All edit operations are gathered in a single edit mode. To our opinion, it is more useful approach (especially for inexperienced users) than division into several modes. However, for adherents of the last case we provide two additional modes. Their usage is optional but in some cases they may be useful: the "creation" mode for object creation and "labels" mode for labels editing.

Other interface features include the following:

- almost unlimited number of undo levels;
- optimized screen update; automatic elimination of objects overlapping;
- automatic vertex size adjusting;
- grid with several parameters;
- a number of options that configure the user interface;
- online help available for each menu, dialog box and editor mode.

3.4 Algorithm animation

To run an algorithm in the Higres system, the user should select an external module in the dialog box. The system starts this module and opens the process window that is used to control the algorithm execution. Higres provides the run-time animation of algorithms. It also caches samples for the repeated and backward animation. A set of

parameters is defined inside a module. These parameters can be changed by the user at any execution step. The module can ask user to input strings and numbers. It can also send any textual information to the protocol that is shown in the process window.

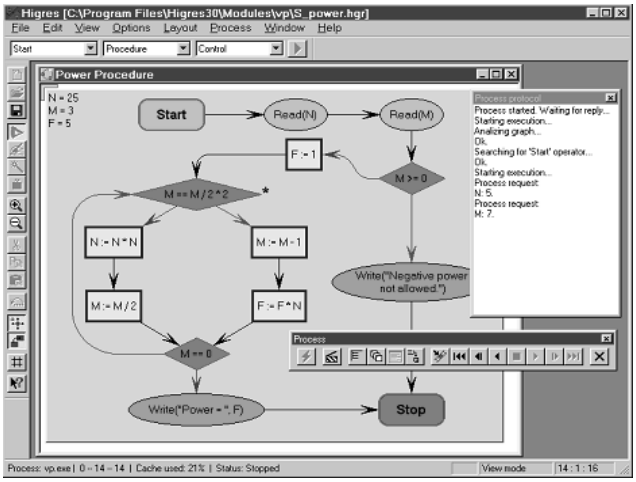

Fig. 3. Simulation of block-scheme representation of a program

A wide range of semantic and graph drawing algorithms can be implemented as external modules. As examples now we have modules that simulate finite automata, Petry nets and imperative program schemes (See Fig.3). The animation feature can be used for algorithm debugging, educational purposes and exploration of iteration processes such as force methods in graph drawing.

A special C++ API that can be used to create external modules is provided. This API includes functions for graph modification and functions that provide interaction with the Higres system. It is unnecessary for programmer, who uses this API, to know the details of the internal representation of graphs and system/module communication interface. Hence, the creation of new modules in the Higres system is a rather simple work.

4 The Visual Graph system

4.1 Visual Graph

Visual Graph is a tool that automatically calculates a customizable multi-aspect layout of hierarchical graph models specified in GraphML (see below). This layout is then displayed, and can be interactively explored, extended and analyzed.

Visual Graph was developed to visualize and explore large graphs that present the internal structured information typically found in compilers. Visual Graph reads a textual and human-readable GraphML-specification and visualizes the hierarchical graph models specified (See Fig. 4). Its design has been optimized to handle large graphs automatically generated by compilers and other applications.

Visual Graph provides tools for analyzing graph structures. *Structural analysis* means solving advanced questions that relate to a graph structure, for instance, determining a shortest path between two nodes.

Simple possibilities to extend the functionality of Visual Graph (for example, to add a new layout, search, analysis or navigating algorithm, a new tool for processing information associated with elements of graph models and so on) are provided.

4.2 GraphML

GraphML (Graph Markup Language) [2] is a comprehensive and easy-to-use file format for graphs. It consists of a language core (known as the Structual Layer) to describe structural properties of one or more graphs and a flexible extension mechanism, e.g. to add application-specific data. Its main features include support of directed, undirected, mixed multigraphs, hypergraphs, hierarchical graphs, multiple graphs in a single file, application-specific data, and references to external data.

Two extensions, adding support of meta-information for light-weight parsers (Parse Extension) and typed attribute data (Attributes Extension) are currently part of the GraphML specification.

Unlike many other file formats for graphs, GraphML does not use a custom syntax. Instead, it is defined as an XML (Extensible Markup Language) sublanguage and hence ideally suited as an exchange format for all kinds of services generating or processing graphs

4.3 Reducing layout time

Visual Graph was designed to explore large graphs that consist of many hundreds of thousands of elements. However, the layout of large graphs may require considerable time. Thus, there are two main ways to speed up the layout algorithm: multi-aspect layout of graph and control of layout algorithms.

The first way in visualizing a large graph is aimed at avoiding computing the layout of parts of the graph that are currently not of interest. Interactive exploring of graph is based on step by step construction of so-called *multi-aspect layout* of

graph being a set of drawings of some subgraphs of the graph. For presentation of multi-aspects layout a set of windows which includes a separate window for visualization of each considered subgraph is used. At each step of the construction a layout algorithm is applied to a subgraph being interested to user at this step. To indicate the interested subgraph the user can select its elements in the active window or in the navigator (see below). The user can also define some condition in the filter or in the search panel (see below). Then the condition will be used for searching of graph elements which will form the interested subgraph. The search can be performed both locally (in some part of graph, e.g. through a subgraph presented in the active window) or globally (around the entire graph). Multi-aspect drawing of graph models makes every visible part of the graph smaller, thus enabling the layout to be calculated faster and the quality of the layout to be improved.

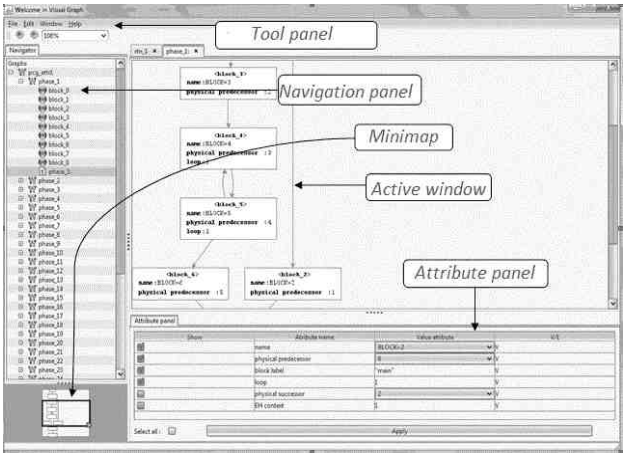

Fig. 4. The Visual Graph system

In order to further reduce layout time, it is possible to control the layout algorithms, e.g. some layout phases can be omitted or the maximum number of iterations of some layout phases can be limited. However, this usually decreases the quality of the layout. The user can improve the layout by hand, e.g. by moving of nodes or changing of their sizes or forms.

4.4 Navigating through a graph

Visual Graph offers several tools for navigating through a graph model: minimap, navigator, attribute panel, filter, search panel, notebook.

The *minimap* visualizes a *working region* of the graph model explored (See Fig. 4). It shows both the whole subgraph from the active windows and its visible part (i.e. such a subgraph part that is allocated in the active window). It is possible to change both the visible part of the subgraph from the working region and scale of its drawing.

The *navigator* visualizes inclusion trees of hierarchical graph models as hierarchies of their vertices. It is possible to open in the active window any vertex of any group of vertices.

The *attribute panel* allows for elements of graph model in the working region choose attributes which should be visualized and ways of their visualization.

The *filter* is used for searching such elements of graph in the working region which satisfy given conditions (See Fig. 5).

The *search panel* is intended to search such elements of either whole graph model or its given part which satisfy given conditions.

The *notebook* can load files with additional information (e.g. with a source program being compiled) and associate it with graph elements.

5 The ALVIS system

5.1 Interactive visualization model

A new algorithm visualization model based on the dynamic approach and hierarchical graph models has been created. The main point of the suggested model is that the given algorithm is formulated in some programming language that allows us to use instructions operating with graphs and to execute the program derived from the text of the algorithm after a set of transformations. More details about the model can be found in [8]. The result of the program execution is information which is to be used in creation of the underlying algorithm visualization. An example of such instruction can be adding an edge or a change in the attributes of vertices. The following example shows the breadth-first search algorithm for any graph. In the given case, *Get* and *Set* instructions are used for reading and changing the graph element's attribute values. These instructions have formats *Get*(*Vertex*, *AttributeName*) and *Set*(*Vertex*, *AttributeName*, *A ttributeValue*), respectively. To construct a visualization of the breadth-first search algorithm, the state attribute is appointed to each graph vertex. The value of the state attribute reflects whether the vertex was visited during graph traversal.

```
VertexQueue.Enqueue(Graph.Vertices[0]);
while (VertexQueue.Count > 0)
```

```
{
Vertex v = VertexQueue.Dequeue();
Set(v, ”state”, ”visited”);
foreach(Edge e in v.InEdges)
{
Vertex t =e.PortFrom.Owner;
string c = Get(t, “state”);
if(c != ”visited”)
{
Set(t, ”state”, ”visited”);
VertexQueue.Enqueue(t);
}
}
foreach(Edge e in v.OutEdges)
{
Vertex t = e.PortTo.Owner;
string c = Get(t, “state”);
if(c != ”visited”)
{
Set(t, ”state”, ”visited”);
VertexQueue.Enqueue(t);
}
}
}
VertexQueue.Clear();
```

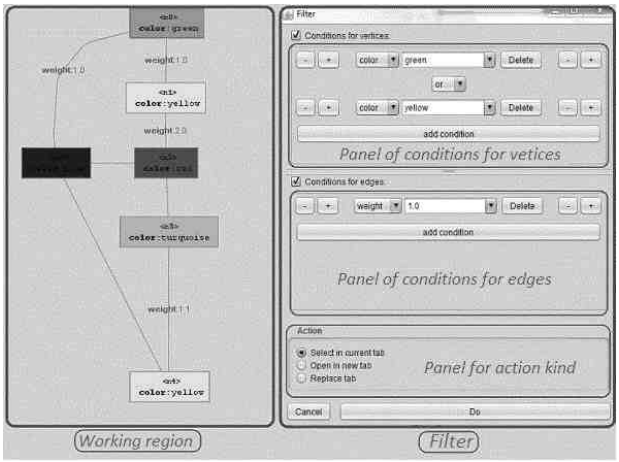

Fig. 5. The filter of the Visual Graph system

Each instruction of the algorithm generates one or more images of the current state of the graph model. The graph model is an annotated hierarchical marked graph. It is useful to highlight the current executing instruction in each image because it allows a user to keep attention on valuable events at this moment. To solve the problem of highlighting the current executing instruction in the image, the following approach is used. Each text line of an algorithm can be interpreted as a function. Also, each text line has a numeric index in all text lines. So that order value is added to arguments of the function corresponding to the text line. This additional parameter is the number of the current executing algorithm instruction. After this transformation, the text of the breadth- first search algorithm from the above example looks like this:

```
VertexQueue.Enqueue(Graph.Vertices[0]);
while (WhileCondition(2, VertexQueue.Count >
0))
{
Vertex v = VertexQueue.Dequeue(3);
Set(4, v.ID, “state”, “visited”);
foreach(Edge e in ForeachCollection(5,
v.InEdges))
{
Vertex t = e.PortFrom.Owner;
string c = Get(7, t, ”state”);
if(If Condition(8, c != ”visited”))
{
Set(10, t, ”state”, ”visited”);
VertexQueue.Enqueue(11, t);
}
}
foreach(Edge e in ForeachCollection(13,
v.OutEdges))
{
Vertex t = e.PortTo.Owner;
string c = Get(16, t, ”state”);
if(If Condition(17, c != ”visited”))
{Set(19, t, ”state”, ”visited”);
VertexQueue.Enqueue(20, t);
}
}
}
VertexQueue.Clear( );
```

The above example shows changes in the attributes of the graph elements, too. This is a typical situation for algorithms implementing only traversal of a graph - a method when all graph vertices are visited one by one. For example, the Prüfer sequence of a given tree is generated by iteratively removing vertices from the tree until only two vertices remain. To perform this operation, the *RemoveVertex*( ) instruction should be used, which leads to generation of a visual effect of the corresponding vertex disappearing. Here is an example of the Prüfer encoding algorithm, how it can be formulated as a parameter of the graph algorithm visualization system:

```
int i = 0;
List<Vertex> Leafs = new List<Vertex>( );
int n = Graph.Vertices.Count;
while(i++ <= n-2)
{
Leafs.Clear();
foreach(Vertex v in Graph.Vertices)
if(v.OutEdges.Count == 0) Leafs.Add(v);
Vertex codeItem =
Leafs[0].InEdges[0].PortFrom.Owner;
Output.Add(codeItem);RemoveVertex(Leafs[0]);
}
```

Each algorithm instruction generates some information during execution of the transformed text of the original algorithm. This information describes the number of the current instruction, the name of an attribute of a graph element, the previous value of the attribute, a new value of the attribute and the identifier of the graph element. This information allows us to get the full log of operations executed over graph elements. This operation log contains the detailed information on the state of the graph model during the algorithm running. Further the log of operations, the input graph and the original text of the algorithm can be used to generate the algorithm visualization. Each operation log entry corresponds to some graphical effect over visual representation of graph elements. The simplest example of the visual effect for the breadth-first search algorithm is to change the color of the graph vertex representation when a state attribute of the vertex has been changed and to change the color of the text of the corresponding instruction.

5.2 Algorithm visualization system

The ALVIS system for graph algorithm visualization on the base of the described model has been constructed.

The system includes two main components: an algorithm execution module and a graph algorithm visualizer. It is assumed that data are passed between these and other components of the system in a text form. It means that components of the system can be implemented on different platforms and with different tools.

The purpose of the algorithm execution module is to generate the execution log. The algorithm running is separated from its visualization. This allows us to perform the algorithm once and after that the operation log can be used to visualize and refine the visualization many times. This can be useful when computationally-intensive algorithms are visualized. In such cases the second cycle of execution of the algorithm is complex.

To provide correct work of the algorithm execution module, it is necessary to meet a significant condition. Since any existing compiler or interpreter can be used to create this module, the algorithm must be formulated in the language supported by the selected compiler or interpreter. Actually this is not a restriction on the algorithm implementation language since many programming languages allow graph structures to be used in the program source code. So, the given algorithm text can be considered as a ready program source code. Also this allows us to transmit the input graph in this compiled program and to generate the log of operations.

Another significant restriction relates to the algorithmic complexity. In this approach, it is reasonable to visualize only efficient algorithms, because it will take much time to build the operation log of execution of an efficient algorithm. We can use a small input graph for this case. This assumption allows us to construct visualization for a reasonable time.

The algorithm execution module takes the given algorithm text in an appropriate programming language, executes it and returns the log of operations generated during the algorithm run on a particular graph. The log of executed operations contains information about all changed attributes of graph elements and about graph elements added or removed during the execution. Further this information is used to generate the algorithm visualization.

Another main component of the ALVIS system is the graph algorithm visualizer. At its input, this component receives the algorithm text, the input graph, the log of operations and additional graphical options. A log information item is added by special instructions created at the stage of preparation of the algorithm text. For example, these special instructions are the functions: *Set*( ), *Get*( ), *IfCondition*( ), *WhileCondition*( ) and *ForeachCollection*( ). Their first argument is the number of the corresponding text line. *IfCondition*( ) and *WhileCondition*( ) do not perform any changes in the graph model state but at least allow us to make a visual selection of the text line where it was inserted. *ForeachCollection*( ) is to be used to generate information which allows highlighting a set of vertices before they will be actually enumerated. To add these functions into appropriate places of the original algorithm, it is sufficient to use a contextual replacement. The purpose of the preparation stage is to eliminate the

need for declarative structures, which have no relation to the actual nature of the algorithm.

A log item may also contain information about the value of an attribute of a graph element. A graph element is a vertex, an edge or a port. If there is a vertex with its incident edge, then a port is a point where the edge enters the vertex. When rendering, it can be useful that the points are allocated for these additional objects. Ports simplify calculation of coordinates of graphical primitives which represent the edge elements. Strictly mathematically, it is possible to simulate a port with a labeled vertex. So the class of graphs with ports is isomorphic to the class of all graphs.

An attribute of a vertex, an edge or a port can have a string name and a string value. The log of operations stores the previous value of the attribute for a particular graph element. This information is also useful for building the visualization, since it is possible to make a smooth visual effect from a previous value of an attribute to its new value.

It is not obvious how to bind information from a log item to the visual effect. In this case, a user needs to interfere in order to set an explicit binding between the set of attributes in the text of the algorithm and the desired visual effects. For example, if the operation of a log item is about changing the coordinates of the graph element reflected with the use of the attribute "position", then it is reasonable to bind the attribute with the visual effect, which leads to a shift of the graph element. Another user example is to bind all log items to the effect of a color mark of a current graph element under processing. It can be a current vertex visited in the algorithm of depth-first search or in any other graph traversal. In this aspect the suggested approach is close to the interesting events approach, where an algorithm instruction is an interesting event.

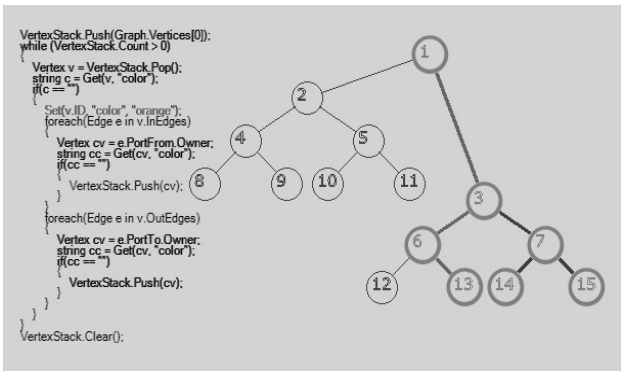

Fig. 6. Visualization of the depth-first search algorithm

Fig. 6 shows an example of visualization of the depth-first search algorithm on the graph, which is actually a binary tree. It is one of the screenshots taken during the process of visualization of the depth-first search algorithm. The left side of the figure displays the text of the algorithm formulated in terms of graph operations. The attribute of a graph vertex state indicates the fact that the vertex has already been visited during the process of the graph traversal. A line of the algorithm text has one of the following states: dark thin, light thin and thick. The first state means that the instruction has been executed at least once. The second state means that the current image and the last shown visual effect is the result of this instruction. The last state means that the instruction has not been executed yet. The right part of the figure displays the graph model, which is a hierarchical graph with attributes. Only if this attribute is set, the corresponding attribute will be created during visualization. In this example, the visited vertices get the state attribute that changes the color of a vertex. Also, this attribute's value corresponds to the increase of line width showing the graph vertex circle. Vertices shown in a thin line have not been visited yet.

Displaying of additional data structures can also be used to improve understanding of visualization of a graph algorithm. For example, the depth-first search algorithm uses a stack and the breadth-first search algorithm visualization uses a queue. The content of a stack or a queue can be represented as a graph. Since the visualization system allows us to use the hierarchical graphs, a stack graph or a queue graph can be included into a graph model for a particular visualization. So the working graph model consists of a graph with two vertices. The first vertex contains a stack graph and the second contains an input graph. Such graph model can be visualized with the created module of the system of graph algorithm visualization. The queue or stack size is changed during execution of the given algorithm and the corresponding vertices are added or removed from the stack graph. Hierarchical graphs are helpful for this purpose. If there is no stack or queue, then a tree of fragments only consists of one fragment, the input graph. For a stack the graph model consists of three fragments: a root and two children. The first child is the input graph and the second is a graph representation of the stack. So, if the given algorithm uses an input graph and N additional structures, then the tree of fragments contains N+2 elements. It is a root element and its N+1 children,

one of which is the input graph and others are graph representations of additional data structures.

6 Conclusion

In the paper, a practical and general graph formalism of hierarchical graphs and graph models was considered. It is suited for visual processing and can be used in many areas where the visualization of structural information is needed

The Higres system being a visualization tool and an editor for attributed hierarchical graphs and a platform for execution and animation of graph algorithms was presented. The Visual Graph system intended to visualize and explore large hierarchical graphs that present the internal structured information typically found in compilers was considered. The ALVIS system which builds the algorithm visualization with the help of a flexible system of visual effects and using a visualized graph algorithm as an input parameter was described.

The described methods and systems will be used in Web-Encyclopedia of Graph Algorithms (WEGA) [33] which is under development on the basis of our book [19] and is aimed to be not only a reference manual on graph algorithms but also an introduction to the graph theory and its applications to computer science.

In contrast to Donald Knuth who used the assembly language of the so-called MIX computer in his fundamental books “The art of computer programming”, we decided to use a high-level and language-independent representation of graph algorithms in our book and system. In our view, such an approach is preferential, as it allows us to describe algorithms in a form that admits direct analysis of their correctness and complexity, as well as a simple translation of algorithms to high-level programming languages without disturbance of their correctness and complexity. Besides, the described approach allows the readers to understand an algorithm at the informative level, to evaluate its applicability to a specific problem, and to make all its modifications needed for correct application of the algorithm.

We also believe that visualization could be very helpful for readers in understanding graph algorithms, and we plan to embed capabilities of interactive animation of graph algorithms which will be described into the WEGA system.

The author is thankful to all colleagues taking part in the projects described. The work was partially supported by the Russian Foundation for Basic Research (grant N 12-07-0091).

References:

[1] aiSee http://www.absint.com/aisee/
[2] U. Brandes, M.S. Marshall, and S.C. North, Graph data format workshop report, *Lecture Notes in Computer Science*, Vol. 1984, 2001, pp. 410–418.
[3] F. Colas, J.-Y. Dieulot, P-J. Barre, P. Borne, Dynamics modeling and causal ordering graph representation of a non minimum phase flexible arm fixed on a cart, *WSEAS Transactions on Computers*, Vol. 5, Issue 1, 2006, p 225-232.
[4] C. Demetrescu, I. Finocchi, J.T. Stasko, Specifying algorithm visualizations: interesting events or state mapping? *Lecture Notes in Computer Science*, Vol. 2269, 2002, pp.16–30.
[5] P. Eades, A heuristic for graph drawing, *Congressus Numerantium*, Vol. 42, 1984, pp. 149-160.
[6] Q.W. Feng, R.F. Cohen, P. Eades, Planarity for clustered graphs, *Lecture Notes in Computer Science*, Vol. 979, 1995, pp. 213-226.
[7] M. Fröhlich, M. Werner, Demonstration of the interactive graph visualization system daVinci, *Lecture Notes in Computer Science*, Vol. 959, 1995, pp. 266-269.
[8] D.S. Gordeev, Graph algorithm visualization: interpretation of algorithm as a program, *Informatics in Research and Education*, Novosibirsk, 2012, pp. 149-160. (In Russian).
[9] D. Harel, On visual formalism, *Comm. ACM*, Vol. 31, No. 5, 1988, pp. 514-530.
[10] M.S. Hecht, *Flow Analysis of Computer Programs*, New York, Elsevier, 1977.
[11] I. Herman, G. Melançon, M.S. Marshall, Graph visualization and navigation in information visualization: a survey, *IEEE Transactions on Visualization and Computer Graphics*, Vol. 6, 2000, pp. 24-43.
[12] Higres http://pco.iis.nsk.su/higres
[13] M. Himsolt, The Graphlet system (system demonstration), *Lecture Notes in Computer Science*, Vol. 1190, 1997, pp. 233-240.
[14] V.N. Kasyanov, Hierarchical graphs and graph models: problems of visual processing, *Problems of Informatics Systems and Programming*, Novosibirsk, 1999, pp. 7-32. (In Russian)
[15] V.N. Kasyanov, Support tools for graphs in computer science, *Proc. of the 15th ACM SIGCSE Conference on Innovation and*

Technology in Computer Science Education (ITiCSE 2010), ACM Press, 2010, p.315.
[16] V.N. Kasyanov, Information visualization on the base of hierarchical graph models, *Advances in Applied Information Science. Proc. of AIC'12 and BEBI'12*, WSEAS Press, 2012, pp. 115-120.
[17] V.N. Kasyanov, Hierarchical graph models and information visualization, *Proceedings of the 2012 Third World Congress on Software Engineering (WCSE 2012)*, IEEE Computer Society, 2012, pp. 79-82.
[18] V.N. Kasyanov, V.A. Evstigneev, *Graph Theory for Programmers. Algorithms for Processing Trees*, Kluwer Academic Publ., 2000.
[19] V.N. Kasyanov, V.A. Evstigneev, *Graphs in Programming: Processing, Visualization and Application*, St. Petersburg, BHV-Petersburg, 2003, 1104 p. (In Russian).
[20] V.N. Kasyanov, E.V. Kasyanova, A Web-based system for distance learning of programming, *Lecture Notes in Electrical Engineering*, Vol. 27, 2009, pp. 453-462.
[21] V.N. Kasyanov, E.V. Kasyanova, *Visualization of Graphs and Graph Models*, Novosibirsk, Siberian Scientific Publ., 2010. (In Russian).
[22] V.N. Kasyanov, E.V. Kasyanova, Information visualization based on graph models, *Enterprise Information Systems*, Vol. 7, N 2, 2013, pp. 187-197.
[23] V.N. Kasyanov, I.A. Lisitsyn, Support tools for hierarchical information visualization, *Human-Computer Interaction: Communication, Cooperation and Application Design*. Vol. 2. Lawrence Erlbaum Associates Publ., London, 1999, pp.117 - 121.
[24] V.N. Kasyanov, I.A. Lisitsyn, Hierarchical graph models and visual processing, *Proceedings of Conference on Software: Theory and Practice. 16th IFIP World Computer Congress 2000*, Beijing, PHEI, 2000, pp. 179-182.
[25] V.N. Kasyanov, A.P. Stasenko, M.P. Gluhankov, P.A. Dortman, K.A. Pyjov, A.I. Sinyakov, SFP - an interactive visual environment for supporting of functional programming and supercomputing, *WSEAS Transactions on Computers*, Vol. 5, Issue 9, 2006, pp. 2063-2069.
[26] A. Kerren and J. Stasko, Algorithm animation - Introduction, *Lecture Notes in Computer Science*, Vol. 2269, 2002, pp. 1-15.
[27] B. Madden, P. Madden, S. Powers, M. Himsolt, Portable graph layout and editing, *Lecture Notes in Computer Science*, Vol. 1027, 1996, pp. 385-395.
[28] A. Mansoor, A graph based method for faster display for distribution networks, *WSEAS Transactions on Computers*, Vol. 7, Issue 6, 2008, pp. 620-629.
[29] F. Neri, Cooperative evolutive concept learning: an empirical study, *WSEAS Transactions on Information Science and Applications*, Vol. 2, Issue 5, 2005, pp. 559-563.
[30] G. Sander, Graph layout through the VCG tool, *Lecture Notes in Computer Science*, Vol. 959, 1995, pp.194-205.
[31] K. Sugiyama, K. Misue, Visualization of structured digraphs, *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 21, No. 4, 1999, pp. 876-892.
[32] Visual Graph http://www.visualgraph.sourceforge.net
[33] WEGA http://pco.iis.nsk.su/WEGA/
[34] yEd http://www.yworks.com/en/

Recognition of Assamese Spoken Words using a Hybrid Neural Framework and Clustering Aided Apriori Knowledge

PALLABI TALUKDAR, MOUSMITA SARMA and KANDARPA KUMAR SARMA
Gauhati University
Department of Electronics and Communication Technology
Guwahati-781014, Assam
INDIA
(pallabiz95, go4mou, kandarpaks)@gmail.com

Abstract: In this paper, an Artificial Neural Network (ANN) based model is proposed for recognition of discrete Assamese speech using a Self Organizing Map (SOM) based phoneme count determination technique. The phoneme count determination technique takes some initial decision about the possible number of phonemes in the word to be recognized and accordingly the word is presented to some N-phoneme recognition algorithm. In this paper recognition algorithm is designed to recognize three phoneme consonant-vowel-consonant (CVC) type Assamese words. The word recognizer is consisted of another SOM block to provide phoneme boundaries and Probabilistic Neural Network (PNN) and Learning Vector Quantization (LVQ) to identify the SOM segmented phonemes. The recognition of constituent phonemes in turn represents the discrimination between incoming words with a minimum success rate of 90%.

Key–Words: Formant, Phoneme, ANN, KMC, LPC, DWT.

1 Introduction

Spoken word recognition is a distinct subsystem providing the interface between low-level perception and cognitive processes of retrieval, parsing, and interpretation of speech. The process of recognizing a spoken word starts from a string of phonemes, establishes how these phonemes should be grouped to form words, and passes these words onto the next level of processing. There are several theories in the literature which focuses on the discrimination and categorization of speech sounds. One of the earliest theory in this contrast is the Motor theory proposed by Alvin Liberman, Franklin Cooper, Pierre Delattre and other researchers in 1950 [1]. Motor theory postulates that speech is perceived by reference to how it is produced. It means that, when perceiving speech, listeners access their own knowledge of how phonemes are articulated. Analysis by Synthesis Model, proposed by Stevens and Halle, 1960 [2], in turn stated that speech perception is based on auditory matching mediated through speech production. Research on the discrimination and categorization of phonetic segments was the key focus of the works on speech perception before 1970s. The processes and representations responsible for the perception of spoken words became a primary object of scientific inquiry with a curiosity of disclosing the cause and methods of how the listener perceives fluent speech. According to Cohort theory (1980) [3][4], various language sources like lexical, syntactic, semantic etc. interact with each other in complex ways to produce an efficient analysis of spoken language. It suggest that input in the form of a spoken word activates a set of similar items in the memory, which is called word initial cohort. The word initial cohort consisted of all words known to the listener that begin with the initial segment or segments of the input words. In 1994, Marslen-Wilson and Warren revised the Cohort theory. In the original version, words were either in or out of the word cohort. But in the revised version, words candidate vary in the activation level, and so membership of the word initial cohort plays a important role in the recognition [3] [5] . According to the TRACE model of spoken word recognition, proposed by McClelland and Elman, in 1986 [6] [4], speech recognition can be described as a process, in which speech units are arranged into levels which interact with each other. There are three levels: features, phonemes, and words. There are individual processing units or nodes at three different levels. Dennis Norris in 1994, have proposed another model of spoken word recognition called the Shortlist model [4]. According to Norris, a shortlist of word candidates is derived at the first stage of the model. The list consists of lexical items that match the bottom-up speech input. This abbreviated list of lexical items enters into a network of word units in the later stage, where lexical units compete

with one another via lateral inhibitory links [4]. From all these theories of spoken word recognition we can arrive at the conclusion that the spoken word recognition is a complex multileveled pattern recognition work performed by neural networks of human brain and the related speech perception process can be modeled as a pattern recognition network. Different levels of the language like lexical, semantic, phonemes can be used as the unit of distribution in the model. All the theories proposed that bottom up and top down processes between feature, phoneme, and word level combines to recognize a presented word. In such a situation, Artificial Neural Network (ANN) models has greatest potential, where hypothesis can be performed in a parallel and high computational approach. ANN models are composed of many non-linear computational elements operating in parallel and arranged in the pattern of biological neural network. The brain's impressive superiority while deal with a wide range of cognitive skills like speech recognition, has motivated researchers to explore the possibilities of ANN models in the field of speech recognition in 1980s [7] with a hope that human neural network like models may ultimately lead to human like performance on such a complex tasks. A few ANN based work on speech recognition are described in . But at the later half of 1990, suddenly ANN based speech research remained dormat, at times terminated [7]. Statistical frameworks like Hidden Markov models (HMMs), Gaussian Mixture Models (GMMs) etc. received attention supporting that supports both acoustic and temporal modeling of speech. However, it should be mentioned that the currently available best systems are far from equaling human like performance and many important research issues are still to be explored. Also, ANN is able to work with model free data unlike HMM and can be continuously learn from the surrounding. Moreover, ANNs can retain this learning and use it for subsequent processing. This way ANNs can be helpful for speech processing applications. Therefore, the value of ANN based research is still large and now a days it is considered as the hot field in the domain of speech recognition [8], [9], [10], [11], [12], [13], [14]. This work presents a novel approach to spoken word recognition, where a phoneme count determination technique is used to take some initial decision about the number of phonemes in the word to be recognized and accordingly the word is directed to the 3-phoneme, 4-phoneme or 5-phoneme word recognition model. Here, we have used the 3-phoneme word recognition algorithm explained in [15], which is designed using a Self Organizing Map (SOM) based phoneme segmentation algorithm and Probabilistic Neural Network (PNN) and Learning Vector Quantization (LVQ) based decision algorithm . The phoneme count determination technique is designed using two different approach, initially using K-means clustering (KMC) and then using SOM based clustering along with a Recurrent Neural Network (RNN) block for decision making. The KMC based technique can take around 83% correct decision about the number of phoneme. If the initial decision about the number of phoneme goes wrong, then the success rate of 3-phoneme word recognition suffers. Therefore, the SOM clustering based phoneme count determination technique is adopted and it provides superior performance in terms of percentage of correct decision. In the CVC word recognition method, SOM is trained with different iteration numbers for the same word and thus different weight vectors are obtained, which is considered as different phoneme boundaries. The phoneme segment obtained by training SOM with various iteration numbers are then matches with the PNN patterns. While taking decision about the last phoneme the algorithm is assisted by LVQ codebook which contains a distinct code for every word of the recognition vocabulary. Assamese CVC words are recorded for this work in a noise free environment from five male and five female speakers of varying age. The description included here is organized as below. Section 2 provides a brief description of the phonemical structure of Assamese language. Section 3 explains the proposed word recognition model which consist of two steps - the phoneme count determination and 3-phoneme word recognition block. The results and the related discussions are included in Section 4. Section 5 concludes the description.

2 Phonemical Structure of Assamese Language

Assamese is an Indo-Aryan language originating from the Vedic dialacts, and therefore, is a sister of all the northern Indian languages. Although the exact nature of the origin and growth of the language is yet to be ascertained with certainity, it is supposed that like other Aryan languages, Assamese was also born from $Apabhra\acute{m}\acute{s}a$ dialects developed from $M\tilde{a}gadhi$ Prakrit of the eastern group of Sanskritic languages [16]. Retaining certain features of its parent Indo-European family it has got many unique phonological characteristics which makes Assamese speech unique and hence requires a study exclusively related to the design of a speech recognition / synthesis system in Assamese [17].

The Assamese phoneme tables obtained from [16] are shown in Figure 1 and Figure 2. There are twenty-three consonants and eight vowel phoneme in the stan-

Consonants

	Bilabial		Alveolar		Palatal	Velar		Glottal
	Vl.	Vd.	Vl.	Vd.	Vd.	Vl.	Vd.	Vd.
Unaspirated	p	b	t	d		k	G	
Aspirated	ph	bh	th	dh		kh	gh	
Spirant			s	z		x		fi
Nasals		m		n			ŋ	
Lateral				l				
Trill				r				
Frictionless Continuant		w			j			

Figure 1: Assamese Consonant Phonemes, Vl.-Voiceless, Vd.-Voiced [16]

Vowels

	Front	Central	Back
High	i		u
Higher-mid	e		o
Lower-mid	ɛ		ɔ
Low		a	ɒ

Figure 2: Assamese Vowel Phonemes [16]

dard colloquial Assamese . The consonants may be grouped into two broad divisions: the stops and the continuants. For the stops there are contrast in three points of articulation- the lips, the alveola, and the velum and four-way contrasts in every point as to the presence or otherwise of voice and aspiration. Therefore, a stop may be voiced or voiceless, aspirated or unaspirated. There are continuants- two frictionless, viz, the semivowels /w,j/, four spirants /s z x h/, one lateral /l/, one trill /r/ and three nasals /m, n/ which are stops as well as continuant both at once [16].
The eight vowels present three different types of contrasts [16]. Firstly, eight-way contrasts in closed syllables, and in open syllables when /i u/ do not follow in the next immediate syllable with intervention of a single consonants except the nasal. Again, it shows six-way contrast in open syllables with /i/ occurring in the immediately following syllable with intervention of any single consonant except the nasals, or except with nasalization and finally, five way contrasts in open syllables when /u/ occurs in the immediately following syllable with a single consonant intervening [16].

3 The Proposed Word Recognition Model

The proposed spoken word recognition model can be described by the block diagram of Figure 3. The model consist of two phases-

- Phoneme count determination and

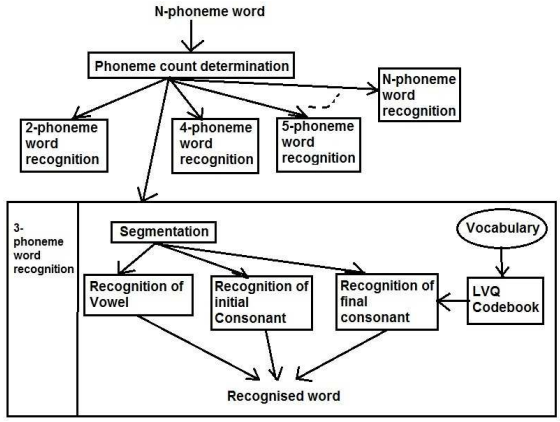

Figure 3: Process logic of the proposed work

- Three phoneme word recognition

The phoneme count determination block determines the number of phonemes in the word and on the basis of that decision, the word can be presented to some N-phoneme word recognition algorithm, where N can be 2, 3, 4, 5 etc. Here, we have described a novel algorithm for recognition of 3-phoneme CVC type words, which uses three different ANN structure, namely SOM, PNN and LVQ. The following section describes both these phases separately.

3.1 Phoneme Count Determination

To develop a discrete speech recognition model for Assamese language, the system should have the capability of dealing with words having different CV combinations. Therefore, in order to enable the model to deal with multiple phoneme case, a block is created, so that the algorithm can take some prior de-

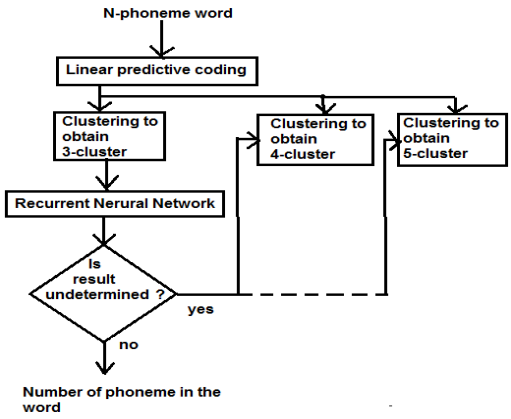

Figure 4: Phoneme Count Determination Block

cision about the number of phonemes in any incoming word. Since, discrete speech signals are consisted of word units separated by some intentional pause or silent part, therefore, it is not very difficult to omit the silent part from the signal in order to separate the words in the speech signal. A simple intensity threshold method can be used for such kind of word- silent separation. The silent part ideally has zero intensity. Thus, a threshold can be put to remove that part of the speech signal which has zero intensity. But the crucial part of discrete speech recognition is to recognize the individual word consisting of multiple phonemes. As a solution to such a problem, here we have proposed an ANN based recognition model where the recognition algorithm initially decides the number of phonemes in the word and then according to that decision the word is presented to recognize the constituent phoneme in some N-phoneme recognition units designed separately. Therefore, a phoneme count determination block is designed which is shown in Figure 4. Two different approaches are carried out to design the phoneme count determination block as explained below.

3.1.1 KMC based Phoneme Count Determination Block

Clustering is the process of partitioning or grouping a given set of patterns into disjoint clusters. This is done such that patterns in the same cluster are alike and patterns belonging to two different clusters are different. Clustering has been a widely studied problem in a variety of application domains including ANNs. The number of clusters k is assumed to be fixed in k-means clustering. In this section, we propose probable technique for generating some apriori knowledge about

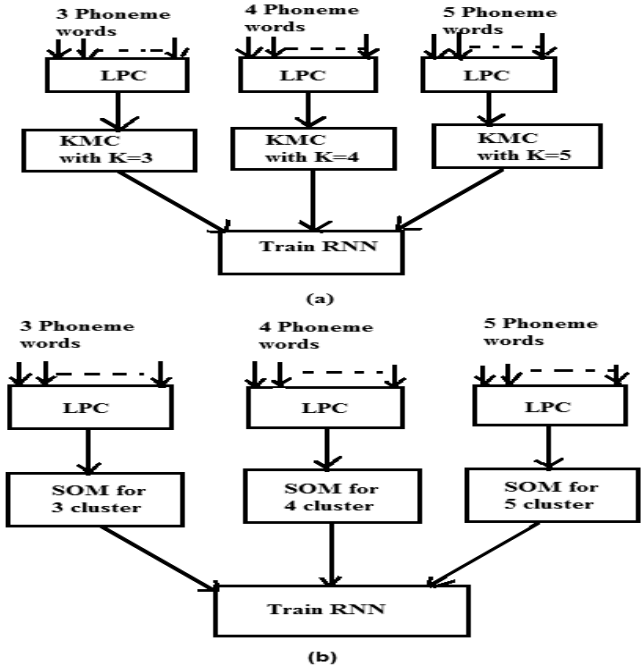

Figure 5: Word Clustering Technique

number of phonemes in an word, without segmentation. The method involves clustering N-phoneme words into N-cluster using KMC. Here, N is the value of k in the KMC. The word clustering technique proposed here, can be visualized from part (a) of Figure 5.

3.1.2 SOM based Phoneme Count Determination Block

The same phoneme count determination block is later redesigned using the SOM. Here, the KMC blocks are replaced by SOM competitive layers. The SOM training algorithm resembles k-means algorithm in the sense that it partitions the input data space into a number of clusters of wining neuron and its neighbors with similar weight vectors. Therefore SOM can be used for clustering data without knowing the class memberships of the input. The SOM algorithm is based on unsupervised, competitive learning, where clusters are formed depending upon a self-organization process of the constituent neurons which groups data depending upon a similarity measure. The similarity measure is decided upon by a euclidian distance between the random connectionist values and the input and finally optimized by a Gaussian spread.The SOM based phoneme count determination technique can be summarized from the part (b) of Figure 5.

3.1.3 Role of RNN in Decision Making of the Proposed Technique

RNN is a special structure of supervised ANN known for data recurrency and is suited for capturing the temporal nature of speech signals [7] . Therefore it has been chosen for taking decision about the number of cluster in a clustered speech data. An RNN is trained with the clustered data to learn the number of cluster so that it can classify any unknown clustered word according to the number of cluster. The sample clustered data set is obtained from KMC or SOM, where 3-phoneme words has 3 clusters, 4-phoneme words has 4 clusters and 5-phoneme words has 5 clusters. Accordingly, the RNN classifies the data into 3, 4 and 5-cluster groups and suppose they are stored as DK3, DK4 and DK5 respectively. Here, we have considered only certain variations like 3, 4 and 5- phonemes. Now, any N-phoneme word obtained from the discrete speech signal, is first clustered with K=3, 4, 5. Next, at first DK3 is presented to the trained RNN to classify into any of the three classes. If RNN fails to classify, then DK4 and DK5 is presented consequently. If any of DK3, DK4 and DK5 do not come under any of the classes defined by RNN, then that word is discarded. The decision making process of the RNN is depicted in Figure 4. The same decision making method is used in case of clustered data set obtained from both KMC and SOM based clustering technique. Comparing the success rate of correct decision for both the techniques it is obtained that the SOM based technique can provide more correct decisions (around 7 $\%$ improvement) and therefore it is used with the 3-phoneme word recognition technique in the proposed model. The result of the both the techniques are shown in Section 4.3

3.2 Recognition of Three Phoneme Words

SOM has a special property of effectively creating spatially organized "internal representations" of various features of input signals and their abstraction [18]. SOMs can be considered as a data visualization technique i.e. it provides some underlying structure of the data [18]. This idea is used in our phoneme segmentation technique, process logic of which is given in Figure 2. The weight vector obtained by training an one dimensional SOM with the LPC features of a word containing the phoneme to be segmented, is used to represent the phoneme. Training the same SOM with various iteration numbers, we get different weight vectors, each of which is considered as a segment of different phonemes constituting the word. The weight vectors thus obtained are classified into vowel, initial consonant and final consonant. The work only covers the unaspirated phoneme family, which have phonemes like /p/, /b/, /t/, /d/, /k/ and /g/ and all the eight vowel of Assamese language [19]. The classification is done by PNNs trained with these clean unaspirated phonemes and vowel phonemes. The PNN is based on statistical principles derived from Bayes decision strategy and non-parametric kernal based estimators of probability density function (pdf)s. It finds pdf of features of each class from the provided training samples using Gaussian Kernel. These estimated densities are then used in a Bayes decision rule to perform the classification. Advantage of the PNN is that it is guaranteed to approach the Bayes optimal decision [20]. The proposed three phoneme word recognition algorithm can be stated in two distinct parts-

- SOM based Segmentation Algorithm and
- PNN and LVQ based Recognition algorithm

The word recognition algorithm is well described in [15]. Although the following sections provides a brief description of the two parts of the algorithm.

3.2.1 SOM based Segmentation Algorithm

The SOM weight vector extraction algorithm can be visualize from the block diagram of Figure 6. The SOM weight vectors thus extracted are stored as SW1, SW2, SW3, SW4, SW5 and SW6. SOMs role is to provide segmentation boundaries for the phonemes. Here six different segmentation boundaries are obtained for six separate sets of weight vectors. The segmented phonemes are next applied to the PNN and LVQ based decision algorithm which performs the role of decision making for constituent vowel and consonant phoneme.

3.2.2 PNN and LVQ based Recognition Algorithm

Here we have performed some two class PNN problem, where three PNNs are trained with two clean unaspirated consonant phonemes and are named as PNN1, PNN2 and PNN3. Similarly, four other PNNs are trained with clean vowel phonemes and are named as PNN4, PNN5, PNN6 and PNN7. That means the output classes of PNN1 are /p/ and /b/, the output classes of PNN2 are /t/ and /d/, the output classes of PNN3 are /k/ and /g/, the output classes of PNN4 are /i/ and /u/ etc. and this way we obtain total seven PNNs which are considered to know the patterns of all the unaspirated consonant phonemes and vowel phonemes of Assamese.

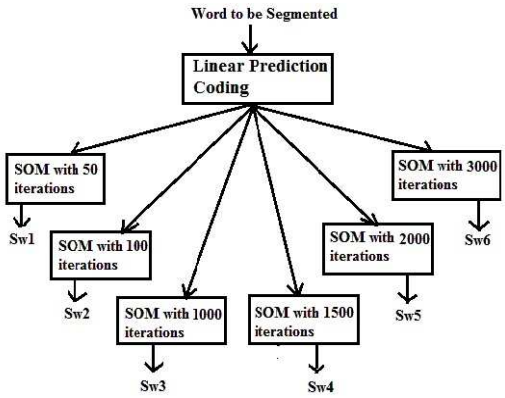

Figure 6: SOM Segmentation Block [**?**] [15]

The reason behind the use of two class PNNs is to increase the success rate by performing a binary level decision. Although it increases the computational complexity and memory requirements, it is tolerable for the sake of increasing success rate. Since PNN is the fastest, use of more than one PNN does not harm much the speed of the algorithm. PNNs handle data that has spikes and points outside the norm better than other ANNs. Therefore, PNN is suitable for problems like phoneme classification [20]

As mentioned earlier, the segmented vowel and consonant phonemes are identified with the help of seven PNNs trained with clean phonemes. An LVQ Codebook is used along with the PNN based recognition algorithm while deciding the last phoneme of the incoming word. The vocabulary used for this work contains words having the last phoneme any of /t/, /k/, /r/ and /n/. One four class PNN is trained with clean phonemes to learn all these four phoneme patterns separately. This PNN is used to recognize the last phoneme from the SOM segmented phonemes of the incoming word. While taking decision about the last phoneme the algorithm is assisted by the LVQ codebook. Learning vector quantization (LVQ) is a method for training competitive layers in a supervised manner. In an LVQ network competitive layer learn to classify input vectors into target classes chosen by the user unlike strictly competitive layer possessed by SOM [21]. The codebook designed by LVQ has a distinct code for every word of the vocabulary. Suppose, the PNN decides about a phoneme, but if no words of the vocabulary end with that phoneme then the decision is discarded. Thus, the codebook assistance assures that most likely decision about the last phoneme can be obtained.

Table 1: Table representing phoneme values for C_i, V_j and C_k

i/j/k	C_i	V_j	C_k
1	$/p/$	$/i/$	$/p/$
2	$/b/$	$/u/$	$/b/$
3	$/t/$	$/e/$	$/t/$
4	$/d/$	$/o/$	$/d/$
5	$/k/$	$/\varepsilon/$	$/k/$
6	$/g/$	$/ao/$	$/g/$
7	$/ph/$	$/a/$	$/ph/$
8	$/bh/$	$/oo/$	$/bh/$
9	$/th/$		$/th/$
10	$/dh/$		$/dh/$
11	$/kh/$		$/kh/$
12	$/gh/$		$/gh/$
13	$/s/$		$/s/$
14	$/z/$		$/z/$
15	$/x/$		$/x/$
16	$/h/$		$/h/$
17	$/m/$		$/m/$
18	$/n/$		$/n/$
19	$/l/$		$/\eta/$
20	$/r/$		$/l/$
21			$/r/$
22			$/w/$
23			$/j/$

4 Experimental Result and Discussion

The work is carried out as per the flow diagram of Figure 3. Mathematically, the problem in hand can be stated as-

Suppose, C_i is the initial consonant phoneme which can be vary within all the phoneme families, V_j be the vowel phoneme and C_k be the last consonant phoneme.
Then any incoming word may have the form
X=$C_iV_jC_k$
where, i = 1 to 20 (excluding /η, w, j/)
j = 1 to 8 and
k = 1 to 23.
Identify, C_i, V_j and C_k.
The values of C_i, V_j and C_k are given in Table 1.

4.1 Experimental Speech Signals

The experimental speech samples are recorded in three phase. At first clean consonant and vowel

Table 2: CVC Type Word List of for Spoken Word Recognition Model

Sl No	Vowel (V)	Initial Consonant(C)	Last Consonant(C)	CVC Word
1	/i/	/x/	/t/	/xit/
2			/r/	/xir/
3			/k/	/xik/
4			/s/	/xis/
5	/u/	/d/	/kh/	/dukh/
6			/r/	/dur/
7			/t/	/dut/
8			/b/	/dub/
9	/e/	/b/	/x/	/bex/
10			/d/	/bed/
11			/s/	/bes/
12			/l/	/bel/
13	/o/	/m/	/n/	/mon/
14			/k/	/mok/
15			/r/	/mor/
16			/t/	/mot/

Table 3: CVC Type Word List of for Spoken Word Recognition Model

Sl No	Vowel (V)	Initial Consonant (C)	Last Consonant (C)	CVC Word
17	/ɛ/	/kh/	/l/	/khɛl/
18			/p/	/khɛp/
19			/d/	/khɛd/
20			/r/	/khɛr/
21	/ao/	/r/	/n/	/raon/
22			/kh/	/raokh/
23			/th/	/raoth/
24			/x/	/raox/
25	/a/	/l/	/bh/	/labh/
26			/z/	/laz/
27			/th/	/lath/
28			/kh/	/lakh/
29	/oo/	/n/	/l/	/nool/
30			/d/	/nood/
31			/kh/	/nookh/
32			/m/	/noom/

phonemes are recorded from five girl speakers and five boy speakers, which results in a total of following broad sets of speech signals: *Girl1*, *Girl2*, *Girl3*, *Girl4*, *Girl5*, *Boy1*, *Boy2*, *Boy3*, *Boy4* and *Boy5*. These are used to train PNNs. The description included in Section 2 simply revels the fact that Assamese consonant phonemes can be classified into six distinct phoneme families as *unaspirated*, *aspirated*, *spirant*, *nasal*, *lateral* and *trill*, excluding the semivowels /w/ and /j/. The sample words selected for this work covers variation in the initial phoneme from different families as well as variation of all the vowel phonemes. For the second phase of collecting samples, a few samples collected are shown in Table 2 and Table 3, which are recorded from the same girl and boy speakers used in the first phase. The third phase covers some more samples recorded with mood variations, for testing the ANNs. Further the third phase samples are extended by adding $\pm 3db$ white gaussian noise to it.For recording the speech signal, a PC headset and a sound recording software, Gold Wave, is used. The recorded speech sample has a duration of 2 seconds, sampling rate of 8000 samples/second and bit resolution of 16 bits/sample.

4.2 Preprocessing

The pre-processing of the speech signal consist of two operations namely-smoothing of the signal by median filter and removal of the silent part by threshold method. Although the speech signals are recorded in a noise free environment, presence of some low frequency distortion is observed. Therefore, a median filtering operation is performed on the raw speech signals, so that the phoneme segmentation does not suffer due to any type of unwanted frequency component [22] [23].

The smoothed signal S_{smooth} contains both speech and non speech part. The non-speech or silent part occurs in a speech signal due to the time spend by the speaker before and after uttering the speech and this time information is considered to be redundant for phoneme segmentation purpose. The silent part ideally have zero intensity. But in practical cases it is observed that even after smoothing, the silent part of the speech signal have intensity about 0.02 in the scale 1. Our silent removing algorithm considers this intensity value as a threshold. Thus, a pure signal containing only the necessary speech part is obtained. These preprocessing operations are applied to every speech signal used in the work.

4.3 Phoneme Count Determination Result

Any word coming to the recognizer is first clustered with KMC algorithm for k-value 3. Then LPC features of the clustered data is extracted and presented to the RNN for classification. The RNN is trained to learn number of clusters in the data. If RNN fails to classify the data into any of the defined class, then the

Table 4: Word List Prepared for Clustering Technique

Sl No	3-phoneme Words	4-phoneme Words	5-phoneme Words
1	/xit/	/xiki/	/xital/
2	/xir/	/xakhi/	/xapon/
3	/xik/	/xati/	/xijal/
4	/xis/	/xari/	/xiphal/
5	/dukh/	/dili/	/datal/
6	/dur/	/dukhi/	/dupat/
7	/dut/	/duni/	/duphal/
8	/dub/	/dora/	/dojal/
9	/bex/	/besi/	/besan/
10	/bed/	/beli/	/betal/
11	/bes/	/burhi/	/betan/
12	/bel/	/bora/	/bixal/
13	/mon/	/mona/	/mohar/
14	/mok/	/mora/	/methon/
15	/mor/	/mula/	/mukut/
16	/mot/	/muthi/	/muazar/

Table 5: Word list Prepared for Clustering Technique

Sl No	3-phoneme words	4-phoneme words	5-phoneme words
17	/khɛl/	/kheda/	/khabar/
18	/khɛp/	/bhada/	/khangal/
19	/khɛd/	/dhara/	/khorak/
20	/khɛr/	/thaka/	/khorang/
21	/raon/	/roza/	/ratan/
22	/raoth/	/ronga/	/rabar/
23	/raokh/	/rati/	/ragar/
24	/raox/	/roa/	/razan/
25	/labh/	/lopha/	/lagan/
26	/laz/	/loa/	/labar/
27	/lath/	/lora/	/lasit/
28	/lakh/	/lani/	/lagar/
29	/nool/	/noga/	/nazar/
30	/nood/	/nila/	/natun/
31	/nookh/	/nura/	/nadan/
32	/noom/	/nija/	/nagar/

word is clustered with KMC algorithm for k-value 4 and the process is repeated. If the RNN again fails to classify the data then the word is clustered with k-value 5 and the same process is repeated again. In this way, the proposed logic is used to determine the possible number of phonemes in the word for 3-, 4- and 5-phoneme words. If the RNN fails to take any decision, then that particular word is discarded.

Selection of proper set of sample words as a representative of all the N-phoneme words, where (N=2, 3 or 4) is an important factor for better success rate of the proposed technique. The RNN should be trained with enough data, so that it can learn the difference between 3, 4 and 5-phoneme words. The work described in this paper experimented the proposed technique only for 3-phoneme, 4-phoneme and 5-phoneme words. Accordingly a word list is prepared. The word list is shown in Table 4 and Table 5. Here, we have considered words having all the Assamese vowel variations. Words with all the possible vowel and consonant combination of Assamese vocabulary if are used for the purpose shall provide better results. The words are recorded from selected speakers with 70 % of the samples used for training and 30 % for testing the KMC technique.

The probability of correct decision depends on several factors. Varying LPC predictor length significantly affects the success rates and training time. Table 6 shows overall success rate for 3-, 4- and 5-phoneme words with varying predictor length and corresponding RNN training time. As can be seen from Table 6, with 50 predictor size, we obtain minimum

Table 6: Performance of RNN v/s Predictor Size for Clustering Technique

Sl No	RNN training Time in sec	Predictor Size	Success Rate
1	2545.012	50	63.2%
2	1546.037	30	75.4%
2	1546.037	10	81%

success rate and with 10 we obtain maximum success rate. Similarly, Table 7 shows success rate for 3-, 4- and 5-phoneme words with fixed predictor size 30.

From the experimental results, it is seen that the KMC based method can give around 83 % success rate while determining number of phonemes in a word using KMC aided apriori knowledge. Obviously, in order to use the logic in the proposed spoken word recognition method, the success rate must be improved; otherwise the whole recognition system shall suffer. Therefore, the SOM based phoneme count determination block is designed as explained in Section 3.1.2. The success rate phoneme count determination using SOM is shown in Table 8 and Table 9 for noise free and noisy signal. The noisy signal set is created by adding 3 dB white Gaussian noise with the noise free signals. It can be seen that the noise free signal set shows around 7 % improvement in comparison to the KMC based method. Further increasing the RNN training signals the success rate of the noisy signal set can also be improved.

Table 7: Success Rate of Number of Phoneme Determination with KMC based method

Sl No	Value of N in N-phoneme word	Success Rate without noise	Success Rate with noise
1	3	83%	76.2%
2	4	67.3%	62.1%
3	5	78.3%	71%

Table 8: Phoneme count determination Success Rate for noise free signals using SOM based method

Sl No	Word	Correct Decision	False Decision
1	3-phoneme	92%	6%
2	4-phoneme	89%	8%
3	5-phoneme	90%	8%

4.4 Vowel Segmentation and Classification Results

The segmented phonemes are then checked one by one to find which particular segment represents the vowel phoneme by matching pattern with the trained PNNs. It is observed that the recognition success rate for various vowel is around 95% with the SOM based segmentation technique. Table 10 shows success rate for various vowels.

4.5 Consonant Phoneme Segmentation and Classification Results

The SOM is trained for six different iterations which provides identical number of decision boundaries. For 50 sessions a segment named SW1 is obtained (Figure 1). Similarly, for 100, 1000, 1500, 2000 and 3000 session other five segments are obtained. This is somewhat similar to the segmentation carried out using DWT. The DWT provides various levels of decomposition with faster processing time than the SOM. In case of SOM, some vital time slots are lost in training. But the segments provided by SOM provide better resolution which helps the PNN to provide improved discrimination capability subsequently.Similarly Table 11 shows success rate for various consonant phoneme as initial phoneme and last phoneme. Table 12 summarizes this performance.
Overall 3-phoneme word recognition success rate using the phoneme count determination block and without using phoneme count determination block can be summarized by the Table 13.
The experimental results shows that the success rate of phoneme count determination block has to be improved further, so that the word recognition model

Table 9: Phoneme count determination Success Rate for noisy signals using SOM based method

Sl No	Word	Correct Decision	False Decision
1	3-phoneme	84%	10%
2	4-phoneme	86.2%	8.5%
3	5-phoneme	87%	6.2%

Table 10: Success Rate of Vowel Phonemes

Sl No	Word	Success rate of SOM
1	/i/	98%
2	/u/	95%
3	/e/	94%
4	/o/	92.5%
5	/ɛ/	97%
6	/ao/	96.7%
7	/a/	92%
8	/oo/	94.5%

can show better success rate. But the novelty of the work is that it successfully integrates a phoneme count block as part of a discrete speech recognition system using a hybrid neural framework with a minimum success rate of 90%.

5 Conclusion and Future Direction

In this work, we have described a word recognition model with a phoneme count determination block, which can take some initial decision about the number of phonemes in any incoming word, so that the algorithm control can move from 2-phoneme to 3-phoneme word recognition algorithm, 3-phoneme to 4-phoneme word recognition algorithm and so on. The novelty of the work is that it acts as a self sustaining, fully automated mechanism for spoken word recognition with 3-, 4- and 5-phoneme variation with no manual intervention. By developing similar kind of word recognition method for 2-phoneme, 4 phoneme or any N- phoneme word a complete discrete speech recognition model for Assamese language can be developed.

References:

[1] R. L. Diehl, Lori L. Holt, “Speech Perception”, Annu. Rev. Psychol. 2004.

[2] R. Mannell, “Speech Perception Background and Some Classic Theories”, Department of

Table 11: Success rate for Consonant Phoneme Recognition

Sl .No	Phoneme	Success Rate of Initial Phoneme	Success Rate of Last Phoneme
1	/p/	×	85%
2	/b/	90%	100%
3	/t/	×	90%
4	/d/	97%	90%
5	/k/	×	85%
6	/g/	×	×
7	/ph/	×	×
8	/bh/	×	89%
9	/th/	×	100%
10	/dh/	×	×
11	/kh/	91%	100%
12	/gh/	×	
13	/s/	×	90%
14	/z/	×	85%
15	/x/	91%	95%
16	/h/	×	×
17	/m/	93%	87%
18	/n/	91%	100%
19	/l/	93%	95%
20	/r/	93%	85%

Table 12: Overall Success rate of Phoneme Recognition

Sl No	Phoneme	Success Rate
1	$Vowel$	95.25%
2	$InitialConsonant$	94%
3	$LastConsonant$	93.2%

Table 13: Overall Success Rate for 3 Phoneme Words

Sl No.	System	Success Rate
1	With Phoneme Count Determination Block	90%
2	Without Phoneme Count Determination Block	98%

Linguistics, Feculty of Human Sciences, Macquarie University, Downloaded on September, 2010.

[3] M. W. Eysenck; Psychology-an international persepective; books.google.co.in; 2004;

[4] P. W. Jusczyk, P. A. Luce, Speech Perception and Spoken Word Recognition: Past and Present, Departments of Psychology and Cognitive Science, Johns Hopkins University, Baltimore, Maryland; and Department of Psychology and Center for Cognitive Science, University at Buffalo, Buffalo, New York; 2002.

[5] B. Bergen; Linguistics 431/631: Connectionist language modeling; Meeting 10: Speech perception; 2006;

[6] J. L. McClelland, D. Mirman and L. L. Holt; Are there interactive processes in speech perception?;TRENDS in Cognitive Sciences Vol.10 No.8;www.sciencedirect.com; 2006;

[7] Y. H. Hu and J. N. Hwang, Handbook of Neural Network Signal Processing, The Electrical Engineering and Applied Signal Processing Series; CRC Press, USA; 2002.

[8] M. Sarma, K.K. Sarma, "Vowel Phoneme Segmentation for Speaker Identification Using an ANN-Based Framework", Journal of Intelligent Systems, vol.22 , Issue.2; pp.111-130; 2013;

[9] M.Sarma, K.K. Sarma, "An ANN based approach to Recognize Initial Phonemes of Spoken Words of Assamese Language", Elsevier's Applied Soft Computing; Vol. 13, Issue 5, pp. 22812291, 2013.

[10] M.Sarma, K.K. Sarma, "Segmentation of Assamese phonemes using hybrid soft-computational approach", International Journal of Parallel, Emergent and Distributed Systems; Vol. 28, Issue 4, pp.370-382 ; 2013.

[11] W. AL-Sawalmeh, K. Daqrouq and O. Daoud, "The Use of Wavelet Entropy in Conjuction with Neural Network for Arabic Vowels Recognition", WSEAS Transactions on Signal Processing, Issue 3, volume 7, 2011.

[12] J. Karam, "BiorthoganalWavelet Packets and Mel Scale Analysis for Automatic Recognition of Arabic Speech via Radial Basis Functions", WSEAS Transactions on Signal Processing, Issue 1, volume 7, 2011.

[13] R. Thangarajan, A. M. Natarajan and M. Selvam, "Word and Triphone Based Approaches in Continuous Speech Recognition for Tamil Language", WSEAS Transactions on Signal Processing, Issue 3, volume 4, 2008.

[14] F. Neri, Cooperative evolutive concept learning: an empirical study, WSEAS Transaction on Information Science and Applications; vol. 2; issue 5; pp. 559-563; 2005.

[15] M. Sarma and K. K. Sarma, "Recognition of Assamese Phonemes using Three Different ANN Structures", Proceedings of CUBE International IT Conference, Pune, India, 2012.

[16] G. C. Goswami: Structure of Assamese, First Edition, Department of publication, Gauhati University, Guwahati, Assam, India, 1982.

[17] Courtesy: Prof. Gautam Baruah, tdil.mit.gov.in / assamesecodechartoct02.pdf, Dept. of CSE, IIT Guwahati, India.

[18] T. Kohonen,"The Self-Organizing Map", Proceedings of the IEEE, vol 78, no. 9, September, 1990.

[19] G. C. Goswami: "Structure of Assamese", First Edition, Department of publication, Gauhati University, 1982.

[20] D. F. Specht, "Probabilistic Neural Networks; Neural Networks, vol. 3.,no. 7, pp. 109-118, 1990

[21] T. Kohonen, J. Hynninen, J. Kangas, J. Laaksonen and K. Torkkola: LVQ PAK,The Learning Vector Quantization Program Package, Programming Team of the Helsinki University of Technology, Laboratory of Computer and Information Science, Version 3.1, Finland, 1995.

[22] B. T. Tang, R. Lang, H. Schroder, A. Spray and P. Dermody, "Applying Wavelet analysis to speech Segmentation and Classification. H. H. Szu, editor, Wavelet Applications, volume Proc. SPIE pp. 2242:750-761, 1994.

[23] L. R. Rabiner and R. W. Schafer, Digital Processing of Speech signals, Third Impression, Pearson Education, New Delhi, 2009.

Fuzzy Authentication Algorithm with Applications to Error Localization and Correction of Images

OBAID UR-REHMAN and NATASA ZIVIC
Chair for Data Communications Systems
University of Siegen
Hoelderlinstrasse 3, 57076 Siegen,
GERMANY
{obaid.ur-rehman, natasa.zivic}@uni-siegen.de

Abstract: - Images are normally protected using standard messages authentication codes to protect them against tampering and forgeries. One problem with this approach is that when such images are transmitted over a noisy medium, even a single bit error might render the image as un-authentic to the receiver. In this paper, a noise tolerant data authentication algorithm is proposed. The proposed algorithm can perform authentication in the presence of minor errors but at the same time identify forgeries in the data. This algorithm is then extended by demonstrating its applications in image authentication. The extended algorithm is called as fuzzy authentication algorithm. It has the ability to localize errors in an image as well as correct the localized errors using error correcting codes. The proposed algorithm rejects only the potentially erroneous / unauthentic parts of the image and correct or authenticate the remaining parts if the number of errors is below a certain threshold. This reduces the need for retransmission of the complete image and only a few parts might be retransmitted if the application demands. This property is especially useful in real-time communications. It is better to obtain a part of the authentic image, rather than having no image at all. A security analysis of the proposed algorithm is given, and simulation results are presented to demonstrate its error localization and correction capabilities.

Keywords: - Fuzzy Authentication; Image Authentication; Reliability Values; Soft Authentication; Content Based Authentication; Noise Tolerant Authentication

1 Introduction

Modern communication systems are typically composed of compression, channel coding and security modules. While compression reduces the size of transmitted data, channel coding aims to deliver the data reliably to the receiver by using additional parity data. The security module provides authenticity of the origin, protection against eavesdropping and forgery attempts. The goal of compression is to eliminate redundancy, whereas the channel coding and security modules add redundancy to the compressed data in order to achieve their respective goals. Thus the solutions provided by these modules can often be conflicting. An improved coordination and information sharing between these building blocks can achieve the desired individual goals of the modules in a better manner.

In order to provide the above mentioned security features, such as message authenticity, proof of origin, integrity and protection against forgeries, cryptographic methods are used. Standard Message Authentication Code (MAC) is used to ensure the integrity and authenticity of a message. MAC is computed on a message using a shared secret key between the sender and the receiver and is appended to the message. This also helps in proving the authenticity of origin of a message. In standard applications of MAC, hard authentication is used, which ensures that even a single bit modification to the message is detected. Some applications including multimedia transmission like image, voice and streaming audio, are generally noise tolerant [5] and have real-time requirements. In such applications, it might be meaningful to retain the received object (e.g., an image), without the need for a retransmission, even if the hard authentication fails and a modest number of errors exist. For this category of applications, soft authentication algorithms have been proposed in the literature, such as Approximate Message Authentication Code (AMAC), Image Message Authentication Code (IMAC) and Noise Tolerant Message Authentication Code (NTMAC) [5-7]. Some soft, as well as hard, authentication algorithms tailored for multimedia transmission have also been proposed [8-12]. Some parallel work on image retrieval based on image content instead of the image pixel data includes [13-14]. Such algorithms are called content based image retrieval algorithms. Image encryption based on image content and region of interest selection has been shown recently in [15]. However, most of these algorithms work with little or

in no co-ordination with the other modules of a communication system. Thus the received image could be rejected due to a single (additional) error beyond the allowed limit. In this paper an algorithm for data authentication in the presence of noise is proposed. This algorithm is named as Soft Input Decryption (SID) and its proposed variant as Threshold based Soft Input Decryption (TSID).

In a standard image transmission system, the image is divided into non-overlapping (square) blocks, and for each block a discrete cosine transform (DCT) is calculated, which is followed by a quantization and entropy decoder. Among all of the DCT components, the first coefficient plays the most important role and is called the DC coefficient, while the rest are called AC coefficients. It is well known that despite of discarding a certain number of AC coefficients, the image can be reconstructed at the receiver, using the Inverse DCT, without much loss in the quality. The redundancies which are assumed to be discarded are inter-pixel or psychological redundancies and can be discarded without any significant detectable visual effects. A soft authentication algorithm based on the coordination between the channel coding and message authentication modules is proposed in this paper. The proposed algorithm works on compressed images using the Discrete Cosine Transform (DCT) and uses the TSID algorithm as well as the NTMAC [7] algorithm for the soft authentication and error localization as well as error correction. A certain number of errors below a predefined threshold are corrected, while another permissible number of errors are tolerated by the proposed authentication algorithm. DCT is used for the basic feature extraction and image compression. The proposed algorithm is based on the NTMAC, with additional error localization, correction and soft authentication properties.

This paper is organized as follows; In Section 2, the Soft Input Decryption Algorithm is described. In Section 2, the Soft Input Decryption Algorithm with Threshold is discussed. In Section 4, the DCT is briefly reviewed followed by the description of the proposed algorithm. The analysis of the proposed algorithm is given in Section 5. Simulation results are presented in Section 6, followed by the conclusion in Section 7.

2 Soft Input Decryption (SID) Algorithm

SID algorithm is the basis for Joint Channel Coding and Cryptography concept [1], [2]. This concept develops further the idea from [3], [4] on using the soft output (Log Likelihood Ratios or L-values) of the SISO (Soft Input Soft Output) channel decoding in order to try and correct the input of cryptographic mechanisms and therefore improving the results of decryption.

The algorithm of Soft Input Decryption (presented in Fig. 1) deals with blocks of data where each block contains a message (as payload) to which its Cryptographic Check Value (CCV) is concatenated as a security redundancy. At the transmitter, the CCV is generated by passing the message through a cryptographic check function and using a secret key K shared with the receiver. Message M and its CCV are thereafter encoded by the channel encoder and transmitted over a noisy channel. After performing channel decoding at the receiver, the (SISO) channel decoder outputs the estimated message M′, the estimated CCV′ and the *L*-values of all the bits of M′ and CCV′. These three elements are the input to Soft Input Decryption process which can be briefly described as follows: At first, CCV″ = *CCV*(M″ = M′) is calculated by applying the shared secret key K. If CCV″ = CCV′, the verification result is successful / true and M' is accepted as correct.

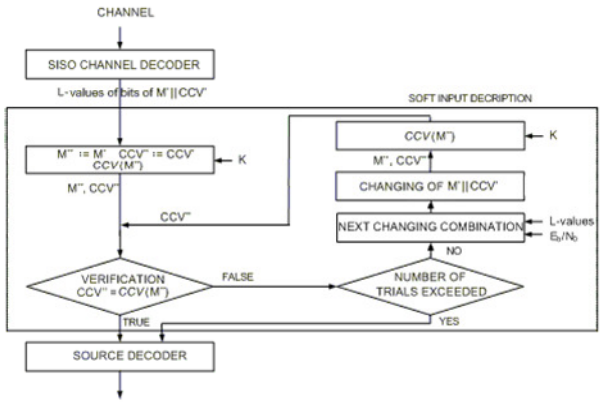

Fig. 1 Soft Input Decryption Algorithm

If the verification result is un-successful / false, the *L*-values of the channel decoder are analyzed. Some bits from M′ and CCV′ with the lowest |*L*|-values are flipped / inverted. There are different strategies for choosing the bits to be inverted, but the simpliest (and also efficient) way is to sort the |*L*|-values in increasing order and then to pick those bits from M′ and CCV′ which correspond to sorted |*L*|-values.

After the bit inversion, which results in M″ and CCV″, the verification process compares CCV″ and *CCV*(M″) for equality. If the verification result is

„false" again, another bit or combination of bits is inverted (corresponding to the binary interpretation of *N*-bit counter incremented by 1). This iterative process continues till either the verification result is „true" or a threshold number of iterations have been performed.

The idea of inversion of the least reliable bits originated from Chase decoding algorithms [3] in 1972, which were the generalization of the GMD (Generalized Minimum Distance) algorithms from 1966 [4] and improved channel decoding. These algorithms have been applied to a binary (n, k) linear block code and are referenced as LRP (Least Reliability Positions) algorithms. The novelty of SID is the inversion of bits iteratively for data protected by cryptographic redundancy with feedback between the decryptor and the channel decoder, since it gives the possibility to check whether the inversion iteration was successful or not (through the process of verification).

It may happen that a pair of M″ and CCV″ generated by the bit inversion passes the verification although the message M″ is not equal to the original message M. The probability of such an event is very small, which will also be taken in consideration in the next Section.

In simulations, both message and CCV have the length of 160 bits. CCV has been calculated using RIPEMD160 with a key *K*. Simulations have been performed using a Convolutional encoder of code rate r = 1/2 and constraint length m = 2 as the simplest one, but used very often in theory and practice. BPSK modulation and an Additive White Gaussian Noise (AWGN) channel are used together with a SISO decoder based on the Maximum A-Posteriori (MAP) algorithm. The MAP decoder was programmed in such a way, that it supports the output of L-values.

The presentation of the results of the simulations (Fig. 2) clearly shows obtained benefits and potentials of SID algorithm. In order to measure the improvement, a parameter named the Cryptographic Check Error Rate (CCER) is defined as follows:

$$CCER = \frac{number\ \ of\ \ incorrect\ \ CCVs}{number\ \ of\ \ received\ \ CCVs}, \quad (1)$$

where an incorrect cryptographic check value is each CCV which didn't pass the verification.

Fig. 2 shows the improvement of CCER with Soft Input Decryption for the cases that up to 8 of the lowest |*L*|-values (graph b) or up to the 16 lowest |*L*|-values are used (graph c). Example: CCER of 10^{-3} at E_b/N_0 ~ 6.2 dB without Soft Input Decryption is the same as for E_b/N_0 ~ 4.2 dB with Soft Input Decryption, or decreased from 10^{-1} to 10^{-3} at E_b/N_0 ~ 4.2 dB. Using up to 16 of the lowest |*L*|-values a coding gain of 2.33 dB can be reached, and CCER > 10^{-1} without Soft Input Decryption can be reduced down to CCER < 10^{-4} at E_b/N_0 ~ 4 dB.

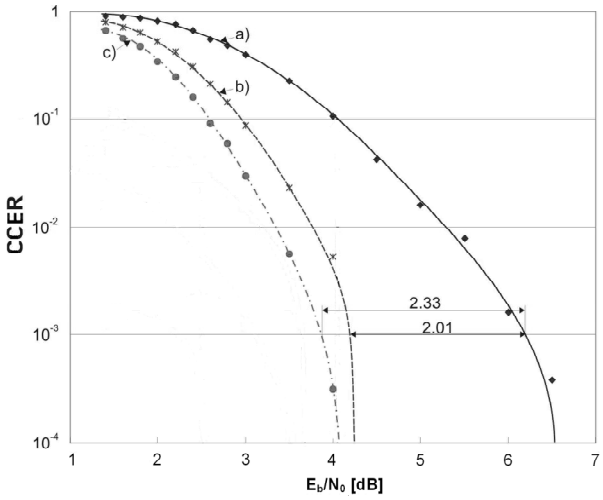

Fig. 2 Results of Soft Input Decryption using up to 8 (b) and 16 (c) lowest |L|-values compared to results without Soft Input Decryption (a)

At this point, a few words need to be said regarding the *L*-values and the type of channel decoder which can be used together with the SID. The channel decoder is assumed to be SISO (Soft Input Soft Output). SISO is a concept of channel decoding, which was originally used in iterative and Turbo coding, because the (soft) output is fed-back internally. Soft output of the channel decoder is used here as the soft input for the cryptographic verification process (called Soft Input Verification). Soft output of the channel decoder is usually expressed as L-value of each output bit u′,

$$L(u') = \ln \frac{P(u=1)}{P(u=0)}, \quad (2)$$

L(u′) represents the reliability of the decision made by the channel decoder, i.e., if the sent bit u was a 1 or 0. The sign of the *L*-value shows the hard output of bit u′ (1 or 0) and the magnitude, i.e., |*L*|, is used as reliability value of the hard decision. Example: if *L* is positive, the hard output is 1, otherwise it is 0. The higher the |*L*|, the more reliable is the hard decision and vice versa: a lower |*L*| means a less reliable decision. When the *L*-Value is equal to 0, the probability of the correctness of the decision is 0.5.

It is obvious that SID method greatly depends on the "quality" of *L*-values, which means that "better information" on bit reliability will give better results

after SID. The decoder which produces *L*-values need not to be exclusively of SISO type, but each *L*-value provided for each particular bit must, in some way, represent its reliability. Such a requirement disqualifies, for example, Viterbi decoding since then, the produced array of *L*-values gives the probabilities of the paths through trellis and not bit reliabilities. From the other side, there is MAP decoder (used in simulations) which internally calculates probabilities of each bit value taking into account the values of all other bits before and after a particular bit. Having all these probabilities already obtained by MAP, using (2) it is easy to get *L*-values which are suitable for the use within SID.

MAP is also a SISO decoder, which, besides the bit array from the output of line decoder (i.e. demodulator), as input can take the soft information on the probabilities of demodulated bits as well. It is important because then MAP has the opportunity to be used as a part of different more efficient (and more complex) decoding schemes with feedback. One of them is Turbo-decoding scheme where two MAP blocks are coupled over "crossed feedbacks", working together in an iterative process. In the above mentioned simulations, the "pure MAP" without feedback is used, which assumes that all the input probabilities of demodulated bits (soft input) have been preset on the value of 0.5.

Having in mind the way on which MAP internally calculates bit probabilities, one can conclude that for better results, the length of input bit array (in our case message M concatenated with its CCV) shouldn't be too small, since each bit probability (and the decision based on it) will be calculated using more neighboring bits. On the other side, the more distant bits have smaller influence on the calculations and also can add numerical noise, so it is expected that there is an optimal length of MAP input array (which is equal to the length of output array). The obtained results are also affected by the quality of *L*-values from the MAP decoder, which as explained, depends on the lengths of message and CCV. After all, the resume is that message and CCV together should have an optimal length (or close to optimal). In such a case where the messages are too long (or a continuous data stream is to be transmitted), they should be fragmented and for each fragment the CCV should be calculated, so that all together have more or less optimal length.

3 Soft Input Decryption Using Threshold

Although it seems that the SID method, especially in the scenario with a feedback, has exploited the whole soft information available from the channel decoder but further improvements of coding gain are still possible. Since the decryption of CCV is very sensitive to errors, the verification process need not to be so "strict" as it used to be. This gives more space for the SID method, which is now able to choose L-values more precisely by setting the verification threshold. Setting the threshold increases the probability of the false verification (collision), but even then it is extremely small.

The efficiency of SID method depends on the obvious factors such as the lengths of the message and the CCV, the number of bits chosen for inversion and the E_b/N_0 ratio. Some results of many simulations for different values of these parameters are presented in the previous Section as well as in [1]. Besides the mentioned parameters, the "quality" of *L*-values produced by channel decoder plays an important role, which indirectly affects the overall efficiency of the process involving SID. Naturally, SID works better with a smaller as compared to a bigger portion of data which is to be corrected.

One way to decrease the data length used by SID is to exclude the bits of cryptographic check value from correction within SID. Namely, if the length of the message is *m* and the length of cryptographic check value is *n*, SID now considers only *m* instead of *m+n* elements. SID picks *N* lowest |*L*|-values and inverts corresponding bits (only from decoded message M') in the iterative process of verification.

The above proposed enhancement of SID method is possible since the cryptographic check value satisfies the so called avalanche criterion which assumes that wrong decoded (i.e., reconstructed) CCV has in average 50% of the bits wrong. This means that if only one bit from the decoded message M′ is erroneous, around *n/2* bits in CCV″ will be erroneous as well. In other words, when a decoded message M′ is incorrect, CCV(M′) must have many wrong bits, much more (i.e. significantly more) than the decoded CCV'. So in the case that CCV″ contains "only a few" incorrect bits, i.e., when the Hamming distance (HD) between CCV′ and CCV″, i.e., HD(CCV', CCV") is "small enough", it is obvious that M′ is correct (M′ equals original M) and that the difference between CCV′ and CCV″ exists only because of the errors in CCV′. Hence, during the verification process within "Thresholded SID" (TSID), there is no need to check if all the bits from CCV″ are equal to the corresponding bits in CCV′, but the criterion for successful verification would be that *d*=HD(CCV',

CCV") is less than a threshold d_{max} which is to be determined.

In order to determine the appropriate value for decision threshold d_{max}, the statistical distribution of HD between CCV′ and CCV″ has to be found. For given BER after decoder (P_e) and the length of the message m, the probability distribution function (*pdf*) over different values of d can be expressed in the form of total probability sum:

$$pdf(d) = P_{M'correct} \cdot pdf_1(d) + P_{M'incorrect} \cdot pdf_2(d), \quad (3)$$

where $P_{M'correct}$ and $P_{M' incorrect}$ are the probabilities that decoded message M′ does not contain or contains errors respectively:

$$P_{M'correct} = (1 - P_e)^m, \quad (4)$$

$$P_{M'incorrect} = 1 - (1 - P_e)^m, \quad (5)$$

and $pdf_1(d)$ and $pdf_2(d)$ are conditional probability distribution functions of the HD after M′ is correct or incorrect.

In the case of successful verification, Hamming distance d = HD(CCV′, CCV″) is expected to have a small value, smaller than the decision threshold d_{max} (which is to be found). Also, CCV″ will be equal to the original CCV (because M′ is equal to original M), so d will be equal to the number of errors in CCV′ only, and d_{max} should be greater than the possibly largest number of errors in CCV′. Since after channel decoder the remaining errors (if exist) are uniformly distributed only over the CCV′ (with the length of n bits), the number of errors in CCV′ has a binomial distribution B(n,P_e), i.e.,

$$pdf_1(d) = \binom{n}{d} P_e^d \cdot (1 - P_e)^{n-d}, \quad 0 \le d \le n, \quad (6)$$

with mean value $n \cdot P_e$ and standard deviation $\sigma^2 = nP_e(1\text{-}P_e)$.

When the verification is unsuccessful, HD(CCV′, CCV″) is "large" (above the decision threshold d_{max}) as a consequence of the characteristics of cryptographic check value. Namely, when the message is wrongly decoded (M′ is incorrect, i.e., M′ contains one or more errors) the number of errors in CCV″ is expected to be n/2 due to the avalanche criterion. In this case, CCV″ can take any of 2^n values (with equal probability).

Definition 1: Bit arrays A and B have length of n elements. Each element a_i and b_i is independent from other bits and has equal probability of taking the values 0 and 1, i.e.,

$$a_i = \begin{cases} 0, & p_0 = 1/2 \\ 1, & p_1 = 1/2 \end{cases} \qquad b_i = \begin{cases} 0, & p_0 = 1/2 \\ 1, & p_1 = 1/2 \end{cases} \quad (7)$$

Definition 2: Y_k and N_k are subsets of set S={1,2,...,n} where: Y_k has D elements, N_k has n-D elements, $Y_k \cup N_k = S$ and $Y_k \cap N_k = \varnothing$.

Lema 1: The Hamming distance d between arrays A and B:

$$d = HD(A,B) = \sum_{i=1}^{n} |a_i - b_i| \quad (8)$$

has the Binomial distribution B(n, p=1/2) i.e.,

$$pdf(d) = \binom{n}{d} \cdot \frac{1}{2^n}, \quad 0 \le d \le n, \quad (9)$$

Proof: The probability that d takes a concrete value D is:

$$P\{d = D\} = P\left\{\sum_{i=1}^{n} |a_i - b_i| = D\right\} = \\ = \sum_{Y_k, N_k} P\{a_i = b_i, \forall i \in Y_k\} \cdot P\{a_i \ne b_i, \forall i \in N_k\} \quad (10)$$

where the summation is applied on each subset pair (Y_k,N_k). The number of those pairs is: $k_{\max} = \binom{n}{D}$, so we have,

$$P\{d = D\} = \\ = \binom{n}{D} \cdot \left[\prod_{i=1}^{D} \left[P\{a_i = 1 \wedge b_i = 1\} + P\{a_i = 0 \wedge b_i = 0\} \right] + \right. \\ \left. + \prod_{i=1}^{n-D} \left[P\{a_i = 1 \wedge b_i = 0\} + P\{a_i = 0 \wedge b_i = 1\} \right] \right] = \\ = \binom{n}{D} \cdot \left[\left(\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \right)^D + \left(\frac{1}{2} \cdot \frac{1}{2} + \frac{1}{2} \cdot \frac{1}{2} \right)^{n-D} \right] = \\ = \binom{n}{D} \cdot \frac{1}{2^n}, \quad (11)$$

which proves the *Lema*.

In the case of an unsuccessful verification, CCV′ and CCV″ can be considered as independent bit arrays A and B according to *Definition 1*, and HD(CCV′, CCV″) will have the probability distribution function as shown in *Lema 1*:

$$pdf_2(d) = \binom{n}{d} \cdot \frac{1}{2^n}, \quad 0 \le d \le n, \qquad (12)$$

Equation (12) can also be explained in simpler way. Namely, when the message is not verified, the expected value of HD(CCV′, CCV″) is equal to the expected value of HD between CCV′ and any other fixed array of bits of the same length. If for simplicity, we choose an array of bits X = 00...0, the HD(CCV′, X) will also have Binomial distribution B(*n*,*p*), where *p*=1/2 since every bit in CCV′ is expected to be 0 or 1 with equal probability, so the *pdf* of HD(CCV′, CCV″) can be written as in (12).

By combining equations (4), (5), (6) and (12) in (3), for the parameter values *m*=160, *n*=160 and P_e=0.01, the probability distribution of *d* = HD(CCV′, CCV″) will have the shape as shown in Fig.3. Two regions are clearly distinguished: the left one for the case of successfully decoded message (i.e., M = M′) and the second − when the decoded message M′ is wrong (i.e., M ≠ M′). It is obvious that the probability distribution over *d* is zero for a great range of values between these regions, which means that the decision threshold d_{max} in the process of verification might take any value from this middle area.

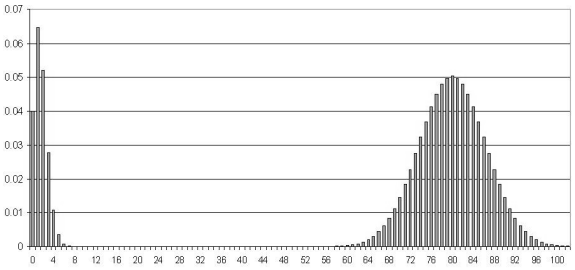

Fig. 3 Probability distribution of d=HD(CCV′, CCV″)

The rate of acceptance of decoded and by flipping corrected messages within verification process will be greater if the d_{max} is set to a greater value. The lower limit of the threshold (d_{max_low}) can be chosen in relation to wanted acceptance rate of message, i.e., regarding the predefined probability of message rejecting. In columns 3, 5 and 7 of Table 1 (see Appendix) the values of d_{max_low} are shown for different lengths of cryptographic check value (*n* = 160, 128, 64) and message (*m* = 160, 192, 256) so that *m*+*n* = 320 and for different BERs (depending on E_b/N_0). The criteria for choice of $d_{max,low}$ was that the probability of message rejecting (when M′ is correct) is less than 10^{-k}, i.e.,

$$\sum_{d=d_{\max\_low}+1}^{n} P_{M'corect} \cdot pdf_1(d) < 10^{-k}, \qquad (12)$$

where $P_{M'correct}$ and $pdf_1(d)$ are defined in (8) and (10), and the condition for message to be accepted as correct is,

$$d = HD(CCV', CCV'') \le d_{\max}. \qquad (13)$$

By setting the parameter *k* to an appropriate value, wanted minimal acceptance rate can be achieved and the matching values of d_{max_low} can be calculated from (12) and (13). The values of d_{max_low} obtained in this way are shown in Table I, where each cell contains four different values: for *k* = 4, 6, 10 and 15 respectively.

On one hand, greater d_{max} and higher acceptance rate of messages means speeding up the verification process, since the expected number of bit-flipping iterations leading to successful verification is smaller. Greater d_{max}, on the other hand, will increase the probability of false verification − the event when the decryptor wrongly decides that a decoded message M′ (or the corrected message M″ after a number of bit-flipping iterations) is correct. This happens when CCV″, which acts as a random variable (since calculated from wrong decoded message M′ and the secret key K), satisfies the condition (7). The probability of false verification becomes significant when the value of decision threshold d_{max} is getting closer to the region on the right side in Fig.3. Similarly as by choosing the lower limit, the upper limit of the threshold ($d_{max,high}$) can be found with regard to the probability of false verification which can be tolerated. This probability can be also defined by the use of parameter *k*, while $d_{max,high}$ will be the maximal integer that satisfies the following condition:

$$\sum_{d=0}^{d_{\max\_high}} P_{M'incorect} \cdot pdf_2(d) < 10^{-k}, \qquad (14)$$

(in (3) and (5) are the definitions of $P_{M'incorrect}$ and $pdf_2(d)$).

Columns 4, 6 and 8 of "Table 1" contain values of $d_{max,high}$ calculated from (8) for *k*=4, 5, 10 and 20 respectively. There is a lot of values within range [$d_{max,low}$+1, $d_{max,high}$] which could be taken as the threshold, for different values of parameters E_b/N_0, P_e, *m* and *n*.

Both SID and TSID methods have been simulated with the message and its CCV / HMAC tag, both of length of 160 bits. HMAC tag has been calculated

using RIPEMID160 hash function. Simulations have been performed using a Convolutional encoder of code rate r (= ½) and a constraint length m = 2, BPSK modulation, AWGN channel and SISO decoding using MAP algorithm.

The results of simulations are expressed through Cryptographic Check Error Rate (CCER), already defined by (1), as the ratio between the number of incorrect CCVs after (T)SID and the whole number of simulations for given set of parameters. In both methods 16 bits with smallest $|L|$-values were being flipped, i.e., maximally 2^{16} trials of soft correction (bit flipping) had been performed in each simulation. The value of decision threshold within TSID had been set to 20% of the CCV length d_{max}= 32.

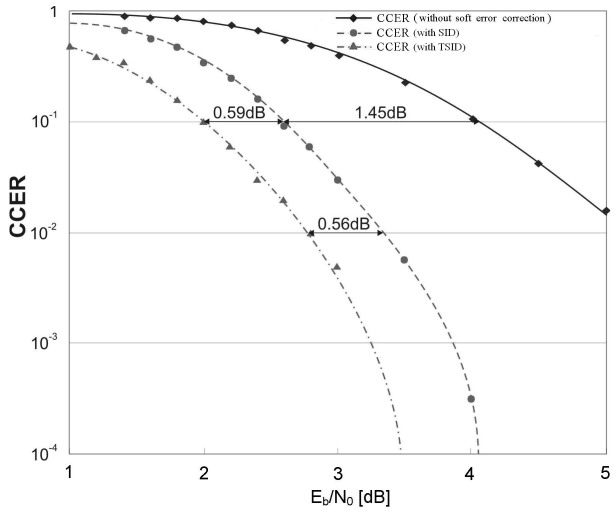

Fig. 4 Achieved coding gains of SID and SID with threshold (TSID)

The results are presented in Fig. 4, showing the achieved coding gain in comparison to standard 1/2 convolutional coding. Using original SID method, more than 1.4 dB of gain is achieved, while TSID obtains additional 0.5-0.6 dB.

4 Authentication Of Images Using TSID And Noise Tolerant MACs (NTMACs) Based On The Discrete Cosine Transform

4.1 Introduction

In this Section the application of TSID algorithm in combination with the Noise Tolerant Message Authentication Code (NTMAC) is investigated in image authentication. For images, the basic features are authenticated rather than authenticating the image itself. For this purpose, the Discrete Cosine Transform is used to extract the block by block features and authenticate the image block-wise. This is also beneficial for the TSID algorithm, which works well over small blocks of data as compared to big ones.

4.2 Discrete Cosine Transform in Image Processing

Discrete Cosine Transform is one of the most widely used techniques in image processing for basic feature extraction and compression. Its application in image processing was pioneered in [16]. Due to its better reconstruction capability, DCT is more suitable to images than other relevant transforms, such as Discrete Fourier Transform (DFT). DCT, like other transforms, tries to eliminate the correlation from image data. After de-correlation, each transform coefficient can be encoded independently without devitalizing the compression efficiency. Many well known image and video compression standards like JPEG and MPEG-1/2/4/H.26x, are based on 2-D DCT.

The Discrete cosine transform of a 2-D vector is defined as follows [16-18]:

$$X(l,k)=\alpha(l)\alpha(k)\frac{2}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}x(m,n)\cos\left[\frac{\pi(2m+1)}{2N}\right]\cos\left[\frac{\pi(2n+1)}{2N}\right]$$

where,

$$\alpha(i)=\begin{cases}1/\sqrt{2}\ , & i=0\\ 1\ , & 1\le i\le N-1\end{cases} \quad (15)$$

It is clear from (15) that the first coefficient (DC) represents the average intensity of the corresponding block and contains most of the energy and perceptual information. Also the inverse of 2-D DCT for l,k = 0, 1, ..., N-1 is defined as follows,

$$x(l,k)=\alpha(l)\alpha(k)\frac{2}{N}\sum_{m=0}^{N-1}\sum_{n=0}^{N-1}X(m,n)\cos\left[\frac{\pi(2m+1)}{2N}\right]\cos\left[\frac{\pi(2n+1)}{2N}\right] \quad (16)$$

Besides the general characteristics of DCT which are defined for every Fourier-like transform, other properties of DCT like de-correlation, energy compactness, symmetry and separability make it a convenient tool for image processing purposes.

4.3 Introduction and Definitions

The algorithm introduced in this paper is based on the DCT transform. It protects the transmitted DCT components of an image by NTMAC and performs soft authentication on received (noisy) images. The algorithm is able to localize errors in the images and to correct a certain number of them if they are below a

certain threshold. The following definition is used in the description of the proposed algorithm.

Definition 3: Let H´ be the received n-bit MAC for a transmitted message M. Let M´ be the received message, H´´ be the MAC recalculated at the receiver and let d be a small non-negative integer ($d << n/2$); then (M´, H´) is said to be d-soft-verified if HD(H´, H´´) ≤ d, where HD is the Hamming Distance between H´ and H´´.

For the sake of simplicity, let's assume the image to be transmitted is N × N pixels. Let m be the block size such that m|N, where both N and m are integers. The sender divides the image into m × m-pixel disjoint blocks (typically m is equal to 8). This is followed by the calculation of DCT for each block.

4.4 Image Authenticating and Correcting Weighted Noise Tolerant MAC (IAC-WNTMAC)

IAC-WNTMAC achieves error localization using the concept of weights. IAC-WNTMAC is based on NTMAC [7] and identifies the locations of potential erroneous blocks with a high probability.

The NTMAC algorithm [7] works by splitting a message / image into smaller components. A MAC is calculated for each of the smaller components and truncated to obtain a sub-MAC. The sub-MACs corresponding to these message components are concatenated to form the NTMAC. The IAC-WNTMAC tag calculation can either operate row-wise or column-wise on the image blocks. It is assumed here that the tag calculation is done row-wise. A DCT matrix is obtained for each block in the source image. Thus there are as many DCT matrices as the number of blocks in the source image. NTMAC is calculated based on the DC components of the DCT matrices taken row-wise. There are N/m such DCT matrices in each row and therefore N/m DC components are used to get one NTMAC (against a row).

The same step is repeated for all the rows, giving N/m NTMACs. This process is also repeated for the first minor diagonal after DC coefficient (called as first minor diagonal for the sake of simplicity) of the DCT matrices, giving another set of N/m NTMACs. This produces a total of 2(N/m) MACs, i.e., N/m + N/m. The usage of NTMAC improves the error localization, whereas the usage of the NTMAC for the minor diagonal increases the quality of reconstructed image at the receiver as explained next. All of these N/m + N/m NTMAC tags are appended together to obtain IAC-WNTMAC tag for transmission.

The image I´ and its IAC-WNTMAC´ tag are received over a noisy channel. The receiver recalculates IAC-WNTMAC tag on I´ to get IAC-WNTMAC´´. Now the received IAC-WNTMAC´ tag is compared with the recalculated IAC-WNTMAC´´ tag. This is done by comparing the corresponding sub-MACs. If the sub-MACs are d-soft-verified according to the definition given above, then the DC component is accepted as authentic and the message block corresponding to the DC component is declared as authentic. Otherwise, the block is marked as un-authentic / suspicious. All the blocks marked as un-authentic / suspicious will be tried for error correction using Chase like iterative error correction algorithm based on the bit reliabilities calculated using the MAP decoder at the receiver. This iterative error correction is repeated for the first minor diagonal as well, so that they can be reconstructed to get a better quality of the reconstructed image. However, the first minor diagonal has a lower weight than the DC component. Lower weight means that the threshold for the maximum number of iterations used for the recovery of the first minor diagonal is smaller than the threshold used for the recovery of the DC component, i.e., a variation of the EC-WNTMAC [10] is used. If T_{iterDC} is the iteration threshold used for the error correction of DC components and T_{itrFMD} is the same used for the first minor diagonal, then the total number of iterations are given by,

$$T_{itr} = T_{itrDC} + T_{itrFMD} \quad (17)$$

The pseudo-code of the IAC-WNTMAC tag generation and verification algorithms is given below for an N × N image. It can be easily extended to the general case where the image is not square. Also here weights are assigned based on the DC and the first minor diagonal elements, which can be easily extended to other minor diagonals. The notation DC is self explanatory, whereas MD represents the Minor Diagonal of the DCT matrix.

Algorithm: IAC-WNTMAC Tag Generation Algorithm

Inputs:

- Source Image (I)
- Image width / height in pixels (N)
- Block length (m)

Algorithm:

blocks = splitImageIntoBlocks(I, N, m)

for i = 1 to N/m

for j = 1 to N/m
DCT = $blocks_{i,j}$
DC = $DCT_{1,1}$
$subMAC_{DCj}$ = calcSubMAC(DC)
$subMAC_{MDj}$ = calcSubMAC($DCT_{1,2}$ || $DCT_{2,1}$)
end
$subMAC_{DCi} = subMAC_{DC1} || \dots subMAC_{DCN/m}$
$subMAC_{MDi} = subMAC_{MD1} || \dots subMAC_{MDN/m}$
end

$NTMAC_{DC} = subMAC_{DC1} || subMAC_{DC2} || \dots || subMAC_{DCN/m}$
$NTMAC_{MD} = subMAC_{MD1} || subMAC_{MD2} || \dots || subMAC_{MDN/m}$

Output:

$NTMAC_{DC} || NTMAC_{MD}$

Pseudo code for tag verification at the receiver:

Algorithm: IAC-WNTMAC Tag Verification at the Receiver

Inputs:

- Received Image (I´)
- Received NTMAC´
- Image width / height (N)
- Block length (m)
- Block LLRs (blockLLRs)

Algorithm:

I´´ = decompressImage(I´)
$subMAC´_{DC}$ = makeDCSubMACs(NTMAC´)
$subMAC´_{MD}$ = makeMDSubMACs(NTMAC´)
dc_LLRs = blockLLRsToDCLLRs(blockLLRs)
md_LLRs = blockLLRsToMDLLRs(blockLLRs)
blocks = makeBlocks(I´´, W, H, m)

for i=1 to N/m
for j=1 to N/m
DCT = $blocks_{i,j}$
DC = $DCT_{1,1}$
$subMAC_{DCi,j}$ = calcSubMAC(DC)
$subMAC_{MDi,j}$ = calcSubMAC($DCT_{1,2}$ || $DCT_{2,1}$)
if(HD($subMAC´_{DCi,j}$, $subMAC_{DCi,j}$) ≤ d)
performErrorCorrection(DC, $DC\_LLRs_{i,j}$)
end
if(HD($subMAC´_{MDi,j}$, $subMAC_{MDi,j}$) ≤ d)
performErrorCorrection($DCT_{1,2}||DCT_{2,1}$, $MD\_LLRs_{i,j}$)
end
end
end

Output: authenticMessageBlocks

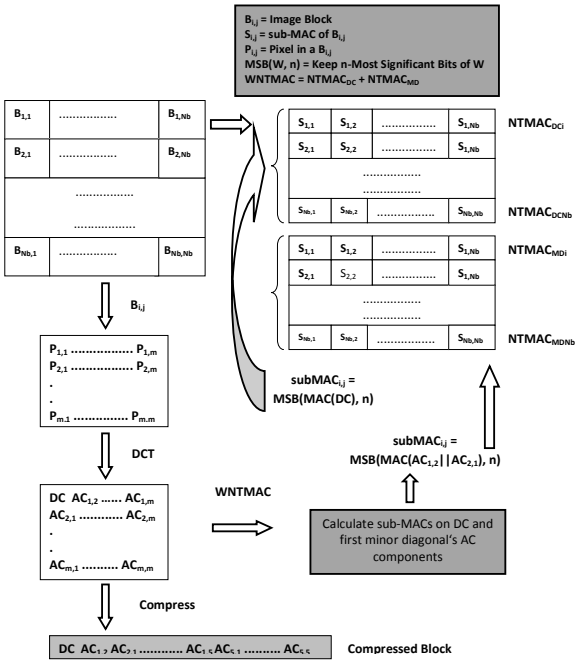

Fig. 5 IAC-WNTMAC Transmitter

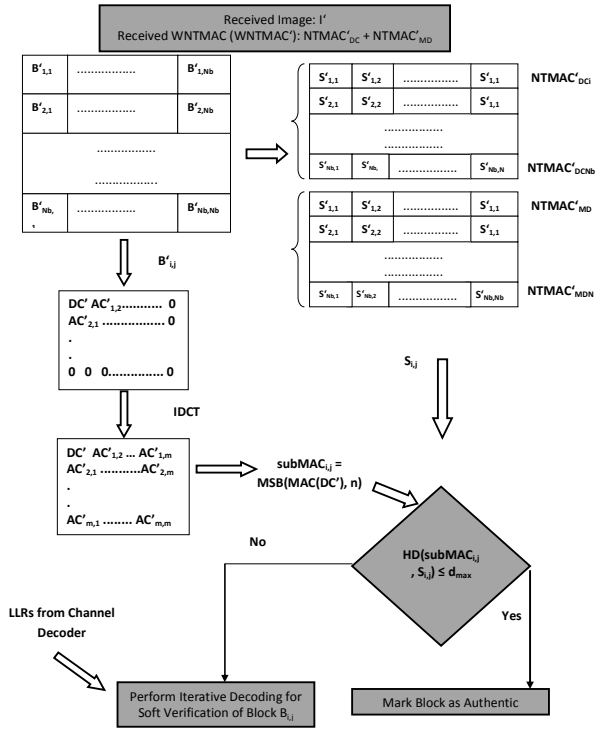

Fig. 6 IAC-WNTMAC Receiver

5 Analysis of the proposed Algorithm

IAC-WNTMAC algorithm is a variant of WNTMAC given in [10]. Therefore the analysis of the IAC-WNTMAC algorithm is based on WNTMAC. It is assumed that an ideal n-bit MAC (n ≥ 256) algorithm is used for each row and each column of concatenated selected DCT elements of the blocks of an image. The total number of concatenated DC coefficients is $(N/m)^2$ of k-bit each. Let the bit error rate of the channel be denoted as BER.

5.1 Performance Study

d-soft-verification is successful, if the difference between the received MAC and the recalculated one is not greater than the threshold value. Two types of errors can exist: "false rejection" (correct image is discarded) and "false acceptance" (wrong image is accepted).

The probability of a false rejection of the whole image (P_{FR}) depends on the policy of the application and the nature of the image blocks. False rejection is not as bad as false acceptance, which causes communication overhead and reduces the efficiency and therefore the security. Therefore the probability of false acceptance will be discussed.

False acceptance happens when there are error(s) in the received image I´, but the received and recalculated tag pair is d-soft-verified. The probability of false acceptance on the block level is:

$$P_{FA}(Block) = \left[1-(1-BER)^k\right]\left(\sum_{i=0}^{d}\binom{n}{i}BER^i(1-BER)^{n-i}\right)^2 \quad (18)$$

5.2 Security Considerations

The most important integrity threat against data is message substitution and forgery. It refers to any attempt for adding, removing and manipulating objects into data in order to fool the receiver to accept the wrong message. This threat in image data can lead to image tampering.

The algorithms introduced in this paper are based on the standard ideal MAC, so the generic attacks on MACs are considered as potential threats. As in the given approaches, the algorithms may tolerate a modest number of error(s) as a consequence of soft verification. The security strength is decreased generally compared to hard authentication MAC schemes by allowing near collisions. This drawback is compensated in both approaches. In the first one each DCT element is supported by two MACs instead of one MAC and the attacker has to forge DCT elements in such a way that both row and column MACs become d-soft-verified. In the second algorithm, the attacker has even more difficult task, the forgery attack requires forgery on protected DC and AC coefficients so that IAC-WNTMAC authentication on both selected DC and AC coefficients becomes successful.

A common approach for approximating the required complexity (data/time) for forgery attack on MACs is given by a "birthday paradox" which is based on finding collisions. In case of soft authentication, the attacker attempts to launch a near-collision attack [19]. Near-collision refers to any message pair whose MACs differ only in few bits from each other. By extending the birthday paradox to the introduced soft verification scheme with threshold d, it is expected to have a near collision (with at most d-bit differences) with the data complexity (C) of,

$$C = \sqrt{\frac{2^{2n}}{\sum_{i=0}^{d}\binom{2n}{i}}} \quad (19)$$

In the presented algorithms, the minimum MAC length used to protect row and columns of DC coefficients is 256 bits. The threshold value is set in such a way that the probability of events like false acceptance and false rejection remains significantly low. There are other experimental methods to find a convenient safe threshold zone through image processing techniques. It can be easily concluded that the security strength compensated by double length of the MAC is much larger than the required security strength of a standard MAC, for low threshold values. The security of the second approach is even higher due to secret partitioning.

6 Simulation Results

The simulation results are presented for the proposed algorithm using image transmission over AWGN channel with BPSK modulation. The results are given in the presence of rate 1/3 Turbo Codes. The extrinsic Log Likelihood Ratios (LLRs) produced by the decoder for Convolutional Turbo Codes (CTC) are used for bit reliabilities values.

The source image is a grayscale image of 128 × 128 pixels (each pixel of 8 bits). The image is split into 8 × 8 pixel non-overlapping blocks, giving a total of 16 × 16 blocks. DCT for each of these blocks is calculated and the DC components of the DCT sub-matrices are protected using the corresponding MACs

as explained earlier. Each element in the DCT matrices requires 2 octets. If the original image is transmitted using the standard MAC based protection, then either the whole image will be authentic or non-authentic and so it will either be accepted or discarded. HMAC-SHA-256 is used as the MAC in the simulations, thus in total 128 × 128 × 8 bits of image data plus 256 bits of MAC needs to be transmitted, which is equal to 131328 bits in total.

Using IAC-WNTMAC, the number of data bits transmitted is calculated as follows. Each WNTMAC tag is 256 bits long. Thus, 256 bits for each of the DC components in the 16 rows as well as another 256 bits for the first minor diagonals for each row are used. Thus 256 × 16 × 2 = 8192 bits of WNTMAC tags are transmitted along the 61440 bits of the data. This is equal to 53% of the whole data transmitted in the standard MAC tag based image transmission.

Each figure shown in the following sub-sections is divided into five constituent sub-images. First, the source image is shown followed by the received image. In the next sub-image, the suspicious block positions (identified through the proposed algorithms) are highlighted in white followed by these suspicious blocks highlighted in the received image. Finally, the resultant image is shown, which is obtained by applying the proposed error correction algorithm over the erroneous image based on the localized errors.

6.1 Simulation Results for IAC-WNTMAC

Images protected using IAC-WNTMACs have better error localization capabilities and so they can be reconstructed in a better manner as compared to the previous algorithm. The simulation results are presented in Fig. 7.

Fig. 7 IAC-WNTMAC at SNR 2.5 with Turbo Codes of rate-1/3

6.2 Image Error Rate (IER)

IER for both the algorithms is shown in Fig 8 at different values of E_b/N_0. The curves represent the IER in the presence of a standard MAC tag based protection scheme and then in the presence of IAC-WNTMAC. Fig. 4 shows that IAC-WNTMAC achieves a coding gain of 1.2 dB at IER of 10^{-4}. Its performance is due to dual error protection and recovery using weighted NTMAC.

7 Conclusion

An algorithm for approximate data authentication (SID) is presented first. This is extended further to TSID which more efficiently perform authentication by iteratively considering only the data part in authentication and doing a threshold number of comparisons till the match criteria is satisfied. Both the algorithms fall into the category of fuzzy authentication algorithms. The application of TSID together with the NTMAC using DCT is demonstrated in image authentication. Thus an algorithm for soft authentication, error localization and correction of images is presented. Soft authentication is performed using the standard MAC together with the threshold value. The main property of the algorithms is its ability of error localization and correction without compromise of security, which is shown in the analysis. Simulation results showing the high error recovery as well as relatively accurate error localization validate the theoretical analysis. In future it would be interesting to extend the proposed work by combining it with artificial intelligence based cooperative learning strategies, e.g., the one proposed in [20]. It is expected to get better content based image retrieval results using such approaches.

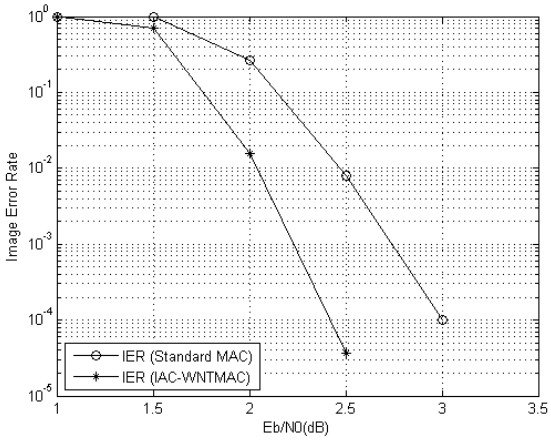

Fig. 8 IER over AWGN channel with BPSK modulation using Turbo codes of rate-1/3

References:

[1] N. Zivic, Joint Channel Coding and Cryptography, Shaker Verlag, Aachen (2008).

[2] C. Ruland, N. Zivic, Soft Input Decryption, 4th Turbocode Conference, 6th Source and Channel Code Conference, VDE/IEEE, Munich, Apr 2006

[3] D. Chase, A Class of Algorithms for Decoding Block Codes with Channel Measurement Information, IEEE Trans. Inform. Theory, IT-18, pp. 170-182, Jan 1972

[4] G. D. Jr. Forney, Generalized Minimum Distance Decoding, IEEE Trans. Inform. Theory, IT-12, pp. 125-131, Apr 1966

[5] R. Gravemen, L. Xie, and G. R. Arce, Approximate image message authentication codes, in Proc. 4th Annu. Symp. Advanced Telecommunications and Information Distribution Research Program, College Park, MD, 2000.

[6] R. Graveman and K. Fu, Approximate message authentication codes, in Proc. 3rd Annual Fed lab Symp. Advanced Telecommunications / Information Distribution, vol. 1, College Park, MD, Feb 1999.

[7] C. Boncelet, The NTMAC for authentication of noisy messages, IEEE Trans. Info. Forensics and Security, vol. 1, no. 1, pp. 35-42, Mar 2006.

[8] D. Onien, R. Safavi-Naini, P. Nickolas and Y. Desmedt, Unconditionally secure approximate message authentication, in Proc. The Second International Workshop on Coding and Cryptology, Springer, 2009.

[9] R. Ge, G. R. Arce and G. D. Crescenzo, Approximate message authentication codes for N-ary alphabets, IEEE Transactions on Information Forensics and Security, vol. 1, no. 1, 2006.

[10] O. Ur-Rehman, N. Zivic, S. Amir Hossein A. E. Tabatabaei, C. Ruland, Error Correcting and Weighted Noise Tolerant Message Authentication Codes, 5th Int. Conference on Signal Processing and Communication Systems (ICSPCS) / IEEE Conference, Hawaii, USA, Dec 2011.

[11] N. Zivic, M. Flanagan, On Joint Cryptographic Verification and Channel Decoding via the Maximum Likelihood Criterion, IEEE Comm. Letters, vol. 6, no. 5, pp. 717-719, May 2012.

[12] O. Ur-Rehman, A. Tabatabaei, N. Zivic, C. Ruland, Soft Authentication and Correction of Images, 9th International ITG Conference on Systems, Communications and Coding (SCC 2013), Jan 2013, Munich, Germany.

[13] L. Zhang, L. Xi, B. Zhou, Image Retrieval Method Based on Entropy and Fractal Coding, WSEAS Transaction on Systems, issue 4, vol. 7, Apr 2008.

[14] C. Aviles-Cruz, A. Ferreyra-Ramirez, J. J. Ocampo-Hidalgo, I. Vazquez-Alvarez, Structured-Image Retrieval invariant to rotation, scaling and translation, WSEAS Transaction on Systems, issue 8, vol. 8, Aug 2009.

[15] N. Doukas, Low Color-Depth Image Encryption Scheme for use in COTS Smartphones, WSEAS Transaction on Systems, issue 9, vol. 11, Sep 2012.

[16] A. Watson, Image compression using Discret Cosine Transform, Mathematical Journal, vol.1, no. 4, pp. 81-88, 1994.

[17] N. Ahmed, T. Natarajan, and K. R. Rao, Discrete Cosine Transform, IEEE Trans. Computers, vol. C-23, pp. 90-93, Jan 1974.

[18] P. Yip, and K. R. Rao, Fast Decimation-in-Time Algorithms for a Family of Discrete Sine and Cosine Transforms, Circuits, Systems and Signal Processing, Vol. 3, pp. 387-408, 1984.

[19] B. Preneel, P. C. van Oorschot, MDx-MAC and building fast MACs, from hash functions, Proc. CRYPTO 1995, LNCS 963, Springer-Verlag, pp. 1-14, 1995.

[20] F. Neri, Cooperative evolutive concept learning: an empirical study, WSEAS Transaction on Information Science and Applications,WSEAS Press (Wisconsin, USA), issue 5, vol. 2, pp. 559-563, May 2005.

Appendix

Table 1
Upper and Lower Limits of the Decision Threshold

E_b/N_0 [dB]	P_e	$\boldsymbol{n = 160}$, m=160		$\boldsymbol{n = 128}$, m=192		$\boldsymbol{n = 64}$, m=256	
		$\boldsymbol{d_{max\_low}}$ (k = 4, 6, 10, 15)	$\boldsymbol{d_{max\_high}}$ (k = 4, 5, 10, 20)	$\boldsymbol{d_{max\_low}}$ (k = 4, 6, 10, 15)	$\boldsymbol{d_{max\_high}}$ (k = 4, 5, 10, 20)	$\boldsymbol{d_{max\_low}}$ (k = 4, 6, 10, 15)	$\boldsymbol{d_{max\_high}}$ (k = 4, 5, 10, 20)
1	0.036	11, 15, 22, 29	56, 52, 40, 23	7, 12, 19, 25	42, 39, 28, 14	0, 6, 12, 17	16, 14, 7, -
1.5	0.0234	10, 13, 19, 25	56, 52, 40, 23	8, 11, 17, 22	42, 39, 28, 14	4, 7, 11, 16	16, 14, 7, -
2	0.0149	8, 11, 16, 21	56, 52 ,40, 23	7, 10, 14, 19	42, 39, 28, 14	4, 7, 10, 14	16, 14, 7, -
2.5	0.00681	6, 8, 12, 16	56, 53, 40, 23	4, 7, 11, 15	43, 40, 28, 14	4, 5, 9, 12	17, 14, 7, -
3	0.00376	5, 7, 10, 14	57, 53, 40, 24	4, 6, 9, 13	43, 40, 29, 14	3, 5, 7, 10	17, 15, 7, -
3.5	0.00142	3, 5, 7, 10	58, 55, 41, 24	3, 5, 7, 10	44, 41, 29, 14	2, 4, 6, 8	18, 15, 8, -
4	0.00037	2, 3, 5, 8	61, 57, 42, 25	2, 3, 5, 7	46, 43, 30, 15	2, 3, 4, 6	19, 16, 8, -
4.5	0.00024	2, 3, 5, 7	61, 57, 43, 25	2, 3, 5, 7	47, 43, 31, 15	2, 2, 4, 6	19, 17, 8, -
5	0.00012	2, 3, 4, 6	63, 58, 43, 26	2, 2, 4, 6	48, 44, 31, 16	1, 2, 4, 5	20, 18, 9, -