The 1998 IEEE International Joint Conference on Neural Networks Proceedings

(V.1) (V.B)

The 1998 IEEE International Joint Conference on Neural Networks Proceedings

IEEE World Congress on Computational Intelligence

May 4 - May 9, 1998
Anchorage, Alaska, USA

Volume 1 Page 1 - 802

The 1998 IEEE International Joint Conference on Neural Networks Proceedings

IEEE Catalog Number 98CH36227
ISBN 0-7803-4859-1 (softbound)
ISBN 0-7803-4860-5 (casebound)
ISBN 0-7803-4861-3 (microfiche)
ISBN 0-7803-4862-1 (CD-ROM)
ISSN 1098-7576

Additional copies of this publication are available from

IEEE Service Center
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
1-800-678-IEEE or 1-732-981-0060

Coordinated and produced by

ALTEC
Technology Driven Print Solutions

in cooperation with

DOCUMATION

Contact ALTEC at 800.997.9921 x 1334

Author's Index

A

B

C

D

E

An HMM/MFNN Hybrid Architecture Based on Stacked Generalization for Speaker Identification

Weiquan Bao Ke Chen and Huisheng Chi
Center for Information Science, National Laboratory of Machine Perception
Peking University, Beijing 100871, China
Email: wqbao@usa.net

Abstract

A hybrid architecture based upon Hidden Markov Models (HMMs) and Multilayer Feed-forward Neural Network (MFNN) is presented for speaker identification. Unlike most of the previous combing methods, the proposed architecture uses HMMs to model individual speaker and uses MFNN to deal with the inter-speaker information for improving performance. Learning in the proposed architecture consists of two phases. In particular, only a small amount of data is needed for training. The HMM/MFNN architecture has been applied to text-independent speaker identification. Simulation has shown that the hybrid architecture yields better identifying rate than that of conventional methods and other hybrid architectures.
Keywords: Speaker identification, Stacked generalization, HMM/MFNN hybrid architecture.

1 Introduction

The Hidden Markov Model (HMM) technique has already been applied to both speech and speaker recognition [1, 2]. However, the HMM has weak discriminative ability for classification [3]. The Multilayer Feed-forward Neural Network (MFNN) has been regarded as a powerful tool for static pattern classification [4] without temporal sequential processing. A rational selection is hybrid architectures, among them the HMM and Neural Network hybrids are promising and have been extensively studied [3, 8]. Basically, these hybrid architectures are classified into two categories; i.e. the MFNN is integrated with the HMM either for pre-processing or for post-processing. For pre-processing [4, 9], activation values of output nodes of the MFNN are used as some kind of posterior probability. This approach results in no assumption on a specific observation probability distribution in the HMM and it eliminates the feature independence assumption. However, much more training data are usually needed for parameter estimation since there exists a great number of adjustable parameters in the MFNN. In another hybrid architecture [10], the MFNN is used as labeler for discrete parameter HMMs. For post-processing, an elaborate representation of outputs of HMMs is sent to the MFNN as its input. A hybrid architecture has been proposed for the on-line handwritten character recognition, in which the vector consisting of all final observation probabilities of a sequence is used as the input of the MFNN to capture the variance of the sequence [5]. However, the method cannot be applied to speaker identification since the representation is unsuitable for conveying speaker-related information [8].

Unlike most of the previous hybrid approaches, we propose a hybrid architecture in which the MFNN is used for post-processing. In speaker identification, each HMM is often used to model an individual speaker's personality, which results in the lack of use of *inter-speaker information*. Empirical studies have shown that the classification errors of HMMs are relatively stable such that an MFNN can be employed to *"learn"* from HMMs for use of inter-speaker information based upon the principle of *stacked generalization* [6]. For this purpose, an elaborate representation of matching scores produced by observation sequences with HMMs is presented as the input of the MFNN. Thus the dimension of the input vector is just equal to the number of HMMs in the system, which results in a fast training of the MFNN due to the lower dimensionality in the input space. Some simulations with 20 population have been conducted on the KING database for text-independent speaker identification. It is evident that the proposed hybrid architecture yields better identifying rate than the conventional HMM method [8].

2 Stacked Generalization

Stacked generalization is a generic term referring to any scheme for feeding information from one set of *generalizers* to another before forming the final guess. A generalizer is a mapping $\{(x_k, y_k), q\} \rightarrow \{g\}$, $1 \leq k \leq m$, $x_k \in$ space R^n, $y_k \in R^n$ and $g \in R$, where (x_k, y_k) is a pair of a learning set of m pairs, q is a question and g is a guess [6]. Full generality would have the guess $\in R^p$, not R. However, for most applications one can replace a generalizer making guesses in R^p with the *Cartesian* product of p separate generalizers making guesses in R. Thus, we can simply take p to equal to 1. The distinguishing feature of stacked generalization is that the information fed up the net of generalizers comes from multiple partitions of the original learning set, all of which split up the learning set into two subsets. Each such pair of subsets is then used to collect information about the biases of the generalizing behavior of the original generalizer(s) with respect to the learning set. Stacked generalization is a means of estimating and correcting for the biases of the constituent generalizer(s) with respect to the provided learning set.

The first step in employing stacked generalization is choosing a set of r partitions, each of which splits a *learning set* Θ into two (usually disjoint) sets. Then label such set of partitions as Θ_{ij}, where $1 \leq i \leq r$ and $j \in \{1, 2\}$. Such set of partitions is called a *partition set*. For instance, for a *cross-validation partition set* (CVPS) [7], r is equal to m , where m is the total pair number of learning set. In CVPS, for all i, Θ_{i2} consists of a single element of Θ, the corresponding Θ_{i1} consists of the rest of Θ, and $\Theta_{i2} \neq \Theta_{j2}$ for $i \neq j$. For simplicity in the following discussion, we shall only consider the CVPS, thus any set Θ_{i2} consists of merely one element.

Now define the space of original learning set Θ as R^{n+1} or "level 0 space". When generalizing directly from Θ in the level 0 space, any generalizer is called a " level 0" generalizer, and the original learning set Θ is called a "level 0" learning set. For any real-world learning set Θ, there are always N generalizers $\{G_j\}$, where $N \geq 1$ (i.e., we are given a set of N separate sequences of functions $\{g_i\}$, $1 \leq i \leq \infty$), that one can use to extrapolate from Θ. Here we'll just discuss 2 levels stacked generalization. For each of the r partitions of Θ, $\{\Theta_{i1}, \Theta_{i2}\}$, look at a set of k numbers determined by a p subset of the $N\{G_j\}$ working together with that partition. Typically these k numbers can be those like the guesses made by the $\{G_j\}$ when taught with Θ_{i1} and presented as a question of the input component of the element Θ_{i2}, i.e., $G_j(\Theta_{i1}$; the input component of $\Theta_{i2})$. Take each such set of k numbers and view it as the input component of a point in a space R^{k+1} . The corresponding output value of each such point is calculated from the output component of the corresponding Θ_{i2} . This space R^{k+1} is called the "level 1 space". Since we have r partitions of Θ, we have r points in the level 1 space. Those r points are known as the "level 1" learning set. Now the common idea is to take a question in the level 0 space, pass it through the transformations which produced the input components of the level 1 learning set to get a level 1 question in the level 1 input space, and then answer that level 1 question by generating from level 1 learning set. This level 1 guess is then transformed back into a level 0 guess. Any generalizing process of this form is known as "stacked generalization". The process as a whole can be iterated, resulting in levels $p \geq 1$, i.e., multiple stackings. An example of how to use stacked generalization to combine generalizers is given as Figure 1 [6].

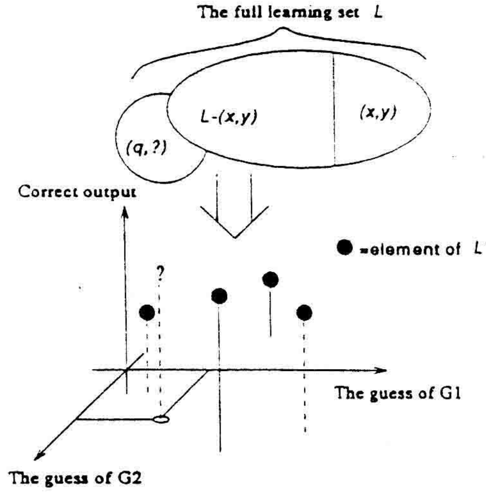

Figure 1: A simple example of how to use stacked generalization to combine generalizers(From [6])

Here we combine two generalizers, G_1 and G_2. The learning set L is represented figuratively by the full ellipse. A question q lying outside of L is also indicated. Finally, a partition of L into two portions is also indicated; one portion consists of the single input-output pair (x, y), and the other portion contains the rest of L. Given this partition, we train both G_1 and G_2 on the portion $\{L-(x, y)\}$. Then we ask both generalizers the question x; their guesses are g_1 and g_2. Generally, since the generalizers have not been trained with the pair (x, y), both g_1 and g_2 will differ from y. Therefore, we have just learned something. When G_1 guess

g_1 and G_2 guess g_2, the correct answer is y. This information can be cast as input-output information in a new space, i.e., as a single point with the 2-dimensional input (g_1, g_2) and the output (y). Choosing other partitions of L gives us other such points. Taken together, these points constitute a new learning set, L'. We now train G_1 and G_2 on all of L and ask them both the question q. Then we take their pair of guesses, and feed that pair as a question to a third generalizer which has been trained on L'. This third generalizer's guess is our final guess for what output corresponds to q. Assuming there is a strong correlation between the guesses made by G_1 and G_2 on the one hand, and the correct guess on the other hand, this implementation of stacked generalization will work well.

According to the principle of stacked generalization, in the speaker recognition procedure with HMM-based method, an HMM can be considered as a generalizer G, and its accumulating matching score can be considered as the guess g. Now there are c HMMs corresponding to c speakers, therefore $G_1G_2 \cdots G_c$ must be considered for the full learning sets corresponding to the guesses $g_1g_2 \cdots g_c$, respectively. The full learning set L can be subdived into two sets L_1 and L_2. The subset L_1 is used to train all HMMs. These HMMs are called level 0 generalizers and the subset L_1 is called level 0 learning set. Question q is that correct speaker under the output guess distribution $\{g_1g_2 \cdots g_c\}$. Since it is not enough to obtain perfect performance merely with HMM, an MFNN can be used as another generalizer to solve the question. All guesses generated by the HMMs in the level 0 learning set and in L_2 are used to train the MFNN. The MFNN is a level 1 generalizer and its guess (i.e., active output) is the correct guess. The guesses $g_1g_2 \cdots g_c$ of subset L_1 and L_2 can also be claimed priori knowledge that includes both right and error classification information. Thus, in our hybrid architecture, we use the HMMs to generate fixed dimensional feature vector $\{g_1g_2 \cdots g_c\}$ from temporal input sequence in the level 0 space and use the MFNN to classify these static time-alignment feature vector in the level 1 space. Apparently, these priori knowledge will give no benefit to the classifier if the distribution of knowledge about error is not correlated. On the contrary, the classifier will achieve powerful adaptive ability if it can learn from the priori knowledge whose distribution abides by somewhat correlative law. An HMM has the capability of accurately modeling statistical variation in spectral features which also represent speaker identity. When the HMM is trained insufficiently, the best matching model (i.e., the correct guess in level 0 space) may be a specific alternative model instead of a random one, e.g., the second-best model related to the true model. In the recognition phase, the input unknown observation sequence is generated by the second-best HMM with great probability, and under the similar training condition, the second-best model keeps relatively stable. When the following unknown observation sequence generates the same incorrect score distribution, the MFNN then can rectify such kind of error.

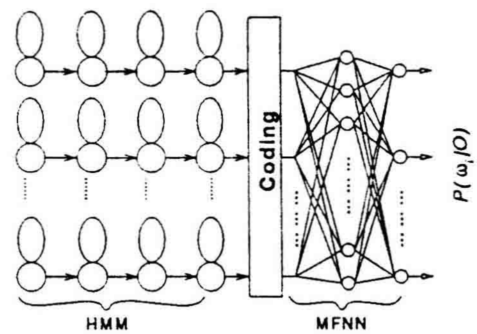

Figure 2: HMM/MFNN hybrid architecture

3 HMM/MFNN Hybrid Architecture

The key idea of the proposed method is to use MFNN to learn from priori knowledge effectively. In the hybrid architecture, we train a full-connected MFNN with Levenberg-Marquardt algorithm for the purpose [7]. HMM/MFNN hybrid architecture is shown as Figure 2.

Given an HMM $\lambda_k = (A_k, B_k, \pi_k)$, $k = 1, 2, \cdots, c$ and an unknown observation sequence $O = (O_1O_2 \cdots O_T)$, with forward-backward algorithm [2], we can obtain a set of $\hat{P}(O|\lambda_k) = \log[P(O|\lambda_k)]$. In order to serve for training MFNN, the logarithmic conditional probability is elaborately encoded first. Then the representation is used as the input of the MFNN. We denote the encoded feature vector as $X = \{p_1, p_2, \cdots, p_c\}$, where p_k, $1 \leq k \leq c$, is the encoded value of the logarithmic conditional probability. Thus the output of MFNN approaches posterior probability

$$P(\omega_i|O) \cong f\{\sum_{j=1}^{H} w_{ji}^{h \to o} f[\sum_{k=1}^{c} w_{kj}^{i \to h} p_k]\} \qquad (1)$$

where $w_{ji}^{h \to o}$ is a weight from the jth hidden node to the ith output node and $w_{kj}^{i \to h}$ is a weight from the kth input node to the jth hidden node. $f(\cdot)$ is a sigmoid function. $P(\omega_i|O)$ is the active probability of class ω_i with input observation sequence O. Instead of simply selecting the model producing the maximum value of $P(O|\lambda_k)$, the proposed architecture makes an MFNN perform the exact classification according to the priori distribution of all $P(O|\lambda_k)$, $1 \le k \le c$. In this architecture, the HMM still elaborately represents the temporal variation while the MFNN yields a kind of static pattern of which the inherent temporal information has been carried. We select the output node label whose active value is maximum as the exact class, i.e.

$$i_{max} = \arg\max_i P(\omega_i|O), \quad 1 \le i \le c \tag{2}$$

During the reestimation procedure of HMM parameters, in order to avoid underflow of iteration computation, a *Scaling* technique was introduced [2],

$$\hat{P}(O|\lambda_k) = \log[P(O|\lambda_k)] = -\sum_{t=1}^{T} \log C_t \tag{3}$$

where C_t is the scaling factor as

$$C_t = \frac{1}{\sum_{i=1}^{N} \alpha_t(i)} \tag{4}$$

and generally the denominator of C_t is less than 1. Thus, for a long sequence, the value of $\hat{P}(O|\lambda_k)$ is very small. Directly input it to MFNN will give MFNN a extreme difficulty to adjust its connection weights to convergence. An encoding rule is introduced to the hybrid architecture,

$$\text{Encoding Rule: } p_k = \frac{P_{max} - \hat{P}(O|\lambda_k)}{P_{max}} \qquad k = 1, 2, \cdots, c \tag{5}$$

where $P_{max} = \max\{\hat{P}(O|\lambda_k)\}$, $1 \le k \le c$. This rule considers the similar degree between every model and the best matching model as the input of MFNN.

Encoded by the rule, each observation sequence can finally obtain one vector as the input of MFNN. The amount of training data of MFNN is reduced extremely after the HMM processing while the temporal information of sequence is still represented accurately by HMM.

4 Experiments

The experiments were primarily conducted using a subset of the KING speech database which includes 5 wide-band sessions (labeled session 1 to session 5 as S-1, S-2, $\cdots$, S-5, respectively) per speaker. 1-2 sessions are used as training data and the remaining sessions as testing data. First, silence and unvoiced parts are removed from the speech priori to feature extraction using an adaptive energy threshold detector, and then a sequence of feature vectors is produced with 256 frame-rate and 16 orders LPC cepstrum analysis [8]. The final average length of speech of each speaker in a session is about 15 to 20 seconds. To evaluate different test utterance lengths, the sequence of feature vectors is divided into overlapping segments of T feature vectors. The first two segments from a sequence would be,

$$\overbrace{\bar{x}_1, \cdots, \bar{x}_L, \bar{x}_{L+1}, \cdots, \bar{x}_T}^{\text{Segment 1}}, \bar{x}_{T+1}, \cdots, \bar{x}_{T+L}$$

$$\bar{x}_1, \cdots, \bar{x}_L, \overbrace{\bar{x}_{L+1}, \cdots, \bar{x}_T, \bar{x}_{T+1}, \cdots, \bar{x}_{T+L}}^{\text{Segment 2}}$$

A test segment length of 3.2 seconds would correspond to $T = 100$ feature vectors for a 32 ms frame rate. L is the number of shifting frames between two adjacent segments. Here L is equal to 10, i.e. 320 ms speech. Each segment of T vectors is treated as a separate test utterance. The final recognition rate is then computed as the percentage of correctly identified T length segments over all test utterances.

The HMM for text-independent speaker identification is generally modeled as an ergodic continuous Gaussian mixture model. In our experiments, an ergodic Gaussian mixture CHMM with 8 states and 8 mixtures is adopted to provide the input vector for MFNN. The training phase of the ergodic CHMM is slightly different with that of left-right CHMM, as follows: The first is to produce an initial model through a *k-means* clustering algorithm by using all training data. The cluster number is identical to the number of states of CHMM, i.e. 8. Then the members of each state are clustered into 8 mixtures. The mean vector and variance matrix can be computed through these clusters and all training vectors. The initial transition probabilities are estimated by counting transitions between states in the training data. Although such an algorithm is not globally optimal, it is an effective way to generate an ergodic HMM. The second is the reestimation of the parameters. Standard *Segmental k-mean* algorithm is used for training the initial model until convergence [8]. In order to evaluate the performance according to our modeling method for different test utterance length, we carried out some experiments for 1.6s, 3.2s, 4.8s, 6.4s, 8.0s, 9.6s test length. Where S-1 and S-2 are used for training ergodic CHMM, S-3, S-4 and S-5 for testing. In the hybrid architecture, The training sets for MFNN are selected as follows: 30 segments are extracted from S-1 and S-2 randomly as well as 10 segments from S-3 and 5 segments from S-4. Thus, total 75 segments are used for training MFNN. S-5 and the remainder of S-3 and S-4 are used for testing the hybrid architecture performance. A comparison of average identifiying rates of S-4 and S-5 using ergodic CHMM and ergodic CHMM/MFNN is shown in Table 1.

Table 1: Average identifying accuracies(%) of S-4 and S-5 for different test utterance lengths with ergodic CHMM and ergodic CHMM/MFNN

Test length(s)	1.6	3.2	4.8	6.4	8.0	9.6
CHMM	74.55	81.98	85.03	85.75	86.83	86.55
CHMM/MFNN	76.64	86.44	90.86	94.29	97.51	94.11

As expected, with increase test utterance length, identification performance increases. Identification rate for the shortest test utterance length shows the greatest improvement. It is also evident that the necessary test utterance length is no less than 5 seconds in order to achieve a better performance. Furthermore, from the classification results provided by ergodic CHMMs, we find a general confusion error, i.e. in this session, some speaker is always recognized as another speaker incorrectly, and in another session, the same error occurs repeatedly. Such kind of regular priori knowledge about error classification benefits for a HMM/MFNN hybrid.

From the experimental results, merely a little better performance can be yielded for short test utterances, e.g. 1.6 and 3.2 seconds. By analyzing the confusion matrix of recognition results to different speakers, we found that the distribution of priori knowledge about error classification is very relaxed for short test utterances. It is briefly owing to the fact that short test utterance is not enough for elaborately representing speaker identity by CHMM, the matching score generated by CHMM naturally cannot provide benefit priori knowledge for MFNN. For 8.0 seconds test utterance length, using our hybrid architecture, we achieved the best identifying accuracy compared with conventional CHMM method. With these identification methods, it is evident that without enough training data, HMM/MFNN hybrid architecture method is far superior to a conventional CHMM method.

5 Conclusions

In this paper, we have presented a hybrid architecture based upon HMM and MFNN for speaker identification. The idea, that is, multiparts of a learning set are used to train a set of level 0 generalizers to obtain priori knowledge and such priori knowledge is used to train a level 1 generalizer to get the correct classification, can be generalized other classification problems. In particular, training HMM and MFNN individually reduces the computational burdens of the whole identification system while encoding the output parameters of HMM does not change the accurate representation to temporal sequence but speed up training of MFNN. Experimental results show that this hybrid architecture is a promising architecture and yields better performance than HMM-based method.

6 Acknowledgments

This work was jointly supported by the Climbing Program-National Key Project for Fundamental Research in China with Grant NSC 92097 and NNSF Grant 69475007, 69635020. The authors gratefully acknowledge the colleague Z.Wu for his benefit discussion about this work, and also thank L.Yang for providing us his Levenberg-Marquardt algorithm program.

References

[1] S. Furui, An Overview of Speaker Recognition Technology, Proc. ESCA Workshop on Automatic Speaker Recognition, Martigny, Switzerland, April 4-7, 1994, pp.1-9.

[2] L.R.Rabinner, A Tutorial on Hidden Markov Models and Selected Application in Speech Recognition. Proc.. IEEE, Vol.77, No.2, 1989, pp.257-286.

[3] H.Bourland and C.J.Wellekens, Links Between Markov Models and Multilayer Perceptrons, IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. 12, No. 12, Dec. 1990, pp.1167-1178.

[4] R.P.Lippmann, Pattern Classification Using Neural Networks, IEEE Commun. Mag. Vol.27, No.11, Nov. 1989, pp.47-64.

[5] Sung-Bae Cho and J.H.Kim, An HMM/MLP Architecture for Sequence Recognition, Neural Computation, 7,1995, pp.358-369.

[6] D.H.Wolpert, Stacked Generalization, Neural Networks, 5, 1992, pp.241-259.

[7] Li, Ker-Chan, From Stein's Unbiased Risk Estimates to The Method of Generalized Cross-Validation, The Annals of Statistics, 13, 1985, pp.1352-1377.

[8] Weiquan Bao, An HMM/MFNN Hybrid Architecture for Speaker Identification, M.E. Thesis, Center for Information Science, Peking University, 1997

[9] J.M.Naik and D.M.Lubensky, A Hybrid HMM-MLP Speaker Verification Algorithm for Telephone Speech, ICASSP'94, pp.I-153-156.

[10] P. L. Cerf, W. Ma and D. V. Compernolle, Multilayer Perceptrons as Labelers for Hidden Markov Models, IEEE Trans. Speech and Audio Processing, Vol.2, No.1, 1994, pp.185-193.

Neural Networks for Gesture-based Remote Control of a Mobile Robot

Hans-Joachim Boehme, Anja Brakensiek, Ulf-Dietrich Braumann, Markus Krabbes, Horst-Michael Gross
Department of Neuroinformatics, Technical University of Ilmenau, 98684 Ilmenau, Germany.
E-mail: hans@informatik.tu-ilmenau.de

Abstract— **We present a neural network architecture for gesture-based interaction between a mobile robot and its user, thereby spanning a bridge from the *localisation of the user* over the *recognition of its gestural instruction* to the *generation of the appropriate robot behavior*. Since this system is applied under real-world conditions, especially concerning the localisation of a human user, some proper techniques are needed which have an adequate robustness. Hence, the combination of several components of saliency towards a multi-cue approach, integrating structure- and color-based features, is proposed. At the moment, the gestures itself are very simple and can be described by the spatial relation between face and hands of the person. The organisation of the appropriate robot behavior is realised by means of a mixture of neural agents, responsible for certain aspects of the navigation task. Due to the complexity of the whole system, above all we use "standard neural network models", which are modified or extended according to the task at hand. Preliminary results show the reliability of the overall approach as well as the sufficient functionality of the already realised submodules.**

I. Introduction and Scenario

Fig. 1. The mobile robot MILVA. Provided with the necessary on-board equipment (68040-VME-system, 2 PC-systems, CNAPS-board, framegrabber) and different sensors (3 cameras, laserscanner, ultrasound and infrared distance measures, bumpers) MILVA serves as the testbed for the human-machine-interaction.

Figure 1 shows our robot platform MILVA (Multisensory Intelligent Learning Vehicle in neural Architecture). A two-camera-system with 7 degrees of freedom (for each camera pan, tilt and zoom, additional pan for both cameras) serves for the interaction with a possible user and actively observes its operational environment. An additional camera, mounted at the front of the robot, provides the visual information for navigation.

The use of our system as an intelligent luggage carrier, for instance at a railway station or an airport, was chosen as a hypothetic scenario for the following reasons: First, we must take into account the capabilities of our robot which does not have manipulators and can only move itself. Second, the scenario is to naturally motivate a gesture-based dialogue between the user and the serving system. At a railway station with a lot of people and a high amount of surrounding noise a gesture-based dialogue seems to be the only possible way for interaction.

Recently there is an increasing interest in video based interface techniques, allowing more natural interaction between users and systems than common interface devices do. A considerable number of approaches for the design of intelligent and adaptive human-machine-interfaces have been proposed (see for instance [8], [15], [7]).

The superior goals of our research concerning the proposed architecture (GESTIK-project[1]) are the highest possible robustness of the intelligent visual interface under highly varying environmental conditions as well as the sufficient organisation of the appropriate robot behavior, achieved by continuous interaction between robot and human user.

II. Neural architecture for user localisation and gesture recognition

Figure 2 provides a coarse sketch of the neural architecture for user localisation and gesture recognition. The several components of the architecture will be described in the following subsections.

[1]Supported by the Thuringian Ministry of Science, Research and Culture (TMWFK, GESTIK-Project)

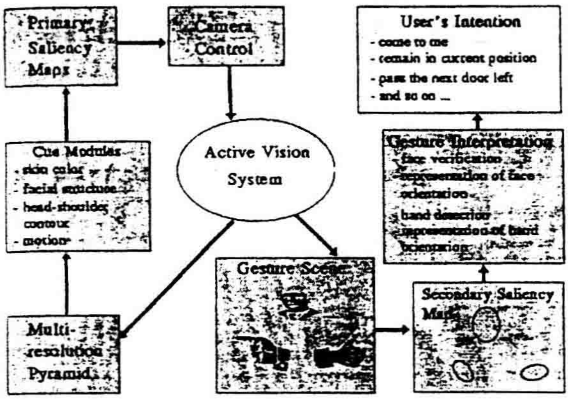

Fig. 2. Building blocks of the neural architecture for user localisation and gesture recognition

A. *Multi-cue approach for user localisation*

Initially both cameras of the two-camera-system operate in wide-angle-mode in order to cover the greatest possible area of the environment. Multiresolution pyramids transform the images into a multiscale representation. Four cue modules which are sensitive to *skin color, facial structure, structure of a head-shoulder-contour* and *motion*, respectively, operate at all levels of the two pyramids. The utility of the different, parallel processing cue modules is to make the whole system robust and more or less independent of the presence of one certain information source in the images. Hence, we can handle varying environmental circumstances much easier, which, for instance, make the skin color detection difficult or almost impossible. Furthermore, high expense for the development of the cue modules can be avoided (see [4], [3], [11], too).

a) Skin color

For the generation of a skin color training data set, portrait images of different persons (of our lab) were segmented manually. The images were acquired under appropriate lighting conditions (typical for our lab environment).

In order to obtain almost constant color sensation, first we map the RGB color space into a fundamental color space and employ a color adaptation method (see [21]). Then, we return into the RGB color space and define a 2-dimensional Gaussian function via calculation of the mean and the covariance of that skin color data set to model the obtained skin color distribution roughly. Furthermore, if a face region could be verified, a new Gaussian model is created, more specific for the illumination and the skin type at hand. Via this model the detection of skin colored regions, especially hands, can be improved. This is very important because the hand regions cannot be segmented by structural information (see [13], too).

A more detailed description of our skin color investigations can be found in [5] and [6].

b) Facial structure

In our scenario we assume that a person is an intended user if its face is oriented towards the robot.
The detection of facial structure uses the gray value image and employs eigenfaces generated by a principal component analysis (PCA) of the images contained in the ORL data set (http://www.cam-orl.co.uk/facedatabase.html; see [19], too). The image regions (15 x 15 pixels) used for the PCA were extracted manually and were normalised by their mean and standard deviation (see also [24], [23]). Then, the input image is processed with 3 eigenfaces (largest eigenvalues). Besides the preprocessing steps, the classification of the obtained fit values remains a difficult problem. The best results we achieved with a supervised Growing-Neural-Gas-Network (GNG, [10]), performing a mapping from the fit values to 2 classes (face, no face). For the training of the GNG a data set of 174 positive (face) and 174 negative (no face) examples was created. To improve the generalisation ability of the network, we implemented a bootstrap algorithm [23] which encloses false classified image regions into the set of the negative examples automatically.

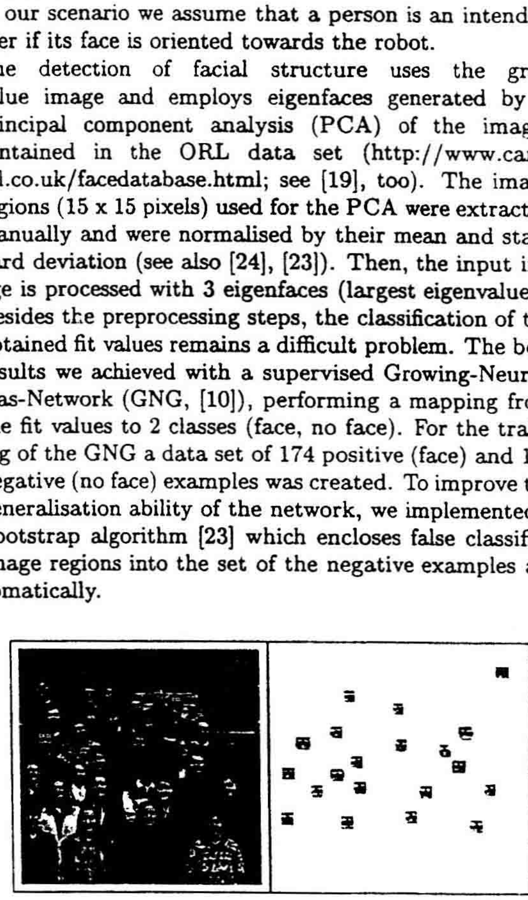

Fig. 3. Detection of frontally aligned faces. The detected faces are marked in the right image (likelihood higher than 0.7).

The performance of the face detection is demonstrated in figure 3, where an image taken from [23] was processed. False positive detected regions cannot be avoided entirely (top right), but this region very likely covers no skin color, and therefore, by combining skin color and facial structure such mislocalisations can be rejected.

c) Head-shoulder-contour

Similar to the detection of facial structure, the localisation of a head-shoulder-contour operates on the gray level image of each level of the multiresolution pyramids. The basic idea is to use an appropriate spatial configuration of Gabor filters (filter arrangement, see figure 4) and to classify the obtained filter outputs by a specially tuned distance measure (Hamming distance) between the actual filter outputs and a prototype.

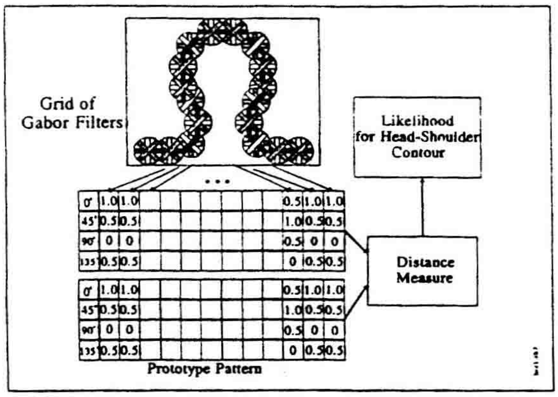

Fig. 4. Processing scheme for detection of a head-shoulder-contour.

d) Motion

Our favoured approach was proposed in [2] and [9]. Based on image differentiation motion is detected in the first step, leading to a binary motion energy image. The second step accumulates this motion energy over a certain period of time resulting in a motion history image. This approach is reliable especially for the following reason: The detection as well as the accumulation of motion could be realised via dynamic neural fields, and by means of different sets of parameters of such fields, different task specific aspects of motion information can be obtained.

e) Dynamic neural fields for generation of primary saliency maps

All cue modules supply input for the primary saliency maps at each level of the multiresolution pyramid, as shown in figure 5.

To achieve a good localisation a selection mechanism is needed to make a definite choice. This is not limited to a two-dimensional position. Since we use five resolutions (fine to coarse) we actually can localise persons even in different distances. Therefore, a neural field (array) for selection of the most salient region should be three-dimensional.

Those fields can be described as recurrent nonlinear dynamic systems (cf. [1], [14]). Regarding to the selection task we need a dynamic behavior which leads to

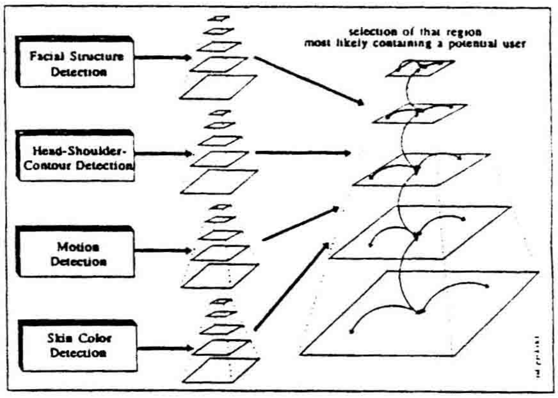

Fig. 5. Generation of a scale space pyramid of primary saliency maps

one local region of active neurons successfully competing against the others, i. e. the formation of one single blob of active neurons as an equilibrium state of the field. The following equations describe the system:

$$\tau \frac{d}{dt} z(\vec{r}, t) = -z(\vec{r}, t) - c_h h(t)$$

$$+ c_l \int_R w(\vec{r} - \vec{r}\,') y(\vec{r}\,', t) d^2\vec{r}\,' + c_i x(\vec{r}, t) \qquad (1)$$

$$\text{with} \quad w(\vec{r} - \vec{r}\,') = 2 \exp(\frac{-3|\vec{r} - \vec{r}\,'|^2}{2\sigma^2})$$

$$- \exp(\frac{-|\vec{r} - \vec{r}\,'|^2}{\sigma^2}) \quad , \qquad (2)$$

$$y(\vec{r}, t) = \frac{1}{1 + exp(-z(\vec{r}, t))} \quad \text{and} \qquad (3)$$

$$h(t) = \int_R y(\vec{r}\,'', t) d\vec{r}\,'' \qquad (4)$$

Herein $\vec{r} = (x, y, z)^T$ denotes the coordinate of a neuron, $z(\vec{r}, t)$ is the activation of a neuron $\vec{r}$ at time t, $y(\vec{r}, t)$ is the activity of this neuron, $x(\vec{r}, t)$ denotes the external input, $h(t)$ is the activity of a global inhibitory interneuron, $w(\vec{r} - \vec{r}\,')$ denotes the function of lateral activation of neuron $\vec{r}$ from the surrounding neighbourhood R. Further, τ is the time constant of the dynamical system and σ is the deviance of the gaussians determining the function of lateral activation. For the computation we used the following values for the constants: $c_h = 0.025$, $c_l = 0.1$, $c_i = 0.1$, $\sigma = 2$ (halved z-direction), $\tau = 10$ with $\Delta T = 1$ (ΔT: sampling rate). The range R of the function of lateral activation reachs over 5 pixels and 3 pixels in z-direction, respectively (anisotropic neighbourhood).

The results of the systems shall be qualitatively illustrated in figure 6. The presented results are exemplary, the usage of the shape of contour provides one solution for the person localisation problem, even under quite different conditions. In our ongoing work, the same principle is extended to the whole saliency pyramid, integrating all cue modules. The novel approach with a three-dimensional dynamic neural field can be assessed as robust method for the selection process, very reliable for the task at hand.

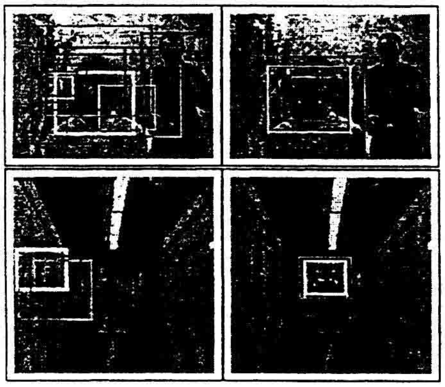

Fig. 6. Input images with marked head-shoulder-contours, obtained at the different levels of the multiresolution pyramid by the proposed method; *left:* without dynamic selection; *right:* by means of dynamic selection (white rectangles mark the highest likelihood).

B. Control of the two-camera-system

A camera control module, based on a neural approach proposed in [22], was extended for the control of the two-camera system. The basic idea is that a definite configuration of the cameras is assumed. Therefore, after a possible user (face region) was located in either camera image, the second camera is directed towards this user. This is realised by means of controlling the pan/tilt of this camera as well as the additional pan for both cameras. Therefore, the initial camera configuration (especially the base distance) remains stable.

As soon as a possible user (face region) is detected in one of the camera images, this camera serves as *general-view-camera*, whereas the second camera becomes the *gesture-camera*. The necessary distance estimation is provided by the cue modules detecting structural information (face and head-shoulder-contour). The resulting *gesture-scene* should contain all gesture-relevant parts of the intended user. Furthermore, the gesture-camera is controlled such that the expected face region will appear on a predefined position in the image with a predefined scale, too. Hence, we do not have to ensure scale invariance by the further processing steps.

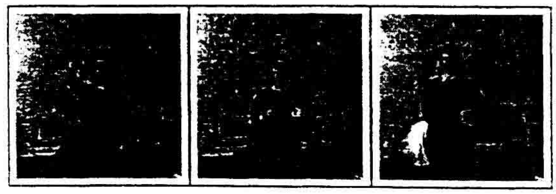

Fig. 7. Possible intuitive gestures (poses); from left to right they could carry the following meanings for the robot: hello, stop, move right

C. Detection and interpretation of gestures

a) Definition of a gesture set
For complexity reasons, we have predefined a gesture alphabet and have assumed only static gestures (poses), which are stable for a certain period of time (see fig. 7). The mapping between the gestures to be recognized and the associated actions of the robot is predefined, too (see also [15]).

b) Generation of the secondary saliency map
A secondary saliency map is created for the gesture-scene, which determines the sequential processing of this scene. Similar to the primary saliency map we utilise topographically organised neural fields.

To simplify the task, we mainly employ the skin color information as the input for this field, thereby assuming that the skin color segmentation is robust enough. Because of the camera control, the prominent position and size of a hypothetic face region is known. So, by means of specially tuned field parameters (coupling width and strength) the emergence of an activity blob covering the face region is highly supported. Therefore, the face region will be the first area to be analysed in detail (see the following section). The hand regions become salient, too.

c) Face verification and representation of gestures
The next processing step must provide a face verification, that means we have to decide if there is a face at all, and if it is oriented towards the robot.

To obtain this information, a very simple method for direct mapping of grey value image parts to the corresponding object orientation (both, faces and hands), based on a MLP network, was tested. Preliminary results show the sufficient functionality of such an approach under certain constraints (unstructured background). At present, the approach is examined under real world conditions. Furthermore, the detailed analysis of faces and hands via a regular grid of Gabor filters and a following classification of the Gabor filter outputs with a neural classifier (see also [17], [18]), will be taken