Isolated Vowel Recognition Using Linear Predictive Features and Neural Network Classifier Fusion

Jeff Byorick, Ravi P. Ramachandran and Robi Polikar
Department of Electrical and Computer Engineering
Rowan University
201 Mullica Hill Road
Glassboro, New Jersey 08028

Abstract - *In this work, various linear predictive feature vectors were used to train three different automated neural networks type classifiers for the task of isolated vowel recognition. The features used included linear prediction filter coefficients, reflection coefficients, log area ratios, and the linear predictive cepstrum. The three neural network classifiers used are the multilayer perceptron, radial basis function and the probabilistic neural network. The linear predictive cepstrum of dimension 12 is the best feature especially when training is done on clean speech and testing is done on noisy speech. Three different classifier fusion strategies (linear fusion, majority voting and weighted majority voting) were found to improve the performance. Linear fusion with varying weights is the best method and is most robust to noise.*

1 Introduction

Speech recognition plays an increasingly important role in Voice Web technologies, which allows users to access a web site from a phone using spoken commands. One of the main applications of speech processing is caller authentication, which is achieved either by using the voiceprint of a particular caller or using a pre-assigned password that allows access to the service. The password access has the same basis as vowel recognition.

Isolated vowel recognition is fundamentally a pattern recognition problem. A vowel is a vibration of the vocal cords that produces quasi-periodic pulses of air that excite the vocal tract. A particular vowel is represented as a discrete signal. The goal of vowel recognition is to find a feature vector that is a compact representation of the signal and to ensure that it will facilitate discrimination among different vowels. The best compact representation of a vowel is its formants because each vowel produces different formants. The formants are embedded in the frequency response of the linear prediction (LP) filter [1]. The LP filter contains the vocal tract information by describing the spectral envelope of the speech.

The LP filter is an all-pole filter of order p as given by

$$H(z) = \frac{1}{1 - \sum_{k=1}^{p} d(k) z^{-k}} \qquad (1)$$

The filter or predictor coefficients $d(k)$ are denoted by the p-dimensional vector **d**. In order to compute the LP filter coefficient vector **d**, the auto-correlation of the signal $t(n)$ is first computed as :

$$R_i = \sum_{n=0}^{N-1-i} t(n) t(n+i) \qquad (2)$$

where N is the number of samples in the signal and i is the time lag of the autocorrelation function.

The Levinson Durbin algorithm [1] is used to compute the filter and reflection coefficients (denoted as **refl**). The reflection coefficients form a one to one correspondence with the LP filter coefficients. The algorithm is shown in Figure 1.

After the analysis is complete, both the filter coefficient vector **d** and the reflection coefficient vector **refl** will describe the spectral envelope of the signal, but they will not contain the fine spectrum details. This is a powerful compression of the signal because the important vocal tract information is still captured. These two feature vectors are used in this study.

Another good representation of the vocal tract information is the cepstrum [1]. It is the sequence whose z-transform is equal to log H(z). It provides a measure of the logarithm of the spectral envelope and has been used as a common feature in speaker identification [2]. The variable $cep(n)$ for $n > 0$ denotes the cepstrum. It is computed using Equation (3). The notation **cep** denotes the vector of cepstrum coefficients.

$$cep(n) = d(n) + \sum_{k=1}^{n-1} (1 - \frac{k}{n}) cep(n-k) \qquad (3)$$

Algorithm Levinson-Durbin

$$energy = R(0) \quad (4)$$

For $i = 1$ to p (LP order)

$$sum = \sum_{j=1}^{i-1} dprev(j)R(i-j) \quad (5)$$

$$refl(i) = \frac{R(i) - sum}{energy} \quad (6)$$

$$d(i) = refl(i) \quad (7)$$

For $j = 1$ to i-1

$$d(j) = dprev(j) - refl(i)dprev(i-j) \quad (8)$$

End

$$energy = energy\left[1 - (refl(i))^2\right] \quad (9)$$

For $j = 1$ to p (LP order)

$$dprev(j) = d(j) \quad (10)$$

End

End

Figure 1. Levinson Durbin Algorithm.

The last representation that is examined uses the log area ratios. They are less spectrally sensitive than the reflection coefficient feature. This is useful if the noise corrupts the signal. The variable *lar(i)* denotes the ith log area ratio. It is computed from the reflection coefficients using Equation (11). The notation **lar** denotes the vector of log area ratios.

$$lar(i) = \ln\left[\frac{1 - refl(i)}{1 + refl(i)}\right] \quad (11)$$

Once all of the data are collected, the classifier is ready for training. In this case, a multilayer perceptron neural network (MLP) is chosen because it has successfully been used in similar type classification problems [3]. Given adequate training data, MLP neural networks can generate complex decision boundaries that can solve an arbitrarily complex classification task.

MLP is a three-layer network that consists of an input layer, a hidden layer, and an output layer, which are connected to each other by a matrix of weights. The initial weights are assigned randomly. The output of each hidden layer node is the weighted sum of the input or feature vector passed through a nonlinear function. The nonlinear function that is commonly used is the sigmoid function. This function works well in classification type problems because it forces the output to approach 0 or 1. The output of the network is the weighted sum of the hidden layer node outputs passed through another nonlinear function.

The weight update rule that is used is backpropagation with a variable learning rate and momentum. After each epoch, the weights are updated until a predetermined criterion is met. A small portion of the previous weight is added in the direction of the steepest descent. This value is computed differently for each layer. The momentum term is incorporated into the weight update rule to help in convergence. Its goal is to help prevent converging to a local minimum. It acts as a filter by smoothing the error surface. This prevents a noisy error surface from hindering network training. Detailed information on the MLP can be found in [5-8].

Various classifier fusion experiments were performed. These include fusing the outputs of three different neural network structures using linear fusion, majority voting, and weighted majority voting. A more detailed explanation of the fusion experiments is provided later. In addition to the multilayer perceptron (MLP), the radial basis function (RBF) network and the probabilistic neural network (PNN) are used in the experiments.

The RBF network is a fully connected three-layered network that consists of an input layer, a pattern layer, and an output layer. The weights in the pattern layer are equal to the values of the training data. The weights in the output layer are initialized using small random values. The output of the pattern layer is computed by finding the distance between the input data and the pattern layer weights. It is then passed through a nonlinear Gaussian function. The outputs of the pattern layer are then sent to the output layer and their weighted sum is computed.

There are various learning rules for the RBF network. The one that was chosen consisted of adding additional pattern nodes until the mean square error reached a minimum criterion. More details on this network architecture are provided in [5 - 8].

In a PNN, the input layer is fully connected to the pattern layer, but the pattern layer is sparsely connected to the output layer. A PNN has a training algorithm that only

requires one pass of the data through the network. The weights in the pattern layer are normalized versions of the input vectors. A pattern layer node is created for each data sample. It is connected to the output corresponding to its correct class. For classification, PNN computes the weighted sum of the pattern layer outputs. The outputs of the pattern layer are then passed through a nonlinear Gaussian function and stored according to the class that is associated with the particular node. More information on this network is found in [4,7].

2 Results on Comparison of Features

The vowels 'a', 'e', 'i', and 'u' were synthesized with a signal length of 240 and a sampling rate of 8 kHz. This produced 4 formants for each vowel as shown in Figure 2, where the horizontal axis is normalized frequency and the vertical axis is the absolute magnitude of the spectral components.

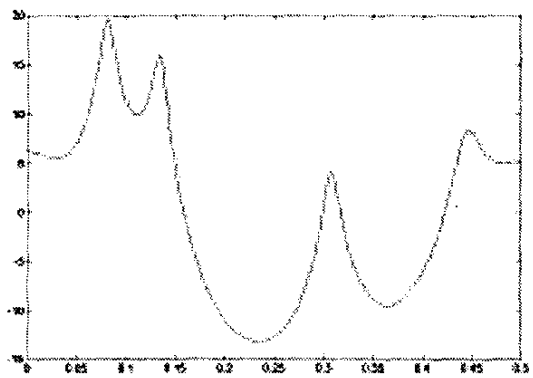

Figure 2. Spectral envelope of the vowel 'a' as determined by the spectrum of the LP filter. The peaks that appear in the image represent the formants of this particular vowel.

In order to produce sufficiently variable data, the pitch was varied on each of the vowels to produce an adequate amount of data. The pitch was varied from 2.5ms to 25ms. The training data consisted of 37 instances, whereas the test data contained 18 instances. This process was repeated for each of the four different feature vectors.

2.1 Effect of Feature Dimension

The effect of dimensionality was investigated to determine if the amount of relevant information could be obtained with a lower order analysis. This provides insight into the minimum feature dimension needed for maximum classification performance. The feature vector dimensions ranged from two to twenty. Note that the particular feature vector dimension corresponded to the LP order p. A weaker classifier was used to train the neural network so as to put more emphasis on how feature vector dimension affected the performance of the network. The MLP was trained with an error goal of .1 and 25 hidden layer nodes. Tables 1, 2, 3 and 4 show the recognition accuracy on the test data for each feature.

Table 1. Recognition Accuracy (%) With Varying Dimension of Predictor Coefficients

Predictor Coefficient (d) dimension	Recognition Accuracy (%)
2	72.22
4	75.00
6	73.61
8	75.00
10	75.00
12	79.16
14	84.72
16	83.33
18	86.11
20	80.55

Table 2. Recognition Accuracy (%) With Varying Dimension of Cepstrum

Cepstrum Dimension	Recognition Accuracy (%)
2	75.00
4	73.61
6	75.00
8	75.00
10	80.55
12	88.88
14	91.66
16	87.50
18	93.05
20	83.33

Table 3. Recognition Accuracy (%) With Varying Dimension of Log Area Ratios

Log Area Ratio Dimension	Recognition Accuracy (%)
2	75.00
4	77.77
6	79.16
8	84.72
10	90.27
12	94.44
14	84.72
16	80.55
18	84.72
20	83.33

Table 4. Recognition Accuracy (%) With Varying Dimension of Reflection Coefficients

Reflection Coefficient Dimension	Recognition Accuracy (%)
2	66.67
4	70.83
6	70.83
8	77.77
10	76.38
12	77.77
14	83.33
16	83.33
18	84.72
20	79.16

A weaker classifier was used to evaluate the generalization performance of feature vectors of different dimensions. This shows how the classifier performs with less training and a different set of features and varying dimension. This is a better representation of how the different dimensions affected the performance because the current classification problem was not complex. The log area ratio performed the best with the least dimension (94.44 % for a dimension of 12). The cepstrum performed fairly close but requiring a higher dimension. The trend of each LP feature was to increase performance as the dimension was increased. This is expected because more of the spectral features are included as the order of the LP analysis is increased. This trend continued until an optimum feature dimension was reached. A drop in performance was noticed at higher dimensions. This is expected because increased dimensionality produced more complicated decision boundaries, resulting in less separation between each of the vowels. For the rest of the experiments, we focus on a dimension of 12.

2.2 Noise Perturbation of Feature Vectors

The effect of introducing additive random noise was examined. The neural networks were trained with twelve dimensional clean data. The twelve dimensional test data was corrupted with random additive noise. This was accomplished using a uniform random number generator. The rationale was to test how the performance was affected by the addition of noise to the feature vector. This noise is equivalent to introducing a random perturbation to each component of the feature vector. Stronger classifiers were used to conduct this experiment. The parameters of the network are shown in Tables 5 and 6, where EG represents the error goal and HLN indicates number of hidden layer nodes. Table 5 shows the recognition accuracy when no perturbation is introduced. Table 6 shows the recognition accuracy when perturbations to the feature vector are introduced. Table 7 shows the decrease in performance due to feature vector perturbation.

Table 5. Recognition Accuracy (%) with no Perturbations to Feature Vectors

12th order	EG	HLN	EG	HLN
	.05	25	.01	25
Cepstrum	100		100	
Predictor Coefficient	93.05		98.60	
Reflection Coefficient	91.66		98.61	
Log Area Ratio	98.6		100	

Table 6. Recognition Accuracy (%) Perturbations to Feature Vectors

12th order	EG	HLN	EG	HLN
	.05	25	.01	25
Cepstrum	93.05		95.83	
Predictor Coefficient	88.88		97.22	
Reflection Coefficient	86.11		94.44	
Log Area Ratio	93.05		98.61	

Table 7. Decrease in Performance (%) due to Feature Vector Perturbation

12th order	EG	HLN	EG	HLN
	.05	25	.01	25
Cepstrum	6.95		4.17	
Predictor Coefficient	4.48		1.39	
Reflection Coefficient	6.05		4.22	
Log Area Ratio	5.62		1.39	

Since a good portion of the spectrum information was contained in a 12th order Linear Predictive Analysis, a dimension of 12 was used to train a stronger classifier. Using mean square error goals of .05 and .01, the cepstrum had the best performance on the test data when no perturbations were introduced to the cepstrum vector. It classified all of the test data correctly. The log area ratio was a close second and the reflection and predictor coefficients were quite similar.

When perturbations were introduced to the feature vectors, an expected drop in the performance was observed. This is reasonable because the noise perturbation could shift a vector across a decision boundary and result in a misclassification. The additive noise did not have a tremendous effect on the outcome of the classifier, however, it only had a maximum performance decrease of 6.95%. Since the distributions were reasonably well separated in the feature space, the

additive noise did not have a significant impact on the classification performance. This would have had a larger impact on a more difficult classification problem.

2.3 Performance Testing on Noisy Speech

The next experiment was to train the classifier on clean speech but test on noisy speech. The noisy speech was obtained by corrupting the original clean speech signal with random Gaussian noise. Linear prediction on the noisy signals was computed to get the twelve dimensional feature vectors. The Signal to Noise Ratio (SNR) was varied from 0 to 30 dB. This examines the effect noise had on the performance given a mismatch between training and testing. A MLP with 30 hidden layer nodes and error goal of 0.01 was used. Table 8 gives the results.

Table 8. Recognition Accuracy (%) when Test Speech is Corrupted with Noise of Varying SNR (MLP Error Goal of 0.01 Used)

SNR	d	Refl	Cep	LAR
30dB	100	100	100	100
20dB	95.83	100	100	100
10dB	68.05	90.28	100	97.22
5dB	51.39	79.16	86.11	84.72
0dB	45.83	54.17	77.78	72.22

The increase in performance for a SNR of 30 dB is practical. The reason is that the small amount of noise shifted the newly created feature vectors closer to the ones that were used in training. As the SNR decreased a noticeable and expected decrease in performance was observed. The log area ratio and the cepstrum are less sensitive and hence, more robust to the noise. The reflection and filter coefficients are more affected and hence, less robust to noise. Overall, the cepstrum performs the best.

3 Fusion Experiments

The combination of different sources of information has been explored within fields known as data fusion, consensus building, team decision theory, combination of multiple experts, along with numerous other titles [9]. Here, we will refer to the combination of information from various sources as data fusion or merely fusion. Fusion has been used in the contexts of both speaker recognition [10, 11] and handwriting character recognition [12, 13]. Fusion has also been applied to augment performance for vehicle reidentification and surveillance in intelligent transportation systems [14].

In the context of this paper, fusion comprises the combination of scores from different neural network classifiers that each trained with a particular feature. We use the twelve dimensional cepstrum. For fusion to be beneficial, it is desired that the errors of one classifier are corrected by the others. If all classifiers are in agreement upon an error, then no combination will rectify the error. However, as long as there is some degree of uncorrelation among the errors, performance can be improved with the proper combination. It is these various combinations that are studied in this paper.

Three classifier fusion experiments were performed. Training is done on clean speech. Noise is added to the test speech, and the SNR is varied in the same way as previously stated. The cepstrum feature vector is chosen for its robustness when noise is added to the signal. The outputs of a MLP, RBF, and PNN were fused together. The fusion methods considered were linear fusion, majority voting, and weighted majority voting. Since the RBF and the PNN were not used in the previous experiments, their performances on the twelve dimensional cepstrum vector alone are shown in Table 9. The MLP is also shown for comparison purposes. The RBF is trained with an error goal and spread constant value of 1. The PNN is trained with a spread constant of .3.

Table 9. Recognition Accuracy (%) on Test Speech for Different Neural Network Structures with Cepstrum of Dimension 12 as the Feature

SNR	MLP	RBF	PNN
Clean signal	100	100	100
30dB	100	100	100
20dB	100	100	100
10dB	100	100	100
5dB	86.11	91.67	93.05
0dB	77.78	80.55	68.06

3.1 Linear Fusion

The outputs of each network provide a value, between zero and one, for each vowel class. Values closer to one reflect the choice of a particular class, while values closer to zero represent the incorrect class. The decision of the network is computed as follows :

$$y_i = \arg\max[g_i(x)] \qquad (12)$$

where $g_i(x)$ represents the value at each output node of the network. The vowel corresponding to y_i is identified. The decision from the Linear Fusion (LF) of the networks is computed as follows

$$LF = \arg\max\left[\sum_{i=a}^{c}\sum_{j=1}^{n}\frac{g_{ij}(x)}{3}\right] \qquad (13)$$

where c is the number of classes and n is the number of neural networks (in our case 3). This method takes an equal portion of the decision making from each of the networks. The vowel corresponding to LF is identified. The results of this experiment are shown in Table 10.

Table 10. Recognition Accuracy (%) on Test Speech Using Linear Fusion with Equal Weights to Each Neural Network

SNR dB	Performance with Linear Fusion
Clean signal	100
30	100
20	100
10	100
5	93.06
0	68.06

The performance with linear fusion having equal weights to each neural network behaves similarly to the PNN network by itself. On this database, the PNN tended to output values closer to one than the other two networks. Since there is an equal bias to each network, the one that produces the value closest to one has more of an impact on the final decision using linear fusion.

In order to analyze the effect of varying weights on the performance, equation (13) is written as follows in a more general form

$$LF = \arg\max\left[\sum_{i=a}^{c}\sum_{j=1}^{n} w_j\, g_{ij}(x)\right] \qquad (14)$$

The weights w_j add up to 1 and are all between 0 and 1. For the results of Table 10, each weight is equal to 1/3.

Table 11. Recognition Accuracy (%) on Test Speech Using Linear Fusion with Five Best Weights to Each Neural Network Obtained from an Exhaustive Search. The weight sequence corresponds to the MLP weight, RBF weight and PNN weight

Weights	Clean	30dB	20dB	10dB	5dB	0dB
[.3 .6 .1]	100	100	100	100	100	77.78
[.4 .5 .1]	100	100	100	100	98.11	80.56
[.5 .4 .1]	100	100	100	100	97.22	80.56
[.6 .2 .2]	100	100	100	100	97.22	77.78
[.7 .1 .2]	100	100	100	100	98.61	77.78

An exhaustive search of all possible weight triplets was performed to find the triplet(s) that maximize recognition accuracy when training is done on clean speech and testing is done on noisy speech. Table 11 gives the results for the best weight triplets. For clean speech and speech corrupted by noise of 30 dB, 20 dB and 10 dB SNR, many weight combinations achieved a 100 percent accuracy. To pick one combination that does well for all test conditions, the [0.4,0.5,0.1] triplet stands out. Note the relatively low weight given to the PNN classifier. Also, note the superiority to merely applying equal weights to all classifiers.

3.2 Majority Voting

The final decision is determined by finding the class that that received the highest vote. This method tries to deliver an output that represents the majority of the decisions of the individual networks. The results of this experiment are shown in Table 12.

Table 12. Recognition Accuracy (%) on Test Speech Using Majority Voting

SNR	Performance with Majority Voting
Clean signal	100
30dB	100
20dB	100
10dB	100
5dB	95.83
0dB	80.56

The performances fell below 100% only when the SNR dropped to 5 dB. A better performance is produced using the majority voting, rather than using only an individual neural network's output. The performance at 0 dB was similar to the performance of RBF by itself. This is still a good characteristic because it performed at the same level as the best network for this particular SNR. This method performs only slightly worse than linear fusion with the weight triplet [0.4,0.5,0.1].

3.3 Weighted Majority Voting

The weighted majority scheme of decision-making is more biased toward the classifier that performed the best on its training data [15]. The weights, denoted as β, were computed as follows :

$$\beta = 1 - RA \qquad (15)$$

where RA is the recognition accuracy. β is a value between 0 and 1. This will assign lower values of β (and hence, a higher weight) to the networks that perform well, while assigning higher values of β (and hence, a lower weight) to the networks that perform poorly. The weighted majority vote (WMV) is computed as follows [15] :

$$WMV(x) = \arg\max_{y \in Y} \sum_{k=1}^{c} \sum_{t:g_t(x)=y} \log \frac{1}{\beta_t} \quad (16)$$

Smaller values of β will create larger values using Equation (16). Hence, the stronger networks have small β values and have more impact on the final decision. The results for this experiment are shown in Table 13.

Table 13. Recognition Accuracy (%) on Test Speech Using Weighted Majority Voting

SNR dB	Performance with Weighted Majority Voting
Original signal	100
30	100
20	100
10	100
5	95.83
0	80.56

The weighted majority voting method produced the same exact results as the majority voting method. This is because there are only a few networks whose outputs are fused together. The purpose of this method is to to allow classifiers who perform well to have more of an impact on the final decision. It is unlikely that a single network overrules a majority vote because the performances of each network do not differ significantly. An alternative method to using three strong classifiers is to use an ensemble of weak classifiers and perform the same weighted majority vote.

4 Conclusions

In this paper, isolated vowel recognition was studied with 3 different neural network classifiers, namely, the MLP, RBF and the PNN. Various linear predictive features were examined (direct form predictor coefficients, reflection coefficients, log-area ratios and the cepstrum). Each linear predictive (LP) feature shows a different vowel recognition performance. Also, varying the feature vector dimension affects the performance of the classifier. A too small or too large dimension does result in poorer results. The LP cepstrum appears to perform the best in classifying the vowels correctly. This is especially true when training is done on clean speech and testing is done on noisy speech. Three different classifier fusion strategies (linear fusion, majority voting and weighted majority voting) were found to improve the performance. Linear fusion with varying weights is the method of choice and is robust to noise. This is in accordance with the result in [16] that states that linear fusion performs at least as well as the best single classifier. The future applications for this experiment would be in isolated words recognition.

References

[1] Rabiner, Schafer, R. W., "Digital Processing of Speech Signals", Prentice-Hall, 1978.

[2] Badran, E.F.M.F.; Selim, H.," Speaker Recognition Using Artificial Neural Networks Based on Vowel Phonemes," Signal Processing Proceedings, 2000.

[3] Heck, L. P., ,Konig, Y., Sönmez, M. K., Weintraub, M. "Robustness to telephone handset distortion in speaker recognition by discriminative feature design," Speech Communication, Volume 31, Issues 2-3, June 2000, Pages 181-192.

[4] Lim, C.P.; Woo, S.C.; Loh, A.S.; Osman, R. ," Speech Recognition Using Artificial Neural Networks," Web Information Systems Engineering, 2000.

[5] Lippmann, Richard P., "An Introduction to Computing with Neural Nets," IEEE ASSP Magazine, April 1987.

[6] Hush, Don R.; Horne, Bill G., "Progress in Supervised Neural Networks," IEEE Signal Processing Magazine, Jan 1993.

[7] Duda, R. O.; Hart, P. E. ; Stork, D. G., "Pattern Classification," John Wiley & Sons Inc., 2001.

[8] Haykins, Simon, "Neural Networks: A Comprehensive Foundation," Prentice Hall, 1999.

[9] Ramachandran. R. P.; Farrell, K. R.; Ramachandran, R.; Mammone, R. J. "Speaker Recognition – General Classifier Approaches and Data Fusion Methods," Pattern Recognition, to appear.

[10] Farrell, K. R. Ramachandran. R. P.; Mammone, R. J. "An Analysis of Data Fusion Methods for Speaker Verification," IEEE International Conference on Acoustics, Speech and Signal Processing, Seattle, Washington, pp. 1129-1132, May 1998.

[11] Farrell, K. R. "Model Combination and Weight Selection for Speaker Verification," IEEE Workshop on Neural Networks for Signal Processing, Madison, Wisconsin, August 1999.

[12] Xu, L.; Krzyzak, A.; Suen, C. Y. ,"Methods of combining multiple classifiers and their applications to hand-written character recognition", IEEE Trans. on Systems, Man and Cybernetics, vol. 23, pp. 418--435, 1992.

[13] Ho T. K.; Hull, J. J.; Srihari, S. N., ``Decision combination in multiple classifier systems" IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 16, pp. 66--75, 1994.

[14] Ramachandran, R. P.; Arr, G.; Sun, C.; Ritchie, S. G., "A Pattern Recognition and Feature Fusion Formulation for Vehicle Reidentification in Intelligent Transportation Systems", IEEE International Conference on Acoustics, Speech and Signal Processing, to appear.

[15] Littlestone, N.; Warmuth M., "Weighted Majority Algorithm," Information and Computation, vol 108, pp. 212-261, 1994.

[16] Rao N. S. V.; ``On fusers that perform better than best sensor", IEEE Trans. on Pattern Analysis and Machine Intelligence, vol. 23, pp. 904--909, August 2001.