A Particle Filter Compensation Approach to Robust Speech Recognition

Aleem Mushtaq

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/51532

1. Introduction

The speech production mechanism goes through various stages. First, a thought is generated in speakers mind. The thought is put into a sequence of words. These words are converted into a speech signal using various muscles including face muscles, chest muscles, tongue etc. This signal is distorted by environmental factors such as background noise, reverberations, channel distortions when sent through a microphone, telephone channel etc. The aim of Automatic Speech Recognition Systems (ASR) is to reconstruct the spoken words from the speech signal. From information theoretic [1] perspective, we can treat what is between the speaker and machine as a distortion channel as shown in figure 1.

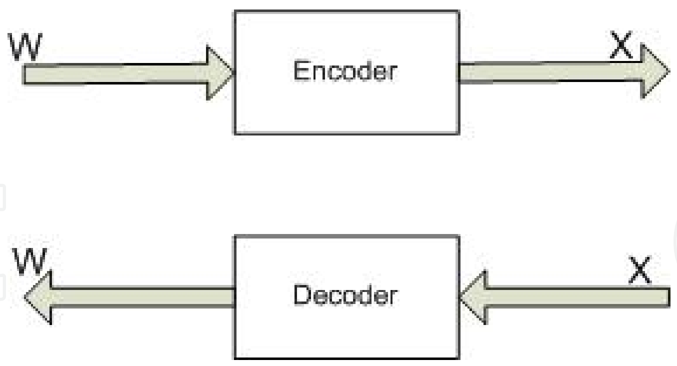


Figure 1. Information theoretic view of Speech Recognition

Here, W represent the spoken words and X is the speech signal. The problem of extracting W from X can be viewed as finding the words sequence that most likely resulted in the observed signal X as given in equation (1)

$$\hat{W} = \arg\max_{W} p(X \mid W) \tag{1}$$

Like any other Machine Learning/Pattern Recognition problem, the posterior $p(X|W)$ plays a fundamental role in the decoding process. This distribution is parametric and its parameters are found from the available training data. Modern ASR systems do well when environment of speech signal being tested matches well with that of the training data. This is so because the parameter values correspond well to the speech signal being decoded. However, if the environments of training and testing data do not match well, the performance of the ASR systems degrade. Many schemes have been proposed to overcome this problem but humans still outperform these systems, especially in adverse conditions.

The approaches to overcome this problem falls under two categories. One way is to adapt the parameters of $p(X|W)$ such that they match better with the testing environment and the other is to choose features X such that they are more robust to environment variations. The features can also be transformed to make them more suited to the parameters of $(X|W)$, obtained from training data.

1.1. Typical ASR system

Typical ASR systems for small vocabulary are comprised of three main components as shown in figure 2. Speech data is available in waveform which is first converted into feature vectors. Mel Frequency Cepstrum Coefficients (MFCC) [2] features have been widely used in speech community for the task of speech recognition due to their superior discriminative capability.

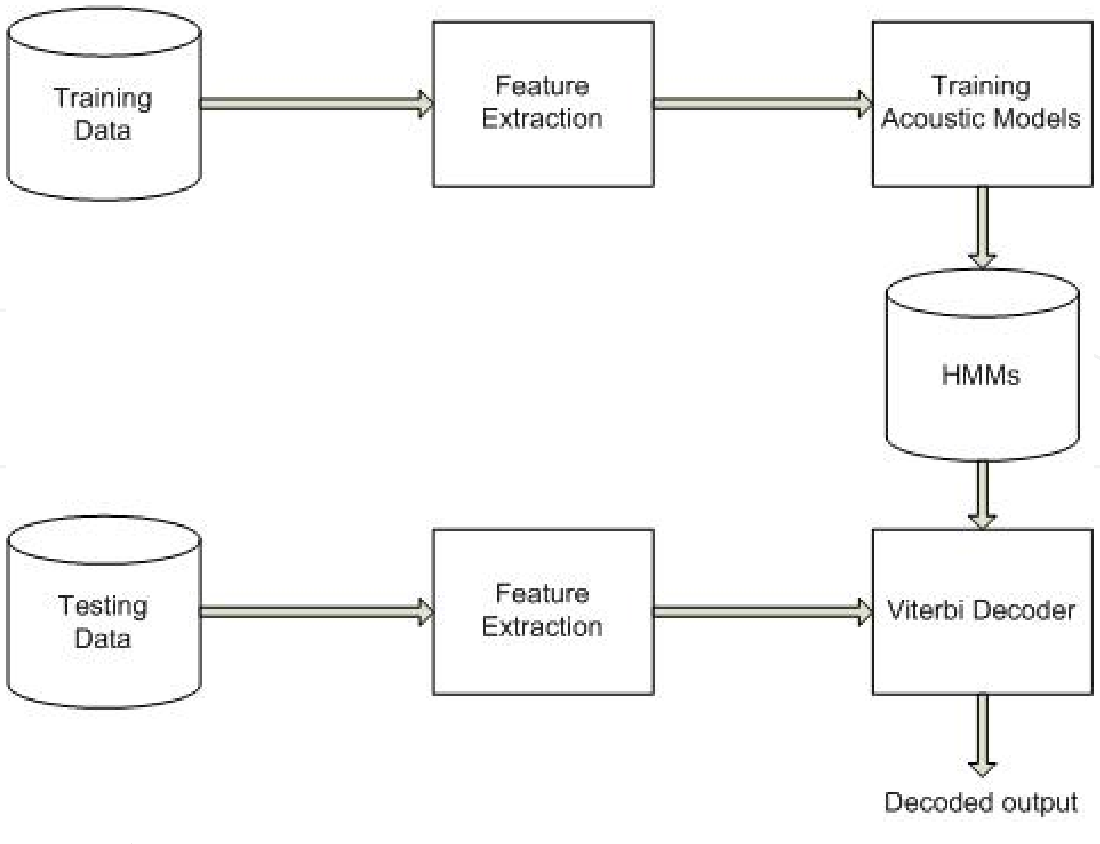


Figure 2. Typical ASR System

The features from an available training speech corpus are used to estimate the parameters of Acoustic Models. An acoustic model for a particular speech unit, say a phoneme or a word is the likelihood of observing that unit based on the features as given in equation 1.1. Most commonly used structure for the acoustic models in ASR systems is the Hidden Markov Models (HMM). These models capture the dynamics and variations of speech signal well. The test speech signal is then decoded using Viterbi Decoder.

1.2. Distortions in speech

The distortions in speech signal can be viewed in signal space, feature space and the model space [3] as shown in figure 3. Resilience to environmental distortions can be added in the feature extraction process, by modifying the distorted features or adapting the acoustic models to better match the environment from which test signal has emanated. S_X and $F_X$represent speech signal and speech feature respectively. M_X represent the acoustic models.

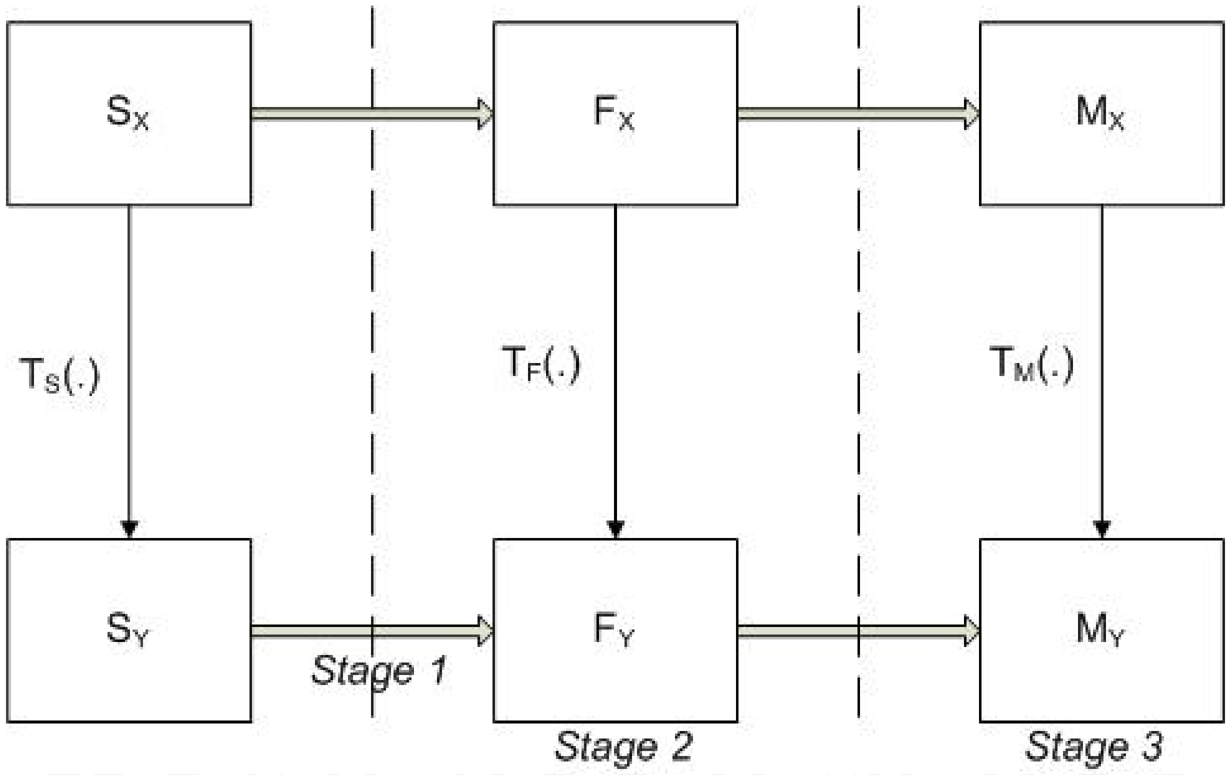


Figure 3. Stages where noise robustness can be added

In stage 1, the feature extraction process is improved so that the features are robust to distortions. In stage 2, the features are modified to match them better with the training environment. The mismatch in this stage is usually modeled by nuisance parameters. These are estimated from the environment and test data and their effect is minimized based on some optimality criteria. In stage 3, the acoustic models are improved to match better with the testing environment. One way to achieve this is to use Multi-Condition training i.e. use data from diverse environments to train the models. Another way is find transform the models where transformation matrix is obtained from the test environment.

1.3. Speech and noise tracking for noise compensation

A sequential Monte Carlo feature compensation algorithm was initially proposed [4-5] in which the noise was treated as a state variable while speech was considered as the signal corrupting the observation noise and a VTS approximation was used to approximate the clean speech signal by applying a minimum mean square error (MMSE) procedure. In [5] extended Kalman filters were used to model a dynamical system representing the noise which was further improved by using Polyak averaging and feedback with a switching dynamical system [6]. These were initial attempts to incorporate particle filter for speech recognition in more indirect fashion as it was used for tracking of noise instead of the speech signal itself. Since the speech signal is treated as corrupting signal to the noise, limited or no information readily available from the HMMs or the recognition process can be utilized efficiently in the compensation process.

Particle filters are powerful numerical mechanisms for sequential signal modeling and is not constrained by the conventional linearity and Gaussianity [7] requirements. It is a generalization of the Kalman filter [8] and is more flexible than the extended Kalman filter [9] because the stage-by-stage linearization of the state space model in Kalman filter is no longer required [7]. One difficulty of using particle filters lies in obtaining a state space model for speech as consecutive speech features are usually highly correlated. Just like in the Kalman filter and HMM frameworks, state transition is an integral part of the particle filter algorithms.

In contrast to the previous particle filter attempts [4-6] we describe a method in this chapter where we treat the speech signal as the state variable and the noise as the corrupting signal and attempt to estimate clean speech from noisy speech. We incorporate statistical information available in the acoustic models of clean speech, e.g., the HMMs trained with clean speech, as an alternative state transition model[10-11]. The similarity between HMMs and particles filters can be seen from the fact that an observation probability density function corresponding to each state of an HMM describes, in statistical terms, the characteristics of the source generating a signal of interest if the source is in that particular state, whereas in particle filters we try to estimate the probability distribution of the state the system is in when it generates the observed signal of interest. Particle filters are suited for feature compensation because the probability density of the state can be updated dynamically on a sample-by-sample basis. On the other hand, state densities of the HMMs are assumed independent of each other. Although they are good for speech inference problems, HMMs do not adapt well in fast changing environments.

By establishing a close interaction of the particle filters and HMMs, the potentials of both models can be harnessed in a joint framework to perform feature compensation for robust speech recognition. We improve the recognition accuracy through compensation of noisy speech, and we enhance the compensation process by utilizing information in the HMM state transition and mixture component sequences obtained in the recognition process. When state sequence information is available we found we can attain a 67% digit error reduction from multi-condition training in the Aurora-2 connected digit recognition task. If the missing parameters are estimated in the operational situations we only observe a 13% error reduction in the current study. Moreover, by tracking the speech features, compensation can be done using only partial information about noise and consequently good recognition performance can be obtained despite potential distortion caused by non-stationary noise within an utterance.

The remainder of the chapter is organized as follows. In section 2, a tracking scheme in general is described followed by the explanation of the well known Kalman filter tracking algorithm. Particle Filters, which form the backbone of PFC are also described in this section. In section 3, the steps involved in tracking and then extracting the clean speech signal from the noisy speech signal are laid out. We also discuss various methods to obtain information required to couple the particle filters and the HMMs in a joint framework. Finally, the experimental results and performance comparison for PFC is given before drawing the conclusions in section 4.

2. Tracking algorithms

Tracking is the problem of estimating the trajectory of an object in a space as it moves through that space. The space could be an image plane captured directly from a camera or it could be synthetically generated from a radar sweep. Generally, tracking schemes can be applied to any system that can be represented by a time dynamical system which consists of a state space model and an observation

$$\begin{aligned} x_t &= f(x_{t-1}, w_t) \\ y_t &= h(x_t, n_t) \end{aligned} \tag{2}$$

Where n_t is the observation noise and w_t is called the process noise and represents the model uncertainties in the state transition function $f(.)$. What is available is an observation y_t which is function of x_t.We are interested in finding a good estimate of current state given observations till current time t i.e. $p(x_t|y_t,y_{t-1},y_{t-2}, \dots\ y_0)$. The state space model $f(.)$ represents the relation between states adjacent in time. The model in equation (2) assumes that state sequence is one step Markov process

$$f(x_{t+1} \mid x_t, x_{t-1}, \dots x_0) = f(x_{t+1} \mid x_t) \tag{3}$$

It is further assumed that observations are independent of one another

$$f(y_{t+1} \mid x_{t+1}, y_t, \dots y_0) = f(y_{t+1} \mid x_{t+1}) \tag{4}$$

Tracking is a two step process. The first step is to obtain density x_t at time $t-1$. This is called the prior density of x_t. Once it is available, we can construct a posterior density upon availability of observation y_t. The propagation step is given in equation (5). The update step is obtained using Bayesian theory (equation (6)).

$$f(x_t \mid y_{t-1}, \dots, y_0) = \int f(x_t \mid x_{t-1}) f(x_{t-1} \mid y_{t-1}, \dots, y_0) dx_{t-1} \tag{5}$$

$$f(x_t \mid y_t, y_{t-1}, ..., y_0) = \frac{f(y_t \mid x_t, y_{t-1}, ..., y_0) f(x_t \mid y_{t-1}, ..., y_0)}{f(y_t \mid y_{t-1}, ..., y_0)} \tag{6}$$

2.1. Kalman filter as a recursive estimation tracking algorithm

Kalman Filter is the optimal recursive estimation solution for posterior density $p(x_{t+1}|y_t, \dots, y_0)$ if the time dynamical system is linear

$$\begin{aligned} x_{t+1} &= A_t x_t + w_t \\ y_t &= C_t x_t + n_t \end{aligned} \tag{7}$$

where A_t and C_t are known as state transition matrix and observation matrix respectively. Subscript t indicates that both can vary with time. Under the assumption that both process noise w_t and observation noise n_t are Gaussian with zero mean and covariance Q_t and R_t respectively, $p(x_{t+1}|x_t)$ can be readily obtained.

$$\begin{aligned} mean(x_{t+1} \mid x_t) &= E(A_t x_t + w_t) = A_t x_t \\ covariance(x_{t+1} \mid x_t) &= E(w_t w_t^T) = Q_t \end{aligned} \tag{8}$$

and therefore

$$p(x_{t+1} \mid x_t) \sim N(A_t x_t, Q_t) \tag{9}$$

To obtain the propagation step, we need $p(x_t|y_t, \dots, y_0)$ in addition to $(x_{t+1}|x_t)$. Since this is an iterative step, the estimate of x_t given observations up to time t is available at $t+1$ and let's call it $x_{t|t}$. Let covariance of $x_{t|t}$ be $P_{t|t}$. Then

$$p(x_t \mid y_t, y_{t-1}, ..., y_0) \sim N(\hat{x}_{t|t}, P_{t|t}) \tag{10}$$

where $P_{t|t}$ is the covariance of $x_t|y_t, \dots, y_0$ and is given by $E[(x_t - E[x_t])(x_t - E[x_t])^T|y_t, \dots, y_0]$. Now both components of the integral in equation (5) are available in equation (9) and (10). Solving the integral using expanding and completing the squares [12] we get

$$p(x_{t+1} \mid y_t, y_{t-1}, ..., y_0) \sim N(A_t \hat{x}_{t|t}, A_t P_{t|t} A_t^T + Q_t) \tag{11}$$

This is the propagation step and is sometimes is also written as

$$p(x_{t+1} \mid y_t, y_{t-1}, ..., y_0) \sim N(\hat{x}_{t+1|t}, P_{t+1|t}) \tag{12}$$

To get the update step, we note that the distributions of $x_{t+1}|y_t,\dots,y_0$ and y_{t+1} are both Gaussian. For two random variables say x and y that are jointly Gaussian, the distribution of one of them given the other for example $x|y$ is also Gaussian. Consequently, $x_{t+1}|y_{t+1}, y_t,\dots,y_0$ is a Gaussian distribution with following mean and variance

$$\hat{x}_{t+1} \mid x_{t+1} = E[x_{t+1} \mid y_{t+1}, y_t, ..., y_0] = \hat{x}_{t+1} \mid x_t + R_{xy} R_{yy}^{-1} (y_{t+1} - E[y_{t+1} \mid y_t, ..., y_0]) \tag{13}$$

where

$$\begin{aligned} R_{xy} &= E[(x_{t+1} - E[x_{t+1}])(y_{t+1} - E[y_{t+1}])^T \mid y_t, y_{t-1}, \ldots, y_0] \\ &= E[(x_{t+1} - \hat{x}_{t+1|t})(C_{t+1}(x_{t+1} - \hat{x}_{t+1|t}) + n_{t+1})^T \mid y_t, y_{t-1}, \ldots, y_0] \\ &= P_{t+1|t} C_{t+1}^T \end{aligned} \tag{14}$$

Similarly

$$R_{yy} = C_{t+1} P_{t+1|t} C_{t+1}^T + R_{t+1} \tag{15}$$

Back substituting equation (14) and equation (15) in equation (13), we get

$$\hat{x}_{t+1} \mid x_{t+1} = \hat{x}_{t+1|t} + K_{t+1}(y_{t+1} - C_{t+1}\hat{x}_{t+1|t}) \tag{16}$$

where $K$is called the Kalman gain and is given by

$$K_{t+1} = P_{t+1|t} C_{t+1}^T (C_{t+1} P_{t+1|t} C_{t+1}^T + R_{t+1})^{-1} \tag{17}$$

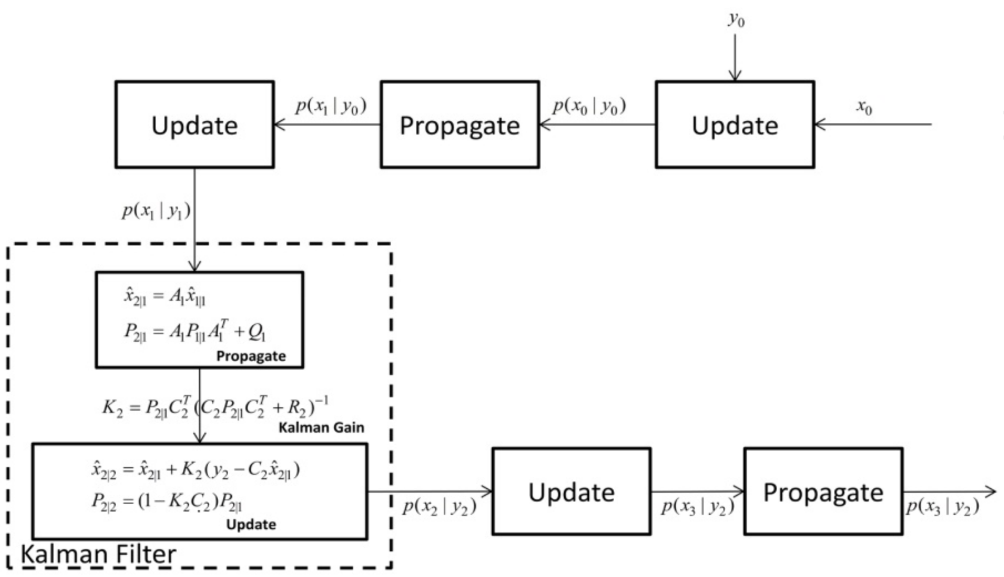
Covariance can also be obtained by referring to the fact that covariance of $x|y$, the two jointly Gaussian random variables, is given by

$$\text{cov}(X \mid Y) = R_{xx} - R_{xy} R_{yy}^{-1} R_{yx} \tag{18}$$

we thus obtain the covariance of estimate of $\hat{x}_{t+1|t+1}$ as follows

$$\begin{aligned} P_{t+1|t+1} &= P_{t+1|t} - P_{t+1|t} C_{t+1}^T (C_{t+1} P_{t+1|t} C_{t+1}^T + R_{t+1})^{-1} C_{t+1} P_{t+1|t} \\ &= (1 - K_{t+1} C_{t+1}^T) P_{t+1|t} \end{aligned} \tag{19}$$

The block diagram in Figure 4 below shows a general recursive estimation algorithm steps starting from some initial state estimate x_0. The block labeled Kalman filter summarizes the steps specific to Klaman filter algorithm.



y_0
x_0
Update
$p(x_0 \mid y_0)$
Propagate
$p(x_1 \mid y_0)$
Update
$p(x_1 \mid y_1)$
$\hat{x}_{2|1} = A_1 \hat{x}_{1|1}$
$P_{2|1} = A_1 P_{1|1} A_1^T + Q_1$
Propagate
$K_2 = P_{2|1} C_2^T (C_2 P_{2|1} C_2^T + R_2)^{-1}$
Kalman Gain
$\hat{x}_{2|2} = \hat{x}_{2|1} + K_2(y_2 - C_2 \hat{x}_{2|1})$
$P_{2|2} = (1 - K_2 C_2) P_{2|1}$
Update
Kalman Filter
$p(x_2 \mid y_2)$
Update
$p(x_3 \mid y_2)$
Propagate
$p(x_3 \mid y_2)$

Figure 4. Recursive Estimation Algorithm

2.2. Grid based methods

It is hard to obtain analytical solutions to most recursive estimation algorithms. If the state space for a problem is discrete, then we can use grid based methods and can still obtain the optimal solution. Considering that state x takes N_s possible values, we can represent discrete density $p(y|x)$ using N_s samples[7].

$$p(x_k \mid y_k, y_{k-1}, \ldots, y_0) = \sum_{i=1}^{N_s} w_{k|k} \delta(x_k - x_k^i) \tag{20}$$

where the weights are computed as follows

$$\begin{aligned} w_{k|k}^i &\sim \frac{1}{C} w_{k|k-1}^i p(y_k \mid x_k^i) \\ w_{k|k-1}^i &\sim \sum_{j=1}^{N_s} w_{k-1|k-1}^i p(x_k^i \mid x_{k-1}^j) \end{aligned} \tag{21}$$

Here C is the normalizing constant to make total probability equal one. The assumption that state can be represented by finite number of points gives us the ability to sample the whole state space. The weight w_k^i represents the probability of being in state x_k^i when observation at time k is y_k. In grid based method we construct the discrete density at every time instant in two steps. First we estimate the weights at k without the current observation $w_{k|k-1}^i$ and then update them when observation is available and obtain $w_{k|k}^i$. In the propagation step we take into account probabilities (weights) for all possible state values at $k-1$ to estimate the weights at time k as shown in figure 5.

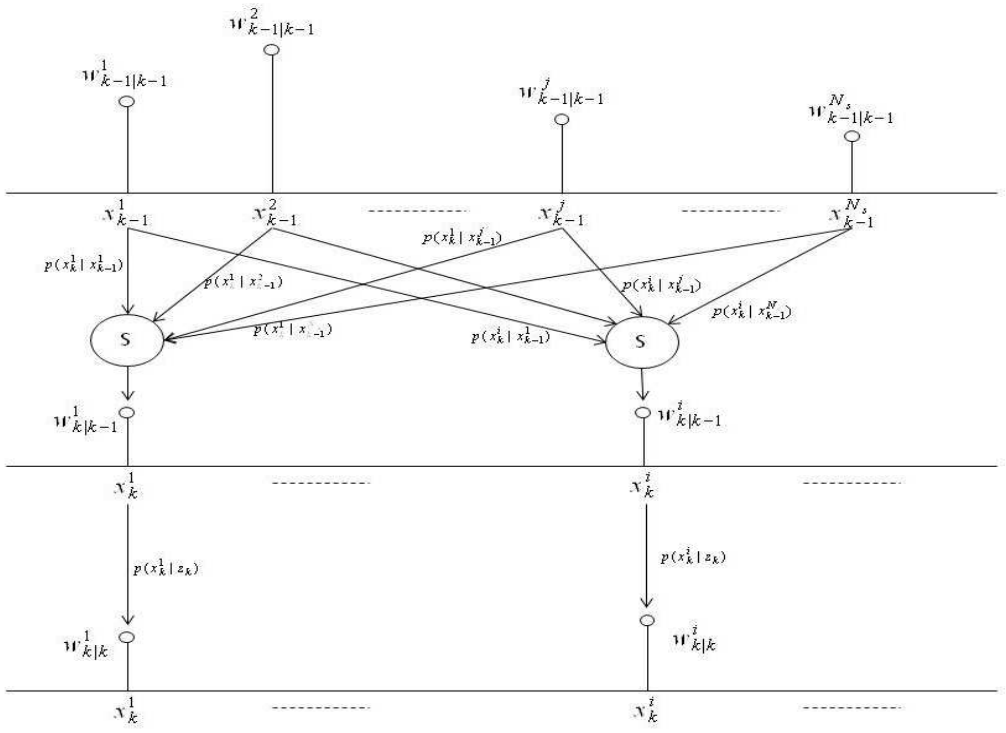


Figure 5. Grid based method

If the prior $p(x_k^i|x_k^j)$ and the observation probability $p(z_k|x_k)$ are available, the grid based method gives us the optimal solution for tracking the state of the system. If the state of the system is not discrete, then we can obtain an approximate solution using this method. We divide the continuous space into say J cells and for each cell we compute the prior and posterior in a way that takes into account the range of the whole cell:

$$\begin{aligned} p(x_k^i \mid x_{k-1}^j) &= \int_{x \in x_k^i} p(x \mid \bar{x}_{k-1}^j) dx \\ p(y_k \mid x_k^i) &= \int_{x \in x_k^i} p(y_k \mid x) dx \end{aligned} \tag{22}$$

where $\bar{x}_k$ is the center of jth cell at time $k-1$. The weight update in equation (21) subsequently remains unchanged.

2.3. Particle filter method

Particle filtering is a way to model signals emanating from a dynamical system. If the underlying state transition is known and the relationship between the system state and the observed output is available, then the system state can be found using Monte Carlo simulations [13]. Consider the discrete time Markov process such that

$$\begin{aligned} &X_1 \sim \mu(x_1) \\ &X_t \mid X_{t-1} = x_t \sim p(x_t \mid x_{t-1}) \\ &Y_t \mid X_t = x_t \sim p(y_t \mid x_t) \end{aligned} \tag{23}$$

We are interested in obtaining $\mathrm{p}(\mathrm{x_t}|\mathrm{y_t},\dots,\mathrm{y_0})$ so that we have a filtered estimate of $\mathrm{x_t}$ from the measurements available so far, $\mathrm{y_t},\dots,\mathrm{y_0}$. If the state space model for the process is available, and both the state and the observation equations are linear, then Kalman filter described above can be used to determine the optimal estimate of $\mathrm{x_t}$ given observations $\mathrm{y_t},\dots,\mathrm{y_0}$. This is so under the condition that process and observation noises are white Gaussian noise with zero mean and mutually independent. In case the state and observation equations are nonlinear, the Extended Kalman Filter (EKF) [9], which is a modified form of the Kalman Filter can be used. Particle filter algorithm estimates the state's posterior density, $\mathrm{p}(\mathrm{x_t}|\mathrm{y_t},\dots,\mathrm{y_0})$ represented by a finite set of support points [7]:

$$p(x_t \mid y_t, y_{t-1}, \dots, y_1) = \sum_{i=1}^{N_s} w_t^i \delta(x_t - x_t^i) \tag{24}$$

where x_t^i for $i = 1, \dots, N_s$are the support points and w_t^i are the associated weights. We thus have a discretized and weighted approximation of the posterior density without the need of an analytical solution. Note the similarities with Grid based method. In that, support points for discrete distribution were predefined and covered the whole space. In particle filter algorithm, the support points are determined based on the concept of importance sampling in which instead of drawing from $p(.)$, we draw points from another distribution q(.) and compute the weights using the following:

$$w^i = \frac{\pi(x^i)}{q(x^i)} \tag{25}$$

where $\pi(.)$ is the distribution of $p(.)$and $q(.)$is an importance density from which we can draw samples. For the sequential case, the weight update equation can be computed one by one,

$$w_t^i = w_{t-1}^i \frac{p(y_t \mid x_t^i) p(x_t^i \mid x_{t-1}^i)}{q(x_t^i \mid x_{t-1}^i, y_t)} \tag{26}$$

The density $q(.)$ propagates the samples to new positions at t given samples at time $t-1$ and is derived from the state transition model of the system.

3. Tracking algorithms for noise compensation

State transition information is an integral part of the particle filter algorithm and is used to propagate the particle samples through time transitions of the signal being processed. Specifically, the state transition is important to be able to position the samples at the right locations. To solve this problem, statistics from HMMs can be used. Although we only have discrete states in HMMs, each state is characterized by a continuous density Gaussian mixture model (GMM) and therefore it enables us to capture part of the variation in speech features to generate particle samples for feature compensation. Using particle filter algorithms with side information about the statistics of clean speech available in the clean HMMs we can perform feature compensation. If the clean speech is corrupted by an additive noise, n, and a distortion channel, h, then we can represent the noise corrupted speech with an additive noise model [14], assuming known statistics of the noise parameters,

$$y = x + h + \log(1 + \exp(n - x - h)) \tag{27}$$

where $y = \log(S_y(m_p))$, $x = \log(S_x(m_p))$ and $h = \log(|H(m_p)|^2)$ and $S(m_p)$ denotes the p^{th} mel spectrum.

$$S_y(m_p) = S_x(m_p) \mid H(m_p) \mid^2 + S_N(m_p) \tag{28}$$

The additional side information needed for feature compensation is a set of nuisance parameters, Φ similar to *stochastic matching* [3], we can iteratively find Φ followed by decoding as shown in Figure 6:

$$\Phi' = \arg\max_{\Phi} P(Y' \mid \Phi, \Lambda) \tag{29}$$

where Y' is the noisy or compensated utterance.

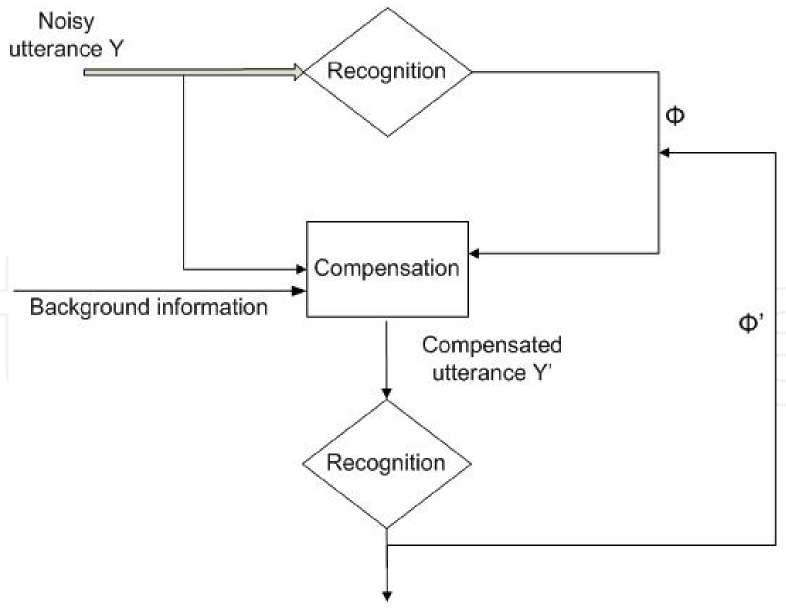


Figure 6. General feature compensation scheme

The clean HMMs and the background noise information enable us to generate appropriate samples from $q(.)$ in equation (26). The parameters Φ in equation (30) in our particle filter compensation (PFC) implementation, correspond to the corresponding correct HMM state sequence and mixture component sequence. These sequences provide critical information for density approximation in PFC. As shown in Figure 6 this can be done in two stages. We first perform a front-end compensation of noisy speech. Then recognition is done in the second stage to generate the side information Φ so as to improve compensation. This process can be iterated similar to what's done in maximum likelihood stochastic matching [3]. During compensation, the observed speech y is mapped to clean speech features x. For this purpose clean speech alone cannot be represented by a finite set of points and therefore HMMs by themselves cannot be used directly for tracking of x. Now if an HMM λ_m is available that adequately represents the speech segment under consideration for compensation along with an estimated state sequence $s_1, s_2, \ldots, s_T$ that correspond to T feature vectors to be considered in the segment, then we can generate the samples from the i^{th} sample according to

$$p(x_t \mid x_{t-1}^i) \sim \sum_{k=1}^{K} c_{k,s_t} N(\mu_{k,s_t}, \Sigma_{k,s_t}) \tag{30}$$

where $N(\mu_{k,s_t}, \Sigma_{k,s_t})$ is the k^{th} Gaussian mixture for the state s_t in λ_m and c_{k,s_t} is its corresponding weight for the mixture. The total number of particles is fixed and the contribution from each mixture, computed at run time, depends on its weight. We have chosen the importance sampling density, $q(x_t|x_{t-1}^i, y_t)$ in equation (26) to be $p(x_t|x_{t-1}^i)$ in equation (31). This is known as the sampling importance resampling (SIR) filter [7]. It is one of the simplest implementation of particle filters and it enables the generation of samples independently from the observation. For the SIR filter, we only need to know the state and the observation equations and should be able to sample from the prior as in Eq. (3). Also, the resampling step is applied at every stage and the weight assigned to the i-th support point of the distribution of the speech signal at time t is updated as:

$$w_t^i \propto p(y_t \mid x_t^i) \tag{31}$$

The procedure for obtaining HMMs and the state sequence will be described in detail later. To obtain $p(y_t|x_t^i)$, the distribution of the log spectra of noise for each channel is assumed Gaussian with mean μ_n and variance σ_n^2. Assuming there is additive noise only with no channel effects

$$y = x + \log(1 + e^{n-x}) \tag{32}$$

We are interested in evaluating $p(y|x)$ where x represents clean speech and n is the noise with density $N(\mu_n, \sigma_n)$. Then

$$\begin{aligned} p[Y < y \mid x] &= p[x + \log(1 + e^{N-x}) < y \mid x] \\ p(y \mid x) &= F'(u) = p(u)\frac{e^{y-x}}{e^{y-x} - 1} \end{aligned} \tag{33}$$

Where $F(\mu)$ is the Gaussian cumulative density function with mean μ_n and variance σ_n^2 and $u = \log(e^{y-x} - 1) + x$. In the case of MFCC features, the nonlinear transformation is [14]

$$y = x + D\log(1 + e^{D^{-1}(n-x)}) \tag{34}$$

Consequently,

$$p(y \mid x) = p_N(g^{-1}(y))J_{g^{-1}}(y) \tag{35}$$

where $P_N(.)$ is a Gaussian pdf, $J_{g^{-1}}(y)$ is the corresponding Jacobian and D is a discrete cosine transform matrix which is not square and thus not invertible. To overcome this problem, we zero-pad the y and x vectors and extend D to be a square matrix. The variance of the noise density is obtained from the available noise samples. Once the point density of the clean speech features is available, we estimate of the compensated features using discrete approximation of the expectation as

$$x_t = \sum_{i=1}^{N_s} w_t^i x_t^i \tag{36}$$

where N_s is the total number of particle samples at time t.

3.1. Estimation of HMM side information

As described above, it is important to obtain $\Phi \in \{\lambda_m, S\}$ where λ_m is an HMM that faithfully represents the speech segment being compensated and $S = s_1, s_2, \dots, s_T$ is the state sequence corresponding to the utterance of length T. To obtain λ_m for the m^{th} word W_m in the utterance, we chose the N-best models $\lambda_{m_1}, \lambda_{m_2}, \dots, \lambda_{m_N}$ from HMMs trained using 'clean speech data'. The N models are combined together to obtain a single model λ_m as follows.

3.1.1. Gaussian Mixtures Estimation

To obtain the observation model for each state j of model λ_m, we concatenate mixtures from the corresponding states of all component models,

$$\hat{b}_j^{(m)}(o) = \sum_{l=1}^{L} \sum_{k=1}^{K} c_{k,j}^{(m_l)} N(\mu_{k,j}^{(m_l)}, \Sigma_{k,j}^{(m_l)}) \tag{37}$$

where K is the number of Gaussian mixtures in each original HMM and L is the number of different words $m_1, m_2, \dots, m_L$ in the N-best hypothesis. $\mu_{k,j}^{m_l}$ and $\Sigma_{k,j}^{m_l}$ are mean and covariance from the k-th mixture in the j-th state of model m_l. The mixture weights are normalized by scaling them according to the likelihood of the occurrence of the model, from which they come from,

$$c_{k,j}^{(m_l)} = c_{k,j}^{(m_l)} \times p(W_m = \lambda_{m_l}) \tag{38}$$

The mixture weight is an important parameter because it determines the number of samples that will be generated from the corresponding mixture. The state transition coefficients for $\lambda_m$are computed using the following:

$$\begin{aligned} \hat{a}_{ij}^{(m)} &= \sum_{l=1}^{L} p[s_t^{(m_l)} = i, s_{t-1}^{(m_l)} = j \mid W_m = \lambda_{m_l}] p[W_m = \lambda_{m_l}] \\ \hat{a}_{ij}^{(m)} &= \sum_{l=1}^{L} [a_{ij}^{(m_l)} \mid W_m = \lambda_{m_l}] p[W_m = \lambda_{m_l}] \end{aligned} \tag{39}$$

3.1.2. State sequence estimation

The recognition performance can be greatly improved if a good estimate of the HMM state sequence S is available. But obtaining this sequence in a noisy operational environment in ASR is very challenging. The simplest approach is to use the decoded state sequence obtained with multi-condition trained models in an ASR recognition process as shown in the bottom of Figure 6. However, these states could often correspond to incorrect models and deviate significantly from the optimal one. Alternatively, we can determine the states (to generate samples from) sequentially during compensation. For left-to-right HMMs, given the state s_{t-1} at time $t-1$, we chose s_t using equation (41) as follows:

$$\begin{aligned} s_t &\sim a_{s_t, s_{t-1}} \\ s_t &= \arg\max_i (a_{ij}) \end{aligned} \tag{40}$$

where a comes from the state transition matrix for λ_m. The mixture indices are subsequently selected from amongst the mixtures corresponding to the chosen state.

3.1.3. Experiments

To investigate the properties of the proposed approach, we first assume that a decent estimate of the state is available at each frame. Moreover, we assume that speech boundaries are marked and therefore the silence and speech sections of the utterance are known. To obtain this information, we use a set of digit HMMs (18 states, 3 Gaussian mixtures) that have been trained using clean speech represented by 23 channel mel-scale log spectral feature. The speech boundaries and state information for a particular noisy utterance is then captured through digit recognition performed on the corresponding clean speech utterance. The speech boundary information is critical because the noise statistics have to be estimated from the noisy section of the utterance. To get the HMM needed for particle filter compensation L models $\lambda_1, \lambda_2, \ldots, \lambda_L$ are selected based on the N-best hypothesis list. For our experiments, we set $L = 3$. We combine these models to get λ'_m for the m-th word in the utterance. Best results are obtained if the correct word model is present in the pool of models that contribute to λ'_m. Upon availability of this information, the compensation of the noisy log spectral features is done using the sequential importance sampling. To see the efficacy of the compensation process, we consider the noisy, clean and compensated filter banks (channel 8) for the whole utterances shown in Figure 7. The SNR for this particular case is 5 dB. It is clear that the compensated feature matches well with the clean feature. It should be noted however that such a good restoration of the clean speech signal from the noisy signal is achievable only when a good estimate of the side information about the state and mixture component sequences is available.

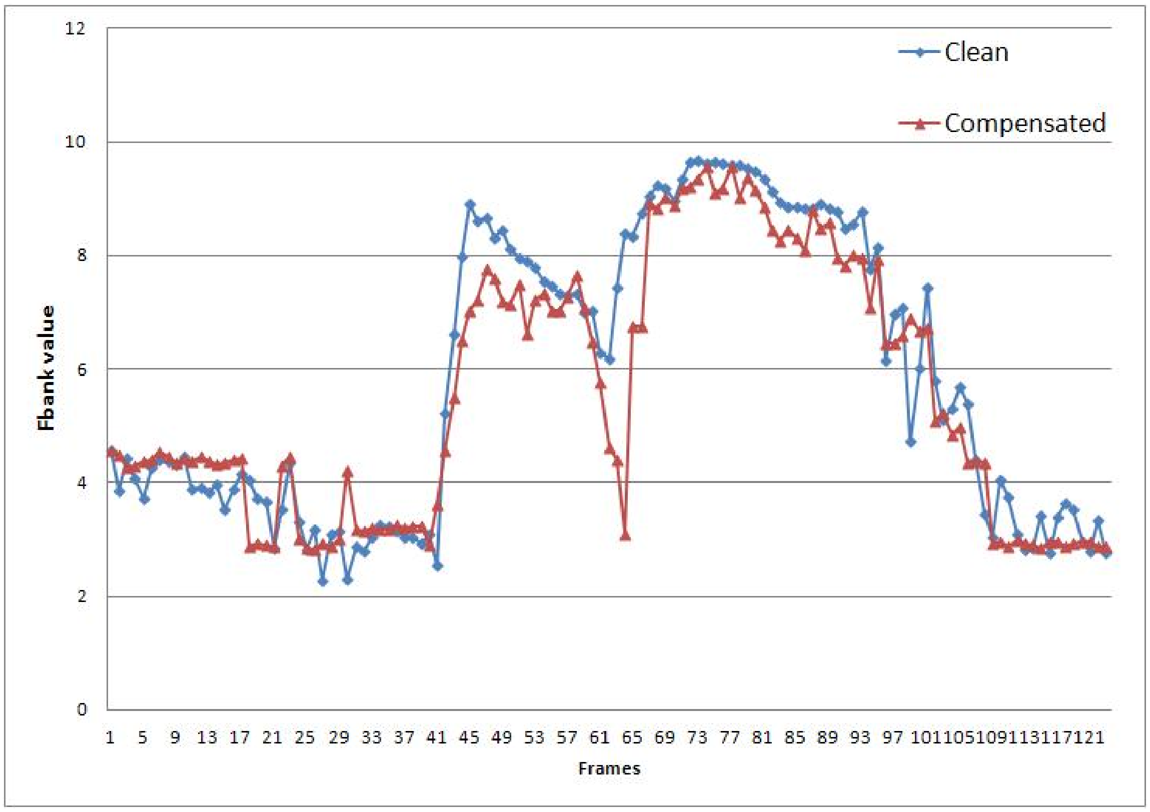


Figure 7. Fbank channel 8 corresponding underlying clean and compensated speech (SNR = 5 dB).

Assuming all such information were given (the ideal oracle case) recognition can be performed on MFCCs (39 MFCCs with 13 MFCCs and their first and second time derivatives) extracted from these compensated log spectral features. The HMMs used for recognition are trained with noisy data that has been compensated in the same way as the testing data. The performance compared to multi-condition (MC) and clean condition training (Columns 5 and 6 in Table 1) is given in Column 2 of Table 1 (Adapted Model I). It is clearly noted that a very significant 67% digit error reduction was attained if the missing information were made available to us.

Word Accuracy	Adapted Models I	Adapted Models II	Adapted Models III	MC Training	Clean Training
clean	99.10	99.10	99.10	98.50	99.11
20dB	97.75	96.46	97.38	97.66	97.21
15dB	97.61	95.98	96.47	96.95	92.36
10dB	96.66	94.00	94.40	95.16	75.14
5dB	95.20	90.64	88.02	89.14	42.42
0dB	92.13	82.62	68.28	64.75	22.57
-5dB	89.28	72.13	32.92	27.47	NA
0-20dB	95.86	90.23	88.91	88.73	65.94

Table 1. ASR accuracy comparisons for Aurora-2

In the case of the actual operational scenarios, when no side information is available, models were chosen from the N-Best list while the states were computed using Viterbi decoding. Of course, the states would correspond to only one model which might not be correct, and there might be a significant mismatch between actual and computed states. Moreover the misalignment of words also exacerbated the problem. The results for this case (Adapted Model III as shown in Table 1 Column 4) were only marginally better than those obtained with the multi-condition trained models. To see the effects of the improvements for the case where the states are better aligned, we made use of whatever information we could get. The boundaries of words were extracted from the N-Best list using exhaustive search and the states for the words between these boundaries were assigned by splitting the digits into equal-sized segments and assigning one state to each segment. This limited the damage done by state misalignment, and it can be seen that a 13% digit error reduction from MC training was observed (Adapted Model II in Table 1 Column 3).

3.2. A clustering approach to obtaining correct HMM information

HMM states are used to spread the particles at the right locations for subsequent estimation of the underlying clean speech density. If the state is incorrect, the location of particles will be wrong and the density estimate will be erroneous. One solution is to merge the states into clusters. Since the total number of clusters can be much less than the number of states, the problem of choosing the correct information block for sample generation is simplified. A tree structure to group the Gaussian mixtures from clean speech HMMs into clusters can be built with the following distance measure [15]:

$$d(m,n) = \int g_m(x) \log \frac{g_m(x)}{g_n(x)} dx + \int g_n(x) \log \frac{g_n(x)}{g_m(x)} dx \tag{41}$$

$$= \sum_i \Big[\frac{\sigma_m^2(i) - \sigma_n^2(i) + (\mu_n(i) - \mu_m(i))^2}{\sigma_n^2(i)} + \frac{\sigma_n^2(i) - \sigma_m^2(i) + (\mu_n(i) - \mu_m(i))^2}{\sigma_m^2(i)} \Big] \tag{42}$$

where $\mu_m(i)$ is the i-th element of the mean vector μ_m and $\sigma_m^2(i)$ is the i-th diagonal element of the covariance matrix Σ_m. The parameters of the single Gaussian representing the cluster, $g_c^k(X) = N(X|\mu_k, \sigma_k^2)$, is computed as follows:

$$\mu_k(i) = \frac{1}{M_k} \sum_{m=1}^{M_k} E(x_m^{(k)}(i)) = \frac{1}{M_k} \sum_{m=1}^{M_k} \mu_m^{(k)}(i) \tag{43}$$

$$\begin{aligned} \sigma_k^2(i) &= \frac{1}{M_k} \sum_{m=1}^{M_k} E((x_m^{(k)}(i) - \mu_k(i))^2 \\ &= \frac{1}{M_k} \sum_{m=1}^{M_k} \sigma_m^{2(k)}(i) + \sum_{m=1}^{M_k} \mu_m^{(k)2}(i) - M_k \mu_k^2(i) \end{aligned} \tag{44}$$

Alternatively, we can group the components at the state level using the following distance measure [16]:

$$d(n,m) = -\frac{1}{S} \sum_{s=1}^{S} \frac{1}{P} \sum_{p=1}^{P} \log[b_{ms}(\mu_{nsp})] + \log[b_{ns}(\mu_{msp})] \tag{45}$$

where S is the total number of states in the cluster, P is the number of mixtures per state and b(.) is the observation probability. This method makes it easy to track the state level composition of each cluster. In both cases, the clustering algorithm proceeds as follows:

1. Create one cluster for each mixture up to k clusters.
2. While $k > M_k$, find n and m for which $d(n, m)$ is minimum and merge them.

Once clustering is complete, it is important to pick the most suitable cluster for feature compensation at each frame. The particle samples are then generated from the representative density of the chosen cluster. Two methods can be explored. The first is to decide the cluster based on the N-best transcripts obtained from recognition using multi-condition trained models. Denote the states obtained from the N-best transcripts for noisy speech feature vectors at time t as $s_{t1}, s_{t2,\ldots,} s_{tN}$. If state s_{ti} is a member of cluster c_k, we increment $M(c_k)$ by one, where $M(c_k)$ is a count of how many states from the N-best list belong to cluster c_k. We choose the cluster based on $\text{argmax}_k\, M(c_k)$ and generate samples from it. If more than one cluster satisfies this criterion, we merge their probability density functions. In the second method, we chose the cluster that maximizes the likelihood of the MFCC vector at time t, O_t, belonging to that cluster as follows:

$$C \sim \arg\max_{k} g_{mc}(O_t \mid C_k) \tag{46}$$

It is important to emphasize here that g_{mc} is derived from multi-condition speech models and has a different distribution from the one used to generate the samples. The relationship between clean clusters and multi-condition clusters is shown in figure 1. Clean clusters are obtained using methods described in section 3. The composition information of these clusters is then used to build a corresponding multi-condition cluster set from multi-condition HMMs. A cluster C_j in clean clusters represents statistical information of a particular section of clean speech. The multi-condition counterpart C_j represents statistics of the noisy version of the same speech section.

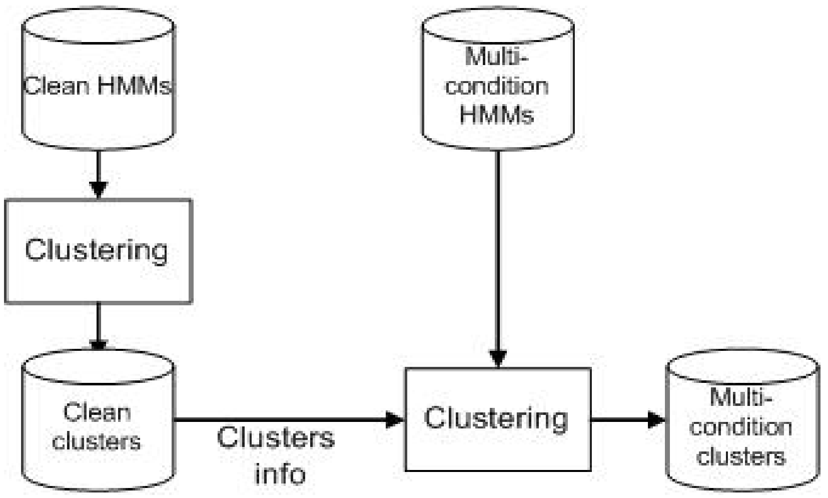


Figure 8. Clustering of multi-condition trained HMMs

Clean clusters are necessary to track clean speech because we need to generate samples from clean speech distributions. However, they are not the best choice for estimating equation (46) because the observation is noisy and has a different distribution. The best candidate for computing equation (46) is the multi-condition cluster set. It is constructed from multi-condition HMMs that match more closely with noisy speech. A block diagram of the overall compensation and recognition process is shown in Figure 9. We make inference about the cluster to be used for observation vector $O_t$using both the N-best transcripts and equation (46) combined together. Samples at frame t are then generated using the pdf of chosen cluster. The weights of the samples are computed using equation (46) and compensated features are obtained using equation (36). Once the compensated features are available for the whole utterance, recognition is performed again using retrained HMMs with compensated features.

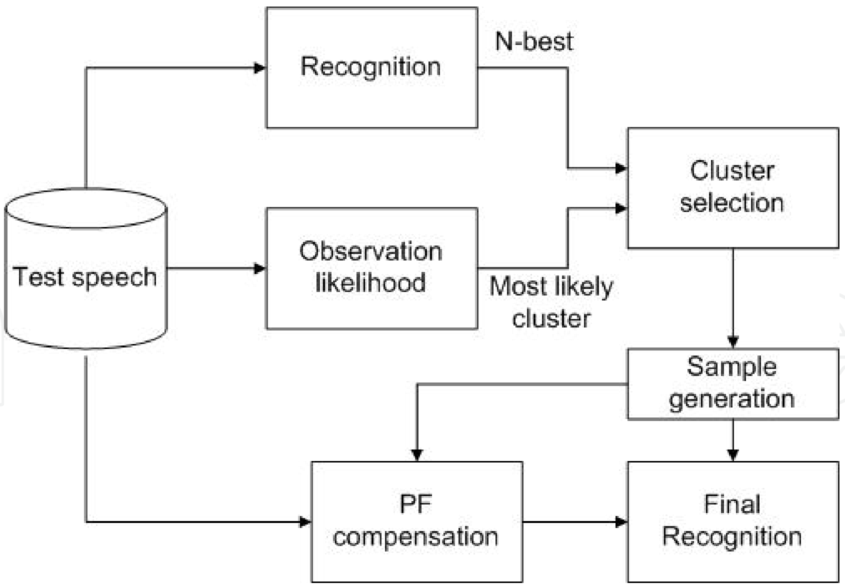


Figure 9. Complete recognition process

3.2.1. Experiments

To evaluate the proposed framework we experimented on the Aurora 2 connected digit task. We extracted features (39 elements with 13 MFCCs and their first and second time derivatives) from test speech as well as 23 channel filter-bank features thereby forming two streams. One-best transcript was obtained from the MFCC stream using the multi-condition trained HMMs. PFC is then applied to the filter-bank stream (stream two). We chose two clusters, one based on 1-best and the other selected with equation (46). The multi-condition clusters used in equation (46) were from 23 channel fbank features so that the test features from stream two can be directly used to evaluate the likelihood of the observations. For results in these experiments, clusters were formed using method two, i.e., tracking the state-wise composition of each cluster. The number of clusters and particles were varied to evaluate the performance of the algorithm under different settings. From the compensated filter-bank features of stream two, we extracted 39-element MFCC features. Final recognition on these models was done using the retrained HMMs, i.e., multi-condition training data compensated in a similar fashion as described above.

Word Accy	20 Clust.	25 Clust.	30 Clust.	MC Trained	Clean Trained
clean	**99.11**	**99.11**	**99.11**	**98.50**	**99.11**
20dB	**97.76**	**98.00**	**97.93**	**97.66**	**97.21**
15dB	**97.00**	**97.14**	**96.69**	**96.80**	**92.36**
10dB	**95.21**	**95.41**	**93.88**	**95.32**	**75.14**
5dB	**89.48**	**89.59**	**87.08**	**89.14**	**42.42**
0dB	**70.16**	**70.38**	**68.84**	**64.75**	**22.57**
-5dB	**36.30**	**36.63**	**36.94**	**27.47**	**NA**
0-20dB	**89.92**	**90.10**	**88.88**	**88.73**	**65.94**

Table 2. Variable number of clusters (100 particles)

The results for a fixed number of particles (100) are shown in Table 1. The number of clusters was 20, 25 or 30. To set the specific number of clusters, HMM states were combined and clustering was stopped when the specified number was reached. HMM sets for all purposes were 18 states, with each state represented by 3 Gaussian mixtures. For the 11-digit vocabulary, we have a total of approximately 180 states. In case of, for example, 20 clusters, we have a 9 to 1 reduction of information blocks to choose from for plugging in the PF scheme.

It is interesting to note that best results were obtained for 25 clusters. Increasing the number of clusters beyond 25 did not improve the accuracy. The larger the number of clusters, the more specific speech statistics each cluster contains. If the number of clusters is large, then each cluster encompasses more specific section of the speech statistics. Having more specific information in each cluster is good for better compensation and recognition because the particles can be placed more accurately. However, due to the large number of clusters to choose from, it is difficult to pick the correct cluster for generation of particles. More errors were made in the cluster selection process resulting in degradation in the overall performance.

This is further illustrated in Figure 10. If the correct cluster is known, having large number of clusters and consequently more specific information per cluster will only improve the performance. The results are for 20, 25 and 30 clusters. In the known cluster case, one cluster is obtained using equation (46) and the second cluster is the correct one. Correct cluster means the one that contains the state (obtained by doing recognition on the clean version of the noisy utterance using clean HMMs) to which the observation actually belongs to. For the unknown cluster case, the clusters are obtained using equation (46) and $1-$best. It can readily be observed from the known cluster case that if the choice of cluster is always correct, the recognition performance improves drastically. Error rate was reduced by 54%, 59% and 61.4% for 20, 25 and 30 clusters, respectively. Moreover, improvement faithfully follows the number of clusters used. This was also corroborated by the fact that if the cluster is specific down to the HMM state level, i.e., the exact HMM state sequence was assumed known and each state is a separate cluster (total of approximately 180 clusters), the error rate was reduced by as much as 67% [10].

For the results in Table 2, we fixed the number of clusters and varied the number of particles. As we increased the number of particles, the accuracy of the algorithm improves for set A and B combined i.e. for additive noise. The error reduction is 17% over MC trained models. Using a large number of particles implies more samples were utilized to construct the predicted densities of the underlying clean speech features, which is now denser and thus better approximated. Thus, a gradual improvement in the recognition results was observed as the particles increased. In case of Set C, however, the performance was worse when more particles were used. This is so because the underlying distribution is different due to the distortions other than additive noise.

	Set A	Set B	Set C	Average
100 particles	**90.02**	**91.03**	**89.26**	**90.1**
500 particles	**90.03**	**91.10**	**89.07**	**90.07**
1000 particles	**90.02**	**91.13**	**89.07**	**90.07**
MC Trained	**88.41**	**88.82**	**88.97**	**88.73**
Clean Trained	**64.00**	**67.46**	**65.39**	**65.73**

Table 3. Variable number of particles (25 clusters)

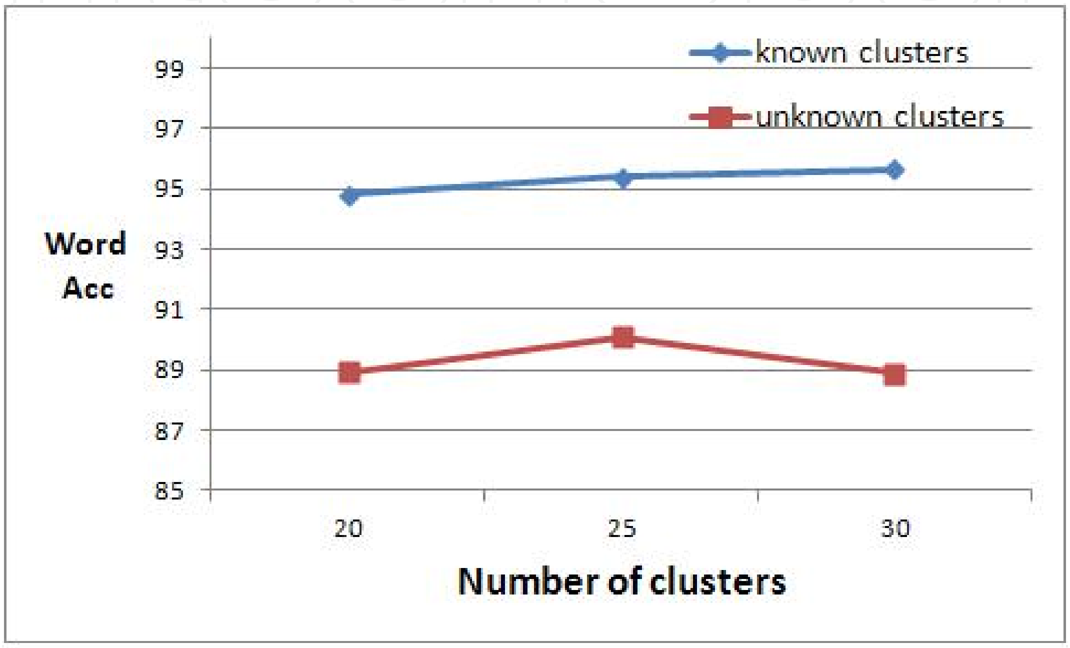


Figure 10. Accuracy when correct cluster known vs. unknown

4. Conclusions

In this chapter, we proposed a particle filter compensation approach to robust speech recognition, and show that a tight coupling and sharing of information between HMMs and particle filters has a strong potential to improve recognition performance in adverse environments. It is noted that we need an accurate alignment of the state and mixture sequences used for compensation with particle filters and the actual HMM state sequences that describes the underlying clean speech features. Although we have observed an improved performance in the current particle filter compensation implementation there is still a considerable performance gap between the oracle setup with correct side information and what's achievable in this study with the missing side information estimated from noisy speech. We further developed a scheme to merge statistically similar information in HMM states to enable us to find the right section of HMMs to dynamically plug in the particle filter algorithm. Results show that if we use information from HMMs that match specifically well with section of speech being compensated, significant error reduction is possible compared to multi-condition HMMs.

Author details

Aleem Mushtaq
School of ECE, Georgia Institute of Technology, Atlanta, USA

5. References

[1] C.-H. Lee and Q. Huo, “On adaptive decision rules and decision parameter adaptation for automatic speech recognition”, *Proc. IEEE,* vol. 88, pp. 1241-1269, 2000.

[2] S.Davis and P. Mermelstein, “Comparison of parametric representations for monosyllable word recognition in continuously spoken sentences,” Proc. ICASSP 1980ol. 28, no.4, pp. 357-366, 1980.

[3] A. Sankar and C.-H. Lee, “A maximum-likelihood approach to stochastic matching for robust speech recognition,” IEEE Trans. Speech Audio Processing, vol. 4, pp.190-202, May.1996.

[4] B. Raj, R. Singh, and R. Stern, “On tracking noise with linear dynamical system models.” Proc. ICASSP, 2004.

[5] M. Fujimoto and S. Nakamura, “Particle Filter based non-stationary noise tracking for robust speech recognition,” Proc. ICASSP, 2005.

[6] M. Fujimoto and S. Nakamura, “Sequential non-stationary noise tracking using particle filtering with switching dynamical system,” Proc. ICASSP, 2006.

[7] M .S. Arulampalam, S. Maskell, N. Gordon, and T. Clapp, “A Tutorial on Particle Filters for Online Nonlinear/Non-Gaussian Bayesian Tracking,” IEEE Trans. Signal Proc., 2002.

[8] Robert Grover Brown and Patrick Y. C. Hwang. 1996. Introduction to Random Signals and Applied Kalman Filtering, 3rd edition, Prentice Hall.

[9] Simon Haykin. 2009. Adaptive Filter Theory, 4th edition, Prentice Hall.

[10] A. Mushtaq, Y. Tsao and C.-H. Lee, “A Particle Filter Compensation Approach to Robust Speech Recognition.” Proc. Interspeech, 2009.

[11] A. Mushtaq and C.-H. Lee, “An integrated approach to feature compensation combining particle filters and Hidden Markov Model for robust speech recognition.” Proc. ICASSP, 2012.

[12] Todd K. Moon and Wynn C. Stirling. 2007. Mathematical Methods and Algorithms for Signal Processing, Pearson Education.

[13] N Arnaud, Doucet, and Johansen, "A tutorial on particle filtering and smoothing: Fifteen years later," Tech. Rep., 2008. [Online]. http://www.cs.ubc.ca/~arnaud/doucet_johansen_tutorialPF.pdf

[14] A. Acero, L. Deng, T. Kristjansson, and J. Zhang, “HMM adaptation using vector Taylor series for noisy speech recognition,” Proc. ICSLP, pp. 869-872, 2002.

[15] T. Watanbe, K. Shinoda, K. Takagi, and E. Yamada, “Speech recognition using tree-structured probability sensity function,” in Proc. Int. Conf. Speech Language Processing ’94, 1994, pp. 223-226.

[16] S. J. Young, J. J. Odell, and P. C. Woodland, “Tree-based state tying for high accuracy acoustic modeling, “ Proc. ARPA Human Language Technology Workshop, pp. 307–312, 1994.