IDIAP COMMUNICATION

IDIAP

Martigny - Valais - Suisse

Development of a DTW based Speech Recognition System over the telephone line

Frank Formaz [a], Manish Goyal [a], Olivier Bornet [a]

IDIAP–Com 01-05

September 2001

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 – 27 – 721 77 11
fax +41 – 27 – 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[a] Dalle Molle Institute for Perceptive Artificial Intelligence PO Box 592 CH-1920 Martigny, Switzerland

Contents

1 Introduction

A voice dialing system has been implemented using speaker dependent speech recognition based on phone-like units as models. Also this system incorporates garbage spotting and keyword spotting abilities. This document gives a brief explanation of the theory behind the working of the system and also gives a description of the manner in which the code written for the application is to be used.

2 The Application

The main goal of this project is to rebuild an existing demonstration which uses a speech package called STRUT with a completely rewritten code to compare the performances between the original code and a simplier algorythm called DTW.

The Voice Dialing application chosen allows to connect a keyword defined by a speaker to a phone number as available now on almost all the new mobile phones. The original code was written at IDIAP few years ago using a SUN ISDN-board and an telephone library called XTL [2]. The speech recognition was done using C++ methods from the STRUT package developped at Mons [1]. A change of the main C++ code has to be done in order to deal not any more with methods but with simplier scripts which simplify the testing phase.

The figures 1,2,3,4 describe the complete application. The different functionalities of the application can be access either via the enroled keywords or, generally only for the first use, by using DTMF (0:new key, 1:delete key, 2:list of existing keys).

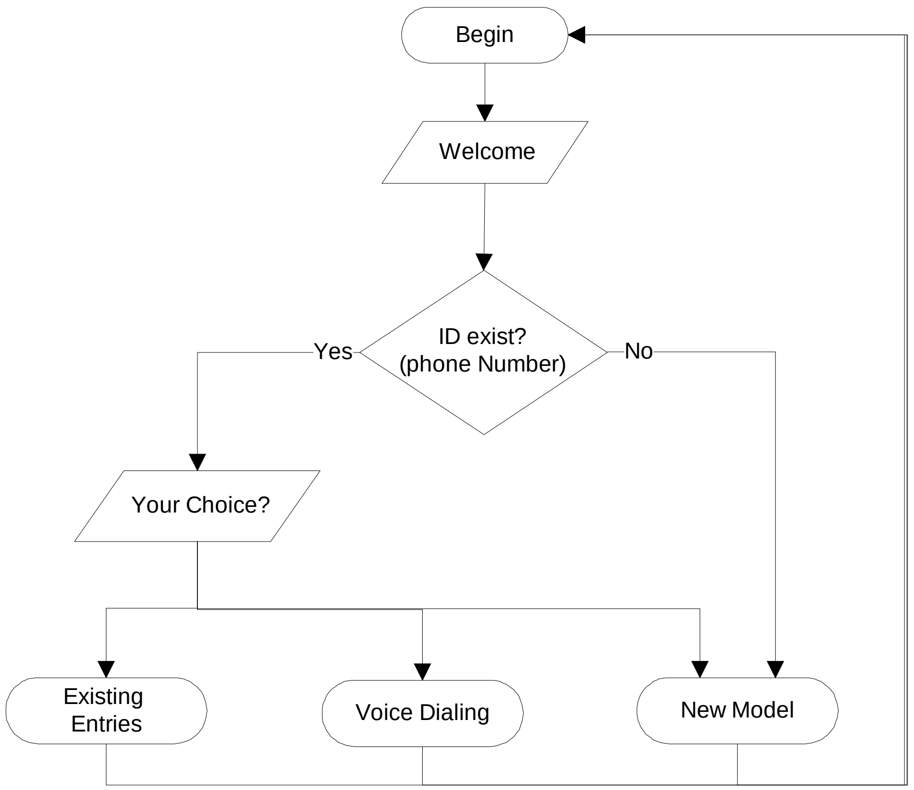

Figure 1: Complete Voice Dialing System

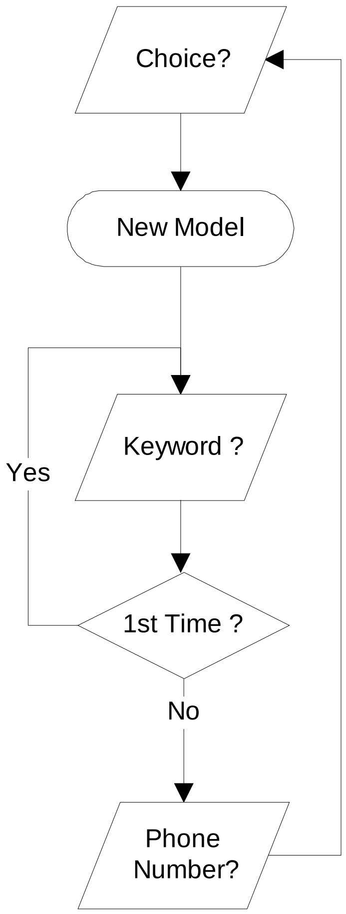

Figure 2: New Model

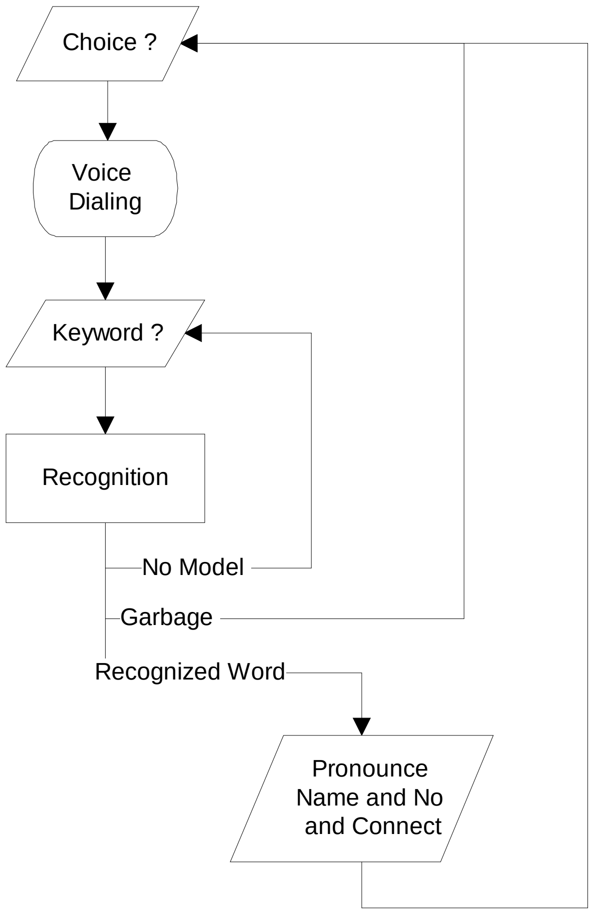

Figure 3: Voice Dialing

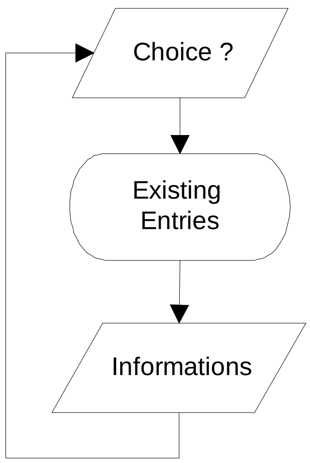

Figure 4: Existing Entries

3 General Overview

A well-studied approach to automatic speech recognition is based on the storage of one or more acoustic patterns (templates) for each word in the recognition vocabulary. The recognition process then consists of matching the incoming speech with stored templates. The template with the lowest distance measure from the input pattern is the recognised word. One algorithm used to find the best match (lowest distance measure) is based upon Dynamic Programming (DP). The aim of this project was to implement a Dynamic Time Warping (DTW) word recogniser. The generation of templates is done using a pre-enroled vectors space splitted into N areas. This will be explained into the section Vector Quantization. The figure 5 represent an overview of the whole system (training/enrolement/recognition). The main Voice Dialing or keyword spotting consists of two phases:

1. Enrollment Phase: In this case the user pronounces the keyword (we have used one template per word here) and provides the system with the associated phone number. Since we have used only one utterance per word this enrollment procedure is fast and flexible. The system builds up word models and stores them top be matched later. We have used Vector Quantization to generate the templates to be used for the models.

2. Recognition Phase: The user pronounces a keyword and the system automatically dials the associated phone number. This is done by using the Dynamic Time Warping Algorithm to compare the utterance with all the templates to see which matches. the closest (or recognized as garbage) and dial its associated phone number.

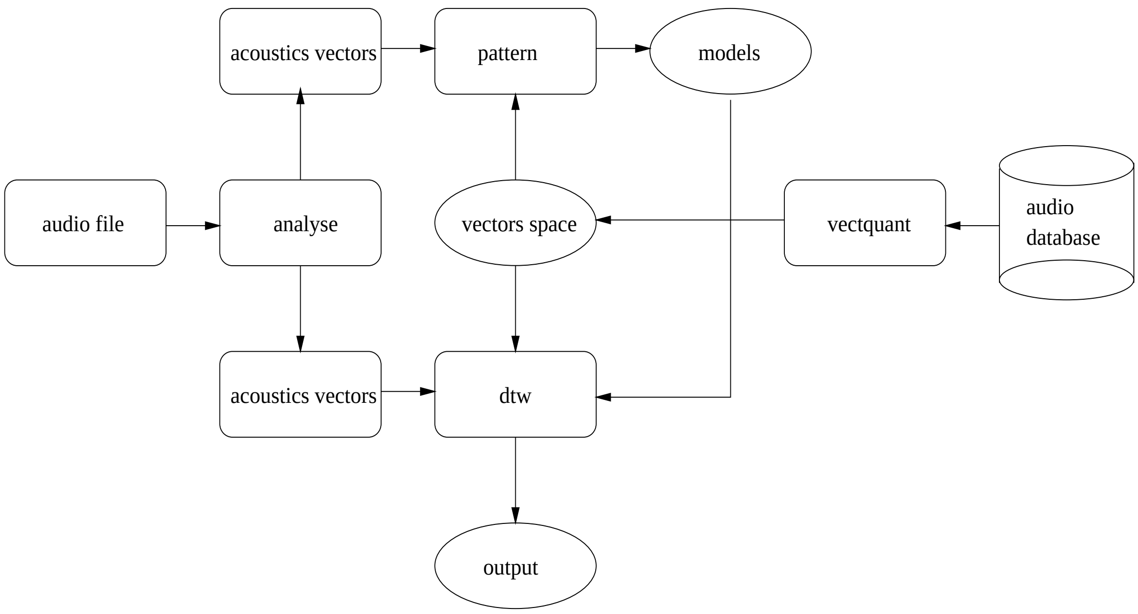

Figure 5: Overwiew of the Whole System

4 Acoustic Vectors

The acoustic features which have been used have been generated from the Acoustic Front End developed at IDIAP (analyse a part of the STRESS project). This software compute the Mel Frequency Cepstral Coefficients (MFCCs) extracted from 30ms speech frames shifted by 10ms. The vectors used are 13 dimensional vectors, consisting of 12 static cepstral components and a log-energy component. Other features could be chosen as for example the RASTA coefficients which seems to be more robust in noisy environement but, we decided to chose the MFCC because it was a real test of the STRESS package into which the new DTW modules were added.

5 Vector Quantization

The vectors are quantized using the standard K-means algorithm (Lloyd's generalized algorithm). The standard Euclidean distance has been used for dividing the acoustic parameter space into different regions (bigger the number of regions is, more precise but slower the recognition will be). The Figure 6 represent this. The data used to generate this vector space are coming from the Swiss French Polyvar Database developed at IDIAP. For the training phase we have used telephone recordings from about 140 speakers. Each speaker recorded between 1 and 229 sessions.The utterances consisted of 10 phonetically rich sentences for each speaker, making it a total of about half an hour of spoken speech.

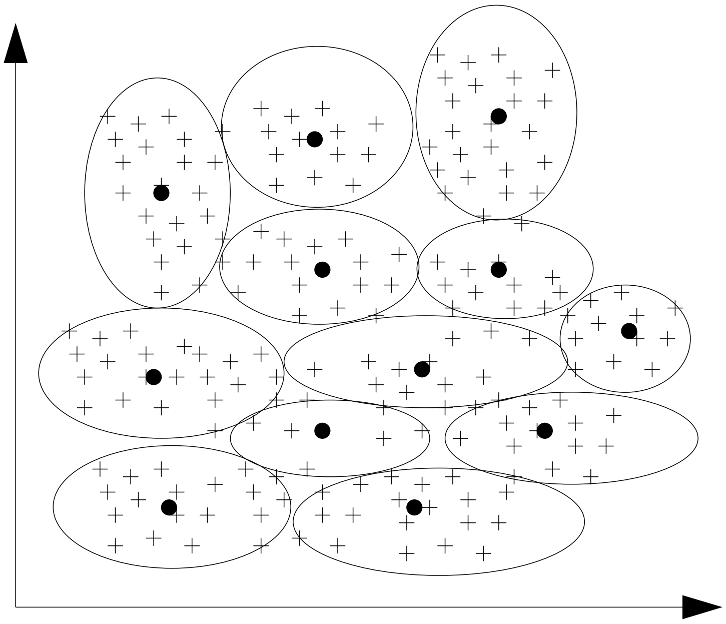

Figure 6: Cluster of 12 areas which splits the complete Cepstral Vector Space. Each + represents one vector of 13 coefficients as described previously. The center of each area is stored as a centroid and used for enrolment/recognition phase.

5.1 Explanation of code:

All the acoustic vectors for the training have been put into a single file. The program for calculating the centroids uses this archive of acoustic vectors and stores the results into another file which contains N vectors of M parameter, N beeing the number of clusters and M the number of parameters used by the features extraction software (generaly 13, 12 coefficients and the log-energy). At the beginning, N points are arbitrarily fixed somewhere into the vectors space (define as codebook). Then the next two steps are iterate until some convergence criterion is met.

- Minimum distortion labeling: Each input vector is associated to one entry of the given codebook. The association is done in fact by choosing the closest point.

- Centroid computation: For each group of vector with the same label (associated to the same point of the codebook), compute a new centroid which minimize the average distortion for the members of the group. This is done by computing the mean of all the vectors associated to one center.

There are two solutions to define the convergence criterion. A first one is the setup of a minimal change under it, changes are concidered as negligible and so, stop the computing process and a simplier one (the one used into our software) which fix the number of iterations without regarding.

5.2 Use of code:

```
vectquant: clusters definition.
VERSION:   1.0 - IDIAP - 1999
AUTHORS:   M. Goyal - O. Bornet - F. Formaz

Options:
        --help
        --CepstralVectorSize[13]=value
        --VectorsRegionNumber[128]=value
        --RecalculationNumber[500]=value

Arguments:
        TrainingVectorsFile -> value
```

- CepstralVectorSize: Define the number of cepstral coefficients which are computed. The default value is 13 constituted into 12 cepstral coefficients vectors and the log energy cofficient.
- VectorsRegionNumber: Define the number of the vectors regions which are defined with a default value of 128.
- RecalculationNumber: Define the number of recomputation of the centroids.

6 Generation of Templates

In our case we have used only one utterance per word to create a single model for each word. Each frame is analysed, compared to the general vector space defined before and associated to one centroid (the closest one). A number is given for each centroid and so, each frame is defined as a number which represent it's position into the vectors space. The word models will be represent as strings of numbers included between 1 and N (N beeing the number of centroids).

6.1 Explanation of code:

The program "pattern.c" is used for generating the word models. It reads in the centroids of the quantized regions from QUANTIZED"nn".DAT" (where nn is the number of defined centroids). It also reads in a list of filenames (utterances) for which word models are to be constructed and also a list of filenames into which the templates are finally to be stored. During the enrolement, word models are built up by first replacing each vector by the label of the closest prototype (centroid of the vectors space). A compression algorithm has also been implemented following a simple rule: sequence of the same label are reduced to sequences of length n, indicating stationary part of speech, while transition parts are left unchanged. The resulting compressed label sequence was stored as the word model. For example, if we supposed n=2, the label sequence 2 2 2 7 4 4 4 4 4 will be turned into 2 2 7 4 4 4. The parameter n will be refered as reduction factor and by default with a very big value in order to be inactive. This reduction doesn't change a lot the quality of the recognition but can be really usefull when used on a machine with very few RAM or CPU capabilities such as, for example, a PDA. The gain in CPU time for the DP is proportional to the storage gain. Also as already shown [3], this kind of modelling could also have a smoothing effect which can improve the recognition performances.

6.2 Use of code:

```
pattern: Templates Generation.
VERSION: 1.0 - IDIAP - 1999
AUTHORS: M. Goyal - O. Bornet - F. Formaz

Options:
        --help
        --CepstralVectorSize[13]=value
        --VectorsRegionNumber[128]=value
        --ReductionFactor[MAXSHORT]=value

Arguments:
        InputListFile -> value
        OutputListFile -> value
```

- CepstralVectorSize: Define the number of cepstral coefficients which are computed. The default value is 13 constituted into 12 cepstral coefficients vectors and the log energy cofficient.
- VectorsRegionNumber: Define the number of the vectors regions which are defined with a default value of 128.
- ReductionFactor: Define the maximum number of same concecutive coefficients in a new model. By default there is no compression and this may be use if the storage size is very small as for example in a PDA.
- InputListFile: contains a list of all the files for which models are to be generated.
- OutputListFile: contains a list of all the files into which the models (templates)are to be stored.

7 Recognition

The Dynamic time warping program is the main recognition software which "compare" a test file to a set of templates already enroled.

7.1 Explanation of code:

Recognition is done using the Dynamic Programming (DP) Algorithm. When applied to template-based speech recognition, it is often referred to as Dynamic Time Warping (DTW). To do this, we have to concider for each reference Y^k a D matrix of dimension $(N * J(k)$ (where N and $J(k)$ are vectors number into the test and reference sequence respectively). For each input (n, j) of this matrix, a local distance $d(n, j)$, defined as the Euclidian distance (1), is associated for each label of the reference vector. In order to find the optimal distance $D(X, Y^k)$ (also called global distance or distortion) between the test sequence X and the reference sequence Y^k, we only have to find the path into this matrix which minimize the sum of local distances encountered from the begin of the sequences (generaly $(1, 1)$) to the end of the sequences (generaly $(N, (Jk))$).

$$d(x_n, y_j^k) = \|x_n - y_j^k\|^2 = \sqrt{\sum_{i=1}^{d}(x_{ni} - y_{ji}^k)^2} \tag{1}$$

If $D(n,j)$ is the global (or accumulated) distance up to (n,j) and the local distance at this point is given by $d(n,j)$ then we can show that the accumulated distance can be computed using the following recurency

$$D(n,j) = d(n,j) + min[D(p(n,j))] \tag{2}$$

into which $p(n,j)$ is the set of all the posible predecessors. In our case $p(n,j)$ was defined as $(n-1,j-1),(n-1,j),(n,j-1)$. Other possibilities can be chose but as the choice between them doesn't follow a clear rule, we decided to use the easiest one. More detailed informations on the other solutions can be find in [4].
The DP algorithm works in a time-synchronous manner: each column of the time/time matrix is considered in succession (equivalent to processing the input frame-by-frame) so that, for a template of length N, the maximum number of paths being considered at any time is N. As illustrated in Figure 7, (2) define the optimal (minimal) path between the input sequence X and a reference Y^k. Applied to isolated word recognition, this programming is done for each reference words (template) $Y^k(k=1,...,K)$ and the recognized word is associated to the minimal accumulated distance but only after checking to see that it is not an Out of vocabulary word. This is done by comparizon between the accumulated distance $D(N,(Jk))$ obtained for each template Y^k and an accumulated garbage G which is defined as the accumulation of all the local garbages g_n of each input vector. This local garbage g_n is defined as the average of the M best local distances from the test vector to all the quantized vectors except the best one. Doing this can be statistically explained to allow a path optimization [5] [6].
No quantization is done at the time of recognition. Fictitious garbage states have been added to incorporate recognition of Out of vocabulary words and also to facilitate Keyword Spotting. This enables our system to work for cases when the speaker might say"Please give me Mr. X" or "I want Mr.X" or "Dial Mr.X" or something else to that effect.

7.2 Use of code:

The sequence of instructions to be used is:

```
   dtw1: recognition module.
VERSION: 1.0 - IDIAP - 1999
AUTHORS: M. Goyal - O. Bornet - F. Formaz

Options:
        --help
        --CepstralVectorSize[13]=value
        --VectorsRegionNumber[128]=value
        --GarbageSensitivity[12]=value

Arguments:
        ModelsListFile -> value
        SignalsListFile -> value
```

- ModelsListFile consists of a list of all the filenames containing the templates which are to checked.
- SignalsListFile contains the list of all the files which are to be recognized (the models already enroled).

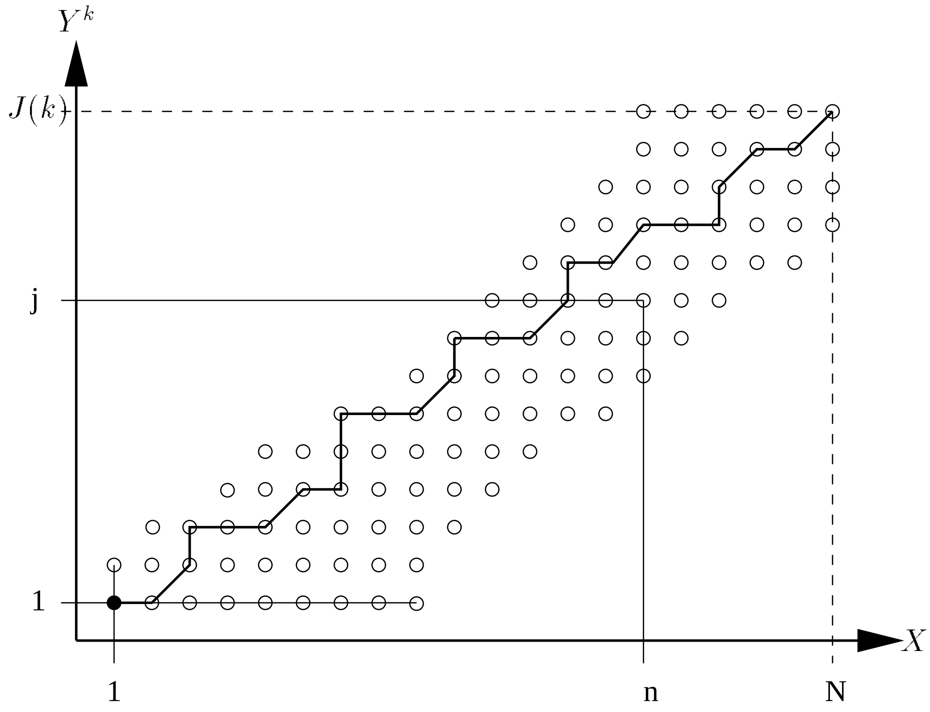

Figure 7: DTW comparizon between input X and reference sequence Y^k. At each point (n,j), a local distance d(n,j) and a accumulated distance D(n,j) are associated. An optimal path is then computed in order to minimize the accumulated distance $D(X, Y^k)$

- As output, the program returns either the filename which contains the word utterance or, in the case of garbage being detected, nothing

7.3 Memory requirements

Assuming 64 classes and an utterance length of about 1 sec.
Since we have vectors being generated every 10 msec, we have 100 labels to store per utterance.
Since each label will be of one byte the total storage per word=100*1=100bytes.
Assuming that we have 5 speakers each with 17 words (as has been the case in the tests done here) we need a total of 85 words in the vocabulary.
thus total storage =85*100=8500 bytes.
We have 13 dimensional vectors.We need to store 64 such vectors for the
centers of the 64 different regions.
Since each dimension is in the range of +-1000 we need 2 bytes of memory per dimension.
Thus the memory required for storing the centers is :
2*13*64=1664 bytes.
Total memory required for an 85 word vocabulary with 64 region vector
quantization is: 8500+1664=10164 bytes = 10 KB
Thus if we have N words in the vocabulary,and quantize into K regions,the number of bytes required would be:
26K + 100N.

7.4 Computational Complexity of the DTW algorithm

Key to the symbols used:
Number of regions into which the vector space is divided = K.
Number of words in the vocabulary = W
Average number of vectors in each word = V. Normally 100 considering a word would normally be uttered within 1 sec and we are generating acoustic vectors every 10 millisecs.
Dimension of the acoustic vectors used = d. 13 in our case.
For keyword spotting, number of distances taken for computing the garbage values = N
For each input vector for the pattern, we first calculate the distance of the vector from each of the centroids.
Since each of the vectors is d dimensional,the distance computation for each centroid requires–2*d +d additions = 3d additions(using simple Euclidean distances).Since we have K regions the number of additions required is 3d*k = 3dK additions.
In case we are also using Keyword Spotting/Garbage models in the algorithm then we also need to sort the resulting distances in order to get the value of the N best for the garbage models. The complexity of this will depend on the sorting algorithm used. It will be O(K*K) for Bubble Sort/Insertion Sort and O(KlogK)for Merge Sort/Heap Sort. Once the sorting is done,we need to add the N best regions.This will take N additions. Thus,if we include Garbage Models we require KlogK + N additions or alternatively,K*K +N additions per pattern vector to be recognized.
Next,we need to compute the best path for the matching. For this we need to look at only the present column and the previous column.
Since we are using the symmetrical DTW algorithm,each template vector would require 3 comparisons(since these are the only 3 paths for alignment)and this is to be repeated for all the template vectors. Thus the computation required is 3*W*V.
Finally we need to check the best match among all the word models. This is done only at the end of the entire pattern to be matched and not after every input vector. This takes W comparisons(2W comparisons if checks for Garbage are also done).
Since W¡3 *W*V this can be neglected.
Hence the computations required per input vector for the DTW algorithm is

1. without garbage model: $3dK + 3WV + W = O(dK + WV)$
2. with garbage model: $3dK + KlogK + N + 3W*V = O(dK + KlogK + N + WV)$ or alternatively $O(dK + K*K + N + WV)$ depending on the sorting algorithm used.

In the algorithm that has been implemented, Merge Sort has been used as the default sorting algorithm.
Plugging in possible values, d=13, K=64, N=12, V=100, W=20
without garbage models =3*13*64 + 3*20*100 + 20=2496 + 6000 + 20=8516 computations ie. around 10000 computations.
with garbage models=3*13*64 + 12 + 3*20*100 + 2*20=4096 + 12 + 6000 + 40=12644 computations. ie around 15000 computations.

8 Conclusions

A final application using the tools described into this report has been set up at IDIAP using the original SUN hardware. We also have an other version which is actually running on WindowsNT with a Dialogic Board. This work [7] wad done for a diploma work and wasn't possible without the rewriting of such basic speech tools because of the platform and the closed speech code. So, in conclusion we can say that this work was profitable for different reason and is a first stone in the building of a more complete development speech library at IDIAP. To be the most complete, a "scientific" test of those tools and so, a comparizon with the original STRUT code should also be done but the time was a little too short in order to run and include it into this report.

References

[1] Faculte Polytechnique de Mons "Speech Training and Recognition Unified Tool (STRUT)". http://tcts.fpms.ac.be/asr/strut.html

[2] SunSoft "XTL Application Programmer's Guide". Montain View, CA 1994

[3] Bourlard H., Ney H. and Wellekens C.J. "Connected Digit Recognition using Vector Quantization". Intl. Conf. on Acoustics, Speech and Signal Processing, pp. 16.10.1-4 1984

[4] Rabiner L. and Juang B. H. "Fundamentals of Speech Recognition". *PTR Prentice-Hall, Inc.*, Englewood Cliffs, NJ, USA. 1993

[5] Boite, J.-M., Bourlard, H., and Haesen, M. "A New Approach Towards Keyword Spotting". Proc. EUROSPEECH'93 (Berlin, Germany), pp. 1273-1276. 1993

[6] Bourlard, H., D'hoore, B., and Boite, J.-M. "Optimizing Recognition and Rejection Performance in Wordspotting Systems". Proc. of IEEE Intl. Conf. on Acoustics, Speech, and Signal Processing (Adelaide, Australia), pp. I:373-376. 1994

[7] Bressoud F. and Wang H. "Personnal Voice Dialing over PC". IDIAP-COM 2000-05 2000