Computer Engineering Four Assignment One

Michael Still
964076

Table of Contents

Part A

Perform time-domain analysis with a frame size of 512

Input the file, play it back and state what the spoken phrase is

The spoken phrase is "Human Computer Communication".

Compute the energy contour in dB

The following Matlab script was used to generate the energy contour in dB, including using the first frame as a reference of 0 dB.

Code: matlab/econtour.m

```
function [energy, edb] = econtour(filename)

framesize = 512;
numsamples = wavread(filename, 'size');

% Read the whole file into one big vector
% Check the wavread doco for a better way of doing this
[samples, fsample, nbits] = wavread(filename, numsamples(1));

% Compute the energy contour
fcount = 1;
while fcount < numsamples(1) / framesize
    temp = samples((fcount - 1) * framesize + 1: fcount * framesize);

    % Calculate the energy, and the reference if required. Apply that reference
    energy(fcount) = temp' * temp;
    edb(fcount) = 10 * log10(energy(fcount));
    if(fcount == 1)
        eref = edb(1);
    end
    edb(fcount) = edb(fcount) - eref;

    % Next frame
    fcount = fcount + 1;
end
```

Compute the normalised autocorrelation contour

The following Matlab script was used to calculate the autocorrelation contour:

Code: matlab/autocorr.m

```
function [acorr, ffreq] = autocorr(filename, energy)

framesize = 512;
numsamples = wavread(filename, 'size');
vthresh = 0.375;

% Read the whole file into one big vector
% Check the wavread doco for a better way of doing this
[samples, fsample, nbits] = wavread(filename, numsamples(1));

% We need to know kmin and kmax before we can continue
kmin = fsample / 200;
```

```
kmax = fsample / 80;

% Compute the autocorrelation
fcount = 1;
while fcount < numsamples(1) / framesize
    temp = samples((fcount - 1) * framesize + 1: fcount * framesize);

    for count = kmin:kmax
        acalc(count) = temp(1:framesize - count)' * temp(1 + count: framesize);
    end

    % Now normalize
    [amax, indexmax] = max(acalc);
    acorr(fcount) = amax / energy(fcount);

    if (acorr(fcount) > vthresh)
        ffreq(fcount) = fsample / indexmax;
    end

    % Next frame
    fcount = fcount + 1;
end
```

Set a suitable autocorrelation value for the voiced/unvoiced threshold and compute the fundamental frequency contour

The fundamental frequency is calculated by the autocorrelation function in the code in the previous section. The values kmin and kmax were derived for the maximum and minimum frequencies for the voice / unvoice threshold as provided by Professor Wagner.

Plot all three of the above contours with a common time axis and on that plot mark the phoneme boundaries and label the phonemes

Please refer the to next page for this graph, with annotation. The phonemes were selected based on reference to the Cambridge online dictionary (http://dictionary.cambridge.org), and the various notes distributed in class.

Documentation

The two scripts presented earlier in this section to calculate the values required are tied together by the following Matlab script:

Code: matlab/questionOne.m

```
function [] = questionOne(filename)

% Generate the energy contour (part b)
[en, edb] = econtour(filename);

% Generate the autocorrelation (part c)
[ac, ff] = autocorr(filename, en);



% Plot all of these
subplot(3, 1, 1), plot(edb)
grid on
```

```
ylabel('Energy (dB)');
title('Energy, with first frame as reference');

subplot(3, 1, 2), plot(ac)
grid on
title('Normalized autocorrelation');

subplot(3, 1, 3), plot(ff)
grid on
title('Fundamental frequency');
```

Perform frequency-domain analysis on a frame in the centre of the first vowel /u/ and on a frame in the centre of the consonant /sh/

The following scripts were used to perform these calculations:

Code: matlab/questionTwo.m

```
function [ ] = questionTwo( filename )

% The framesize is 512
framesize = 512;

% Find the /u/ frame and perform a frequency domain analysis
[sdata, fsample] = ph_u(filename);
[lms, cep, smoothed] = fd_analyse(sdata, fsample, framesize);

% Plot all of this
figure(1);
subplot(3, 1, 1), plot(lms);
grid on
title('Log magnitude spectrum of /u/');

subplot(3, 1, 2), plot(cep);
grid on
title('Cepstrum of /u/');

subplot(3, 1, 3), plot(smoothed);
grid on
title('Smoothed cepstrum of /u/');


% Find the /sh/ frame and perform a frequency domain analysis
[sdata, fsample] = ph_sh(filename);
[lms, cep, smoothed] = fd_analyse(sdata, fsample, framesize);

% Plot all of this
figure(2);
subplot(3, 1, 1), plot(lms);
grid on
title('Log magnitude spectrum of /sh/');

subplot(3, 1, 2), plot(cep);
grid on
title('Cepstrum of /sh/');

subplot(3, 1, 3), plot(smoothed);
grid on
title('Smoothed cepstrum of /sh/');
```

Code: matlab/ph_u.m

```
function [ sdata, fsample ] = ph_u( filename )

% The framesize is 512
framesize = 512;

% Read in the wav file again
numsamples = wavread(filename, 'size');
[samples, fsample, nbits] = wavread(filename, numsamples(1));

% I have imperically determined that the following frames are used for /u/
start = 6 * framesize;
finish = (12 * framesize) - 1;

% This can be verified by plotting the /u/ sound. The hints for the as-
signment say that for /u/
% we should have periodicity of about 100Hz -- therefore I generate a graph with a pe-
riod of 100Hz
%plot(samples(start:finish))
%grid on

% Therefore, a frame in the center is 8 * 512 to 9 * 512
sdata = samples(start:finish);
```

Code: matlab/ph_sh.m

```
function [ sdata, fsample ] = ph_sh( filename )

% The framesize is 512
framesize = 512;

% Read in the wav file again
numsamples = wavread(filename, 'size');
[samples, fsample, nbits] = wavread(filename, numsamples(1));

% I have imperically determined that the following frames are used for /sh/
start = 70 * framesize;
finish = (74 * framesize) - 1;

% This can be verified by plotting the /sh/ sound. The hints for the as-
signment say that for /sh/
% the plot exhibits non-periodicity and a high-frequency "ragged" waveform
%plot(samples(start:finish))
%grid on

% Therefore, a frame in the center is 8 * 512 to 9 * 512
sdata = samples(start:finish);
```

Code: matlab/fd_analyse.m

```
function [ lms, cep, smoothed ] = fd_analyse( sdata, fsample, framesize )

% Generate a hamming window and then apply that window
hwind = hamming(framesize);
fcount = 1;
while fcount < framesize
    hammed(fcount) = sdata(fcount) * hwind(fcount);

    % Next frame
    fcount = fcount + 1;
end
```

```
% Perform a FFT
fdata = abs(fft(hammed));

% Square each value and then perform a log
fcount = 1;
while fcount < length(fdata)
    lms(fcount) = 20 * log10(fdata(fcount));

    % Next frame
    fcount = fcount + 1;
end

cep = real(ifft(lms));

% Now we need the smoothed cepstrum
tcut = fsample / 400;
scep = cep;
for i = tcut:(framesize + 2 - tcut)
    scep(i) = 0;
end

smoothed = real(fft(scep));
```

/u/ plot

Please refer to the next page for this plot.

/sh/ plot

Please refer to the page following that for the /sh/ plots.

Part B

This part documents the speech recongition experiment.

Compute the codebook

The following code produces a LBG VQ codebook of size 32 from the 9 training MFC files.

Code: c++/gencode.cpp

```
// Generate a codebook

#include "world.h"

int
main (int argc, char *argv[])
{
  mfc data;
  float delta = 0.01;
  float errTarget = 0.0;

  cout << "Delta is " << delta << endl;
  data.clear ();
  load (data, "s014d001.mfc");
  load (data, "s014d002.mfc");
```

```
  load (data, "s014d003.mfc");
  load (data, "s017d001.mfc");
  load (data, "s017d002.mfc");
  load (data, "s017d003.mfc");
  load (data, "s029d001.mfc");
  load (data, "s029d002.mfc");
  load (data, "s029d003.mfc");
  cout << "Total frame count: " << data.size () << endl << endl;

  // Initialize the codebook
  codebook cb;
  cb.init (data);
  cb.cloud (data);

  // Distortion information
  float distortion (0.0);
  for (size_t i = 0; i < data.size (); i++)
    {
      float closest (2000000.0);
      for (size_t j = 0; j < cb.size (); j++)
{
  if (data.getFrame (i) - cb.getLine (j) < closest)
    closest = data.getFrame (i) - cb.getLine (j);
}

      distortion += closest;
    }

  cout << "Distortion for " << cb.size () << " entry codebook: " <<
    "total: " << distortion << " average: " <<
    (distortion / data.size ()) << endl << endl;

  // We fork five times, because we want 32 output codebook entries
  int i;
  for (i = 0; i < 5; i++)
    {
      cout << "------------------------------------------------------
" <<
 endl;
      cout << "- CALCULATE THE NEXT SEQUENCE OF CODEBOOK VECTORS    -
" <<
 endl;
      cout << "------------------------------------------------------
" <<
 endl;
      cb.stats ();

      // Dump the final output of this round
      cout << "Codebook entries are currently:" << endl;
      for (size_t i = 0; i < cb.size (); i++)
{
  cout << cb.getLine (i) << endl;
}
      cout << endl;

      cb.fork (delta);
      cb.cloud (data);

      // Recenter whilst the error is outside acceptable bounds
      float err = 42;
      while (err > errTarget)
{
  err = cb.recenter ();
  cb.cloud (data);
}
```

```
      // Donate and recenter again if we have to
      bool needRecenter (false);
      while (cb.emptyCount () > 0)
 {
   cout << "Donating" << endl;
   cb.donate (delta);
   cb.cloud (data);
   needRecenter = true;
 }

      if (needRecenter)
 {
   cout << "Recentering because of donations" << endl;
   err = 42;
   while (err > errTarget)
     {
       err = cb.recenter ();
     }
 }

      // Dump the final output of this round
      cout << endl << "Final results from round:" << endl;
      for (size_t i = 0; i < cb.size (); i++)
 {
   cout << cb.getLine (i) << endl;
 }

      // Distortion information
      float distortion (0.0);
      for (size_t i = 0; i < data.size (); i++)
 {
   float closest (2000000.0);
   for (size_t j = 0; j < cb.size (); j++)
     {
       if (data.getFrame (i) - cb.getLine (j) < closest)
  closest = data.getFrame (i) - cb.getLine (j);
     }

   distortion += closest;
 }

      cout << "Distortion for " << cb.size () << " entry codebook: " <<
 "total: " << distortion << " average: " <<
 (distortion / data.size ()) << endl << endl;
    }

  cout << "Final codebook size is: " << cb.size () << endl;
  cb.stats ();
  cb.cloudSize ();
  cout << endl;

  cout << cb;
  cout << "-------------------------------------------------------" << endl;

  // We need to save the codebook to disc
  cb.save ("codebook.mfc");

  return (0);
}

void
load (mfc & data, string filename)
{
  string result;

  result = data.add (filename);
```

```
  if (result != "")
    {
      cout << "File load error: " << filename << ": " << result << endl;
      exit (-1);
    }
}
```

Code: c++/lbg.cpp

```
// LBG implementation

#include "world.h"

codebook::codebook ():
m_recenters (0), m_donations (0)
{
}

codebook::codebook (mfc & codes):
m_recenters (0), m_donations (0)
{
  unsigned int i;

  m_table.clear ();
  for (i = 0; i < codes.size (); i++)
    {
      m_table.push_back (codes.getFrame (i));
    }
}

size_t codebook::size ()
{
  return m_table.size ();
}

void
codebook::init (mfc & data)
{
  // If the target size is zero, then it should be one
  // Otherwise, we need to split the current points in twain...
  if (size () == 0)
    {
      m_table.push_back (data.centroid ());
    }
}

void
codebook::fork (float delta)
{
  vector < mfcFrame > oldtable = m_table;
  size_t i;

  m_table.clear ();
  for (i = 0; i < oldtable.size (); i++)
    {
      cout << "\tForking " << oldtable[i] << endl;
      cout << "\t        " << mfcFrame (oldtable[i], 1.0 + delta) << endl;
      cout << "\t        " << mfcFrame (oldtable[i],
     1.0 - delta) << endl << endl;

      m_table.push_back (mfcFrame (oldtable[i], 1.0 + delta));
      m_table.push_back (mfcFrame (oldtable[i], 1.0 - delta));
    }
}
```

```
void
codebook::cloud (mfc & data)
{
  // We now build the clouds around the codebook entries
  cout << "Build clouds" << endl;
  size_t i, j;

  // Clear out the old clouds
  m_clouds.resize (size ());
  for (i = 0; i < size (); i++)
    {
      m_clouds[i].clear ();
    }

  for (i = 0; i < data.size (); i++)
    {
      // Find which point this data point is closest to
      size_t found = 0;
      float dist = 2000000;

      for (j = 0; j < size (); j++)
 {
   float newdist = m_table[j] - data.getFrame (i);

   if (newdist < dist)
     {
       found = j;
       dist = newdist;
     }
 }

      // Add the point to that cloud
      m_clouds[found].add (data.getFrame (i));
    }

  cloudSize ();
}

void
codebook::cloudSize ()
{
  // Display information about cloud sizes
  unsigned int i;

  cout << "Cloud sizes: ";
  for (i = 0; i < size (); i++)
    {
      cout << m_clouds[i].size () << " ";
    }
  cout << endl;
}

// Now we change the table entries to match the centroids of that cloud
float
codebook::recenter ()
{
  m_recenters++;

  float maxerr = 0;
  unsigned int i;

  cout << "Recentering cloud centroids" << endl << endl;

  for (i = 0; i < size (); i++)
    {
```

```
      if (m_clouds[i].size () != 0)
 {
   float err;

   err = m_table[i] - m_clouds[i].centroid ();
   cout << "\tErr: " << err << endl;
   cout << "\tOld: " << m_table[i] << endl;
   m_table[i] = m_clouds[i].centroid ();
   cout << "\tNew: " << m_table[i] << endl << endl;

   if (err > maxerr)
     maxerr = err;
 }
      else
 {
   cout << "\tSkipping error on zero sized cloud" << endl << endl;
 }
    }

  cloudSize ();
  return maxerr;
}

int
codebook::emptyCount ()
{
  unsigned int i;
  int count = 0;

  for (i = 0; i < size (); i++)
    {
      if (m_clouds[i].size () == 0)
 count++;
    }

  return count;
}

void
codebook::donate (float delta)
{
  m_donations++;

  // Find the biggest cloud
  int maxI;
  unsigned int i, maxSize = 0;

  for (i = 0; i < size (); i++)
    {
      if (m_clouds[i].size () > maxSize)
 {
   maxSize = m_clouds[i].size ();
   maxI = i;
 }
    }

  // Find the first empty cloud
  for (i = 0; i < size (); i++)
    {
      if (m_clouds[i].size () == 0)
 {
   cout << "\tDonate from cloud " << maxI << " to cloud " << i << endl
     << endl;

   mfcFrame f = m_table[maxI];
   m_table[maxI] = mfcFrame (f, 1.0 + delta);
```

```
   m_table[i] = mfcFrame (f, 1.0 - delta);
   return;
 }
    }
}

void
codebook::stats ()
{
  cout << "Statistics" << endl;
  cout << "\tRecenters performed: " << m_recenters << endl;
  cout << "\tDonations performed: " << m_donations << endl << endl;
}

mfcFrame & codebook::getLine (size_t i)
{
  return m_table[i];
}

void
codebook::save (string filename)
{
  fstream file;

  // Open the file
  file.open (filename.c_str (), ios::out);
  if (!file)
    {
      cerr << "Could not open file to save codebook to" << endl;
      exit (42);
    }

  // Most of these are dodgy hard coded values
  file << size () << " 100000 44 14106" << endl;

  unsigned int i;
  for (i = 0; i < size (); i++)
    {
      for (int j = 0; j < 9; j++)
 file << m_table[i].getC (j) << " ";
      file << m_table[i].getE () << endl;
    }
}

// Encode the vectors
vector < int >
codebook::encode (mfc & target)
{
  unsigned int i;
  vector < int >output;

  output.resize (target.size ());
  for (i = 0; i < target.size (); i++)
    {
      // Find the table entry which is closest to this target entry
      float dist = 2000000;
      unsigned int j, tindex;

      for (j = 0; j < size (); j++)
 {
   if (m_table[j] - target.getFrame (i) < dist)
     {
       dist = m_table[j] - target.getFrame (i);
       tindex = j;
     }
 }
```

```
      output[i] = tindex;
    }

  return output;
}

// Used for display
ostream & operator << (ostream & stream, codebook cb)
{
  size_t i;

  for (i = 0; i < cb.size (); i++)
    {
      stream << (i +
   1) << " of " << cb.size () << ": " << cb.getLine (i) << endl;
    }

  return stream;
}
```

Code: c++/lbg.h

```
// Header for the LBG implementation

#include "world.h"

#ifndef LBG_H
#define LBG_H

class codebook
{
public:
  codebook ();
  codebook (mfc & codes);

  size_t size ();
    mfcFrame & getLine (size_t i);

  void init (mfc & data);
  void fork (float delta);
  void cloud (mfc & data);
  void cloudSize ();
  float recenter ();
  int emptyCount ();
  void donate (float delta);
  void stats ();
  void save (string filename);
    vector < int >encode (mfc & target);

private:
    vector < mfcFrame > m_table;
    vector < mfc > m_clouds;

  long m_recenters;
  long m_donations;
};

ostream & operator<< (ostream & stream, codebook cb);

#endif
```

Code: c++/mfc.cpp

```
// Parse the MFC files into memory

#include "world.h"

mfcFrame::mfcFrame ()
{
}

mfcFrame::mfcFrame (mfcFrame & other, float delta)
{
  int i;

  for (i = 0; i < 10; i++)
    {
      setC (i, other.getC (i) * delta);
    }
  setE (other.getE () * delta);
}

void
mfcFrame::setC (size_t index, float val)
{
  m_c[index] = val;
}

void
mfcFrame::setE (float val)
{
  m_e = val;
}

float
mfcFrame::getC (size_t index)
{
  return m_c[index];
}

float
mfcFrame::getE ()
{
  return m_e;
}

float
mfcFrame::operator- (mfcFrame other)
{
  float dist = 0;
  size_t i;

  for (i = 0; i < 10; i++)
    {
      dist += pow (getC (i) - other.getC (i), 2);
    }
  dist += pow (getE () - other.getE (), 2);

  return sqrt (dist);
}

// Returns a non empty string if there was an error
string mfc::add (mfcFrame & frame)
{
  m_frames.push_back (frame);
  return string ("");
}
```

```
// Returns a non empty string if there was an error
string mfc::add (string & filename)
{
  fstream
    file;

  // Open the file
  file.open (filename.c_str (), ios::in);
  if (!file)
    {
      return string ("Failed to open the file " + filename);
    }

  // Read until end of file
  int
  stage (0),
  col (0),
    numframes,
    timescale,
    bytes,
    type;
  char
    c;
  string
  line ("");
  mfcFrame
    f;

  while (!file.eof ())
    {
      file.read (&c, 1);

      if ((c != '\n') && (c != ' '))
 line += c;
      else if (line.length () > 0)
 {
   // The first four fields in the file are information about the file
   switch (stage)
     {
     case 0:
       numframes = atoi (line.c_str ());
       break;
     case 1:
       timescale = atoi (line.c_str ());
       break;
     case 2:
       bytes = atoi (line.c_str ());
       break;
     case 3:
       type = atoi (line.c_str ());
       break;
     default:
       // The rest go in the cycle of ten MFC values, then a log energy
       if (col < 10)
 {
   f.setC (col, atof (line.c_str ()));
   col++;
 }
      else
 {
   f.setE (atof (line.c_str ()));
   m_frames.push_back (f);
   col = 0;
 }
      break;
```

```
     }

   line = "";
   ++stage;
 }
    }

  // Print out some information about the file...
  cout << filename << ": " << numframes << " frames" << endl;
  return string ("");
}

mfcFrame & mfc::getFrame (size_t index)
{
  return m_frames[index];
}

size_t mfc::size ()
{
  return m_frames.size ();
}

mfcFrame mfc::centroid ()
{
  mfcFrame
    cent;
  size_t
    i;
  int
    j;

  for (i = 0; i < 10; i++)
    cent.setC (i, 0.0);
  cent.setE (0.0);

  for (i = 0; i < m_frames.size (); i++)
    {
      for (j = 0; j < 10; j++)
 cent.setC (j, cent.getC (j) + m_frames[i].getC (j));
      cent.setE (cent.getE () + m_frames[i].getE ());
    }

  for (i = 0; i < 10; i++)
    cent.setC (i, cent.getC (i) / m_frames.size ());
  cent.setE (cent.getE () / m_frames.size ());

  return cent;
}

void
mfc::clear ()
{
  m_frames.clear ();
}

ostream & operator<< (ostream & stream, mfcFrame frame)
{
  stream << "Frame: ";

  int i;
  for (i = 0; i < 10; i++)
    stream << frame.getC (i) << " ";

  stream << "energy: " << frame.getE ();
  return stream;
}
```

Code: c++/mfc.h

```
// Header for the MFC file parser

#include "world.h"
#ifndef MFC_H
#define MFC_H

class mfcFrame
{
public:
  mfcFrame ();
  mfcFrame (mfcFrame & other, float delta);

  void setC (size_t index, float val);
  void setE (float val);
  float getC (size_t index);
  float getE ();

  // Returns the "distance" between two frames
  float operator- (mfcFrame other);

private:
  float m_c[10];
  float m_e;
};

class mfc
{
public:
  string add (string & filename);
  string add (mfcFrame & frame);
    mfcFrame & getFrame (size_t index);

  // Returns the number of frames in the data pool
  size_t size ();

  // Return the centroid of this dataset
  mfcFrame centroid ();

  // Clear the data
  void clear ();

private:
    vector < mfcFrame > m_frames;
};

// Allow out stream stuff
ostream & operator<< (ostream & stream, mfcFrame frame);

#endif
```

Code: c++/world.h

```
#ifndef WORLD_H
#define WORLD_H

#include <iostream>
#include <fstream>
#include <string>
#include <vector>
#include <math.h>
#include <stdlib.h>
```

```
#include "mfc.h"
#include "lbg.h"

void load (mfc & data, string filename);

#endif
```

Code: c++/Makefile

```
all: codebook encode hmminit

codebook: gencode.o mfc.o lbg.o
 c++ gencode.o mfc.o lbg.o -o codebook

encode: encode.o mfc.o lbg.o
 c++ encode.o mfc.o lbg.o -o encode

gencode.o: gencode.cpp world.h
 c++ -c -Wall -ggdb gencode.cpp -o gencode.o

encode.o: encode.cpp world.h
 c++ -c -Wall -ggdb encode.cpp -o encode.o

mfc.o: mfc.cpp mfc.h world.h
 c++ -c -Wall -ggdb mfc.cpp -o mfc.o

lbg.o: lbg.cpp lbg.h world.h
 c++ -c -Wall -ggdb lbg.cpp -o lbg.o

hmminit: hmminit.cpp
 c++ hmminit.cpp -o hmminit

clean:
 rm -f *.o codebook encode
```

Output codebook

The output codebook is as follows:

```
32 100000 44 14106
17.5184 -9.61972 10.9811 12.5487 -20.3334 -13.3807 -14.0741 -21.5065 7.33428 0.97422
12.5844 -1.20952 12.7599 13.1692 -9.94967 -7.91391 -10.9386 -15.8806 3.07113 0.807378
20.5158 -9.7351 3.62021 15.8527 -24.6896 -2.2363 -18.3472 -13.0605 -0.807636 0.969249
21.1208 -7.20986 -6.09996 10.3476 -20.2856 4.05776 -10.0198 -11.3466 3.20897 0.877923
12.7301 -6.21918 16.1583 8.78379 -3.94845 -0.884173 -11.3109 -29.3998 -
0.658228 0.953744
17.4091 -2.94272 8.44412 4.90336 -9.91341 5.90885 -4.71809 -15.744 -7.91073 0.787865
24.4306 -4.54745 -13.7423 6.37143 -1.79758 -7.51856 -8.8303 -21.0768 5.84822 0.963372
20.2375 -3.00456 -3.20923 10.8505 -8.9082 -5.12877 -15.1718 -14.5714 -
0.96914 0.913506
12.7753 -2.90817 3.44214 21.644 8.59306 -8.61043 -10.8078 -3.65806 -13.1106 0.644082
7.49598 -3.80248 -0.681117 7.12422 6.64038 -7.75067 -12.3045 -8.14882 -
6.8389 0.357454
14.3265 1.23288 0.93998 12.1902 -1.74486 -3.24943 -1.53636 -6.63878 -
1.68243 0.762487
9.73785 2.76178 -3.26917 9.82138 7.27378 6.89372 0.0347545 -11.515 -0.161303 0.480143
17.1504 10.8616 5.68798 2.52324 -9.59537 -8.08671 -0.787987 -0.12787 -
2.41403 0.707007
13.7426 3.82317 7.19483 -1.00011 -10.8809 12.7695 -1.134 -3.28702 -4.06922 0.627633
9.87925 9.02714 5.27229 5.81882 -3.25409 5.50761 9.71562 6.36749 -2.35262 0.501449
4.26506 1.68982 8.98186 0.131684 5.45508 7.18203 -0.586789 1.3915 2.81816 0.45498
19.38 -4.89252 -6.95936 -10.8722 -25.5888 8.79123 8.70056 -5.64287 2.01727 0.956863
```

```
19.1046 -12.0034 -7.16167 -5.12748 -19.8278 13.0812 1.49002 -12.9461 4.81698 0.97126
19.7832 -4.90871 2.37965 -11.7441 -16.1058 -7.33486 17.7386 -12.8284 9.30193 0.98171
16.0383 -9.76116 -6.13497 -7.36874 -4.05799 4.88475 -1.10305 -13.1399 12.0321 0.969295
13.4222 -3.7279 19.9909 14.2036 -32.2308 -3.38185 -13.2581 -0.982058 -
1.17585 0.842049
16.4038 -13.5475 15.0297 13.9928 -27.922 -6.2536 -14.8237 -3.37106 -0.948236 0.941473
2.49029 7.76125 24.9521 11.7434 -19.848 -1.41258 -2.41829 -3.4535 -3.77819 0.575459
5.15913 -5.33255 3.61373 -0.564554 -20.3849 -1.13483 4.96248 -8.21963 -
17.2715 0.104859
-8.04763 -1.96283 -1.69638 -17.3481 2.20104 -0.701295 -6.1682 -3.87097 2.23939 0.46651
-9.46387 -5.03008 -1.72295 -10.2171 -3.26096 -8.83133 -0.602713 -6.10837 7.26646 0.4582
-8.71179 -13.2229 -0.0720176 0.952793 -1.81995 -4.7104 1.21669 0.577704 -
3.39437 0.463447
-9.06997 -5.50893 4.08051 -4.84227 -5.47672 -1.5031 -6.38152 -1.35329 3.62614 0.409615
3.92364 -0.625753 -3.69331 -4.94929 -0.22928 -5.23897 -6.88927 -7.48523 -
14.2767 0.0335002
3.67109 -3.62674 -3.78139 -12.243 -5.13097 -1.72663 -0.482378 -2.60503 -
7.34609 0.249361
4.91593 2.84004 -5.67108 -4.29014 -1.65023 5.66779 4.59955 5.64348 -3.67165 0.161052
-1.31697 2.82501 -4.08869 0.238118 -0.299205 -1.08468 2.61686 4.26842 1.67924 0.0271404
```

Code: c++/codebook.mfc

Distortion values

The following distortion values are calculated for the various phases in the codebook generation:

```
Distortion for 1 entry codebook: total: 11519.6 average: 27.758
Distortion for 2 entry codebook: total: 9968.5 average: 24.0205
Distortion for 4 entry codebook: total: 8666.67 average: 20.8835
Distortion for 8 entry codebook: total: 7508.66 average: 18.0932
Distortion for 16 entry codebook: total: 6309.63 average: 15.2039
Distortion for 32 entry codebook: total: 5168.91 average: 12.4552
```

Encoding the nine training files

This script was used to encode the nine training files:

#!/bin/bash for item in ‘ls s*.mfc‘ do ./encode codebook.mfc $item ‘echo $item | sed ’s/mfc$/enc/’‘ done

Encoded files

These are the encoded files:

s014d001

28 27 27 24 28 22 23 25 27 25 25 25 9 9 9 9 13 13 13 13 13 13 11 11 11 11 11 11 11 5 5 5 5
24 12 27 27 10 12 25 13 13 13 13 13 13 10 10 10 10 10 9 13 25 25 28 28 28

s014d002

25 25 24 24 24 24 24 24 24 24 27 13 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 25 15 15 25 11 9
14 28 28 28 28 28

s014d003

24 28 24 21 24 24 28 25 28 28 11 21 11 28 28 21 28 28 8 22 22 22 27 27 27 13 9 10 10 10 10 10 10 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 20 20 28 20 20 28 28 28 28

s017d001

28 28 11 11 11 11 3 11 5 3 5 5 5 5 11 11 11 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 13 27 19 27 26 26 27

s017d002

22 22 24 28 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 10 28 10 10 28 10 24 26 9 9

s017d003

28 26 28 21 28 28 28 22 27 27 27 12 12 0 0 0 0 0 0 0 0 0 0 20 20 20 20 20 20 20 20 20 20 20 20 28 11 22 22 22 22 22 22 22 22 26

s029d001

28 9 9 9 11 11 11 11 11 11 11 5 11 11 11 11 11 12 12 12 0 0 0 0 0 0 0 24 24 24 24 15 24 25 24 25 25 26 26 25 24 22 25 25 26

s029d002

21 21 21 24 28 28 27 27 24 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 12 0 12 24 11 25 25 25 25 25 23 22 22

s029d003

27 26 27 27 28 26 26 26 26 21 28 22 22 27 25 24 12 12 12 12 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 20 20 20 20 20 15 15 9 13 25 25 25 22 25 26 25

s031d00x

24 24 28 28 28 28 28 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 24 13 13 13 13 10 25 25 28

Verfication

The LBG VQ algorithm used in this part was verified by the use of selected dummy MFC files which a much smaller number of samples. This allowed me to manually calculate the correct cloud centroids et cetera, and then verify this against the program.

HMM initialization

The following code was used for HMM initialization:

Code: c++/hmminit.cpp

```
#include <string>
#include <vector>
#include <iostream>
#include <fstream>

typedef vector < float >matrixcol;

matrixcol readFile (string filename);
void readInFile (string filename, vector < matrixcol > &mat, int &size);

int
main (int argc, char *argv[])
{
  vector < matrixcol > obs, a, b;
  matrixcol p, file;
  int nobs (0);
  float rprob;

  // Read in the encoded files (not that these columns are not padded, this
  // is implicitly done later)
  readInFile (argv[1], obs, nobs);
  readInFile (argv[2], obs, nobs);
  readInFile (argv[3], obs, nobs);

  // What was the maximum length of a vector read in
  cout << "Maximum length: " << nobs << endl;

  // Generate p (aka pi)
  for (int i = 0; i < nobs - 1; i++)
    {
      p.push_back (0.0);
    }
  p.push_back (1.0);

  // Generate A
  a.resize (nobs);
  for (int i = 0; i < nobs; i++)
    {
      a[i].resize (nobs);

      for (int j = 0; j < nobs; j++)
 {
   if (i == j)
     a[i][j] = 0.5;
   else if (i - 1 == j)
     a[i][j] = 0.5;
   else
     a[i][j] = 0.0;
 }
    }
  cout << nobs * nobs << " values inserted into A matrix" << endl;

  // Initialise B
  b.resize (nobs);
  for (int i = 0; i < nobs; i++)
    {
      b[i].resize (nobs);

      for (int j = 0; j < nobs; j++)
 {
   b[i][j] = 1 / 320;
 }
    }
  cout << nobs * nobs << " values inserted into B matrix" << endl;
```

```
 // Calculate remaining probability
 rprob = 1.0 - (32.0 / 320.0);
 cout << "Remaining probability is: " << rprob << endl;

 // Regenerate B
 vector < matrixcol > count;
 count.resize (nobs);
 vector < int >totalcount;
 totalcount.resize (nobs);

 for (int v = 0; v < nobs; v++)
   {
     count[v].resize (nobs);

     for (int o = 0; o < nobs; o++)
{
  int i = o / nobs * obs.size ();
  count[i][v]++;
  totalcount[i]++;
}
   }

 // Put the new values into B
 for (int i = 0; i < nobs; i++)
   {
     for (int j = 0; j < nobs; j++)
{
  b[i][j] += count[i][j] * rprob / totalcount[j];
}
   }

 // Write this out to a file
 fstream files;

 files.open (argv[4], ios::out);
 if (!files)
   {
     cerr << "Could not open output file: " << argv[4] << endl;
     exit (3);
   }

 // PI
 files << "pi = ";
 for (int i = 0; i < nobs; i++)
   {
     files << p[i] << " ";
   }
 files << endl << endl;

 // A
 files << "A = " << endl;
 for (int i = 0; i < nobs; i++)
   {
     for (int j = 0; j < nobs; j++)
{
  files << a[i][j] << " ";
}
     files << endl;
   }
 files << endl << endl;

 // B
 files << "B = " << endl;
 for (int i = 0; i < nobs; i++)
   {
```

```
      for (int j = 0; j < nobs; j++)
 {
   files << b[i][j] << " ";
 }
      files << endl;
    }
  files << endl << endl;
}

// Read in a file into the matrix
void
readInFile (string filename, vector < matrixcol > &mat, int &size)
{
  matrixcol temp;

  temp = readFile (filename);
  mat.push_back (temp);

  cout << "Read a vector of length: " << temp.size () << endl;
  if (temp.size () > size)
    {
      size = temp.size ();
    }
}

// Read in the given file into a vector
matrixcol
readFile (string filename)
{
  fstream file;
  matrixcol retval;

  // Open the file
  file.open (filename.c_str (), ios::in);
  if (!file)
    {
      cerr << "Could not open file " << file << " for reading" << endl;
      exit (2);
    }

  // Read all of the values from the file
  while (!file.eof ())
    {
      float temp;
      file >> temp;

      retval.resize (retval.size () + 1);
      retval[retval.size ()] = temp;
    }

  // Cleanup and return
  file.close ();
  return retval;
}
```