基于 DTW 算法的孤立词语音识别

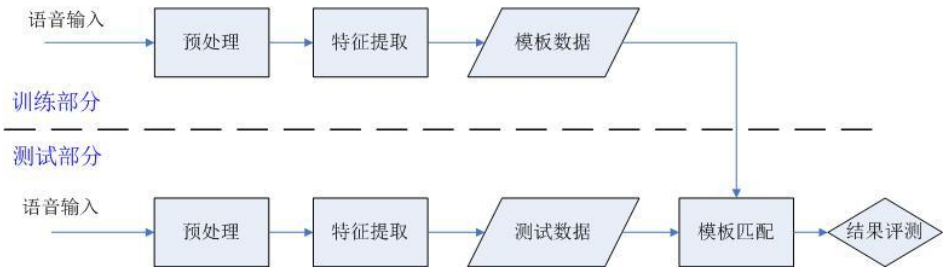
一、前言

让计算机像人类一样去理解自然语言一直是科学研究者们的梦想，近三十年来，语音识别技术取得了显著的进步。现如今，语音识别技术正在逐步应用于工业、家电、通信、汽车电子、医疗、家庭服务、消费电子产品等各个领域。2011 年 10 月发布的 iphone 4s Siri 技术，更是将语音识别推向了一个新的应用高潮。目前各类手机智能语音助手层出不穷，比如 Google Now、讯飞灵犀、百度手机语音助手等。即将上市的 Google Glass 也将语音识别作为标配。

我们在多媒体技术课程中，学习了语音识别的基础知识。语音识别的算法包括基于统计框架的隐马尔科夫模型（Hidden Markov Model）、神经网络（Neural Network）、动态时间弯折（Dynamic Time Warping, DTW）等。本实验以较为简单的孤立词语音识别系统[1]的实现作为基础，要求学生了解 Cool Edit 软件和剑桥大学 HTK 工具包，学会利用 Cool Edit 录制和编辑音频文件，学会利用 HTK 的 HCopy 提取语音特征提取工具提取语音的特征，生成特征文件，并深入理解和应用 DTW 算法实现孤立词语音识别，最终对识别算法进行实验评测。DTW 是早期的基于模板匹配的语音识别算法，主要用于孤立词语音识别领域。DTW 在数据挖掘领域也有着广泛的应用。

二、实验内容

任务描述：编程实现能够识别 0-9 这十个数字的孤立词语音识别系统。如用户通过麦克风语音输入“0”，系统能够给出识别结果。



实验内容（上图为一个孤立词识别系统搭建示意图）：

1. 数据准备：录制 0-9 这十个数字的音频文件（采样率 16KHz，量化位数 16bit，单声道）
2. 预处理（端点检测）：检测语音的开始端点和结束端点，仅保留语音部分
3. 特征提取：提取音频文件的 MFCC 特征，用工具 HCopy 生成*.mfc 特征文件。

[1] 语音识别可分为孤立词语音识别和连续语音识别，前者识别的对象为事先指定的孤立词汇，如数字等，后者识别的对象则是连续词汇流，如一个句子等。

4. 模板匹配：完成动态时间规整算法（Dynamic Time Warping, DTW 算法），实现测试数据与模板的匹配，得出测试数据的识别结果
5. 识别结果测试评价：对识别结果进行统计评测

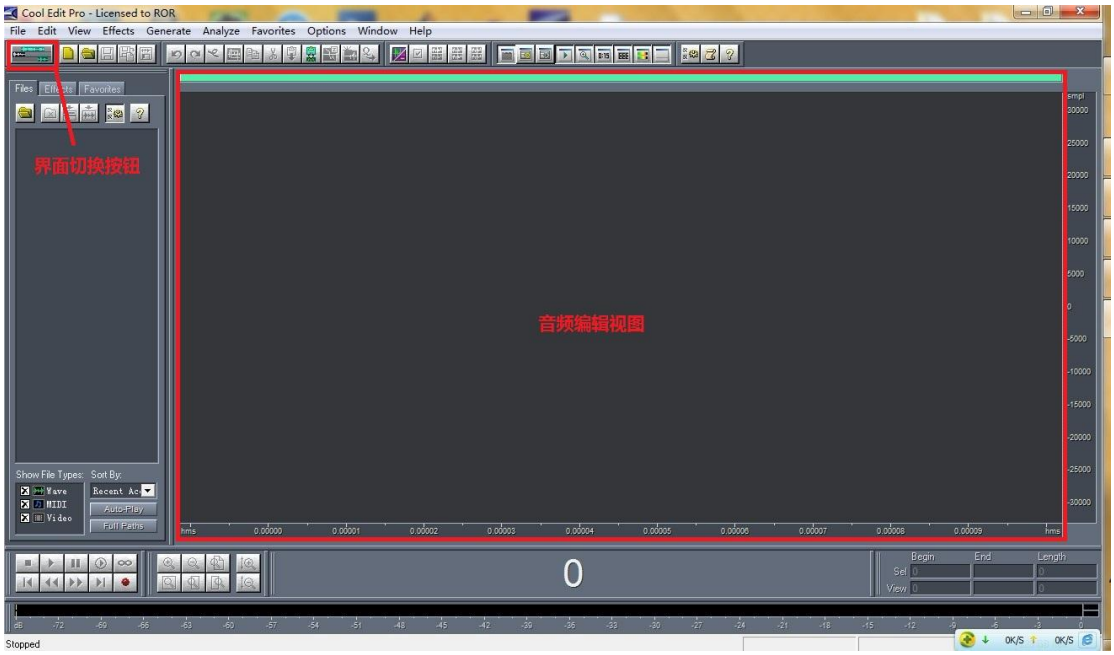
三、实验步骤

1、数据准备和预处理

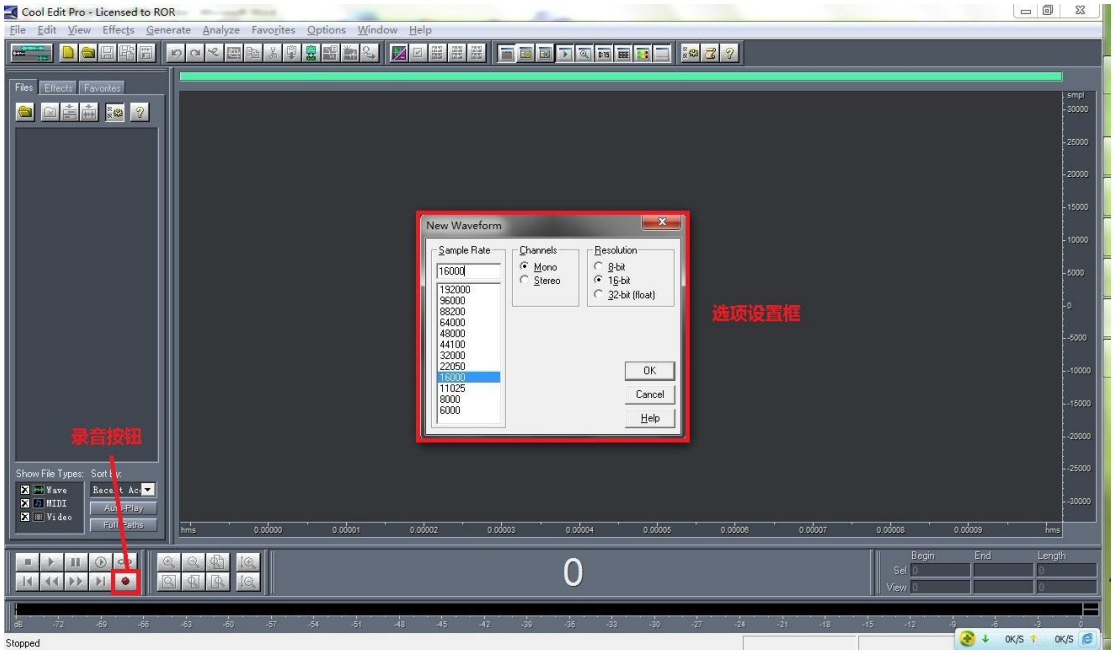
利用音频处理软件 Cool Edit 录制 0-9 这十个数字的音频文件，并只保留语音部分。请预先安装附件的 Cool Edit Pro 2.1（参考安装说明）。以录制数字“0”为例：

a. 打开 Cool Edit 软件，点击左上角的界面切换按钮使主界面切换到编辑视图，如图一所示。
b. 点击左下角的录音按钮，在弹出的选项设置框中选择需要录制的音频类型：采样率（sample rate）16000Hz，单声道（mono），量化位数（resolution）16bit，如图二所示。
c. 点击选项设置框上的 OK，开始对麦克风说话，可以看到主界面上有波形产生，如图三所示，点击左下角的停止（Stop）按钮完成该次录音（快捷键为空格键 Sapce）。如对该次录音不满意，可选择该次录音内容（快捷键 Cltl+A），删除该次录音（快捷键 Delete）。点击录音按钮再次录制。
d. 从图三中我们可以看到，只有图中鼠标选中的区域才是包含语音的部分，其它区域通常为静音等其它音频类型，我们需要去除它们以避免对语音识别造成干扰。因此在进行语音识别前，通常需要检测已录制的音频中语音的开始结束位置，这个过程被称为端点检测。在语音处理中，我们常常使用短时能量、短时过零率等音频特征来自动地完成端点检测。在这里，为简单起见，我们手动完成这个过程：用鼠标选择非语音区域并删除（快捷键 Delete），如图四所示。
e. 保存处理完的音频文件，如图五所示，注意用合适的文件命名规则。

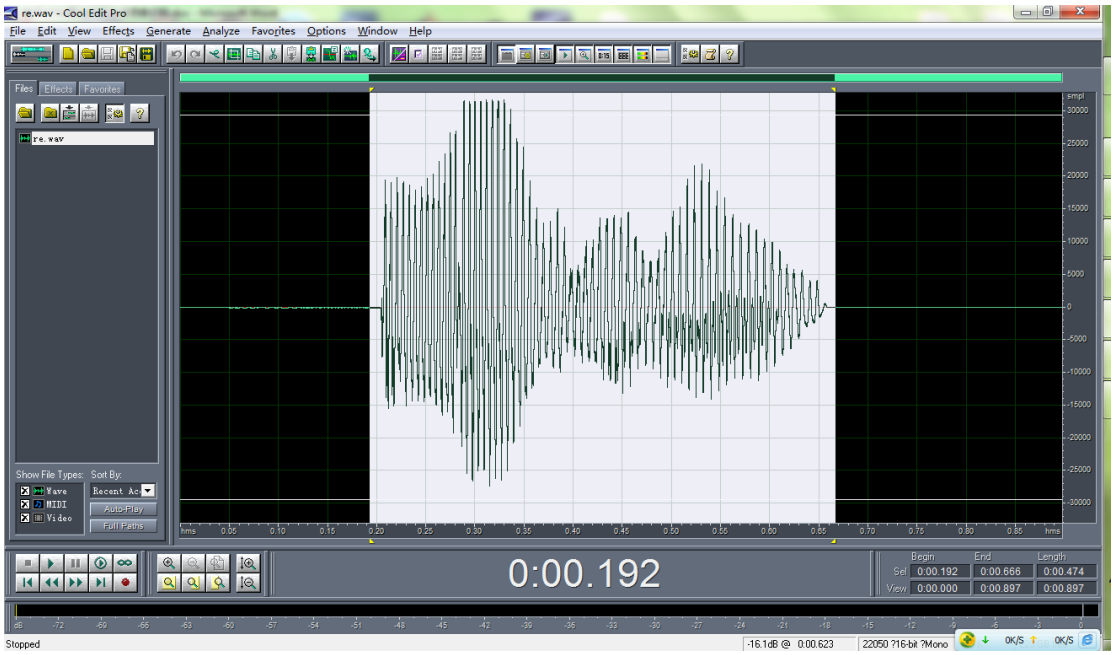
文件命名规则：好的命名方式有利于更好地管理文件。我们首先要录制 0-9 这十个数字各一遍作为模板（template）数据，可命名为 number_template.wav，如 0_template.wav。然后我们需要再录制测试（test）数据，0-9 每个数字各 4 遍，可命名为 number_test1/2/3/4.wav，如 0_test1.wav, 0_test2.wav, … , 0_test4.wav。注意每一组数（0-9 各一次）的录制要在同一时段内完成，因为人的声音在不同的时段是不同的。


Cool Edit Pro - Licensed to ROR
File Edit View Effects Generate Analyze Favorites Options Window Help
界面切换按钮
音频编辑视图
Stopped

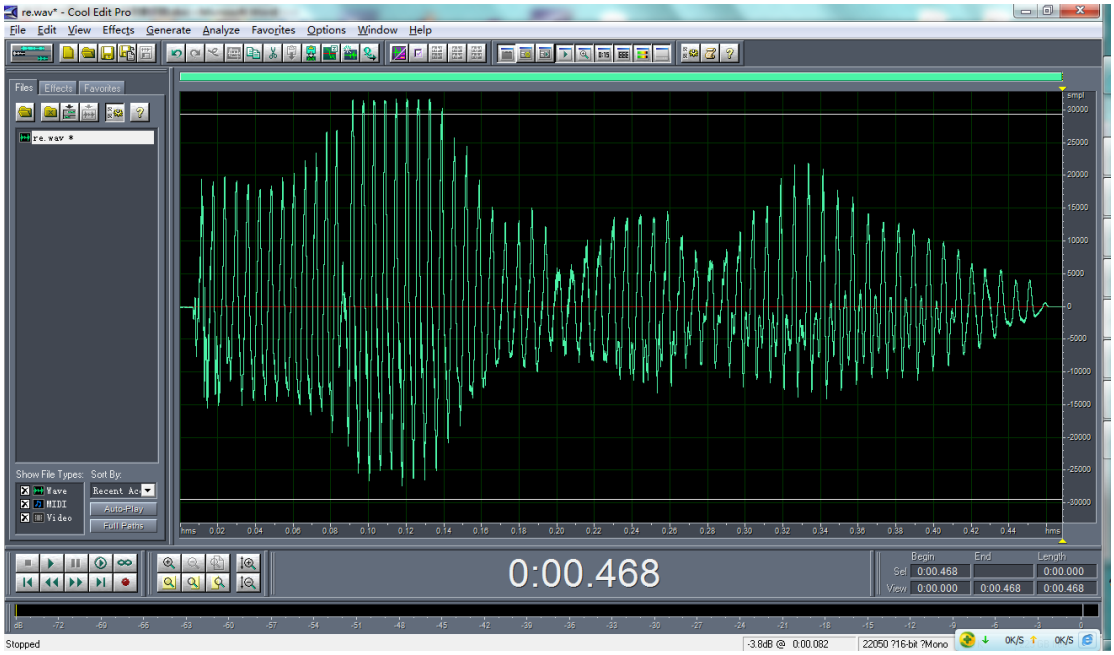
图一


Cool Edit Pro - Licensed to ROR
New Waveform
Sample Rate
Channels
Resolution
选项设置框
录音按钮
Stopped

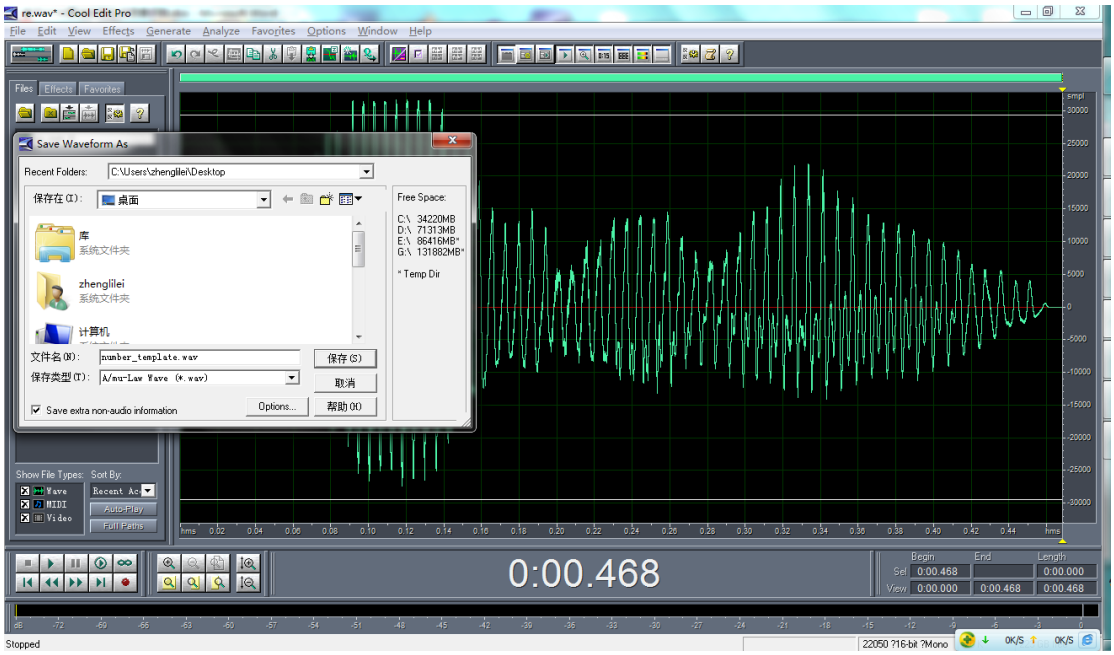
图二


re.wav - Cool Edit Pro
File Edit View Effects Generate Analyze Favorites Options Window Help
0:00.192
Stopped

图三


re.wav* - Cool Edit Pro
File Edit View Effects Generate Analyze Favorites Options Window Help
0:00.468
Stopped

图四



图五

2、特征提取

利用 HTK 工具包中的 HCopy 提取音频文件的 MFCC（Mel-Frequency Cepstral Coefficients，梅尔频率倒谱系数）特征。我们常说，声音的频率越高，那么我们听到的声音越高。但是，人耳所听到的声音的高低与声音的频率并不成线性正比关系，而使用梅尔频率尺度则更符合人耳的听觉特性[2]。MFCC 特征正是模拟人耳通过频谱分析分辨不同语音的一种常用于语音识别的语音特征形式。

我们使用 HTK 工具包提供的 HCopy 实现 MFCC 特征的提取。我们在这里需要使用的命令格式为：

```
HCopy –C config_MFCC.cfg –S audios.scp
```

其中-C 表示读取配置（config）文件 config_MFCC.cfg，-S 表示读取列表文件 audios.scp。其中配置文件 config_MFCC.cfg 对需要提取的 MFCC 特征进行了参数设定[3]；列表文件则给出了需要提取特征的音频文件的路径以及得到的特征文件存放的路径，该文件的每一行都有如下形式:

```
wav 音频文件所在路径[制表符\t]mfc 特征文件存放路径[换行符\n]
```

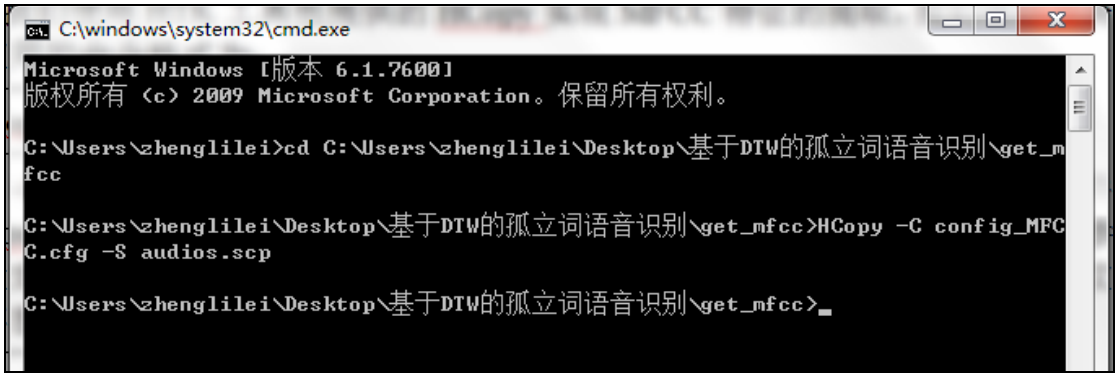
例如:

[2] 关于语音信号的相关知识，有兴趣的同学可以阅读附件中的《语音信号处理》。梅尔频率倒谱系数的相关内容在第 54 页

[3] 用文本文件可查看该文件内容，各参数的意义详见《The HTK Book》。下载地址：http://htk.eng.cam.ac.uk/docs/docs.shtml

```
audio\0_template.wav    mfc\0_template.mfc
audio\1_template.wav    mfc\1_template.mfc
….
```

提取特征过程的示例：在 HCopy 的同级路径下建立两个文件夹 audio 和 mfc，将所有的 wav 音频文件都拷贝到 audio 文件夹下，并编辑好列表文件 audios.scp，从命令行进入 HCopy 所在路径，调用 HCopy，就可在 mfc 文件夹下得到音频文件对应的*.mfc 特征文件：



MFCC 特征是按语音帧计算的。这是因为研究发现，语音信号在短时间内频谱特性保持平稳，即具有短时平稳特性。所以在实际处理时通常将语音信号分成很小的时间段(约 10~30ms)，称之为“帧”，作为语音信号处理的最小单位，帧与帧的非重叠部分称为帧移，而将语音信号分成若干帧的过程称为分帧。在本实验的设置中，帧长为 25ms，帧移为 10ms。以一段 1s 的音频为例，可被划分为 98 帧（思考为什么不是 100 帧？），每一帧提取 39 个反映不同语音特性的参数，即该段音频可以用 98 个 39 维的帧向量表示。

3、模板匹配

DTW 原理

在用帧向量表示语音信号之后，我们就可以把测试数据的帧向量与 10 个模板数据的帧向量依次进行比对，并根据比对的结果判断测试数据与哪一个模板最相似，以该模板对应的数值作为测试数据的识别结果。在本实验中，我们使用经典的 DTW 算法实现帧向量的比对。

考虑到人在说同一个词语时存在的说话速度不均匀的问题，日本学者 Itakura 提出了一种把时间规整和距离计算结合起来的非线性规整技术，称之为 DTW 算法。假设某测试数据由 I 帧向量表示，某模板数据由 J 帧向量表示，则 DTW 算

法就是要寻求一个时间规整函数 $j=w(i)$, (1<=i<=I, 1<=j<=J)，它将测试数据的第 i 帧向量非线性地映射到模板数据的第 j 帧向量，并使该函数满足：

$$D = \min_{w(i)} \sum_{i=1}^{I} d(T(i), R(w(i)))$$

其中 $d(T(i), R(w(i)))$ 是第 i 帧测试向量 $T(i)$ 和第 j 帧模板向量 $R(w(i))$ 之间的欧式距离。D 则是处于最优时间规整条件下的向量累积距离。

由于 DTW 不断地计算向量间的距离以寻找测试数据和模板数据之间的最优匹配（最佳规整），得到的数据匹配是有最小累积距离的规整函数，所以可以反映出测试数据和模板数据之间的最大声学相似性。

DTW 算法采用动态规划算法（Dynamic Programming, DP）实现，其原理如下图：

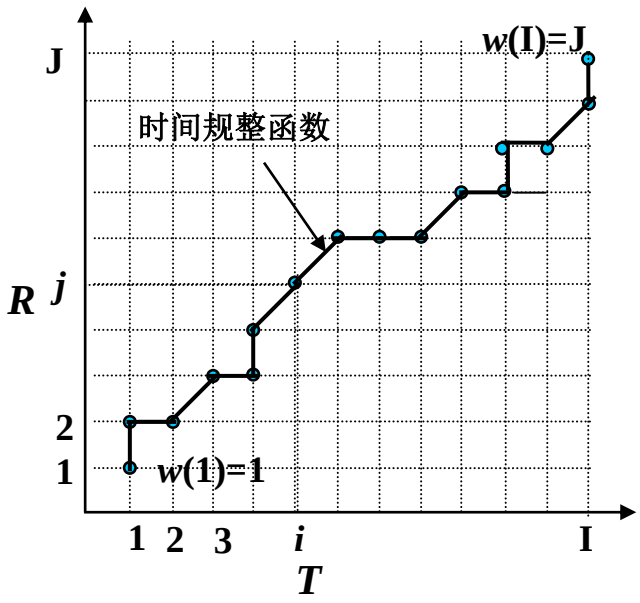


图 a DTW 算法原理图

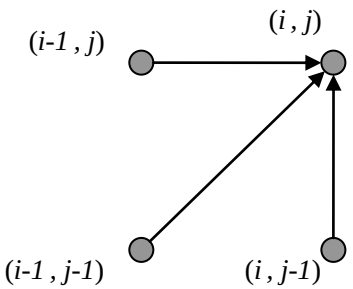


图 b 局部约束条件

可以从图中看到，规整函数的起始点为（1,1），终点为（I,J）。寻找最佳规整的过程可分为两步：一是计算测试数据和模板数据各帧向量之间的距离 $d(T(i), R(j))$，即求出帧间距离矩阵；二是在这个距离矩阵中找出一条最佳路径，使这条路径上的累加距离和最小。搜索最佳路径的过程可描述如下：从点（1,1）出发，在局部约束条件下（如上图 b），路径上某个点（i,j）的前一个点只能是（i-1,j），（i-1,j-1）或（i,j-1），且点（i,j）选择这三个点中有最小累加距离的点作为其前向点：

$$D(i, j) = d(T(i), R(j)) + \min\{D(i-1, j), D(i-1, j-1), D(i, j-1)\}$$

搜索直到（I,J）结束，且 D(I,J) 就是最佳路径所对应的累加距离。这个累加距离越小，说明测试数据与模板数据越相似。

C 语言实现

本实验建议使用 C 语言实现，主要过程可分为三部分：*.mfc 特征文件的读取，模板匹配（DTW 算法）和结果输出。

a. *.mfc 特征文件的读取

前文中我们知道*.mfc特征文件存储着语音的帧向量。HTK规定了专属的二进制数据存储方式，在本实验中我们直接提供读取*.mfc的C语言代码（binaryread.h）。该部分对应函数为：

```
int ReadHtk( string InputFile, float FrameV[][DVECTOR+1])
```

实现的功能是：将指定的*.mfc 文件 InputFile 读入到一个二维数组 FrameV 中，数组的行表示语音的帧数，而每一列为一个 39 维的帧向量。

b. 模板匹配

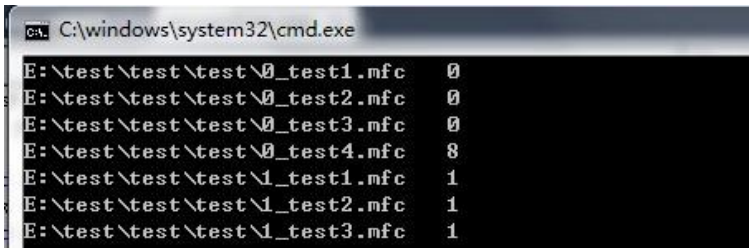
实现测试数据和模板数据之间的匹配算法。该部分对应两个函数：

```
float CalculateDistance( float Vector1[], float Vector2[] )
float Run_DTW( float TemplateV[][DVECTOR+1], int NFTemplate, float
TestV[][DVECTOR+1], int NFTest )
```

后者调用前者计算帧向量之间的欧式距离。这两个函数要求自己独立完成代码的编写。

c. 结果输出

将识别结果输出到屏幕上。在主函数 main()中给出了此部分代码，可供参考。输出结果形式为： [测试文件名 识别结果]



可以看到，0_test4.mfc 的识别结果为 8，错误！而其它文件的测试结果都是正确的。

4、结果评测

对所有的识别结果进行统计评测，可参考如下表格：

数字	0	1	2	3	4	5	6	7	8	9	0~9
正确数	3	3	1	4	3	4	4	4	4	4	34
错误数	1	1	3	0	1	0	0	0	0	0	6
正确率	0.75	0.75	0.25	1.00	0.75	1.00	1.00	1.00	1.00	1.00	0.85

需要汇报的结果包括：

a) 使用课程提供的模板与测试数据，进行识别测试，测试结果总结成上述表格。
b) 录制自己的模板与测试数据，进行识别测试，测试结果总结成上述表格。
c) 使用课程提供的模板与自行录制的测试数据，进行识别测试，测试结果总结

成上述表格。

d) 比较与分析 a)、b)和 c)的评测结果。

5、书面报告撰写内容

a) 封面写清楚姓名、班级、学号。
b) 核心代码内容，即 CalculateDistance 和 Run_DTW 两个函数的详细代码，并加以详细注释。
c) 按照结果评测要求，汇报与分析各项实验结果。
d) 请回答：DTW 是作为经典的早期语音识别算法，我们可以看到，识别效果仍有改善空间，你有何种想法可以提升识别率？如有时间，可以尝试编写代码实现自己的想法，并进行测试，检验效果。
e) 经过三个多媒体课程实验，你有何心得体会？