МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ
ФЕДЕРАЛЬНОЕ АГЕНТСТВО ПО ОБРАЗОВАНИЮ
ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ
ТЕХНОЛОГИЧЕСКИЙ ИНСТИТУТ
ЮНОГО ФЕДЕРАЛЬНОГО УНИВЕРСИТЕТА В Г. ТАГАНРОГЕ

Факультет АВТОМАТИКИ И ВЫЧИСЛИТЕЛЬНОЙ ТЕХНИКИ
Кафедра СИСТЕМ АВТОМАТИЧЕСКОГО УПРАВЛЕНИЯ

К защите допустить:.
Зав. кафедрой ___ д.т.н., проф. Финаев В. И.
« » 2008 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ
На академическую степень бакалавр техники и технологии
на тему:

СИСТЕМА УПРАВЛЕНИЯ РАСПОЗНАВАНИЕМ РЕЧЕВОЙ ИНФОРМАЦИИ

Руководитель работы
д.т.н., проф.В.И.Финаев
Студент гр. А-14 Келускар Пунам Нарян
(фамилия, имя, отчество, группа)
«____»_____2008г.

Таганрог 2008

УДК 621.395

АННОТАЦИЯ

Проект содержит 83 страницы машинописного текста, 12 рисунков, 4 таблиц, 13 источников литературы.

Эта работа посвящена преобразованию устной речи в электронный текст. В ней рассматриваются основные методы автоматического распознавания речевой информации. В этой работе также разработаны и реализованы алгоритм, модель и информационное обеспечение для распознавания изолированных слов. Пользовательский интерфейс выполнен в среде visual C#.net, а программная реализация - в пакете Matlab.

UDK 621.395

SUMMARY

This project contains 83 pages of text, 12 diagrams, 4 tables and 13 literature sources.

This bachelor’s project is dedicated to problem of automatic speech recognition. Described basic speech recognition methods such as HMM(Hidden Markov Model) and neural network method. To solve this real-world problem algorithm was developed using Matlab and was further implemented in visual C#.net.

УДК 621.395

РЕФЕРАТ

СКРЫТАЯ МОДЕЛЬ МАРКОВА, УПРАВЛЕНИЕ РЕЧЕВОЙ ИНФОРМАЦИЕЙ

В выпускной работе разработана система автоматического распознавания речевой информации. Так же была разработана программа, написанная для системы MATLAB, распознающая речевую информацию и строящая графики входных и выходных величин. С помощью этой программы был исследован алгоритм распознавания речи.

Решены задачи экономической целесообразности проекта.

Решены задачи обеспечения безопасности труда инженера-проектировщика.

Оглавление

ВВЕДЕНИЕ

В настоящее время научное сообщество вкладывает гигантское количество денег в развитие ноу-хау и научно-исследовательские разработки для решения проблем автоматического распознавания и понимания речи. Это стимулируется практическими требованиями, связанными с созданием системы военного и коммерческого назначения. Не касаясь первого из них, можно указать, что только в европейском сообществе объем продаж систем гражданского назначения составляет несколько миллиардов долларов. При этом следует обратить внимание на то, что в практическом использовании отсутствуют системы, считающиеся по непонятным причинам вершиной развития систем автоматического распознавания речи. Это системы, которые можно назвать демонстрационными и которые 50 лет назад назывались «фонетическими печатающими машинками». Их целью является перевод речи в соответствующий письменный текст.

Если рассматривать классическую схему «наука – технологии – практические системы», то, прежде всего, надо определить те условия, в которых будет работать практическая система автоматического распознавания или понимания речи. Наиболее серьезные проблемы возникают при условиях:

-произвольный, наивный пользователь;

-спонтанная речь, сопровождаемая аграмматизмами и речевым «мусором»;

-наличие акустических помех и искажений, в том числе меняющихся;

-наличие речевых помех.

С другой стороны необходимо определить важность задачи, ее научную и прикладную фундаментальность, связь с другими областями знаний. При этом необходимо учитывать состояние научно-промышленного потенциала, его возможности. Ни для кого не секрет, что правильно поставленная задача – это уже половина решения.

В настоящее время в среде «речевиков» сложилось представление, что конечной и высшей целью является создание именно «фонетической печатающей машинки», а универсальным методом решения всех речевых проблем являются «скрытые Марковские модели» (СММ).

Остановимся на возможностях и недостатках соответствующих систем автоматического распознавания речи (анонсируемые сегодня возможностью распознавания сотен и даже тысяч слов с надежностью до 98%).

От пользователя требуется предварительная настройка системы на его голос от нескольких десятков минут до нескольких часов предварительного наговаривания текстов.

Так как слова, включенные даже в хорошо и аккуратно произносимый текст, оказываются как бы плавающими в океане омонимии, то количество ошибок (словесных) возрастает приблизительно в 5 раз. Беглое отслеживание таких ошибок, кроме случаев возникновения нелепых текстов, уже затруднительно. Аппарат коррекции ошибок в большинстве демонстрационных систем слабо отлажен.

Были упоминания, что даже для хорошо организованных спонтанно произнесенных текстов вероятность правильного распознавания слов не превышает одной трети.

Наконец, время обработки введенного отрезка речи в таких системах может занимать минуты.

Все сказанное говорит о том, что в качестве конечной цели предлагаемые демонстрационные системы «речь-текст» вряд ли представляют интерес. Это не исключает возможности использования их в качестве полигона для оценки научных идей, но в этом случае должны отчетливо излагаться те модели, которые закладываются в данные системы автоматического распознавания и каким образом должна проверяться их практическая перспективность. Таким образом, мы переходим на противоположный конец триады «практические системы – речевые технологии – речевая наука».

Целью данной бакалаврской работы является распознавание речевой информации с помощью систем управления, использующих системы автоматического распознавания речевых команд на основе скрытых Марковских моделей (СММ) на компьютере. При фиксированной на сегодняшний день аппаратной базе подобных систем распознавания и учитывая тенденции её развития в ближайшем будущем, рассматривается один из наиболее важных блоков таких систем - блок обучения СММ тренировочными последовательностями. От успешного решения им задачи обучения Марковской модели напрямую зависит качество работы системы распознавания. В задаче обучения СММ на данный момент есть две серьёзные проблемы: стандартные методы её решения (метод Баума-Велча или EM-процедура) являются методами локальной оптимизации, (то есть, не способны выйти за пределы локальных экстремумов функции) и сильно зависимы от стартовых параметров.

В поисках решения данной задачи в работе проводится разработка программного обеспечения для систем распознавания речевых команд.

Для достижения поставленной цели в работе решены следующие основные задачи:

• Исследованы алгоритмы обучения СММ тренировочными последовательностями.

• Разработаны методы, направленные на дальнейшее повышение эффективности и качества работы данного алгоритма в контексте рассматриваемой задачи.

В настоящее время работы по распознаванию речи не только не потеряли актуальности, но и развиваются широким фронтом, находя для себя множество областей для практического применения. Сейчас можно выделить 4 сравнительно изолированных направления в области развития речевых технологий :

1. Распознавание речи - т.е. преобразование речевого акустического сигнала в цепочку символов, слов. Эти системы могут быть

охарактеризованы по ряду параметров. Прежде всего это объём словаря: малые объёмы до 20 слов, большие - тысячи и десятки тысяч. Количество дикторов: от одного до произвольного. Стиль произнесения: от изолированных команд до слитной речи и от чтения до спонтанной речи. Коэффициент ветвления, т.е. величина, определяющая количество гипотез на каждом шаге распознавания: от малых величин (<10÷15) до больших (>100÷200). Отношение сигнал/шум от больших (>30 дБ) до низких (<10 дБ). Качество каналов связи: от высококачественного микрофона до телефонного канала. Качество работы систем распознавания речи обычно характеризуется надёжностью распознавания слов, или, что то же самое, процентом ошибок.

2. Определение индивидуальности говорящего. Эти системы делятся на два класса: верификация говорящего (т.е. подтверждение его личности) и идентификация говорящего (т.е. определение его личности из заранее ограниченного числа людей). Оба эти класса далее могут быть разделены на тексто-зависимые и тексто-независимые. Следующий характеристический параметр - объём парольной фразы. Два других (как и в распознавании речи): отношение сигнал/шум и качество канала связи. Качество работы систем верификации/идентификации говорящего характеризуется двумя величинами: вероятностью не опознания «своего» диктора и вероятностью принятия «чужого» диктора за своего.

3. Синтез речи. Практически существует два класса:

1) Воспроизведение записанного в той или иной форме ограниченного числа сообщений;

2) Синтез речи по тексту. Синтезаторы характеризуются по следующим параметрам: разборчивость (словесная или слоговая), естественность звучания, помехоустойчивость.

4. Компрессия речи. Основной (и единственный) классификационный признак этих систем, это степень компрессии: от низкой (32-16 кбит/сек) до высокой (1200-2400 кбит/сек и ниже). Качество работы систем компрессии речи характеризуется, прежде всего, разборчивостью компрессированной

речи. Дополнительными характеристиками очень важными в ряде приложений являются узнаваемость голоса говорящего и возможность определения стрессового уровня говорящего.

В данной работе рассматриваются системы первой группы - системы распознавания речи и их частный случай - системы распознавания речевых команд, т.е. распознавание изолированных слов, а не слитной речи. Такие системы весьма полезны на практике, и возросшая необходимость в них связана в первую очередь с появлением большого количества доступных человеку разнообразных устройств (персональные, мобильные и карманные компьютеры, коммуникаторы и мобильные телефоны, игровые и многофункциональные мультимедийные устройства с достаточной вычислительной мощностью) в сочетании с бурным развитием телекоммуникаций в современном мире. Растёт важность массового внедрения новых интерфейсов взаимодействия человека с техническими системами, поскольку традиционные интерфейсы во многом уже достигли своего совершенства, а вместе с ним и своих пределов. При традиционно высокой значимости информации, поступающей к нам через органы зрения, и её высокой доли среди всей сенсорной информации, считающейся равной порядка 85%, этот канал восприятия человека становится в значительной степени перегружен, и первоочередной альтернативой здесь видится коммуникация именно по акустическому каналу. Кроме того, системы распознавания (а также синтеза) речи также крайне важны для людей с ограниченным зрением, и эта ниша для их применения активно развивается, прежде всего, в области мобильной телефонии, а также в бытовой технике (для управления разнообразными домашними устройствами). Для помощи таким людям производители вводят в свои устройства возможности управления посредством голосовых команд, а также дублирования экранной информации голосом. И в первую очередь от таких продуктов требуется распознавание ограниченного набора команд пользователя, а не слитной речи с большим или неограниченным словарём. Благодаря стандартизации

платформ и операционных систем телефонов расширяется круг сторонних разработчиков программных продуктов с данной функциональностью.

Аппаратная база таких систем также может быть весьма разнообразной и оказывать заметное влияние на итоговую эффективность системы распознавания в целом. Аппаратная часть систем распознавания уже не является самым узким местом и способна выполнять качественную оцифровку речевого сигнала с требуемыми параметрами, а также обеспечивает требуемые вычислительные мощности для реализации необходимых алгоритмов предобработки и работы с моделями слов.

1. АНАЛИЗ ТЕХНИЧЕСКОГО ЗАДАНИЯ

Традиционная модель автоматического распознавания речи (АРР) предполагает, что путем отслеживания акустических параметров и применения одного из средств поиска по набору эталонов фонематических сегментов можно установить фонематические ряды. Затем эти ряды могут быть применены для проведения лингвистического анализа на более высоком ярусе выделения слов, фраз и смысла высказываний. Успешное понимание произнесенных предложений (фраз) включает употребление той или иной лингвистической структуры в сочетании с наиболее достоверной звуковой информацией.

При автоматическом распознавании речи большие трудности представляют собой процессы обнаружения и идентификации некоторых групп фонем.

1.1 Распознавание слов в слитной речи

Для распознавания слов в слитной речи апробированы два различных подхода. В первом случае при глобальном подходе слово, которое необходимо распознать, сравнивается с каждым словом словаря. При сравнении используется, как правило, спектральное представление каждого слова. Среди различных методов данного типа хорошие результаты дал метод динамического программирования.

Во втором случае при аналитическом подходе каждое слово или группа слов сначала сегментируется на меньшие единицы. Сегментами являются слогоподобные или фонемоподобные единицы. Это позволяет проводить распознавание либо на слоговом, либо на фонемном уровне и одновременно хранить в памяти параметры (длительность, энергию и т.п.), относящиеся к просодии и полезные в дальнейшем. Сегментация может быть основана на нахождении гласных высказывания, которые часто располагаются около

максимума интегративной энергии спектра. При таком подходе первым критерием сегментации является изменение энергии во времени. Некоторые согласные, например m, n, l, иногда обладают такой же энергией, как и гласные. Поэтому необходим ввод дополнительных параметров для выяснения наличия гласного звука в каждом ранее определенном сегменте.

Для идентификации согласных, как правило, проводится разделение взрывных и невзрывных согласных. Это достигается путем обнаружения паузы (смычки), соответствующей смыканию перед реализацией взрыва. Задача усложняется для позиции начала высказывания, где сравнительно просто определяется смычка только для звонких взрывных согласных. После обнаружения смычки определяются изменение спектра и вид изменения. Для установления каждой категории звуков обычно пользуются упорядоченными правилами, основанными на информации, зависящей от акустического и фонетического контекстов. В слитной речи фонетическая реализация какого-то конкретного высказывания зависит от нескольких факторов, включая диалект, скорость произнесения речи, манеру произнесения диктора и другие.

1.2 Распознавание изолированных слов

Основные признаки распознавания изолированных слов - иерархическая многоярусная структура и контроль каждого яруса с помощью соответствующих грамматик, чьи символы являются расплывчатыми лингвистическими переменными величинами.

Стратегия распознавания основана на группировке единиц речи в широкие фонетические классы, за которым следует классификация на более детальные группы.

При распознавании слитной речи возникают трудности: распознавание слитной речи намного сложнее распознавания отдельно произнесенных слов, прежде всего, вследствие неявных границ между словами. В результате

трудно определить начало и конец соответствия между фонемной цепочкой слова из словаря и распознаваемой фонемной цепочкой. Система акустико-фонетического анализа слитной речи обычно рассматривается как часть общей системы по автоматическому ее распознаванию.

Предварительная сегментация и классификация звуковых элементов включает определение гласноподобных, фрикативноподобных звуков, взрывных согласных, пауз. Задача сегментации, рассматриваемая как задача деления речевого потока на функционально значимые отрезки, решается по-разному. При разработке систем распознавания речи учитывается важность первой ступени обработки акустического сигнала, что связано с работой акустического процессора. Процесс автоматической сегментации непрерывно связан с маркировкой звуковой последовательности. Разработка автоматической сегментации и маркировки вызвана необходимостью привлечения большой акустико-фонетической базы данных и стремлением к объективизации речевого анализа.

1.2 Проблема автоматического распознавания речи

Проблема АРР может быть решена поэтапно. На первом этапе задача распознавания заключается во внешнем удостоверении внутренне выявленных и только поверхностно охарактеризованных классов акустических событий. Для второго этапа решающее значение имеет обобщение внешних критериев классификации внутренне не выявленных классов, что делает возможным предсказуемость характеристики неизвестного сигнала.

При автоматическом распознавании речи, прежде всего, следует выяснить, является ли сигнал в действительности фонетическим (речевым). Известно деление речевого потока на микро- и макросегменты. Разграничение между двумя макросегментами (фразами синтагмами) носит, как правило, дискретный характер, а между двумя микросегментами

(субзвуками, звуками, слогами) - стертый. Звуки изменяют свои супрасегментные (длительность, интенсивность, частота основного тона) и сегментные (спектральные) характеристики в соответствии с влиянием единиц других ярусов. Например, увеличение длительности гласной в речевом потоке может указывать на семантическую выделенность слова, положение ударения относительно этой гласной, информацию о предшествующей и последующей фонемах и т. д. Следовательно, для предсказания, например, длительности звука, следует учитывать ряд лингвистических факторов.

Знание сочетаемости фонем на стыках слов играет также не последнюю роль при восприятии речи. Разграничительные средства звучащей речи представляют собой сложное явление, состоящее из самых различных компонентов, связанных с фонотактическими особенностями, синтактико-семантическими факторами, ритмикой формирования речевого высказывания.

Следует остановиться на некоторых проблемах сегментации, связанных со спецификой фонетического уровня. К числу трудностей может быть отнесено автоматическое распознавание назальных и плавных фонем слитной речи. Неопределенности, возникающие из-за ограничений любой системы обработки речи и часто из-за плохого произношения, рассматриваются как источники информации для стохастической грамматики или грамматики неопределенного множества.

Имеющиеся в настоящее время способы микросегментации речи (сегментации на субзвуки, звуки, слоги) можно классифицировать следующим образом:

1) использование степени стабильности во времени каких-либо акустических параметров речевого сигнала, таких как концентрация энергии в частотном спектре;

2) накладывание акустических меток на речевой сигнал через регулярно повторяющиеся короткие интервалы;

3) сравнение выборок речевого сигнала в коротких временных окнах при регулярных интервалах с выборками из фонем-прототипов.

Различают контекстно-зависимые и контекстно-независимые методы сегментации. Самым простым методом контекстно-независимой маркировки является сопоставление эталонов. Для этого необходимо, чтобы в запоминающем устройстве для каждой возможной словарной единице хранилось модель. Контекстно-зависимая сегментация допускает связь используемого множества признаков и порогов с фонетическим контекстом.

Для решения проблемы сегментации звучащей речи большое значение имеет обращение к слогу. При этом в современной лингвистике условно разграничиваются фонетический и фонологический типы слога.

При определении, разграничении и определении слога необходимо использовать фонологические критерии. В наиболее общих терминах слог - это речевой сегмент, состоящий из ядра, т.е. гласного (или слогообразующего согласного) и артикуляторно связанных с ним соседних согласных. Слог дает возможность выхода как на более низкий звуковой, так и на более высокий языковой ярус с использованием информации фонотактических особенностей формирование морфем, слов. Большинство способов сегментации на слоги основано на изменениях общей (суммарной) интенсивности сигнала, т.е. энергии. Поскольку теоретически каждый слог должен содержать только один гласный, а гласные обычно имеют преобладающую интенсивность по сравнению с окружающими согласными, можно предположить, что большинство локальных максимумов - гласные. Очевидно, что слоговые границы находятся в минимальной точке между двумя максимумами. Однако этот подход наталкивается на сложность, т.к. при наличии, например, сонанта могут появляться ложные максимумы.

Сегментация может проводиться в два этапа: на слоги, а затем на звуки, их составляющие, в результате чего уточняются границы между слогами. Соотношение между сегментами по ряду параметров позволяет выявить внутреннюю структуру слоговой единицы.

В фонетике точка зрения на акустическую выделенность границ фонетического слова (ритмической структуры) претерпела ряд изменений. Полное отрицание акустических границ слова сменилось утверждением о том, что при определении границ фонетического слова в потоке речи вполне реально опираться на объективные критерии: акустические характеристики звуков на стыке фонетических слов и их аллофоническую вариативность. При разграничении речевого потока на фонетические слова привлечение акустических характеристик стыковых звуков необходимо во всех случаях: как без паузы, так и при наличии последней.

Вероятность появления паузы в речи зависит от характера сочетаний звуков ритмической структуры соседних слов (например, если первое слово кончается ударным слогом, а следующее за ним начинается также с ударного, то появление между этими словами паузы более вероятно, чем в том случае, когда за ударным слогом первого фонетического слова следует безударный слог второго фонетического слова) и места рассматриваемого стыка во фразе.

В потоке речи определение границ фонетического слова сопряжено с рядом трудностей, возникающих в связи с принадлежностью высказывания к стилю произношения и типу произнесения; позицией фонетического слова в тексте, синтагме и фразе.

Одни реализации границ фонетических слов действительно имеют свои акустические признаки, другие их не имеют. Задача не должна ограничиваться исключительно поиском физических и слуховых признаков соседних звуков, а должна быть направлена на определение иерархии (соподчинения) этих признаков.

Информация об ударении, несомненно, также используется для определения числа фонетических слов в сообщении. Важнейшей информацией, однако, используемой человеком при членении речевого потока, является информация о типах наиболее частотных фонетических слов (ритмических структур). При членении слитной речи на семантически

значимые отрезки используется информация различных языковых уровней - от фонологического до семантического. При разработке программ для автоматического членения текста эта информация (о типах ритмических структур, числе и степени ударения и т.д.), безусловно, должна быть принята во внимание. Однако в слитной речи возникают двусмысленные языковые ситуации, декодирование которых может быть осуществлено с привлечением дополнительной информации об акустических признаках членения. Стыковые гласные и согласные обладают определенными акустическими признаками, изменение которых зависит от характера связи между ними.

В случаях, когда доступ к системе, распознающей речь, должен быть обеспечен любому пользователю, целесообразен переход к неадаптивным (независимым от диктора) системам автоматического распознавания. Эти системы гораздо легче реализовать для языков, фонетическая структура которых более изучена (для русского, японского, английского) и гораздо сложнее для языков тонального типа (вьетнамского, китайского, французского).

При создании систем автоматического распознавания звучащей речи огромное значение приобретают эксперименты в области восприятия речи. Результаты таких экспериментов часто лежат в основе функционирования той или иной системы. ЭВМ, распознающие речь, зачастую копируют некоторые не только анализирующие функции человеческого уха, но и запоминающие, а также логические функции человеческого мозга.

Непрерывное совершенствование форм диалога между человеком-оператором и ЭВМ должно привести к оптимизации коммуникации между ними. Диалог «человек-машина» на естественном языке предполагает использование, как соответствующих технических методов, так и определенных лингвистических знаний. Изучение проблемы роли языка общения между человеком и ЭВМ и разработка автоматизированных систем с естественным для человека языком общения находятся на стадии дальнейшего становления.

1.4 Структурная схема устройства выделения признаков речевых сигналов

Ниже будет предложена следующая структурная схема устройства выделения признаков речевых сигналов (рисунок 1.1).

Она состоит из следующих блоков:

1 - микрофон;

2 – блок выделения огибающей;

3 – блок определения начала и конца слова;

4 – блок выделения конечной разности;

5 – блок выделения количества звуков;

6 – линия задержки;

7 – блок выделения интервалов;

8 – блок анализа;

9 – блок данных;

10 – печатающее устройство.

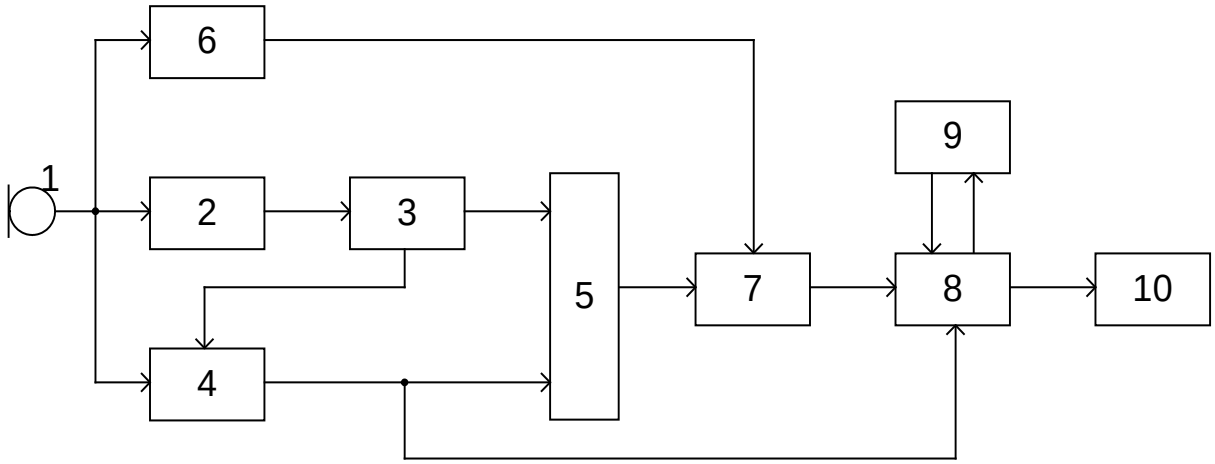


Рисунок 1.1 - Структурная схема устройства выделения признаков речевых сигналов

Задача распознавания речи может быть сведена к задаче распознавания отдельных звуков с последующим использованием алгоритмов,

учитывающих особенности произношения, словопостроения и словосочетания фраз отдельных индивидуумов.

В этом случае задача выделения звуков речи может рассматриваться как задача распознавания образов, количество которых ограничено, хотя и достигает нескольких десятков. При этом сама задача классификации предъявляемых образцов звуков может быть сведена к задаче многоальтернативной проверки гипотез. При этом система распознавания звуков речи может строиться с использованием принципов "обучения с учителем", т.е. предварительного набора информационной базы классифицированных данных, с которыми производится сравнение поступающих на анализ сигналов. Процедура распознавания звуков речи должна учитывать особенности их реализации. Во-первых, эти реализации у каждого звука имеют свой вид. Во-вторых, имеют ограниченную протяженность во времени.

Методы анализа речевых сигналов можно рассматривать с помощью модели, в которой речевой сигнал является откликом системы с медленно изменяющимися параметрами на периодическое или шумовое возбуждающее колебание (рисунок 1.2).

Выходной сигнал голосового тракта определяется сверткой функции возбуждения и импульсного отклика линейного, изменяющегося во времени фильтра, моделирующего голосовой тракт. Таким образом, речевой сигнал s(t) выражается следующим образом:

$$S(t) = \int_{-\infty}^{t} e(\tau)\, v(t, \tau)\, d\tau ,$$

где e(t) - функция возбуждения, v(t,τ) - отклик голосового тракта в момент t на дельта-функцию, подаваемую на вход в момент τ.

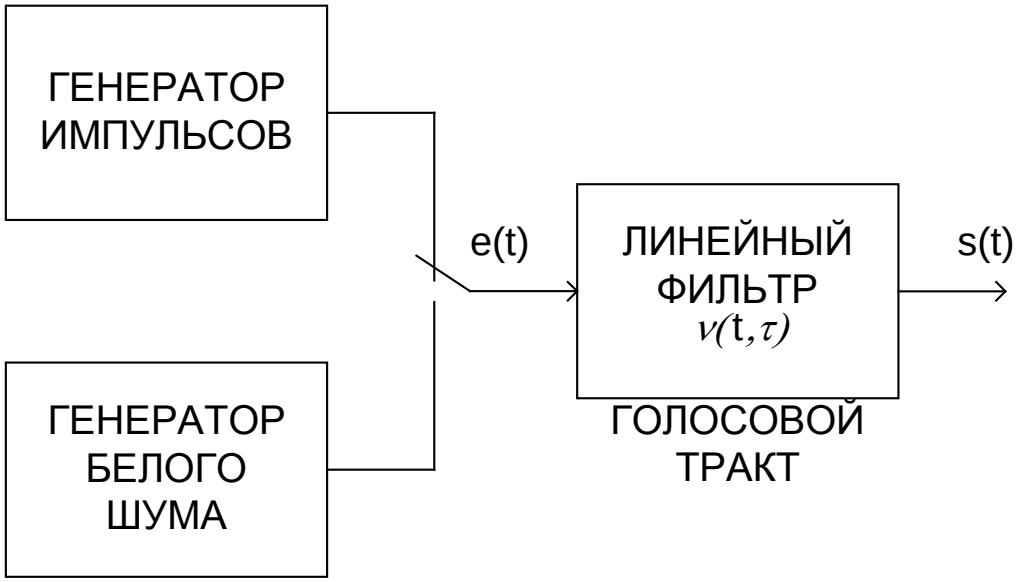


Рисунок 1.2 - Схема функциональной модели формирования речи

Речевой сигнал можно промоделировать откликом линейной системы с переменными параметрами (голосового тракта) на соответствующий возбуждающий сигнал. При неизменной форме голосового тракта выходной сигнал равен свертке возбуждающего сигнала и импульсного отклика голосового тракта. Однако все разнообразие звуков получается путем изменения формы голосового тракта. Если форма голосового тракта изменяется медленно, то на коротких интервалах времени выходной сигнал логично по-прежнему аппроксимировать сверткой возбуждающего сигнала и импульсного отклика голосового тракта. Поскольку при создании различных звуков форма голосового тракта изменяется, огибающая спектра речевого сигнала будет, конечно, тоже изменяться с течением времени. Аналогично при изменении периода сигнала, возбуждающего звонкие звуки, частотный разнос между гармониками спектра будет изменяться. Следовательно, необходимо знать вид речевого сигнала на коротких отрезках времени и характер его изменения во времени.

В системах анализа речевых сигналов обычно пытаются разделить возбуждающую функцию и характеристики голосового тракта. Далее в

зависимости от конкретного способа анализа получают параметры, описывающие каждую компоненту.

В частотной области спектр коротких отрезков речевого сигнала можно представить в виде произведения огибающей, характеризующей состояние голосового тракта, и функции, описывающей тонкую структуру, которая характеризует возбуждающий сигнал. Поскольку основным параметром сигнала, возбуждающего звонкий звук, является разнос гармоник основного тона, а характеристики голосового тракта с достаточной полнотой определяются частотами формант, то при анализе весьма удобно исходить из представления речи в частотной области. При создании различных звуков форма голосового тракта и возбуждающий сигнал изменяются, при этом изменяется и спектр речевого сигнала. Следовательно, спектральное представление речи должно основываться на кратковременном спектре, получаемом из преобразования Фурье.

Рассмотрим дискретизированный речевой сигнал, представленный последовательностью s(n). Его кратковременное преобразование Фурье S(ω,n) определяется как

$$S(\omega,n)=\sum_{k=-\infty}^{\infty}s(k)h(n-k)e^{-j\omega k} \qquad (1.1)$$

Данное выражение описывает преобразование Фурье взвешенного отрезка речевого колебания, причем весовая функция h(n) сдвигается во времени.

Линейное предсказание является одним из наиболее эффективных методов анализа речевых сигналов. Этот метод становится доминирующим при оценке основных параметров речевых сигналов, таких как период основного тона, форманты, спектр, а также при сокращенном представлении речи с целью ее низкоскоростной передачи и экономного хранения. Важность метода обусловлена высокой точностью получаемых оценок и относительной простотой вычисления.

Основной принцип метода линейного предсказания состоит в том, что текущий отсчет речевого сигнала можно аппроксимировать линейной комбинацией предшествующих отсчетов. Коэффициент предсказания при этом определяется однозначно минимизацией среднего квадрата разности между отсчетами речевого сигнала и их предсказанными значениями (на конечном интервале). Коэффициенты предсказания - это весовые коэффициенты, используемые в линейной комбинации. Метод линейного предсказания можно применять для сокращения объема цифрового речевого сигнала.

Основной целью обработки речевых сигналов является получение наиболее удобного и компактного представления содержащейся в них информации. Точность представления определяется той информацией, которую необходимо сохранить или выделить. Например, цифровая обработка может применяться для выяснения, является ли данное колебание речевым сигналом. Сходная, но несколько более сложная задача состоит в том, чтобы классифицировать колебания на вокализованную речь, невокализованную речь и паузу (шум).

В основе большинства методов обработки речи лежит представление о том, что свойства речевого сигнала с течением времени медленно изменяются. Это предположение приводит к методам кратковременного анализа, в которых сегменты речевого сигнала выделяются и обрабатываются так, как если бы они были короткими участками отдельных звуков с отличающимися свойствами.

Одним из наиболее известных методов анализа речи во временной области можно назвать метод, предложенный Л.Рабинером и Р.Шафером в /3/. Он основан на измерении кратковременного среднего значения сигнала и кратковременной функции среднего числа переходов через нуль. Как отмечалось выше, амплитуда речевого сигнала существенно изменяется во времени. Подобные изменения амплитуды хорошо описываются с помощью

функции кратковременной энергии сигнала. В общем случае определить функцию энергии можно как

$$E_n = \sum_{m=-\infty}^{\infty} [x(m)\, w(n-m)]^2$$

Это выражение может быть переписано в виде

$$E_n = \sum_{m=-\infty}^{\infty} x^2(m) h(n-m), \qquad (1.2)$$

где $h(n) = w^2(n)$

Выбор импульсной характеристики h(n) или окна составляет основу описания сигнала с помощью функции энергии.

Чтобы понять, как влияет выбор временного окна на функцию кратковременной энергии сигнала, предположим, что h(n) в (1.2) является достаточно длительной и имеет постоянную амплитуду; значение En будет при этом изменяться во времени незначительно. Такое окно эквивалентно фильтру нижних частот с узкой полосой пропускания. Полоса фильтра нижних частот не должна быть столь узкой, чтобы выходной сигнал оказался постоянным. Для описания быстрых изменений амплитуды желательно иметь узкое окно (короткую импульсную характеристику), однако слишком малая ширина окна может привести к недостаточному усреднению и, следовательно, к недостаточному сглаживанию функции энергии. Влияние ширины временного окна на точность измерения кратковременного среднего значения (средней энергии):

если N (ширина окна в отсчетах) мало (порядка периода основного тона и менее), то En будет изменяться очень быстро, в соответствии с тонкой структурой речевого колебания,

если N велико (порядка нескольких периодов основного тона), то En будет изменяться медленно и не будет адекватно описывать изменяющиеся особенности речевого сигнала.

Это означает, что не существует единственного значения N, которое в полной мере удовлетворяло бы перечисленным требованиям, так как период основного тона изменяется от 10 отсчетов (при частоте дискретизации 10 кГц) для высоких детских и женских голосов и до 250 отсчетов для очень низких мужских. N выберем равным 100, 200, 300 отсчетов при частоте дискретизации 8 кГц.

Основное назначение En состоит в том, что эта величина позволяет отличить вокализованные речевые сегменты от невокализованных. Значение функции кратковременного среднего значения сигнала для невокализованных сегментов значительно меньше, чем для вокализованных.

Характерной особенностью метода анализа речевых сигналов является бинарное квантование входного речевого сигнала. Возможность выделения параметров сигналов, подвергшихся бинарному квантованию, показана в /4/. Используемая математическая модель речевого сигнала имеет вид:

$$S(t) = A(t) \cdot e^{j\Psi(t)}, \qquad (1.3)$$

где A(t) - закон изменения амплитуды речевого сигнала, Ψ(t) - полная фазовая функция речевого сигнала.

Закон изменения амплитуды сигнала не является достаточно информативным параметром для оценки речевого сообщения, так как он не является постоянным для одного и того же слова или фразы, произнесенных с различной интонацией и громкостью. В качестве информативной характеристики речевого сигнала в предлагаемом методе полагается полная фазовая функция речевого сигнала. Полная фазовая функция речевого сигнала представляется в виде разложения в ряд Тейлора:

$$\Psi(t) = \frac{\Psi^{(0)}(t_0)}{0!} t^0 + \frac{\Psi^{(1)}(t_0)}{1!} t^1 + \frac{\Psi^{(2)}(t_0)}{2!} t^2 + \frac{\Psi^{(3)}(t_0)}{3!} t^3 + \dots . \qquad (1.4)$$

Выражение (1.4) можно переписать следующим образом

$$\Psi(t) = \mu_0 + \mu_1 t + \frac{\mu_2 t^2}{2} + \frac{\mu_3 t^3}{6} + \dots . \qquad (1.5)$$

В разложении берутся первые три коэффициента разложения. При этом первый коэффициент μ0, являющийся начальной фазой речевого сигнала, принимается равным нулю, вследствие неинформативности. Тогда полная фазовая функция будет:

$$\Psi(t) = \mu_1 t + 0{,}5\mu_2 t^2, \qquad (1.6)$$

где, μ1 - коэффициент разложения, являющийся средней частотой речевого сигнала, μ2 - коэффициент разложения, являющийся изменением (девиацией) частоты речевого сигнала.

После дискретизации полная фазовая функция имеет следующий вид:

$$\Psi(i \cdot \Delta t) = \mu_1 \cdot (i \cdot \Delta t) + 0{,}5 \cdot \mu_2 \cdot (i \cdot \Delta t)^2, \qquad (1.7)$$

где i - номер текущего отсчета в дискретизированной последовательности, Δt - шаг дискретизации.

Параметры μ1 и μ2 являются характеристиками, которые используются для описания речевого сообщения. В режиме обработки "скользящее окно" вычисляется первая конечная разность полной фазовой функции речевого сигнала, которая является кратковременной функцией среднего числа переходов через нуль речевого сигнала и является грубой оценкой частоты речевого сигнала μ1 с некоторой погрешностью, зависящей от изменения частоты μ2. Для определения μ2 следует вычислить вторую конечную разность полной фазовой функции речевого сигнала, которая также является скоростью изменения функции среднего числа переходов через нуль речевого сигнала. Первая и вторая конечные разности полной фазовой функции имеют следующий вид /4/:

$$\Delta\Psi(i \cdot \Delta t) = \Psi(i \cdot \Delta t) - \Psi((i - L) \cdot \Delta t) = \mu_1 \cdot L \cdot \Delta t + \mu_2 \cdot i \cdot L \cdot \Delta t^2 - 0{,}5 \cdot \mu_2 \cdot (L \cdot \Delta t)^2,$$

$$\Delta^2\Psi(i \cdot \Delta t) = \Delta\Psi(i \cdot \Delta t) - \Delta\Psi((i - L) \cdot \Delta t) = \mu_2 \cdot (L \cdot \Delta t)^2, \qquad (1.8)$$

где L - ширина временного "скользящего" окна выраженная в количестве отсчетов.

Тогда из (1.8) частоту речевого сигнала μ1 и изменение частоты μ2, получим в виде:

$$\mu_2 = \Delta^2\Psi(i\cdot\Delta t)/T^2,$$

$$\mu_1 = (\Delta\Psi(i\cdot\Delta t) - \mu_2\cdot i\cdot T\cdot\Delta t + 0{,}5\cdot\mu_2\cdot T^2)/T,$$

где T=L·Δt - ширина временного "скользящего" окна.

1.5 Разработка структурной схемы устройства определения количества звуков в изолированном слове речи

Структурная схема разрабатываемого устройства, анализирующего информационные признаки речевых сигналов и определяющего начало и конец звука в слове, изображена на рисунке 1.3. Она состоит из следующих блоков:

1 – первый формирователь;

2 – цифровая линия задержки (ЦЛЗ);

3 – первый реверсивный счетчик;

4 – второй РС;

5 – первый сумматор;

6 – третий РС;

7 – четвертый РС;

8 – второй сумматор;

9 – пятый РС;

10 – шестой РС;

11 – третий сумматор;

12 – первый вычислитель модуля;

13 – второй вычислитель модуля;

14 – третий вычислитель модуля;

15 – первое пороговое устройство;

16 – второе ПУ;

17 – третье ПУ;

18 – второй формирователь;

19 – третий формирователь;

20 – четвертый формирователь;

21 – схема ИЛИ.

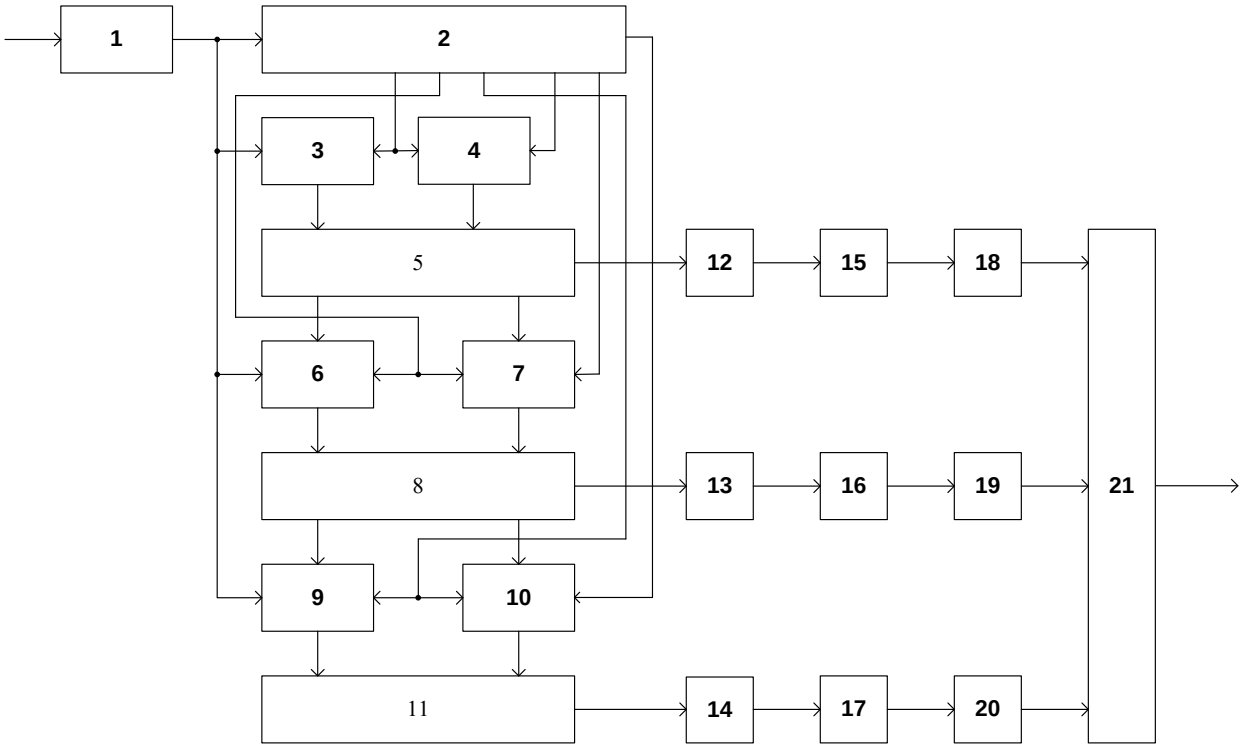


Рисунок 1.3 - Структурная схема устройства определения количества звуков

Речевой сигнал, произносимый человеком, попадает в микрофон. Микрофон служит для преобразования акустических волн, возбуждаемых голосовым трактом человека, в электрические колебания.

Для формирования бинарно-квантованного сигнала из аналогового речевого сигнала применяется АЦП с однобитной словарной организацией. В качестве такого АЦП можно использовать компаратор. Амплитудная характеристика компаратора приведена на рисунке 1.4.

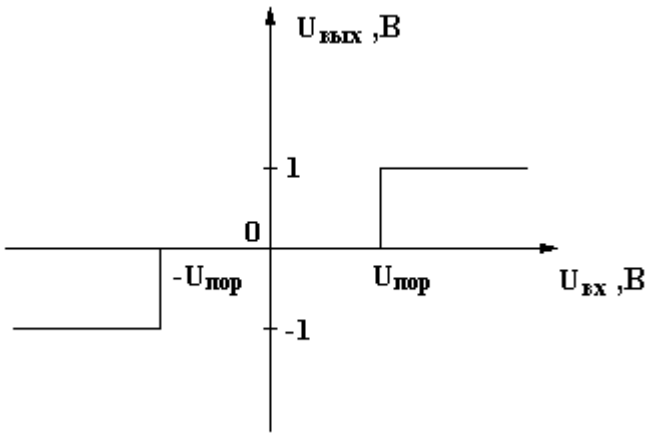


Рисунок 1.4 - Амплитудная характеристика компаратора

Задачей компаратора является отслеживание превышения входным речевым сигналом некоторого порога Uпор (для отрицательной полуволны сигнала –Uпор). Когда речевой сигнал на входе компаратора мал (находится в интервале –Uпор<Uвх<Uпор), то на выходе будет присутствовать уровень сигнала, соответствующий логическому «0». При превышении входным сигналом некоторого порога Uпор (или если сигнал меньше чем –Uпор по амплитуде), на выходе компаратора будет присутствовать уровень сигнала, соответствующий логической «1».

На выходе компаратора формируется сигнал в виде последовательности бинарно-квантованных отсчетов, то есть в виде последовательности логических «0» и «1». Появление сигналов на выходе компаратора определяется частотой поступления на его стробирующий вход стробирующих импульсов. Частота следования стробирующих импульсов, которая также является частотой дискретизации входного речевого сигнала, выбирается из условия выполнения теоремы Котельникова, то есть не менее 2Fmax, где Fmax - это максимальная частота в спектре речевого сигнала.

С выхода компаратора оцифрованный сигнал поступает на первую ЦЛЗ, которая обеспечивает задержку сигнала на 100 отсчетов, и на суммирующий вход первого реверсивного счетчика. Параметр, выделяемый реверсивным счетчиком, носит название первой конечной разности полной

фазовой функции речевого сигнала или функцией среднего числа переходов через нуль. Схема, вычисляющая первую конечную разность, состоит из линии задержки и реверсивного счетчика. Она работает в режиме «скользящее окно». Ширина временного окна составляет 100 отсчетов. Код на выходе реверсивного счетчика показывает количество пересечений через нуль на интервале времени 100 отсчетов. Сдвигаясь на один отсчет, «скользящее окно» выдает новый код, показывающий количество нулевых пересечений.

Вторая ЦЛЗ и второй реверсивный счетчик также вычисляют первую конечную разность, но задержанную на 100 отсчетов относительно той, которая вычисляется первой ЦЛЗ и первым реверсивным счетчиком. Имея две первые конечные разности полной фазовой функции речевого сигнала, можно дать оценку изменения частоты речевого сигнала во времени, т.е. вычислить скорость изменения функции среднего числа пересечений через нуль.

Операция нахождения второй конечной разности выполняется в первом сумматоре, который вычитает из первой конечной разности в текущий момент времени первую конечную разность, задержанную на длину временного окна 100 отсчетов.

Следующие блоки в схеме (четыре реверсивных счетчика и два сумматора) предназначены для 200 и 300 отсчетов.

Так как вторая конечная разность имеет отрицательные значения, то с 1, 2, 3-го сумматоров она поступает на 1, 2, 3-й блоки вычислителя модуля. Затем на 1, 2 и 3-е пороговое устройство и на формирователи. После чего идет схема ИЛИ.

2. ОБЗОР СУЩЕСТВУЮЩИХ СИСТЕМ УПРАВЛЕНИЯ РАСПОЗНАВАНИЕМ РЕЧЕВОЙ ИНФОРМАЦИИ И МЕТОДОВ РЕШЕНИЯ

Распознавание и порождение (синтез) речи компьютером является, безусловно, важной проблемой. Десятилетиями ученые и инженеры искали способы, которые позволили бы людям общаться с компьютером так же, как они общаются между собой, а не заставляли человека подстраиваться под способ общения, приемлемый для машины. Много было сделано, но, пожалуй, и на сегодняшний день можно считать, что вопрос далеко не закрыт, хотя именно в последнее время были достигнуты значительные успехи: уже многие годы голосовые команды являются одной из возможных опций программного обеспечения персональных компьютеров, появление функций распознавания речи уже обычное дело в ряде текстовых процессоров, системы распознавания речи работают там, где требуется оказание справочных услуг и в системах безопасности.

Вопросы цифровой обработки сигналов, отдельные области математической статистики, искусственного интеллекта (теории нейронных сетей), связанные с разработкой движков и приложений распознавания и порождения речи. Приводятся многочисленные отрывки программ на языке Си.

Вопросы обработки речи являются, главным образом, частью дисциплин, именуемых цифровой обработкой сигналов и распознаванием образов.

Методы цифровой обработки сигналов обычно осуществляют преобразование, очистку и трансформацию звукового сигнала в цифровой формат данных и другие представления, которые могут непосредственно обрабатываться системой распознавания речи. Эти задачи включают также фильтрацию шумовых сигналов, которые примешиваются к звуку при передаче акустических сигналов от воспринимающих устройств

(микрофонов) или по сети. Методы же распознавания образов используют при выделении и распознавании отдельных слов или предложений речевого потока или в некоторых случаях для идентификации говорящего.

Кроме того, системы распознавания и синтеза речи затрагивают вопросы лингвистики, в которой заложены фундаментальные концепции и принципы распознавания речи и понимания языка.

Перечислим два подхода решения задачи распознавания голосового сообщении.

2.1 Применение нейронных сетей для распознавания речи.

- Введение в нейронные сети:

Искусственная нейронная сеть — это математическая модель, а также устройства параллельных вычислений, представляющие собой систему соединённых и взаимодействующих между собой простых процессоров (искусственных нейронов). Как математическая модель искусственная нейронная сеть представляет собой частный случай методов распознавания образов или дискриминантного анализа.

Такие процессоры обычно довольно просты, особенно в сравнении с процессорами, используемыми в персональных компьютерах.

Каждый процессор подобной сети имеет дело только с сигналами, которые он периодически получает, и сигналами, которые он периодически посылает другим процессорам. И тем не менее, будучи соединёнными в достаточно большую сеть с управляемым взаимодействием, такие локально простые процессоры вместе способны выполнять довольно сложные задачи.

Понятие возникло при изучении процессов, протекающих в мозге при мышлении, и при попытке смоделировать эти процессы. Полученные модели называются искусственными нейронными сетями (ИНС).

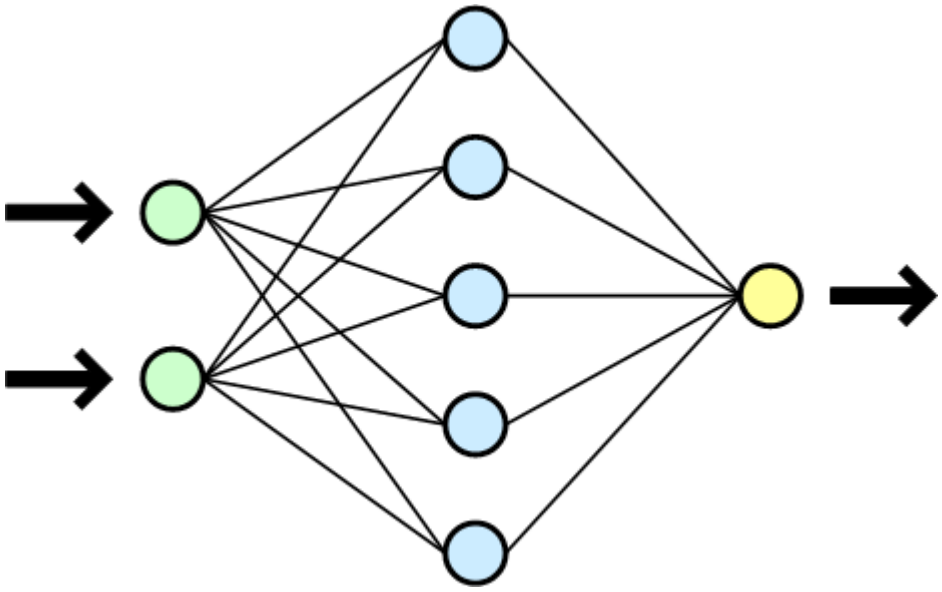


Рис. 2.1. Схема простой нейросети. Зелёным обозначены входные элементы, жёлтым — выходной элемент

Нейронные сети не программируются в привычном смысле этого слова, они обучаются. Возможность обучения — одно из главных преимуществ нейронных сетей перед традиционными алгоритмами. Технически обучение заключается в нахождении коэффициентов связей между нейронами. В процессе обучения нейронная сеть способна выявлять сложные зависимости между входными данными и выходными, а также выполнять обобщение. Это значит, что, в случае успешного обучения, сеть сможет вернуть верный результат на основании данных, которые отсутствовали в обучающей выборке.

- Алгоритмы обратного распространения:

Сложнее обстоит дело с многослойными сетями, так как изначально неизвестны желаемые выходы слоев сети (за исключением последнего) и их невозможно обучить, руководствуясь только величиной ошибок на выходе сети, как это было с однослойной сетью.

Наиболее приемлемым вариантом решения проблемы стала идея распространения сигнала ошибки от выхода сети к ее входу, слой за слоем. Алгоритмы, реализующие обучение сети по этой схеме, получили название

алгоритмов обратного распространения. Наиболее распространенный вариант этого алгоритма мы и рассмотрим и в дальнейшем применим в программной реализации задачи.

Алгоритм требует дифференцируемости активационной (или как ее по-другому называют, сжимающей) функции на всей оси абсцисс. По этой причине, функция единичного скачка не может использоваться и в качестве сжимающей функции обычно применяют упомянутый выше сигмоид (логистическую функцию), хотя существуют и другие варианты.

2.2 Применение скрытых Марковских моделей для распознавания речи.

- Введение в скрытые Марковские модели (СММ). Решение задачи распознавания.

Скрытой Марковской моделью (СММ) называется модель состоящая из N состояний, в каждом из которых некоторая система может принимать одно из M значений какого-либо параметра. Вероятности переходов между состояниями задается матрицей вероятностей A={aij}, где aij – вероятность перехода из i-го в j-е состояние. Вероятности выпадения каждого из M значений параметра в каждом из N состояний задается вектором B={bj(k)}, где bj(k) – вероятность выпадения k-го значения параметра в j-м состоянии. Вероятность наступления начального состояния задается вектором $\pi=\{\pi i\}$, где πi – вероятность того, что в начальный момент система окажется в i-м состоянии.

Таким образом, скрытой Марковской моделью называется тройка $\lambda=\{A,B,\pi\}$. Использование скрытых Марковских моделей для распознавания речи основано на двух приближениях:

1) Речь может быть разбита на фрагменты, соответствующие состояниям в СММ, параметры речи в пределах каждого фрагмента считаются постоянными.

2) Вероятность каждого фрагмента зависит только от текущего состояния системы и не зависит от предыдущих состояний.

Модель называется «скрытой», так как нас, как правило, не интересует конкретная последовательность состояний, в которой пребывает система. Мы либо подаем на вход системы последовательности типа O={o1,o2,…oi} - где каждое oi – значение параметра (одно из M), принимаемое в i-й момент времени, а на выходе ожидаем модель λ={A,B,π}с максимальной вероятностью генерирующую такую последовательность, - либо наоборот подаем на вход параметры модели и генерируем порождаемую ей последовательность. И в том и другом случае система выступает как “черный ящик”, в котором скрыты действительные состояния системы, а связанная с ней модель заслуживает названия скрытой.

Для осуществления распознавания на основе скрытых моделей Маркова необходимо построить кодовую книгу, содержащую множество эталонных наборов для характерных признаков речи (например, коэффициентов линейного предсказания, распределения энергии по частотам и т.д.). Для этого записываются эталонные речевые фрагменты, разбиваются на элементарные составляющие (отрезки речи, в течении которых можно считать параметры речевого сигнала постоянными) и для каждого из них вычисляются значения характерных признаков. Одной элементарной составляющей будет соответствовать один набор признаков из множества наборов признаков словаря.

Фрагмент речи разбивается на отрезки, в течении которых параметры речи можно считать постоянными. Для каждого отрезка вычисляются характерные признаки и подбирается запись кодовой книги с наиболее подходящими характеристиками. Номера этих записей и образуют последовательность наблюдений O={o1,o2,…oi} для модели Маркова. Каждому слову словаря соответствует одна такая последовательность. Далее A – матрица вероятностей переходов из одного минимального отрезка речи (номера записи кодовой книги) в другой минимальный отрезок речи (номер

записи кодовой книги). B – вероятности выпадения в каждом состоянии конкретного номера кодовой книги рис. 2.2

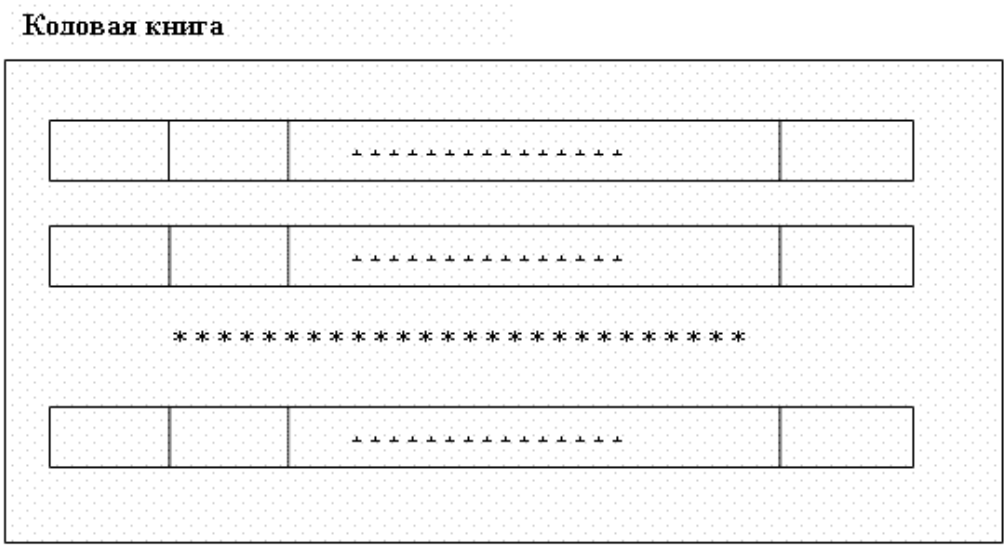


Рис. 2.2 кодовая книга

На этапе настройки моделей Маркова мы применяем алгоритм Баума-Уэлча для имеющегося словаря и сопоставления каждому из его слов матрицы A и B.

При распознавании мы разбиваем речь на отрезки, для каждого вычисляем набор номеров кодовой страницы и применяем алгоритм прямого или обратного хода для вычисления вероятности соответствия данного звукового фрагмента определенному слову словаря. Если вероятность превышает некоторое пороговое значение – слово считается распознанным.

- Алгоритм Баума-Уэлча.

Необходимо подобрать параметры скрытой модели Маркова так, чтобы максимизировать вероятность данной последовательности наблюдений.

Вводятся переменные

$\xi t(i,j) = P(qt=Si,qt+1=Sj|O,\lambda)$

которые показывают вероятность того, что при заданной последовательности наблюдений O система в моменты времени t и t+1 будет

находиться соответственно в состояниях Si и Sj. Используя прямую и обратную переменные запишем:

$$\xi_t(i,j) = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{P(O|\lambda)} = \frac{\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}{\sum_N\sum_N\alpha_t(i)a_{ij}b_j(O_{t+1})\beta_{t+1}(j)}$$

Введем переменные вероятности того, что при заданной последовательности наблюдений O система в момент времени t будет находиться в состоянии Si:

$$\gamma_t(i) = \sum_{j=1}^{N}\xi_t(i,j)$$

При этом мы можем вычислить ожидаемое число переходов из состояния Si: равно

$$\sum_{t=1}^{T-1}\gamma_t(i)$$

а ожидаемое число переходов из состояния Si в состояние Sj

$$\sum_{t=1}^{T-1}\xi_t(i,j)$$

Исходя из этого можно получить формулы для переоценки параметров модели Маркова:

$\pi^*i=\gamma_t(i)$

$$a_{ij}^* = \frac{\sum_{t=1}^{T-1}\xi_t(i,j)}{\sum_{t=1}^{T-1}\gamma_t(i)}$$

$$b_{ij}^*(k) = \frac{\sum_{t=1}^{T-1}\gamma_t(i)}{\sum_{t=1,O_t=k}^{T-1}\gamma_t(i)}$$

Выражение

$$\sum_{t=1,O_t=k}^{T-1}\gamma_t(i)$$

в формуле для b*ij (k) означает что суммируются только те γt(j) , для которых значение состояния равно k, то есть Ot = k.

После переоценки параметры модели либо выясняется, что она уже была оптимальной до переоценки либо обязательно улучшаются ее

параметры (то есть правдоподобность модели после переоценки выше, чем до переоценки во всех случаях, когда модель можно оптимизировать).

3. МОДЕЛИРОВАНИЕ РАБОТЫ БЛОКА ВЫДЕЛЕНИЯ НАЧАЛА И ОКОНЧАНИЯ СЛОВА, КОЛИЧЕСТВА ЗВУКОВ НА ЭВМ

В процессе выполнения дипломного проекта были проведены экспериментальные исследования алгоритма выделения признаков речевых сигналов.

Речевой сигнал с микрофона вводился в ЭВМ с помощью 16-ти разрядного преобразователя аналог-код. Частоту дискретизации выбрали 8 кГц. Далее сигнал подвергался анализу. Запись трех гласных звуков «а-о-е» и слов «Hello», «Start», «Stop», «Next» и «Back» производилась с помощью специального звукового редактора «COOL».

Обработка речевых сигналов производилась в ЭВМ с помощью программного пакета «Matlab7.6.0». Входные данные представляли собой массив дискретизированных чисел и содержались в отдельном файле данных. В созданной программе было произведено моделирование работы компаратора. Было выбрано три уровня квантования «0», «1», «-1» и был установлен шумовой порог, т.к. шумы хоть и были незначительны, но все же могли повлиять на результаты.

Далее в программе производился подсчет количества переходов через нуль на интервале в 100, 200 и 300 отсчетов. Таким образом, моделировалась работа блока состоящего из ЦЛЗ и реверсивного счетчика. Были вычислены первая и вторая конечные разности полной фазовой функции. Более подробно можно рассмотреть на примере фонем «а-о-е» (рисунок 4.1).

По такому же принципу были проведены эксперименты со словами, такими как «Notepad», «Open», «Close», «Pause» и с более сложными словами «Calculator», «Microsoft» и «Explorer».

Далее стояла задача собрать статистические данные по выделению количества звуков в нескольких словах. Для статистики были взяты следующие слова: «Hello» и «Start». Каждое слово было произнесено шестью

людьми по десять раз. Результаты статистических данных приведены в таблицах.

Таблица 4.1 – Статистические данные по слову «Hello»

Номер эксперимента	0	1	2	3	4	5	6	7	8	9
Количество звуков	4	4	4	4	3	5	4	4	3	4

10	11	12	13	14	15	16	17	18	19	20	21
5	5	3	3	4	3	4	4	4	5	4	4

22	23	24	25	26	27	28	29	30	31	32	33
4	4	4	4	4	4	4	4	4	4	4	4

34	35	36	37	38	39	40	41	42	43	44	45
4	4	4	3	4	4	5	4	4	4	4	4

46	47	48	49	50	51	52	53	54	55	56	57
3	3	4	4	4	4	4	4	4	4	4	4

58	59	60
4	4	4

Итого получается, что процент распознавания количества звуков в слове «Hello» равен порядка 79%.

Таблица 4.2 – Статистические данные по слову «Start»

Номер эксперимента	0	1	2	3	4	5	6	7	8	9
Количество звуков	4	4	4	4	4	4	4	4	4	4

10	11	12	13	14	15	16	17	18	19	20	21
4	4	4	4	4	4	4	4	4	4	4	4

22	23	24	25	26	27	28	29	30	31	32	33
4	4	4	4	4	2	3	3	4	3	3	4

34	35	36	37	38	39	40	41	42	43	44	45
4	4	4	4	4	4	4	3	4	3	4	3

46	47	48	49	50	51	52	53	54	55	56	57
4	4	4	4	4	3	3	4	3	3	4	3

58	59	60
4	3	4

Итого получается, что процент распознавания количества звуков в слове «Start» равен порядка 75%.

5. РАЗРАБОТКА ИНОФРМАЦИОННОГО ОБЕСПЕЧЕНИЯ

5.1 Структурно-алгоритмическая организация

В структуре программы можно выделить логические модули. Каждому модулю присущи свои задачи, методы, вызываемые функции. Структурная схема изображена на рисунке 5.1.

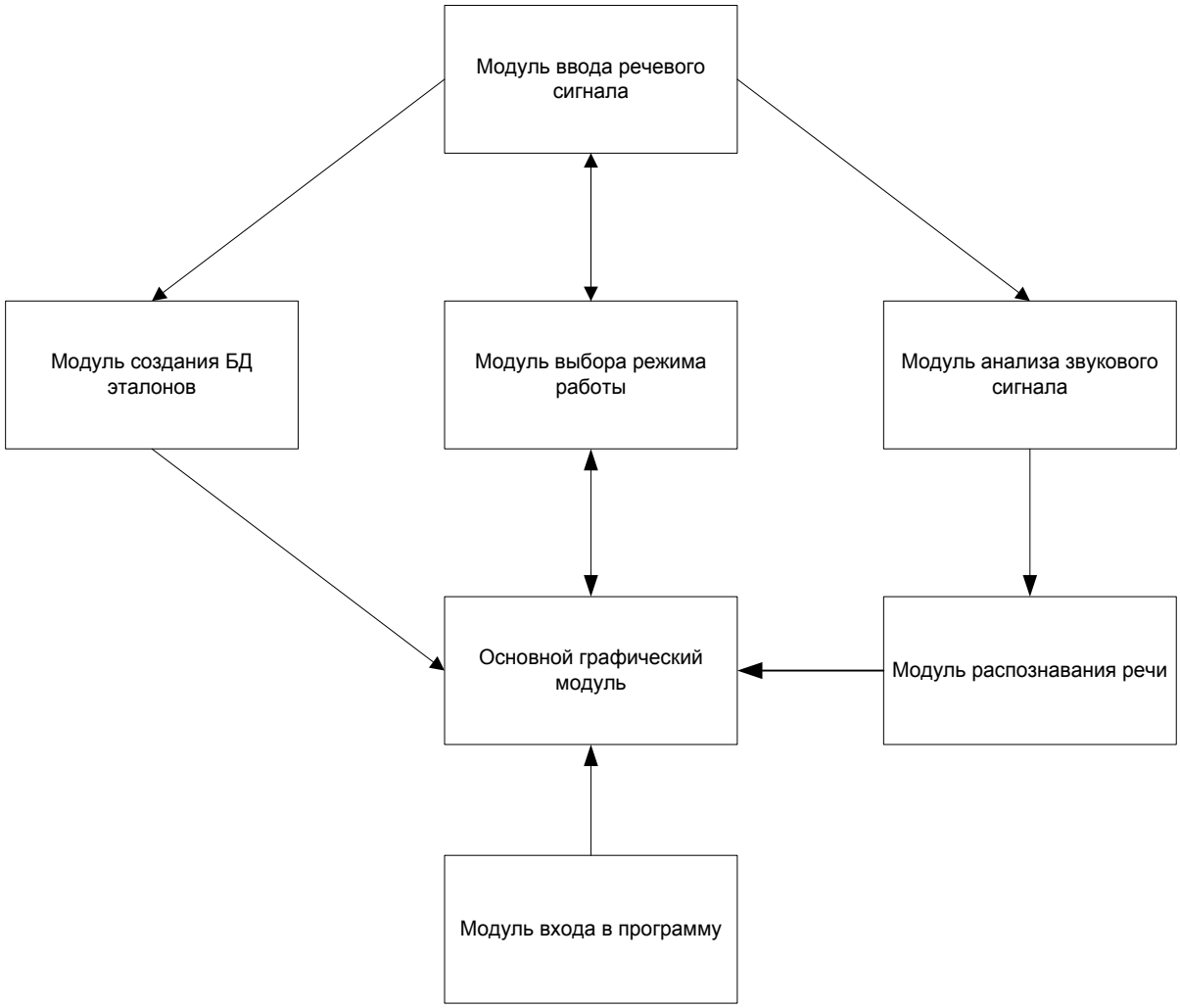


Рисунок 5.1 – Структурная схема программы

Основной графический модуль – это графический интерфейс общения пользователя с программой. Пользователь при нажатии на кнопки вызывает выполнение функций из других модулей.

Модуль выбора режима работы – это модуль для взаимодействия пользователя с программой с целью установки режима работы.

Модуль ввода речевого сигнала – это модуль, который отвечает за запись сигнала с микрофона.

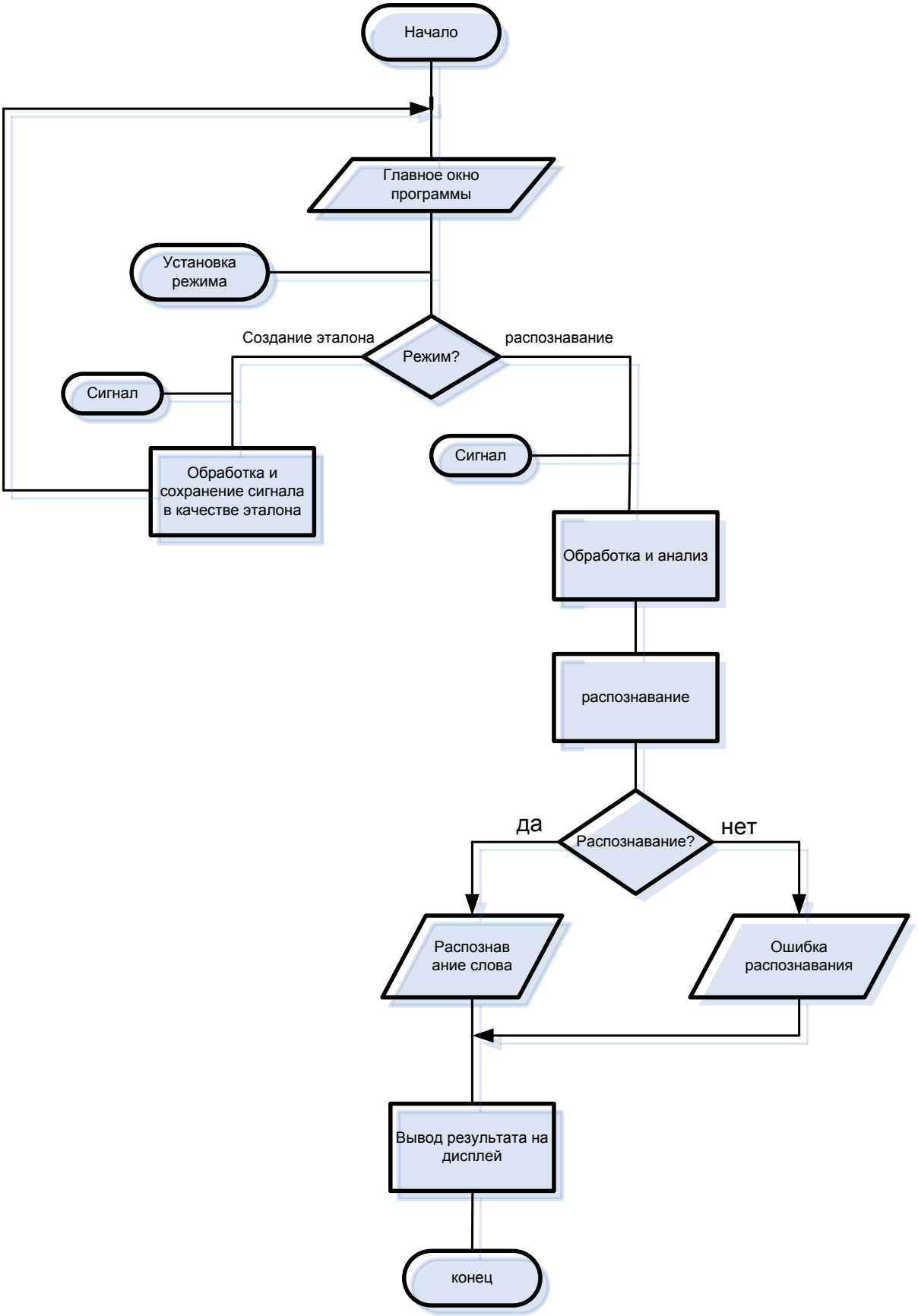
Модуль создания БД эталонов – это модуль, который анализирует входной сигнал в режиме создания эталона, переводит в цифровой вид и создает БД.

Модуль анализа звукового сигнала – это модуль, который анализирует входной сигнал в режиме распознавания, и переводит в цифровой вид.

Модуль распознавания речи – это модуль, который проводит сопоставление входного сигнала и эталона в БД.

5.2 Алгоритм программы

В начале работы на экран выводится главное окно программы. После этого на динамик микрофона подается звуковое сообщение, за который отвечает модуль ввода речевого сигнала. Затем на главном окне пользователь выбирает режим работы программы. Если выбран режим создания эталона, за который отвечает модуль создания БД эталонов, то программа обрабатывает и сохраняет входной сигнал с микрофона и выводит спектр на экран. Если же выбран режим распознавания, то программа обрабатывает результаты и сравнивает с заранее записанным эталоном в БД, сохраняет входной сигнал и переходит к его распознаванию с помощью вычисления первой и второй конечной разности полной фазовой функции, т.е. определяем количество звуков в данном слове, что видно из проделанного ранее моделирования, Определяем начало и конец слова с помощью выделения огибающей, что показано на рис.3.2. Результат распознавания выводится на дисплей.



5.3 Описание интерфейса

Программа имеет дружественный интерфейс и легка в освоении. Графический интерфейс проектируется в среде разработки Visual C#.net. В данной системе можно быстро и качественно разрабатывать графические приложения, используя готовые компоненты.

Интерфейс программы разработан с учетом информативности получаемых пользователем данных. На рисунке 5.2 показано основное окно программы.

В верхней строке окна программы находится панель управления со стандартными вкладками. Ниже располагается панель с вкладками, которые позволяют пользователю выбирать режим обработки поступающего с микрофона звукового сигнала. Непосредственно над графиками представлена информационная панель, которая дает пользователю возможность в реальном режиме времени наблюдать общее количество команд в БД, количество распознанных команд, последнюю распознанную команду и номер испытания.

На верхнем графике изображена частотная характеристика входного звукового сигнала, а на следующем – амплитудная характеристика распознанного сигнала.

В нижней части окна программы имеется поле событий, в котором выводятся сообщения о работе аппаратной части.

5.4 Реализация

5.4.1 Модуль входа в программу

При запуске программы вызывается функция «static void Main()». Эта функция инициализирует приложение путем вызова функции Application.EnableVisualStyles();. Далее запускается функция, которая составляет стиль программы путем вызова следующей функции Application.SetCompatibleTextRenderingDefault(false); . После нее запускается функция, которая создает диалоговое окно программы путем вызова функции

Application.Run(new Form1()); и запускает приложение. В теле этой функции так же предусмотрена обработка исключений try – catch.

5.4.2 Основной графический модуль

Основной графический модуль в исходном коде программы представлен как класс Form1. Он содержит следующие основные графические элементы:

1) PictureBox1 : Вывод графического изображения 1
2) PictureBox2 : Вывод графического изображения 2
3) textBox2 : Вывод распознанного слова

5.4.3 Модуль выбора режима работы

Выбор режима работы программы осуществляется нажатием кнопки «Mode». При ее нажатии вызывается функция-обработчик этого события «private void recognitionToolStripMenuItem_Click(object sender, EventArgs e)» Функция проверяет и сохраняет указанный режим работы.

5.4.4 Модуль ввода речевого сигнала

Запись и обработка звукового сигнала осуществляется функцией «private void toolStripDropDownButton1_Click(object sender, EventArgs e)». В этой функции задается частота дискретизации – 16 кГц. Длина сигнала по времени определяется с помощью функции GetTickCount(). Дискретезированный и отфильтрованный сигнал записывается в буфер matrix[].

5.4.5 Модуль создания БД эталонов

При создании эталона необходимо нажать кнопку «Add Command» в режиме создания эталона. После нажатия этой кнопки вызывается функция-обработчик этого события «private void toolStripButton1_Click(object sender, EventArgs e)». Эта функция активирует кнопку «Add Command» . После этого

вызывается функция «public bool Record()» класса WaveIn, которая начинает запись сигнала с микрофона в буфер и сохраняет этот сигнал как эталон. После того как сигнал записан, нажимается кнопка «Stop». После нажатия этой кнопки вызывается функция-обработчик события «private void toolStripButton1_Click(object sender, EventArgs e)». Эта функция активирует кнопку «Stop» и останавливает запись путем вызова функции WaveIn.StopRecord() класса WaveIn.

5.4.5 Модули анализа звукового сигнала и распознавания речи

Анализ и распознавание звукового сигнала происходит в функции «Void AudioFrame.WaveIn(short *buf,int len)» после того как сигнал записан, дискретезирован и отфильтрован.

5.5 Пример работы программы

Перед началом работы клиента, непосредственно после запуска программы необходимо ее настроить. Для этого потребуется подсоединенный к компьютеру микрофон.

Для настройки системы необходимо после запуска создать эталон слова для последующего распознавания. Для этого нужно в поле выбора режима работы программы нажать кнопку создания эталона т.е. «Add Command», в левом верхнем углу главного окна программы на панели с вкладками. После нажатия кнопки начнется запись сигнала. Далее следует произнести слово и остановить запись путем нажатия кнопки «Stop» в левом верхнем углу экрана.

После того как эталон создан и сохранен, необходимо переключить режим работы программы в «Mode - Recognition». После переключения в «Mode - Recognition» необходимо включить запись сигнала путем нажатия кнопки «File-->Start Recognition», произнести слово, которое требуется распознать, и остановить запись путем нажатия кнопки «Stop».

Распознанное слово будет выведено в текстовом поле после «You just said:».

На рисунке 5.3 показан результат работы программы. Программа работает успешно, было распознано слово «Hello». Программой было определено слово по эталону, сохраненному в базе данных программы в режиме создания эталона.

На рисунке 5.4 показан результат работы программы при помехах. Из-за созданных помех программе не удалось распознать слово. В качестве шума использовался искусственно созданный стационарный шум, возникший в результате действия нескольких независимых источников: шум людей, шум работающего двигателя автомобиля и другие окружающие шумы. Уровень шума, при котором слово не было распознано, составляет выше 85дБ, что говорит о пригодности программы для использования в автомобиле. Такой уровень шума может быть вызван очень большим скоплением автомобилей на дороге, интенсивным трамвайным движением (физический шум).

6 БЕЗОПАСНОСТЬ И ЭКОЛОГИЧНОСТЬ ПРОЕКТА

6.1 Системный анализ безопасности и надежности блока выделения начала и окончания слова, количества звуков при эксплуатации

Анализируемый блок предназначен для работы в составе устройства определения количества звуков в изолированном слове речи. Подробное описание принципа работы устройства приведено в разделе 2. Проведем процессы синтеза и анализа.

Головным является событие, когда устройство неработоспособно. Оно может быть неработоспособно в двух случаях: когда оно продолжает работать, но устранило свои технические характеристики, либо устройство полностью вышло из строя.

Рассмотрим подробно первый случай. Он может быть обусловлен двумя причинами: при подаче сигнала на вход устройства на его выходе периодически пропадают данные о количестве звуков в изолированном слове речи, либо вследствие воздействия внешних причин или неверной работы одного из блоков устройства происходит ложное срабатывание устройства. Внешними причинами могут быть: переотражение акустической волны, воздействие шумов или звуков не относящихся к анализу, а также многие другие причины. Пропуск сигнала может произойти в случае недостаточной амплитуды входного сигнала или неверной работы одного из блоков устройства, которая возможна если:

допущены дефекты в устройстве при производстве;

происходит сильное падение питающего напряжения;

неверно установлен шумовой порог,

либо по каким-нибудь другим причинам.

Второй случай – это полный выход устройства из строя, который возможен при отсутствии напряжения питания либо при выходе из строя

одного или нескольких блоков. Питание может отсутствовать по многим причинам. Часто случающиеся – это выход из строя вставки плавкой или самого блока питания, а также потеря контакта, например, при повреждении электрического кабеля. Выйти из строя могут следующие блоки: ЦЛЗ, УОРС, УУ, компаратор, счетчик, микрофон. Все они могут отказать если на них подействуют механические или электрические причины, в которые входят: возможное падение прибора, неверное обращение с прибором, воспламенение прибора, большие скачки напряжения питания, попадание воды, либо по другим причинам.

6.2 Мероприятия по повышению надежности и безопасности блока

Для повышения надежности и безопасности блока можно произвести следующие мероприятия:

1. Амортизация и демпфирование прибора, а также произведение

контроля за качеством монтажа, для уменьшения поражающего действия вибраций, связанных с падением.

2. Установка в местах, недоступных маленьким детям, дабы предотвратить неверное обращение с прибором.

3. Применение заземления и вставки плавкой для исключения воспламенения прибора при замыкании.

4. Применение стабилизатора напряжения с защитой от перегрузки, для предотвращения выхода устройства из строя при больших подачах напряжения питания.

5. Применение сплошного кожуха для избежание попадания воды на плату устройства.

6. Применение экранирования звукоизоляции в приборе в целях уменьшения воздействия внешних электромагнитных полей, а также компенсации переотражений акустической волны.

7. Применение чувствительного, малошумящего микрофона для получения необходимой амплитуды входного сигнала.

8. Произведение настройки шумового порога только квалифицированным специалистом.

6.3 Безопасность блока для природной среды

Допустимые значения вредных факторов устанавливаются стандартами ССБТ. Наибольшую опасность для окружающей среды представляет собой процесс изготовления. Это объясняется тем, что в период эксплуатации устройство практически не наносит вреда ни гидросфере, ни литосфере, практически не излучает электромагнитных полей, а если и излучает, то их значение ничтожно мало, оно наносит незначительный урон только атмосфере, так как при нагревании элементов выделяются вредные газовые примеси. Процесс утилизации тоже не несет значительного загрязнения окружающей среды ввиду отсутствия вредных, ядовитых веществ в элементах устройства, а также отсутствие газообразных веществ.

Но, процесс изготовления вреден и для атмосферы, так как во время травления печатной платы, при производстве элементов схемы и проведения пайки выделяется значительное количество вредных газообразных примесей; и для литосферы, так как остатки ненужного сырья необходимо утилизировать; и для гидросферы, так как во время производства некоторых элементов схемы возможно применение водяного охлаждения.

7.ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОЭКТА

7.1 Расчет заработной платы разработчиков

Общая продолжительность ТПП складывается из времени на всех этапах с учётом параллельности их выполнения. Фонд заработной платы рассчитан путём умножения трудоёмкости на часовую ставку.

Расходы на заработную плату разработчиков конструкции представлены в таблице 2.

Расчет часовой тарифной ставки производится путем деления месячной тарифной ставки на количество рабочих часов в месяц.

$$L_{ЧАС} = \frac{L_{CP}}{t_{CP}}$$

где $L_{ЧАС}$ - часовая ставка работников;

L_{CP} - средний месячный оклад;

$t_{ЧАС}$ - среднее количество рабочих часов в месяце,

$t_{CP} = \sum K_{ДН} \cdot t_{ДН} = 21 \cdot 8 = 168 часов$

Для программиста, электронщика 14 разряда часовая ставка составит $L_{ЧАС}$ = 4370 /168 = 26 руб./час.

Таким образом, принимая во внимание все выше указанные факторы, составляется таблица по расчету основной заработной платы участников проекта.

Таблица 1

Заработная плата участников разработки

Этап разработки	Исполнитель	Трудо-ёмкость, часы	Часовая тарифная ставка, руб.	Заработная плата, руб.
Разработка функциональной	Электронщик	40	26	1040

схемы				
Разработка программного обеспечения	программист	65	26	1690
Основная заработная плата разработчиков (ОЗП)				2730

Из таблицы 1 видно, что основная заработная плата, т.е. сумма расходов на этапе подготовки производства составит 2730 рубля.

Рассчитаем сумму дополнительной заработной платы из расчета 10% от основной заработной платы, получим:

Здоп=0,1·2730=273 руб.

Отчисления на социальные нужды – 26% от суммы основной и дополнительной заработных плат

Инуж=0,26·(2730+273)=780,78руб.

Рассчитаем накладные расходы. Норматив на накладные расходы составляет от 110 до 200% от суммы основной заработной платы разработчиков. Допустим, что накладные расходы составят 130%.

Инак=1,3·2730=3549руб.

Таблица 2

Затраты на техническую подготовку производства

Вид затрат	Сумма, руб.
Основная заработная плата (ОЗП)	2730
Дополнительная ЗП (ДЗП) разработчиков –10% от основной ЗП	273
Отчисления на социальные нужды - 26% от ОЗП+ДЗП	780,78
Накладные расходы (Н) - 130% от основной ЗП	3549
ИТОГО ($З_{тех}$):	7332,78

Как видно из таблицы 2, затраты на техническую подготовку производства $З_{тех}$ составляют 7332,78рублей.

ЗАКЛЮЧЕНИЕ

В результате работы над бакалаврским проектом был произведен обзор литературы с целью поиска существующих методов анализа речи. Также был проведен патентный поиск устройств, осуществляющих выделение признаков речевых сигналов. Оказалось, что предложенный метод анализа речевых сигналов, базирующийся на обработке сигналов во временной области, на сегодняшний день не имеет аналогов. Особенностью предложенного метода является представление модели речевого сигнала не в аддитивной форме, как в методах спектрального анализа, а в мультипликативной. Это объясняет использование ряда Тейлора при разложении полной фазовой функции речевого сигнала на компоненты, а не ряда Фурье. Характерной особенностью данного метода является выделение скорости изменения частоты речевого сигнала как информативного параметра. Ранее ни в одном методе анализа речи этого не проводилось. Также впервые была получена огибающая речевого сигнала и проведен ее спектральный анализ.

В бакалаврской работе проводилось моделирование работы устройства на ЭВМ. Так же была разработана программа на языках программирования высокого уровня C#.net и Matlab, реализующая изложенный алгоритм моделирования распознавания речевых сигналов. Полученные результаты показали возможность использования выделяемых параметров речевых сигналов для распознавания речи.

В бакалаврской работе было проведено экономическое обоснование целесообразности разработки и рассмотрены вопросы безопасности и экологичности спроектированного устройства.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Искусственный интеллект. Системы общения и экспер тные системы. Кн. 1 / Под ред. Э.В.Попова. - М.: Радио и связь, 1990. - 461 с.

2. Оппенгейн А.В., Шафер Р.В. Цифровая обработка сигналов, М.: Радио и связь, 1979 ., 347 с.

3. Рабинер Л.Р. Шафер Р.В. Цифровая обработка речевых сигналов, М.: Радио и связь, 1981 ., 258 с.

4. Литюк В.И. Методическое пособие № 2231 часть 3 «Методы расчета и проектирование цифровых многопроцессорных устройств обработки радиосигналов», Таганрог, 1995, 48 с.

5. Кузнецов В., Отт А. Автоматический синтез речи. - Таллинн: Валгус, 1989. - 135 с.

6. Методы автоматического распознавания речи / Под ред. У.Ли. - М.: Мир, 1983. - 716 с.

7. Зиндер Л.Р. Общая фонетика. - М.: Высшая школа, 1979. - 312 с.

8. Златоустова Л.В., Потапова Р.К., Трунин-Донской В.Н. Общая и прикладная фонетика. М.: МГУ, 1986. - 304 с.

9. Линдсей П., Нордман Д. Переработка информации у человека. - М.: Мир, 1974. - 550 с.

10. Потапова Р.К. Речевое управление роботом. - М.: Радио и связь, 1989. - 248 с.

11. Бакаева Т.Н. Системный анализ безопасности: Методическая разработка к самостоятельной работе по курсу «Безопасность жизнедеятельности». Таганрог: ТРТУ, 1995, 18 с.

12. Бакаева Т.Н. Безопасность жизнедеятельности. Часть 2: Безопасность в условиях производства: Учебное пособие. Таганрог: ТРТУ, 1997, 318 с.

13. Фрумкин Г.А. «Расчет и конструирование РЭА», Москва: Высшая школа, 1997, 289 с.

Приложение

1. Листинг программы – Speech Recognition

1.1) WaveIn.cs

```
// Speech recognition
// wavein => operations on incoming sound signal
using System;
using System.Threading;
using System.Runtime.InteropServices;
namespace SoundViewer
{
	internal class WaveInHelper
	{
		public static void Try(int err)
		{
			if (err != WaveNative.MMSYSERR_NOERROR)
				throw new Exception(err.ToString());
		}
	}

	public delegate void BufferDoneEventHandler(IntPtr data, int size);

	internal class WaveInBuffer : IDisposable
	{
		public WaveInBuffer NextBuffer;

		private AutoResetEvent m_RecordEvent = new AutoResetEvent(false);
		private IntPtr m_WaveIn;
```

```
        private WaveNative.WaveHdr m_Header;
        private byte[] m_HeaderData;
        private GCHandle m_HeaderHandle;
        private GCHandle m_HeaderDataHandle;

        private bool m_Recording;

        internal static void WaveInProc(IntPtr hdrvr, int uMsg, int dwUser, ref WaveNative.WaveHdr wavhdr, int dwParam2)
        {
            if (uMsg == WaveNative.MM_WIM_DATA)
            {
                try
                {
                    GCHandle h = (GCHandle)wavhdr.dwUser;
                    WaveInBuffer buf = (WaveInBuffer)h.Target;
                    buf.OnCompleted();
                }
                catch
                {
                }
            }
        }

        public WaveInBuffer(IntPtr waveInHandle, int size)
        {
            m_WaveIn = waveInHandle;
```

```
                m_HeaderHandle = GCHandle.Alloc(m_Header,
GCHandleType.Pinned);
                m_Header.dwUser = (IntPtr)GCHandle.Alloc(this);
                m_HeaderData = new byte[size];
                m_HeaderDataHandle =
GCHandle.Alloc(m_HeaderData, GCHandleType.Pinned);
                m_Header.lpData =
m_HeaderDataHandle.AddrOfPinnedObject();
                m_Header.dwBufferLength = size;

        WaveInHelper.Try(WaveNative.waveInPrepareHeader(m_WaveIn, ref
m_Header, Marshal.SizeOf(m_Header)));
            }
            ~WaveInBuffer()
            {
                Dispose();
            }

            public void Dispose()
            {
                if (m_Header.lpData != IntPtr.Zero)
                {
                    WaveNative.waveInUnprepareHeader(m_WaveIn,
ref m_Header, Marshal.SizeOf(m_Header));
                    m_HeaderHandle.Free();
                    m_Header.lpData = IntPtr.Zero;
                }
                m_RecordEvent.Close();
                if (m_HeaderDataHandle.IsAllocated)
                    m_HeaderDataHandle.Free();
```

```
            GC.SuppressFinalize(this);
        }

        public int Size
        {
            get { return m_Header.dwBufferLength; }
        }

        public IntPtr Data
        {
            get { return m_Header.lpData; }
        }

        public bool Record()
        {
            lock(this)
            {
                m_RecordEvent.Reset();
                m_Recording =
WaveNative.waveInAddBuffer(m_WaveIn, ref m_Header,
Marshal.SizeOf(m_Header)) == WaveNative.MMSYSERR_NOERROR;
                return m_Recording;
            }
        }

        public void WaitFor()
        {
            if (m_Recording)
                m_Recording = m_RecordEvent.WaitOne();
            else
```

```
                    Thread.Sleep(0);
                }

                private void OnCompleted()
                {
                    m_RecordEvent.Set();
                    m_Recording = false;
                }
            }

            public class WaveInRecorder : IDisposable
            {
                private IntPtr m_WaveIn;
                private WaveInBuffer m_Buffers; // linked list
                private WaveInBuffer m_CurrentBuffer;
                private Thread m_Thread;
                private BufferDoneEventHandler m_DoneProc;
                private bool m_Finished;

                private WaveNative.WaveDelegate m_BufferProc = new
WaveNative.WaveDelegate(WaveInBuffer.WaveInProc);

                public static int DeviceCount
                {
                    get { return WaveNative.waveInGetNumDevs(); }
                }

                public WaveInRecorder(int device, WaveFormat format, int
bufferSize, int bufferCount, BufferDoneEventHandler doneProc)
                {
```

```
                m_DoneProc = doneProc;
                WaveInHelper.Try(WaveNative.waveInOpen(out
m_WaveIn, device, format, m_BufferProc, 0,
WaveNative.CALLBACK_FUNCTION));
                AllocateBuffers(bufferSize, bufferCount);
                for (int i = 0; i < bufferCount; i++)
                {
                        SelectNextBuffer();
                        m_CurrentBuffer.Record();
                }

        WaveInHelper.Try(WaveNative.waveInStart(m_WaveIn));
                m_Thread = new Thread(new ThreadStart(ThreadProc));
                m_Thread.Start();
        }
        ~WaveInRecorder()
        {
                Dispose();
        }
        public void Dispose()
        {
                if (m_Thread != null)
                        try
                        {
                                m_Finished = true;
                                if (m_WaveIn != IntPtr.Zero)

        WaveNative.waveInReset(m_WaveIn);
                                        WaitForAllBuffers();
                                        m_Thread.Join();
```

```
                    m_DoneProc = null;
                    FreeBuffers();
                    if (m_WaveIn != IntPtr.Zero)

        WaveNative.waveInClose(m_WaveIn);
                }
                finally
                {
                    m_Thread = null;
                    m_WaveIn = IntPtr.Zero;
                }
            GC.SuppressFinalize(this);
        }
        private void ThreadProc()
        {
            while (!m_Finished)
            {
                Advance();
                if (m_DoneProc != null && !m_Finished)
                    m_DoneProc(m_CurrentBuffer.Data,
m_CurrentBuffer.Size);
                m_CurrentBuffer.Record();
            }
        }
        private void AllocateBuffers(int bufferSize, int bufferCount)
        {
            FreeBuffers();
            if (bufferCount > 0)
            {
```

```
                    m_Buffers = new WaveInBuffer(m_WaveIn, bufferSize);
                    WaveInBuffer Prev = m_Buffers;
                    try
                    {
                        for (int i = 1; i < bufferCount; i++)
                        {
                            WaveInBuffer Buf = new WaveInBuffer(m_WaveIn, bufferSize);
                            Prev.NextBuffer = Buf;
                            Prev = Buf;
                        }
                    }
                    finally
                    {
                        Prev.NextBuffer = m_Buffers;
                    }
                }
            }
            private void FreeBuffers()
            {
                m_CurrentBuffer = null;
                if (m_Buffers != null)
                {
                    WaveInBuffer First = m_Buffers;
                    m_Buffers = null;

                    WaveInBuffer Current = First;
                    do
                    {
```

```
                                        WaveInBuffer Next = Current.NextBuffer;
                                        Current.Dispose();
                                        Current = Next;
                                } while(Current != First);
                        }
                }
                private void Advance()
                {
                        SelectNextBuffer();
                        m_CurrentBuffer.WaitFor();
                }
                private void SelectNextBuffer()
                {
                        m_CurrentBuffer = m_CurrentBuffer == null ? m_Buffers : m_CurrentBuffer.NextBuffer;
                }
                private void WaitForAllBuffers()
                {
                        WaveInBuffer Buf = m_Buffers;
                        while (Buf.NextBuffer != m_Buffers)
                        {
                                Buf.WaitFor();
                                Buf = Buf.NextBuffer;
                        }
                }
        }
}
```

1.2) WaveOut.cs

```
// Speech recognition
```

```
// waveout => show graph on screen

using System;
using System.Threading;
using System.Runtime.InteropServices;

namespace SoundViewer
{
 internal class WaveOutHelper
 {
 public static void Try(int err)
 {
 if (err != WaveNative.MMSYSERR_NOERROR)
 throw new Exception(err.ToString());
 }
 }

 public delegate void BufferFillEventHandler(IntPtr data, int size);

        internal class WaveOutBuffer : IDisposable
        {
 public WaveOutBuffer NextBuffer;

 private AutoResetEvent m_PlayEvent = new AutoResetEvent(false);
 private IntPtr m_WaveOut;

 private WaveNative.WaveHdr m_Header;
 private byte[] m_HeaderData;
 private GCHandle m_HeaderHandle;
 private GCHandle m_HeaderDataHandle;
```

```
private bool m_Playing;

internal static void WaveOutProc(IntPtr hdrvr, int uMsg, int dwUser, ref WaveNative.WaveHdr wavhdr, int dwParam2)
{
if (uMsg == WaveNative.MM_WOM_DONE)
{
try
{
GCHandle h = (GCHandle)wavhdr.dwUser;
WaveOutBuffer buf = (WaveOutBuffer)h.Target;
buf.OnCompleted();
}
catch
{
}
}
}

public WaveOutBuffer(IntPtr waveOutHandle, int size)
                {
m_WaveOut = waveOutHandle;

m_HeaderHandle = GCHandle.Alloc(m_Header, GCHandleType.Pinned);
m_Header.dwUser = (IntPtr)GCHandle.Alloc(this);
m_HeaderData = new byte[size];
m_HeaderDataHandle = GCHandle.Alloc(m_HeaderData, GCHandleType.Pinned);
m_Header.lpData = m_HeaderDataHandle.AddrOfPinnedObject();
```

```
        m_Header.dwBufferLength = size;
        WaveOutHelper.Try(WaveNative.waveOutPrepareHeader(m_WaveOut, ref
m_Header, Marshal.SizeOf(m_Header)));
                    }
        ~WaveOutBuffer()
        {
        Dispose();
        }
        public void Dispose()
        {
        if (m_Header.lpData != IntPtr.Zero)
        {
        WaveNative.waveOutUnprepareHeader(m_WaveOut, ref m_Header,
Marshal.SizeOf(m_Header));
        m_HeaderHandle.Free();
        m_Header.lpData = IntPtr.Zero;
        }
        m_PlayEvent.Close();
        if (m_HeaderDataHandle.IsAllocated)
        m_HeaderDataHandle.Free();
        GC.SuppressFinalize(this);
        }

        public int Size
        {
        get { return m_Header.dwBufferLength; }
        }

        public IntPtr Data
        {
```

```
get { return m_Header.lpData; }
}

public bool Play()
{
lock(this)
{
m_PlayEvent.Reset();
m_Playing = WaveNative.waveOutWrite(m_WaveOut, ref m_Header, Marshal.SizeOf(m_Header)) == WaveNative.MMSYSERR_NOERROR;
return m_Playing;
}
}
public void WaitFor()
{
if (m_Playing)
{
m_Playing = m_PlayEvent.WaitOne();
}
else
{
Thread.Sleep(0);
}
}
public void OnCompleted()
{
m_PlayEvent.Set();
m_Playing = false;
}
}
```

```
public class WaveOutPlayer : IDisposable
{
private IntPtr m_WaveOut;
private WaveOutBuffer m_Buffers; // linked list
private WaveOutBuffer m_CurrentBuffer;
private Thread m_Thread;
private BufferFillEventHandler m_FillProc;
                private bool m_Finished;
                private byte m_zero;

private WaveNative.WaveDelegate m_BufferProc = new WaveNative.WaveDelegate(WaveOutBuffer.WaveOutProc);

public static int DeviceCount
{
get { return WaveNative.waveOutGetNumDevs(); }
}

public WaveOutPlayer(int device, WaveFormat format, int bufferSize, int bufferCount, BufferFillEventHandler fillProc)
{
                        m_zero = format.wBitsPerSample == 8 ? (byte)128 : (byte)0;
m_FillProc = fillProc;
WaveOutHelper.Try(WaveNative.waveOutOpen(out m_WaveOut, device, format, m_BufferProc, 0, WaveNative.CALLBACK_FUNCTION));
AllocateBuffers(bufferSize, bufferCount);
m_Thread = new Thread(new ThreadStart(ThreadProc));
m_Thread.Start();
```

```
}
~WaveOutPlayer()
{
Dispose();
}
public void Dispose()
{
if (m_Thread != null)
                        try
                        {
                                m_Finished = true;
                                if (m_WaveOut != IntPtr.Zero)

WaveNative.waveOutReset(m_WaveOut);
                                m_Thread.Join();
                                m_FillProc = null;
                                FreeBuffers();
                                if (m_WaveOut != IntPtr.Zero)

WaveNative.waveOutClose(m_WaveOut);
                        }
                        finally
                        {
                                m_Thread = null;
                                m_WaveOut = IntPtr.Zero;
                        }
GC.SuppressFinalize(this);
}
private void ThreadProc()
{
```

```
        while (!m_Finished)
        {
        Advance();
                                if (m_FillProc != null && !m_Finished)
                                        m_FillProc(m_CurrentBuffer.Data,
m_CurrentBuffer.Size);
                                else
                                {
                                        // zero out buffer
                                        byte v = m_zero;
                                        byte[] b = new byte[m_CurrentBuffer.Size];
                                        for (int i = 0; i < b.Length; i++)
                                                b[i] = v;
                                        Marshal.Copy(b,   0,   m_CurrentBuffer.Data,
b.Length);

                                }
        m_CurrentBuffer.Play();
        }
                                WaitForAllBuffers();
                        }
        private void AllocateBuffers(int bufferSize, int bufferCount)
        {
        FreeBuffers();
        if (bufferCount > 0)
        {
        m_Buffers = new WaveOutBuffer(m_WaveOut, bufferSize);
        WaveOutBuffer Prev = m_Buffers;
        try
        {
```

```
for (int i = 1; i < bufferCount; i++)
{
WaveOutBuffer Buf = new WaveOutBuffer(m_WaveOut, bufferSize);
Prev.NextBuffer = Buf;
Prev = Buf;
}
}
finally
{
Prev.NextBuffer = m_Buffers;
}
}
}
private void FreeBuffers()
{
m_CurrentBuffer = null;
if (m_Buffers != null)
{
WaveOutBuffer First = m_Buffers;
m_Buffers = null;

WaveOutBuffer Current = First;
do
{
WaveOutBuffer Next = Current.NextBuffer;
Current.Dispose();
Current = Next;
} while(Current != First);
}
}
```

```
        private void Advance()
        {
        m_CurrentBuffer = m_CurrentBuffer == null ? m_Buffers :
m_CurrentBuffer.NextBuffer;
        m_CurrentBuffer.WaitFor();
        }
        private void WaitForAllBuffers()
        {
        WaveOutBuffer Buf = m_Buffers;
        while (Buf.NextBuffer != m_Buffers)
        {
        Buf.WaitFor();
        Buf = Buf.NextBuffer;
        }
        }
        }
       }
```

1.3) SignalGenerator.cs

```
// Speech recognition
// singal generator => to generate various signals like sawtooth…

using System;
using System.Collections.Generic;
using System.Text;

namespace SoundViewer
{
 class SignalGenerator
 {
```

```
private string _waveForm = "Sine";
private double _amplitude = 128.0;
private double _samplingRate = 44100;
private double _frequency = 5000.0;
private double _dcLevel = 0.0;
private double _noise = 0.0;
private int _samples = 16384;
private bool _addDCLevel = false;
private bool _addNoise = false;

public SignalGenerator()
{
}

public void SetWaveform(string waveForm)
{
_waveForm = waveForm;
}

public String GetWaveform()
{
return _waveForm;
}

public void SetAmplitude(double amplitude)
{
_amplitude = amplitude;
}

public double GetAmplitude()
```

```
{
return _amplitude;
}

public void SetFrequency(double frequency)
{
_frequency = frequency;
}

public double GetFrequency()
{
return _frequency;
}

public void SetSamplingRate(double rate)
{
_samplingRate = rate;
}

public double GetSamplingRate()
{
return _samplingRate;
}

public void SetSamples(int samples)
{
_samples = samples;
}

public int GetSamples()
```

```
{
return _samples;
}

public void SetDCLevel(double dc)
{
_dcLevel = dc;
}

public double GetDCLevel()
{
return _dcLevel;
}

public void SetNoise(double noise)
{
_noise = noise;
}

public double GetNoise()
{
return _noise;
}

public void SetDCLevelState(bool dcstate)
{
_addDCLevel = dcstate;
}

public bool IsDCLevel()
```

```
{
return _addDCLevel;
}

public void SetNoiseState(bool noisestate)
{
_addNoise = noisestate;
}

public bool IsNoise()
{
return _addNoise;
}

public double[] GenerateSignal()
{
double[] values = new double[_samples];
if (_waveForm.Equals("Sine"))
{
double theta = 2.0 * Math.PI * _frequency / _samplingRate;
for (int i = 0; i < _samples; i++)
{
values[i] = _amplitude * Math.Sin(i * theta);
}
}
if (_waveForm.Equals("Cosine"))
{
double theta = 2.0f * (double)Math.PI * _frequency / _samplingRate;
for (int i = 0; i < _samples; i++)
values[i] = _amplitude * Math.Cos(i * theta);
```

```
}
if (_waveForm.Equals("Square"))
{
double p = 2.0 * _frequency / _samplingRate;
for (int i = 0; i < _samples; i++)
values[i] = Math.Round(i * p) % 2 == 0 ? _amplitude : -_amplitude;
}
if (_waveForm.Equals("Triangular"))
{
double p = 2.0 * _frequency / _samplingRate;
for (int i = 0; i < _samples; i++)
{
int ip = (int)Math.Round(i * p);
values[i] = 2.0 * _amplitude * (1 - 2 * (ip % 2)) * (i * p - ip);
}
}
if (_waveForm.Equals("Sawtooth"))
{
for (int i = 0; i < _samples; i++)
{
double q = i * _frequency / _samplingRate;
values[i] = 2.0 * _amplitude * (q - Math.Round(q));
}
}
if (_addDCLevel)
{
for (int i = 0; i < _samples; i++)
values[i] += _dcLevel;
}
if (_addNoise)
```

```
 {
 Random r = new Random();
 for (int i = 0; i < _samples; i++)
 values[i] += _noise * r.Next();
 }
 return values;
 }
 }
}
```

1.4) AudioFrame.cs

```
// Speech recognition
// audioframe => working on audio frame
using System;
using System.Drawing;
using System.Windows.Forms;

namespace SoundViewer
{
 class AudioFrame
 {
 private Bitmap _canvasTimeDomain;
 private Bitmap _canvasFrequencyDomain;
 private double[] _waveLeft;
 private double[] _waveRight;
 private double[] _fftLeft;
 private double[] _ftRight;
 private SignalGenerator _signalGenerator;
 private bool _isTest = false;
```

```
public AudioFrame(bool isTest)
{
_isTest = isTest;
}

/// <summary>
/// Process 16 bit sample
/// </summary>
/// <param name="wave"></param>
public void Process(ref byte[] wave)
{
_waveLeft = new double[wave.Length / 4];
_waveRight = new double[wave.Length / 4];

if (_isTest == false)
{
// Split out channels from sample
int h = 0;
for (int i = 0; i < wave.Length; i += 4)
{
_waveLeft[h] = (double)BitConverter.ToInt16(wave, i);
_waveRight[h] = (double)BitConverter.ToInt16(wave, i + 2);
h++;
}
}
else
{
// Generate artificial sample for testing
_signalGenerator = new SignalGenerator();
_signalGenerator.SetWaveform("Sine");
```

```
        _signalGenerator.SetSamplingRate(44100);
        _signalGenerator.SetSamples(16384);
        _signalGenerator.SetFrequency(5000);
        _waveLeft = _signalGenerator.GenerateSignal();
        _waveRight = _signalGenerator.GenerateSignal();
        }

        // Generate frequency domain data in decibels
        _fftLeft = FourierTransform.FFTDb(ref _waveLeft);
        _fftRight = FourierTransform.FFTDb(ref _waveRight);
        }

        /// Render time domain to PictureBox
        public void RenderTimeDomain(ref PictureBox pictureBox)
        {
        // Set up for drawing
        _canvasTimeDomain = new Bitmap(pictureBox.Width, pictureBox.Height);
        Graphics offScreenDC = Graphics.FromImage(_canvasTimeDomain);
        SolidBrush brush = new System.Drawing.SolidBrush(Color.FromArgb(0, 0,
0));
        Pen pen = new System.Drawing.Pen(Color.WhiteSmoke);

        // Determine channnel boundries
        int width = _canvasTimeDomain.Width;
        int center = _canvasTimeDomain.Height / 2;
        int height = _canvasTimeDomain.Height;

        offScreenDC.DrawLine(pen, 0, center, width, center);

        int leftLeft = 0;
```

```
int leftTop = 0;
int leftRight = width;
int leftBottom = center - 1;

int rightLeft = 0;
int rightTop = center + 1;
int rightRight = width;
int rightBottom = height;

// Draw left channel
double yCenterLeft = (leftBottom - leftTop) / 2;
double yScaleLeft = 0.5 * (leftBottom - leftTop) / 32768; // a 16 bit sample has values from -32768 to 32767
int xPrevLeft = 0, yPrevLeft = 0;
for (int xAxis = leftLeft; xAxis < leftRight; xAxis++)
{
int yAxis = (int)(yCenterLeft + (_waveLeft[_waveLeft.Length / (leftRight - leftLeft) * xAxis] * yScaleLeft));
if (xAxis == 0)
{
xPrevLeft = 0;
yPrevLeft = yAxis;
}
else
{
pen.Color = Color.LimeGreen;
offScreenDC.DrawLine(pen, xPrevLeft, yPrevLeft, xAxis, yAxis);
xPrevLeft = xAxis;
yPrevLeft = yAxis;
}
```

```
        }

        // Draw right channel
        int xCenterRight = rightTop + ((rightBottom - rightTop) / 2);
        double yScaleRight = 0.5 * (rightBottom - rightTop) / 32768; // a 16 bit sample has values from -32768 to 32767
        int xPrevRight = 0, yPrevRight = 0;
        for (int xAxis = rightLeft; xAxis < rightRight; xAxis++)
        {
        int yAxis = (int)(xCenterRight + (_waveRight[_waveRight.Length / (rightRight - rightLeft) * xAxis] * yScaleRight));
        if (xAxis == 0)
        {
        xPrevRight = 0;
        yPrevRight = yAxis;
        }
        else
        {
        pen.Color = Color.LimeGreen;
        offScreenDC.DrawLine(pen, xPrevRight, yPrevRight, xAxis, yAxis);
        xPrevRight = xAxis;
        yPrevRight = yAxis;
        }
        }

        // Clean up
        pictureBox.Image = _canvasTimeDomain;
        offScreenDC.Dispose();
        }
```

```
        /// <summary>
        /// Render frequency domain to PictureBox
        /// </summary>
        /// <param name="pictureBox"></param>
        public void RenderFrequencyDomain(ref PictureBox pictureBox)
        {
        // Set up for drawing
        _canvasFrequencyDomain = new Bitmap(pictureBox.Width,
pictureBox.Height);
        Graphics offScreenDC = Graphics.FromImage(_canvasFrequencyDomain);
        SolidBrush brush = new System.Drawing.SolidBrush(Color.FromArgb(0, 0,
0));
        Pen pen = new System.Drawing.Pen(Color.WhiteSmoke);

        // Determine channnel boundries
        int width = _canvasFrequencyDomain.Width;
        int center = _canvasFrequencyDomain.Height / 2;
        int height = _canvasFrequencyDomain.Height;

        offScreenDC.DrawLine(pen, 0, center, width, center);

        int leftLeft = 0;
        int leftTop = 0;
        int leftRight = width;
        int leftBottom = center - 1;

        int rightLeft = 0;
        int rightTop = center + 1;
        int rightRight = width;
        int rightBottom = height;
```

```
        // Draw left channel
        for (int xAxis = leftLeft; xAxis < leftRight; xAxis++)
        {
        double amplitude = (int)_fftLeft[(int)(((double)(_fftLeft.Length) / (double)(width)) * xAxis)];
        if (amplitude < 0) // Drop negative values
        amplitude = 0;
        int yAxis = (int)(leftTop + ((leftBottom - leftTop) * amplitude) / 100); // Arbitrary factor
        pen.Color = Color.FromArgb(120, 120, (int)amplitude % 255);
        offScreenDC.DrawLine(pen, xAxis, leftTop, xAxis, yAxis);
        }

        // Draw right channel
        for (int xAxis = rightLeft; xAxis < rightRight; xAxis++)
        {
        double amplitude = (int)_fftRight[(int)(((double)(_fftRight.Length) / (double)(width)) * xAxis)];
        if (amplitude < 0)
        amplitude = 0;
        int yAxis = (int)(rightBottom - ((rightBottom - rightTop) * amplitude) / 100);
        pen.Color = Color.FromArgb(120, 120, (int)amplitude % 255);
        offScreenDC.DrawLine(pen, xAxis, rightBottom, xAxis, yAxis);
        }

        // Clean up
        pictureBox.Image = _canvasFrequencyDomain;
        offScreenDC.Dispose();
```

```
 }
 void WaveIn(short* buf, int len)
 {
 //raspoznavat
 }



 }
 }
```

2. Листинг программы – Speech Recognition (Matlab)

2.1) CMN.m

```
function NormMatrix = CMN(Matrix)
[r,c]=size(Matrix);
NormMatrix=zeros(r,c);
for i=1:c
 MatMean=mean(Matrix(:,i)); %Derives mean for each column i in utterance
 NormMatrix(:,i)=Matrix(:,i)-MatMean; %Subtracts mean from each element in
End
```

2.2) Recognition.m

```
clear all;
close all;
ncoeff = 13; %Required number of mfcc coefficients
N = 20; %Number of words in vocabulary
k = 3; %Number of nearest neighbors to choose
```

```
fs=16000; %Sampling rate
duration1 = 0.1; %Initial silence duration in seconds
duration2 = 2; %Recording duration in seconds
G=2; %vary this factor to compensate for amplitude variations
NSpeakers = 5; %Number of training speakers

fprintf('Press any key to start %g seconds of speech recording...', duration2);
pause;
silence = wavrecord(duration1*fs, fs);
fprintf('Recording speech...');
speechIn = wavrecord(duration2*fs, fs); % duration*fs is the total number of
sample points
fprintf('Finlshed recording.\n');
fprintf('System is trying to recognize what you have spoken...\n');
speechIn1 = [silence;speechIn]; %pads with 150 ms silence
speechIn2 = speechIn1.*G;
speechIn3 = speechIn2 - mean(speechIn2); %DC offset elimination
speechIn = nreduce(speechIn3,fs); %Applies spectral subtraction
rMatrix1 = mfccf(ncoeff,speechIn,fs); %Compute test feature vector
rMatrix = CMN(rMatrix1); %Removes convolutional noise

Sco = DTWScores(rMatrix,N); %computes all DTW scores
[SortedScores,EIndex] = sort(Sco); %Sort scores increasing
K_Vector = EIndex(1:k); %Gets k lowest scores
Neighbors = zeros(1,k); %will hold k-N neighbors

for t = 1:k
 u = K_Vector(t);
 for r = 1:NSpeakers-1
 if u <= (N)
```

```
 break
 else u = u - (N);
 end
 end
 Neighbors(t) = N;

end

%Apply k-Nearest Neighbor rule
Nbr = Neighbors
%sortk = sort(Nbr);
[Modal.Freq] = mode(Nbr); %most frequent value
Word =
strvcat('One','Two','Three','Four','Five','Six','Seven','Eight','Nine','Ten','Yes','No','He
llo','Open','Close','Start','Stop','Dial','On','Off');
if mean(abs(speechIn)) < 0.01
 fprintf('No microphone connected or you have not said anything.\n');
elseif ((k/Freq) > 2) %if no majority
 fprintf('The word you have said could not be properly recognised.\n');
else
 fprintf('You have just said %s.\n',Word(Modal,:)); %Prints recognized word
 end
```

2.3) setTemplates.m

```
ncoeff=13; %Required number of mfcc coefficients
fMatrix1 = cell(1,20);
fMatrix2 = cell(1,20);
fMatrix3 = cell(1,20);
```

```
fMatrix4 = cell(1,20);

for j = 1:20
 q = ['C:\SpeechData\Amir\5_' num2str(j) '.wav'];
 [speechIn1,FS1] = wavread(q);
 speechIn1 = myVAD(speechIn1); %Speech endpoint trimming
 fMatrix1(1,j) = {mfccf(ncoeff,speechIn1,FS1)}; %MFCC coefficients are
 %computed here
end

for k = 1:20
 q = ['C:\SpeechData\Ayo\5_' num2str(k) '.wav'];
 [speechIn2,FS2] = wavread(q);
 speechIn2 = myVAD(speechIn2);
 fMatrix2(1,k) = {mfcvcf(ncoeff,speechIn2,FS2)};
end

for l = 1:20
 q = ['C:\SpeechData\Sameh\5_' num2str(l) '.wav'];
 [speechIn3,F3] = wavread(q);
 speechIn3 = myVAD(speechIn3);
 fMatrix3(1,l) = {mfccf(ncoeff,speechIn3,FS3)};
end

for m = 1:20
 q = ['C:\SpeechData\Jim\5_' num2str(m) '.wav'];
 [speechIn4,FS4] = wavread(q);
 speechIn4 = myVAD(speechIn4);
 fMatrix4(1,m) = {mfccf(ncoeff,speechIn4,FS4)};
end
```

```
for n = 1:20
 q = ['C:\SpeechData\Tope\5_' num2str(n) '.wav'];
 [speechIn5,FS5] = wavread(q);
 speechIn5 = myVAD(speechIn5);
 fMatrix5(1,n) = {mfccf(ncoeff,speechIn5,FS5)};
end

%Converts the cells containing all matrices to structures and save
%structures in matlab .mat files in the working directory.
fields = {'One','Two','Three','Four','Five','Six','Seven','Eight','Nine','Ten','Yes','No','Hello','Open','Close','Start','Stop','Dial','On','Off'};
s1 = cell2struct(fMatrix1, fields, 2);
save Vectors1.mat -struct s1;
s2 = cell2struct(fMatrix2, fields, 2);
save Vectors2.mat -struct s2;
s3 = cell2struct(fMatrix3, fields, 2);
save Vectors3.mat -struct s3;
s4 = cell2struct(fMatrix4, fields, 2);
save Vectors4.mat -struct s4;
s5 = cell2struct(fMatrix5, fields, 2);
save Vectors5.mat -struct s5;
```