РЕФЕРАТ

Метою дослідження є розробка програмного забезпечення для голосового керування тривимірними моделями функціонування промислових роботів.

Основними задачами є аналіз методів цифрової обробки звукових сигналів, аналіз систем розпізнання мовлення, розробка програмного забезпечення для розпізнання команд керування промисловим роботом.

Розглядаються питання обробки звукової інформації, її аналізу та фільтрації. Проаналізовані методи реалізації систем розпізнання мовлення.

Реалізовано програмне забезпечення для запису, відтворення й аналізу звукової інформації. Програмне забезпечення розроблене в середовищі розробки програмного забезпечення Visual C++ 6.0 з використанням мультимедійної бібліотеки mmsystem, а також графічної бібліотеки OpenGL.

СПЕКТРАЛЬНИЙ АНАЛІЗ, ДИСКРЕТНЕ ПЕРЕТВОРЕННЯ ФУР'Є, MMSYSTEM, ГРАФІЧНА БІБЛІОТЕКА, OPENGL, СЕРЕДОВИЩЕ ПРОГРАМУВАННЯ, VISUAL C++.

РЕФЕРАТ

Целью исследования является разработка программного обеспечения для голосового управления трехмерными моделями функционирования промышленных роботов.

Основными задачами является анализ методов цифровой обработки звуковых сигналов, анализ систем распознавания речи, разработка программного обеспечения для распознавания команд управления промышленным роботом.

Рассматриваются вопросы обработки звуковой информации, её анализа и фильтрации. Проанализированы методы реализации систем распознавания речи.

Реализовано программное обеспечение для записи, воспроизведения и анализа звуковой информации. Программное обеспечение разработано в среде разработки программного обеспечения Visual С++ 6.0 с использованием мультимедийной библиотеки mmsystem, а также графической библиотеки OpenGL.

СПЕКТРАЛЬНЫЙ АНАЛИЗ, ДИСКРЕТНОЕ ПРЕОБРАЗОВАНИЕ ФУРЬЕ, MMSYSTEM, ГРАФИЧЕСКАЯ БИБЛИОТЕКА, OPENGL, СРЕДА ПРОГРАММИРОВАНИЯ, VISUAL С++.

THE ABSTRACT

Software development for the voice control by the three-dimensional models of industrial robots functioning is a purpose of research.

Basic tasks are the analysis of the digital sounds signal processing methods, analysis of the speech recognition systems, software development for recognition of control commands by an industrial robot.

The questions of sound information processing, of its analysis and filtration are examined. The methods of implementation of the speech recognition systems are analyzed.

Software for a record, reproducing and analysis of sound information is implemented. Software is developed in the environment of Visual C++ 6.0 with the using of mmsystem multimedia library, and also OPENGL graphic library.

SPECTRAL ANALYSIS, DISCRETE TRANSFORMATION OF FURJE, MMSYSTEM, GRAPHIC LIBRARY, OPENGL, PROGRAMMING ENVIRONMENT, VISUAL C++.

Перечень условных сокращений, обозначений, терминов

ПР - промышленный робот

ЭВМ - электронно вычислительная машина

ДПФ - дискретное преобразование Фурье

БПФ - быстрое преобразование Фур'е

ЛП - линейное предсказание

PCM - Pulse-Code Modulation

Содержание

Введение

1. Цифровая обработка сигналов и её использование в системах распознавания речи

1. Дискретные сигналы и методы их преобразования
2. Основы цифровой фильтрации
3. Особенности акустической фонетики и её\* учёт при обработке речевых сигналов
4. Обработка речевого сигнала во временной области

2. Реализация систем распознавания речи

1. Гомоморфная обработка речи
2. Кодирование речи на основе линейного предсказания
3. Цифровая обработка речи в системах речевого общения человека с машиной

3. Разработка программного обеспечения для распознавания команд управления промышленным роботом

1. Реализация интерфейса записи и воспроизведения звукового сигнала в операционной системе Microsoft Windows
2. Реализация программного обеспечения для записи, воспроизведения и анализа звукового сигнала
3. Реализация функции распознавания голосовых команд голосового управления промышленным роботом

3.4 Реализация голосового управления трёхмерными моделями промышленного робота

Выводы

Перечень ссылок

Приложение А. Элементы текстов программы

Введение

Распознавание человеческой речи является одной из сложных научно-технических задач. В настоящее время пользователями вычислительных машин и средств, оснащенных вычислительными машинами, становятся люди, не являющиеся специалистами в области программирования. Проблема речевого управления возникла, кроме того, в связи с тем, что в некоторых областях применения речь стала единственно возможным средством общения с техникой (в условиях перегрузок, темноты или резкого изменения освещенности, при занятости рук, чрезвычайной сосредоточенности внимания на объекте, который не позволяет отвлечься ни на секунду, и т.д.). Хотя в этой области и достигнуты существенные успехи, тем не менее, системы распознавания еще весьма далеки по своим возможностям от человеческих.

Проблема реализации речевого диалога человека и технических средств - актуальная задача современной кибернетики.

Задача машинного распознавания речи привлекает внимание специалистов уже очень давно. Тем не менее, продвинуться далеко в этом направлении пока не удалось. Чисто формально процесс распознавания речи можно описать буквально в нескольких фразах. Аналоговый сигнал, генерируемый микрофоном, оцифровывается, и далее в речи выделяются так называемые фонемы, то есть элементарные фрагменты, из которых состоят все произносимые слова. Затем определяется, какое слово, какому сочетанию фонем соответствует, и строится соответствующий словарь. Распознать слово - значит найти его в этом словаре по произнесенному сочетанию фонем. По мере развития компьютерных систем становится все более очевидным, что использование этих систем намного расширится, если станет возможным использование человеческой речи при работе непосредственно с компьютером, и в частности станет возможным управление машиной обычным голосом в реальном времени, а также ввод и вывод информации в виде обычной человеческой речи.

В настоящее время всё более актуальным становится управление роботом при помощи голосовых команд. Однако создание программного обеспечения для голосового управления промышленным роботом предусматривает проведение экспериментов во время разработки программы на всех этапах разработки. Проведение таких экспериментов, обеспечивающих устранение недостатков, ошибок программы, является экономически невыгодным в условиях промышленного производства и приводит к повышению стоимости разработки и отладки программного обеспечения. Для уменьшения затрат на создание программного обеспечения целесообразно разработать программу, которая обеспечит трёхмерное моделирование голосового управления промышленным роботом, что приводит к необходимости проведения экспериментов в условиях производства лишь на последнем этапе разработки программного обеспечения.

Темой данного исследования является голосовое управление трёхмерными моделями функционирования промышленных роботов. Его задачами является анализ методов цифровой обработки звуковых сигналов, анализ систем распознавания речи, разработка программного обеспечения для распознавания команд управления промышленным роботом.

1. Цифровая обработка сигналов и её использование в системах распознавания речи

1.1 Дискретные сигналы и методы их преобразования

Акустическое колебание, формируемое в речевом тракте человека, является непрерывно изменяющимся процессом. С математической точки зрения его можно описать функцией непрерывного времени 1. Аналоговые (непрерывные во времени) сигналы будут обозначаться через ха(1). Речевой сигнал можно представить и последовательностью чисел. Последовательности обозначаются через х(п). Если последовательность чисел представляет собой последовательность мгновенных значений, аналогового сигнала, взятых периодически с интервалом Т, то эта операция дискретизации обозначается через ха(пТ). На рис. 1.1 показан пример речевого сигнала в аналоговой форме и в виде последовательности отсчетов, взятых с частотой дискретизации 8 кГц.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 32 мс |  |  | | Г  ^ | |
| .„...„.„ .!-«•-■- и-1" ".|||||11||||И..||| |  | | |  | |
| ...., 256 отсчё! | |||| ■  гов | \ | ||| г | 1 | ► |

Рис. 1.1 – Представление речевого сигнала

Для удобства даже при рассмотрении дискретных сигналов иногда на графике будет изображается непрерывная функция, которая может рассматриваться как огибающая последовательности отсчетов. При изучении систем цифровой обработки речи требуется несколько специальных последовательностей. Единичный отсчет или последовательность, состоящая из одного единичного импульса, определяется как

(1.1)



Последовательность единичного скачка имеет вид

(1.2)



Экспоненциальная последовательность

(1.3)



Если *а* - комплексное число, т. е. , то



(1.4)



Если *z*=1 и , *х*(*n*) - комплексная синусоида; если . *х*(*n*) -действительное; если *z*<1 и , то *х*(*n*) - экспоненциально-затухающая осциллирующая последовательность. Последовательности этого типа часто используются при представлении линейных систем и моделировании речевых сигналов.



Обработка сигналов включает преобразование их в форму, удобную для дальнейшего использования. Таким образом, предметом интерес представляют дискретные системы или, что то же самое, преобразования входной последовательности в выходную. Подобные преобразования далее изображаются на структурных схемах. Многие системы анализа речевых сигналов разработаны для оценивания переменных во времени параметров по последовательности мгновенных значений речевого колебания. Подобные системы имеют многомерный выход, т. е. одномерная последовательность на входе, представляющая собой речевой сигнал, преобразуется в векторную последовательность на выходе.

При обработке речевых сигналов особенно широкое применение находят системы, инвариантные к временному сдвигу. Такие системы полностью описываются откликом на единичный импульс, Сигнал на выходе системы может быть рассчитан по сигналу на входе и отклику на единичный импульс *h*(*n*) с помощью дискретной свертки

(1.5a)



где символ \* обозначает свертку. Эквивалентное выражение имеет вид

(1.5б)



Линейные системы, инвариантные к временному сдвигу, применяются при фильтрации сигнала и, что более важно, они полезны как модели речеобразования.

Анализ сигналов и расчет систем значительно облегчаются при их описании в частотной области. В этой связи полезно кратко остановиться на представлении сигналов и систем в дискретном времени с использованием преобразования Фурье и *z*-преобразования [1].

1.1.1 Прямое и обратное г-преобразование

Прямое и обратное г-преобразование последовательности определяется двумя уравнениями:

(1.6a)



(1.6б)



Прямое z-преобразование *х*(*n*) определяется уравнением (1.6а). В общем случае *Х*(*z*) - бесконечный ряд по степеням z-1; последовательность *х*(*n*) играет роль коэффициентов ряда. В общем случае подобные степенные ряды сходятся к конечному пределу только для некоторых значений z. Достаточное условие сходимости имеет вид

(1.7)



Множество значений, для которых ряды сходятся, образует область на комплексной плоскости, известную как область сходимости. В общем случае эта область имеет вид [2]

(1.8)



1.1.2 Преобразование Фурье

Описание сигнала в дискретном времени с помощью преобразование Фурье задаётся в виде

(1.9a)



(1.9б)



Эти уравнения представляют собой частный случай уравнений (1.6а,б).

Преобразование Фурье получается путём вычисления z -преобразования на единичной окружности, т. е. подстановкой . Частота может быть интерпретирована как угол на z - плоскости. Достаточное условие существования преобразования Фурье можно получить, подставляя в (1.7)



(1.10)



Важная особенность преобразования Фурье последовательности состоит в том, что оно является периодической функцией со с периодом 2к. С другой стороны, поскольку представляет собой значение Х(z) на единичной окружности, оно должно повторяться после каждого полного обхода этой окружности, т. е. когда со изменится на рад [1].



1.1.3 Дискретное преобразование Фурье

Как и в случае аналоговых сигналов, если последовательность периодическая с периодом *N*, т. е.

(1.11)



то *х*(*n*) можно представить в виде суммы синусоид, а не в виде интеграла. Преобразование Фурье для периодической последовательности имеет вид

(1.12а)



(1.12б)



Это точное представление периодической последовательности. Однако, основное преимущество данного описания заключается в возможности несколько иной интерпретации уравнений (1.12). Рассмотрим последовательность конечной длины *х*(*n*), равную нулю вне интервала B этом случае *z*-преобразование имеет вид



(1.13)



Если записать *X*(*z*) в *N* равноотстоящих точках единичной окружности, т. е. , *k*= 0, 1,…,*N*-1, то получим



(1.14)



Если при этом построить периодическую последовательность в виде бесконечного числа повторений сегмента *х*(*n*),

(1.15)



то отсчеты (), как это видно из (1.12а) и (1.14), будут представлять собой коэффициенты Фурье периодической последовательности *х*(*n*) в (1.15). Таким образом, последовательность длиной *N* можно точно описать с помощью дискретного преобразования Фурье (ДПФ) в виде



(1.16)



(1.17)



Следует иметь в виду, что все последовательности при использовании ДПФ ведут себя так, как если бы они были периодическими функциями, т. е. ДПФ является на самом деле представлением периодической функции времени, заданной (1.15). Несколько иной подход при использовании ДПФ заключается в том, что индексы последовательности интерпретируются по модулю *N*. Это следует из того факта, что если *х*(*n*) имеет длину *N*, то



Введение двойных обозначений позволяет отразить периодичность, присущую представлению с помощью ДПФ. Эта периодичность существенно отражается на свойствах ДПФ. Очевидно, что задержка последовательности должна рассматриваться по модулю *N*. Это приводит, например, к некоторым особенностям выполнения дискретной свертки.

Дискретное преобразование Фурье со всеми его особенностями является важным способом описания сигналов по следующим причинам: 1) ДПФ можно рассматривать как дискретизированный вариант z -преобразования (или преобразования Фурье) последовательности конечной длительности; 2) ДПФ очень сходно по своим свойствам (с учетом периодичности) с преобразованием Фурье и *z*-преобразованием; 3) *N* значений *Х*(*k*) можно вычислить с использованием эффективного (время вычисления пропорционально *NlogN*) семейства алгоритмов, известных под названием быстрых преобразований Фурье (БПФ).

Дискретное преобразование Фурье широко используется при вычислении корреляционных функций, спектров и при реализации цифровых фильтров, а также часто используется и при обработке речевых сигналов [1-5].

1.1.4 Спектральный анализ

Спектральный анализ – это метод обработки сигналов, который позволяет выявить частотный состав сигнала. Поскольку анализируемые сигналы во многих случаях имеют случайный характер, то важную роль в спектральном анализе играют методы математической статистики. Частотный состав сигналов определяют путем вычисления оценок спектральной плотности мощности (СПМ). Задачами вычисления СПМ являются обнаружение гармонических составляющих в анализируемом сигнале и оценивание их параметров. Для решения указанных задач требуется соответственно высокая разрешающая способность по частоте и высокая статистическая точность оценивания параметров. Эти два требования противоречивы. Аргументы в пользу выбора высокого разрешения или высокой точности оценки СПМ зависят от того, что интересует исследователя: устойчивые оценки в пределах всего диапазона частот или высокая степень обнаруживаемости периодических составляющих.

Все методы цифрового спектрального анализа можно разделить, на две группы [6-7]: классические методы, базирующиеся на использовании преобразований Фурье, и методы параметрического моделирования, в которых выбирается некоторая линейная модель формирующего фильтра и оцениваются его параметры. К первой группе относят корреляционный и периодограммные методы. Ко второй группе относят методы оценивания СПМ на основе авторегрессии скользящего среднего и др.

Периодограммный метод обеспечивает вычисление оценки СПМ непосредственно по числовой последовательности *х*[*nТ*0], формируемой путем дискретизации стационарного эргодического случайного процесса *x*(*t*). Периодограммная оценка СПМ равна [6-7]

(1.18)



Выражение (1.18) соответствует возможности вычисления СПМ с помощью преобразования Фурье непосредственно по реализации исходного сигнала.

Вычисленная с помощью (1.18) оценка СПМ является несостоятельной, т.е. с увеличением *N* она не улучшается. Для получения состоятельной оценки ее необходимо сглаживать. Кроме этого, при выполнении преобразования Фурье последовательности *х*[*nТ*0] конечной длины /*V* происходит «размывание» спектра, которое также оказывает влияние на состоятельность оценки СПМ.

Ограничение последовательности *х*[*nТ*0] конечным числом значений равносильно умножению исходной бесконечной последовательности *х*0 [*nТ*0] на другую последовательность

(1.19)



которую называют прямоугольным окном. Тогда можно записать

(1.20)



Преобразование Фурье последовательности *х*[*nТ*0] равно свертке преобразований Фурье последовательности *х*0 [*nТ*0] и прямоугольного окна *w*[*nТ*0]

(1.21)

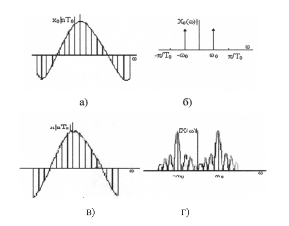


Рисунок 1.2 – Размывание спектра: (а – исходная бесконечная последовательность; б – модуль преобразования Фурье от *х*0 [*nТ*0]; в - последовательность *х*0 [*nТ*0], умноженная на прямоугольное окно; г - модуль преобразования Фурье последовательности *х*[*nТ*0])

При выборе оконных функций используются следующие параметры: ширина основного лепестка, максимальный уровень боковых лепестков, скорость спадания уровня боковых лепестков [6,7,10].

Здесь ширина основного лепестка определена на уровне 3 дБ ниже его максимума и измерена в единицах разрешения преобразования Фурье, т.е. 2π/N, где *N* - длина окна.

Для повышения состоятельности оценки (1.18) выполняют её сглаживание. Имеется несколько методов сглаживания: Даньелла, Бартлетта, Уэлча [6,7].

Метод Даньелла основан на осреднении значений СПМ в пределах смежных спектральных частот.

В соответствии с методом Бартлетта состоятельность оценки СПМ повышают усреднением оценок СПМ коротких реализаций, полученных из

одной реализации длиной *N* отсчетов. Пусть дана реализация длиной *N* отсчетов. Она разбивается на *ns* неперекрывающихся сегментов, длиной *Ns=N/s* отсчетов. Для каждого сегмента по формуле (1.18) вычисляется выборочная оценка СПМ. Сглаженная оценка СПМ получается путем усреднения по всем *n*, сегментам

(1.30)



Если последовательность *х*[*n*] представляет нормальный стационарный эргодический процесс, то сглаженная оценка имеет дисперсию обратно пропорциональную числу сегментов *n*.

Спектральное разрешение оценки задается приближенным равенством

(1.31)



В методе Уэлча подход Бартлетта применяется к перекрывающимся сигментам исходной последовательности *х*[*n*], и каждый сегмент взвешивается с помощью оконной функции для уменьшения смещения оценок из-за эффекта «просачивания» энергии в боковые лепестки. Цель перекрытия сегментов - увеличить число усредняемых участков при фиксированной длине последовательности и тем самым повысить точность оценок СПМ. Метод Уэлча - один из самых распространенных периодограммных методов [6,7].

Обозначим через величину сдвига между сегментами, которая должна удовлетворять условию , где - максимальное имя корреляции анализируемого процесса. При выполнении этого условия получим *p*=*int*[(*N*-*Ns*)/+l]слабо коррелированных сегментов. Отсчеты каждого сегмента взвешиваются окном *w*[*n*]



(1.32)



Выборочное значение СПМ сегмента *р* оценивается по формуле

(1.33)



где

(1.34)



(1.35)



Сглаженная оценка периодограммы Уэлча вычисляется по формуле

(1.36)



Введение перекрытия сегментов в методе Уэлча позволяет уменьшить изменчивость оценки СПМ, Так же как и в методе Бартлетта, дисперсия оценки СПМ по методу Уэлча обратно пропорциональна числу сегментов, но благодаря большему числу сегментов, значение дисперсии будет меньше.

* 1. Основы цифровой фильтрации

Цифровой фильтр представляет собой систему с постоянными параметрами (инвариантную к сдвигу), работающую в дискретном времени. Напомним, что для таких систем сигнал на входе и выходе связан дискретной сверткой (1.5). Соответствующее соотношение между z-преобразованиями имеет вид

(1.37)



Прямое *z*-преобразование отклика на единичный импульс H(z) называется передаточной функцией системы. Преобразование Фурье отклика на единичный импульс называется частотной характеристикой. Обычно представляет собой комплексную функцию со, которую можно записать в виде



(1.38)



или через модуль и фазу

(1.39)



Инвариантная к сдвигу линейная система называется физически реализуемой, если *h*(*n*)=0 при *n*<0. Линейная система устойчива, если для любой ограниченной по уровню входной последовательности выходная последовательность также ограничена. Необходимым и достаточным условием устойчивости линейной системы с постоянными параметрами является

(1.40)



Это условие аналогично (1.10) и оказывается достаточным для существования .



Сигналы на входе и выходе линейных инвариантных к сдвигу систем, таких, например, как фильтры, связаны дискретной сверткой (1.5) и кроме того, разностным уравнением

(1.41)



Вычисляя *z*-преобразование от обеих частей, можно получить

(1.42)



Сравнивая (1.41) и (1.42), полезно отметить следующее. Если задано разностное уравнение вида (1.41), то H(z) можно получить непосредственной подстановкой коэффициентов при входном сигнале в числитель передаточной функции к соответствующим степеням z-1, а коэффициенты при выходном сигнале - в знаменатель к соответствующим степеням z-1.

Передаточная функция в общем случае является дробно рациональной. Таким образом, она определяется положением нулей и полюсов на z-плоскости. Это означает, что H(z) можно представить в виде

(1.43)



При рассмотрении *z*-преобразования отмечалось, что физически реализуемые системы имеют область сходимости вида |*z|*>*R*1. Если система, кроме того, еще и устойчива, то *R*1 должно быть меньше единицы, таким образом единичная окружность входит в область сходимости. Иначе говоря, для устойчивой системы все полюсы H(z) должны лежать внутри единичной окружности.

Достаточно определить два типа линейных систем с постоянными параметрами. Это системы с конечной импульсной характеристикой (КИХ) и системы с бесконечной импульсной характеристикой (БИХ). Эти два класса обладают отличными друг от друга свойствами, которые будут рассмотрены ниже.

Если все коэффициенты *аk*. в уравнении (1.22) равны нулю, то разностное уравнение принимает вид

(1.44)



Сравнивая (1.44) с (1.56), можно отметить, что

(1.45)



Системы с КИХ обладают рядом важных свойств. Передаточная функция H(z) таких систем представляет собой полином по степеням z-1 и, таким образом, не имеет ненулевых полюсов, а содержит только нули. Системы с КИХ могут обладать строго линейной фазо-частотной характеристикой (ФЧХ). Если *h*(*n*) удовлетворяет условию

(1.46)



то

(1.47)



где - действительная или чисто мнимая величина в зависимости от знака в (1.48).



Возможность получения строго линейной ФЧХ является очень важным обстоятельством применительно к речевым сигналам в тех случаях, когда требуется сохранить взаимное расположение элементов сигнала. Это свойство систем с КИХ существенно облегчает решение задачи их проектирования, поскольку все внимание можно уделять лишь аппроксимации амплитудно-частотной характеристики (АЧХ). За это достоинство фильтра с линейной ФЧХ, приходится расплачиваться необходимостью аппроксимации протяженной импульсной реакции в случае фильтров с крутыми АЧХ. Хорошо разработаны три метода проектирования КИХ-фильтров с линейными ФЧХ: взвешивания, частотной выборки и проектирования оптимальных фильтров с минимаксной ошибкой [1-3]. Второй и третий методы являются оптимизационными и используют итеративный (в отличие от замкнутой формы) подход для определения коэффициентов фильтра. Несмотря на простоту метода взвешивания, широкое применение нашли все три метода. Это обусловлено завершенностью глубоких исследований оптимальных КИХ-фильтров а, кроме того, наличием подробно описанных программ, позволяющих пользователю легко рассчитать любой фильтр [1,2,11].

При рассмотрении вопросов реализации цифровых фильтров полезно изображать их в виде схем. Разностное уравнение (1.25) изображено на рис. 1.4. Подобные схемы, называемые структурными, описывают в графической форме те операции, которые необходимо проделать над входной последовательностью для получения сигнала на выходе.

Для фильтра после подстановки (1.49) в(1.51) и выполнения интегрирования получается

(1.52)



Импульсная характеристика (1.32) определена при любых целых значениях к и является бесконечной, поэтому ограничивают значения *k*. Пусть |*k*|≤*К*, где *К*=(*N*-1)/2 для фильтров с нечётными значениями *N* и *К*=*N*/2 для фильтров с чётными значениями *N*. Тогда конечная импульсная характеристика, соответствующая (1.52), записывается в виде

(1.53)



Импульсная характеристика (1.53) является некаузальной, так как имеет ненулевые значения при *k*<0. Чтоб получить каузальную характеристику, необходимо задержать на *К* тактов, т.е.



(1.54)



Синтезированный таким образом фильтр будет иметь значительные пульсации вблизи частоты среза. Указанные пульсации представляют эффект Гиббса, проявляющийся вблизи точек разрыва заданной частотной характеристики. Эффект Гиббса обусловлен ограничением длительности бесконечной импульсной характеристики .



Конечная импульсная характеристика , получаемая из , может быть также представлена в виде



(1.55)



где *w*[*k*] - прямоугольное весовое окно длиной *N* отсчётов. Умножение заданной импульсной характеристики на прямоугольное окно приводит к искажению частотной характеристики синтезируемого фильтра. Частотная характеристика, соответствующая (1.55), будет равна свёртке заданной частотной характеристики прямоугольного окна. Таким образом, эффект Гиббса связан с пульсирующим поведением частотной характеристики прямоугольного весового окна.

Для уменьшения отрицательного влияния эффекта Гиббса применяют весовые окна, отличные от прямоугольного. Здесь имеет место аналогия со спектральным анализом, когда для уменьшения утечки энергии в боковые лепестки применяют оконные функции. С целью обеспечения линейности фильтра низких частот указанные оконные функции должны быть симметричными *w*[*k*]=*w*[*N*-*k*-1] и определены для значений *k*, лежащих диапазоне 0≤*k*≤*N*-1 [6].

1.3 Особенности акустической фонетики и её учёт при обработке речевых сигналов

1.3.1 Механизм речеобразования

Речь состоит из последовательности звуков. Звуки и переходы между ними служат символическим представлением информации. Порядок следования звуков (символов) определяется правилами языка. Изучение этих правил и их роли в общении между людьми составляет предмет лингвистики, анализ и классификация самих звуков речи — предмет фонетики. При обработке речевых сигналов с целью повышения их информативного содержания либо для выделения содержащейся в сигнале информации полезно располагать как можно большим количеством сведений о структуре сигнала, например, о способе кодирования информации в сигнале [1].

Голосовой тракт начинается с прохода между голосовыми связками,называемого голосовой щелью, и заканчивается у губ. Голосовой тракт, таким образом, состоит из гортани (от пищевода до рта) и рта, или ротовой полости. У взрослого мужчины общая длина голосового тракта составляет примерно 17 см. Площадь поперечного сечения голосового тракта, которая определяется положением языка, губ, челюстей и небной занавески, может изменяться от нуля (тракт полностью перекрыт) до примерно 29 см2. Носовая полость начинается у нёбной занавески и заканчивается ноздрями. При опущенной небной занавеске носовая полость акустически соединена с голосовым трактом и участвует в образовании носовых звуков речи. На рис. 1.4 показано подробное схематическое изображение речеобразующей системы. Для полноты в диаграмму включены и такие органы, как легкие, бронхи и трахея, расположенные ниже гортани. Совокупность этих органов и служит источником энергии для образования речи. Речь представляет собой акустическую волну, которая вначале излучается этой системой при выталкивании воздуха из легких и затем преобразуется в голосовом тракте. Основные особенности колебания легко объяснить на основе подробного анализа механизма образования речи. Звуки речи могут быть разделены на три четко выраженные группы по типу возбуждения. Вокализованные звуки образуются проталкиванием воздуха через голосовую щель, при котором периодически напрягаются и расслабляются голосовые связки и возникает квазипериодическая последовательность импульсов потока воздуха, возбуждающая голосовой, тракт.

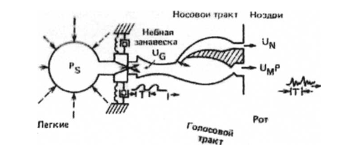


Рисунок 1.4 – Схема речеобразующих органов человека [1,13]

Фрикативные или невокализованные звуки генерируются при сужении голосового тракта в каком-либо месте (обычно в конце рта) и проталкивании воздуха через суженное место со скоростью, достаточно высокой для образования турбулентного воздушного потока. Таким образом, формируется источник широкополосного шума, возбуждающего голосовой тракт.

При произнесении взрывных звуков голосовой тракт полностью закрывается (обычно в начале голосового тракта). За этой смычкой возникает повышенное сжатие воздуха. Затем воздух внезапно высвобождается. Область малого уровня соответствует периоду полного закрытия голосового тракта. Голосовой тракт и носовая полость показаны на рис. 1.4 в виде труб с переменной по продольной оси площадью поперечного сечения. При прохождении звуковых волн через эти трубы их частотный спектр изменяется в соответствии с частотной избирательностью трубы. Этот эффект похож на резонансные явления, происходящие в трубах органов и духовых музыкальных инструментов. При описании речеобразования резонансные частоты трубы голосового тракта называют формантными частотами или просто формантами. Формантные частоты зависят от конфигурации и размеров голосового тракта: произвольная форма тракта может быть описана набором формантных частот. Различные звуки образуются путем изменения формы голосового тракта. Таким образом, спектральные свойства речевого сигнала изменяются во времени в соответствии с изменением формы голосового тракта.

Переменные во времени спектральные характеристики речевого сигнала с помощью звукового спектрографа могут быть высвечены в виде графика. Этот прибор позволяет получить двумерный график, называемый спектрограммой, на которой по вертикальной оси отложена частота, а по горизонтальной – время. Плотность зачернения графика пропорциональна энергии сигнала. Таким образом, резонансные частоты голосового тракта имеют вид затемненных областей на спектрограмме. Вокализованным областям сигнала соответствует появление четко выраженной периодичности временной зависимости, в то время как невокализованные интервалы выглядят почти сплошными [1].

1.3.2 Акустическая фонетика

Многие языки, в том числе и английский, можно описать набором отдельных звуков или фонем. Изучать фонему можно по-разному. Лингвисты, например, изучают отличительные характеристики фонем [1,2]. Четыре широких класса звуков образуют гласные, дифтонги, полугласные и согласные. Каждый из классов разбит на подклассы по способу и месту образования звука в голосовом тракте. Каждая фонема может быть отнесена к классу протяжных или кратковременных звуков. Протяжные звуки образуются при фиксированной (инвариантной ко времени) форме голосового тракта, который возбуждается соответствующим источником. К этому классу относятся гласные, фрикативные (вокализованные и невокализованные) носовые согласные. Остальные звуки (дифтонги, полугласные, аффрикаты и взрывные согласные) произносятся при изменяющейся форме голосового тракта. Они образуют класс кратковременных звуков.

Гласные. Гласные образуются при квазипериодическом возбуждении голосового тракта неизменной формы импульсами воздуха, возникающими вследствие колебания голосовых связок. Как будет показано ниже, зависимость площади поперечного сечения голосового тракта от координаты (расстояния) вдоль его продольной оси определяет резонансные частоты тракта (форманты) и характер произносимого звука. Эта зависимость называется функцией площади поперечного сечения. Функция площади поперечного сечения для каждой гласной зависит в первую очередь от положения языка; вместе с тем на характер звука оказывают влияние положения челюстей, губ и, в меньшей степени, небной занавески. Например, при произнесении звука |а|, голосовой тракт открыт в начале, а в его конце тело языка образует сужение. Наоборот, при произнесении звука |и|, язык образует сужение в начале голосового тракта и оставляет его открытым в конце. Таким образом, каждому гласному звуку может быть поставлена в соответствие форма голосового тракта (функция площади поперечного сечения), характерная для его произношения. Очевидно, что это соответствие неоднозначное, так как у разных дикторов голосовые тракты различны; Другим представлением гласного звука является его описание с помощью набора резонансных частот голосового тракта. Это описание также зависит от диктора. Петерсон и Барней [1] провели измерения формантных (резонансных) частот с помощью звукового спектрографа для гласных, произнесенных различными дикторами.

На спектрограммах четко выделяются различные резонансные области, характерные для каждой гласной. Акустические колебания, иллюстрируя периодичность вокализованных звуков, позволяют также путем анализа одного периода выявить грубые спектральные характеристики. Например, акустическое колебание звука |и| состоит из низкочастотного затухающего колебания, на которое накладывается относительно высокочастотная составляющая. Это соответствует низкой частоте первой форманты и высоким частотам второй и третьей формант. Два резонанса, расположенных на близких частотах, расширяют спектр колебания. Наоборот, в акустическом колебании гласной |у| энергия высокочастотных составляющих относительно мала, что соответствует низким частотам первой и второй формант. Подобный анализ может быть проведен для всех гласных.

Дифтонги. Дифтонгом называется участок речи, соответствующий одному слогу, который начинается с одной гласной и затем постепенно переходит в другую. На основе этого определения можно выделить следующие дифтонги: |эй|, |оу|, |ау|, |ой|, |ай|.

Дифтонги образуются путём плавного изменения формы голосового тракта.

Полугласные. Группу звуков, содержащих |в|, |й| описать довольно трудно. Эти звуки называются полугласными, гак как по своим свойствам они напоминают гласные звуки. Обычно их характеризуют плавным изменением функции площади поперечного сечения голосового тракта между смежными фонемами. Таким образом, акустические характеристики этих звуков существенно зависят от произносимого текста. Удобно рассматривать эти звуки как переходные, сходные с гласными. Их структура близка к структуре гласных и дифтонгов.

Носовые звуки (сонорные). Носовые согласные |м|, |н| и |л| образуются при голосовом возбуждении. В полости рта при этом возникает полная смычка. Небная занавеска опущена, поэтому поток воздуха проходит через носовую полость и излучается через ноздри. Полость рта, которая вначале закрыта, акустически соединена с гортанью. Таким образом, рот служит резонансной полостью, в которой задерживается часть энергии при определенных частотах воздушного потока. Эти резонансные частоты соответствуют антирезонансам или нулям передаточной функции тракта речеобразования [2]. Более того, для носовых согласных и гласных (т. е. гласных, расположенных перед носовыми согласными) характерны менее выраженные резонансы, чем для гласных. Расширение резонансных областей происходит из-за того, что внутренняя поверхность носового тракта напрягается и при этом носовая полость имеет большое отношение площади поверхности к площади поперечного сечения. Вследствие этого потери за счёт, теплопроводности и вязкости оказываются большими, чем обычно.

Три носовых согласных различаются местом расположения полной смычки. При произнесении звука |м| смычка образуется между губами, |н| - у внутренней стороны зубов.

Глухие фрикативные звуки (шипящие, свистящие). Глухие фрикативные звуки |ф|, |с|, |ш|, образуются путем возбуждения голосового тракта турбулентным воздушным потоком, возникающим в области смычки голосового тракта. Расположение смычки характеризует тип фрикативного звука. При произнесении звука |ф| смычка возникает около губ, |с| - в середине полости рта и |ш| - в конце полости рта. Таким образом, система образования глухих фрикативных звуков содержит источник шума, расположенный в области смычки, которая разделяет голосовой тракт на две полости. Звуковая волна излучается через губы т. е. через переднюю полость. Другая полость служит, как и в случае произнесения носовых звуков, для задерживания акустического потока, и таким образом в речеобразующем тракте возникают антирезонансы [1].

Звонкие фрикативные звуки. Звонкие фрикативные звуки |в|, |з| и |ж| являются прототипами глухих звуков |ф|, |с|, |п| и |ш| соответственно. Место расположения смычки для этих пар звуков совпадает. Однако звонкие фрикативные звуки отличаются от своих аналогов тем, что при их образовании участвуют два источника возбуждения. При образовании звонких звуков голосовые связки колеблются и, таким образом, один источник возбуждения находится в гортани. Однако, так как в голосовом тракте образуется смычка, поток воздуха в этой области становится турбулентным. Можно ожидать, что в спектре звонких фрикативных звуков будут две различные составляющие. Сходство структуры звонкого |в| и глухого |ф| также легко установить путем сравнения соответствующих спектрограмм. Аналогично можно сравнить и спектрограммы звуков |ш| и |ж|.

Звонкие взрывные согласные. Звонкие взрывные согласные |б|, |д| и |г| являются переходными непротяжными звуками. При их образовании голосовой тракт смыкается в какой-нибудь области полости рта. За смычкой воздух сжимается и затем внезапно высвобождается. При произнесении звука |б| смычка образуется между губами, |д| - с внутренней стороны зубов, |г| -вблизи небной занавески. В течение периода, когда голосовой тракт полностью закрыт, звуковые волны практически не излучаются через губы. Однако слабые низкочастотные колебания излучаются стенками горла (эту область иногда называют голосовымг затвором — «voice bar»). Колебания возникают из-за того, что голосовые связки могут вибрировать даже тогда, когда голосовой тракт перекрыт.

Так как структура взрывных звуков изменчива, их свойства существенно зависят от последующего гласного [1]. В этой связи характер временных колебаний несёт мало сведений о свойствах этих согласных.

Глухие взрывные согласные. Глухие взрывные согласные |п|, |т| и |к| подобны своим звонким прототипам |б|, |д| и |г|, но имеют одно важное отличие. В течение периода полного смыкания голосового тракта голосовые связки не колеблются. После этого периода, когда воздух за смычкой высвобождается, в течение короткого промежутка времени потери на трение возрастают из-за внезапной турбулентности потока воздуха. Далее следует период придыхания (шумовой воздушный поток из голосовой щели возбуждает голосовой тракт). После этого возникает голосовое возбуждение.

Аффрикаты и звук |х|. Остальными согласными произношения являются аффрикаты |ч| и |дж| и фонема |х|. Глухая аффриката |ч| является динамичным звуком, который можно представить как сочетание взрывного |т| и фрикативного согласного |щ|. Звонкий звук |дж| можно представить как сочетание взрывного |д| и фрикативного звука |ж|. Наконец, фонема |х| образуется путем возбуждения голосового тракта турбулентным воздушным потоком, т. е. без участия голосовых связок, но при возникновении шумового потока в голосовой щели. Структура звука |х| не зависит от следующей за ним гласной. Поэтому голосовой тракт может перестраиваться для произнесения следующей гласной в процессе произнесения звука |х| [1].

1.3.3 Распространение звуков

Понятие звука почти совпадает с понятием колебаний. Звуковые волны возникают за счет колебаний. Они распространяются в воздухе или другой среде с помощью колебаний частиц этой среды. Следовательно, образование и распространение звуков в голосовом тракте подчиняется законам физики. В частности, основные законы сохранения массы, сохранения энергии, сохранения количества движения вместе с законами термодинамики и механики жидкостей применимы к сжимаемому воздушному потоку с низкой вязкостью, который является средой распространения звуков речи. Используя эти основные физические законы, можно составить систему дифференциальных уравнений в частных производных, описывающую движение воздуха в речеобразующей системе [1,2]. Составление и решение этих уравнений весьма затруднительны даже для простых предположений относительно формы голосового тракта и потерь энергии в речеобразующей системе. Полная акустическая теория должна учитывать следующие факторы:

* изменение во времени формы голосового тракта;
* потеря энергии на стенках голосового тракта за счет вязкого трения и теплопроводности;
* мягкость стенок голосового тракта;
* излучение звуковых волн через губы;
* влияние носовой полости;
* возбуждение голосового тракта.

Голосовой тракт можно представить в виде неоднородной трубы с переменной во времени площадью поперечного сечения. Для колебаний, длина волны которых превышает размеры голосового тракта (это обычно имеет место на частотах ниже 4000 Гц), можно допустить, что вдоль продольной оси трубы распространяется плоская волна. Дальнейшее упрощение состоит в предположении отсутствия потерь на вязкость и теплопроводность как внутри воздушного потока, так и на стенках трубы. На основе законов сохранения массы, количества движения и энергии с учетом перечисленных допущений Портнов показал, что звуковые волны, в трубе удовлетворяют следующим уравнениям

(1.56а)



(1.56б)



где *р*=*р*(*х*,*t*) – звуковое давление как функция *х* и *t*; *u*=*u*(*х*,*t*) – скорость воздушного потока (volume velocity) как функция *х* и *t*, *ρ* - плотность воздуха в трубе; *с* - скорость распространения звука; *A*=*A*(*x,t*) - «функция площади», т. е. площадь поперечного сечения в направлении, перпендикулярном продольной оси трубы, как функция расстояния вдоль этой оси и времени. Сходная система уравнений была получена Сондхи,

Замкнутое решение уравнений (1.36) получить невозможно даже для простых форм трубы. Однако могут быть получены численные решения. Полное решение дифференциальных уравнений предполагает заданными давление и скорость потока для значений *х* и *t* в области голосовой щели и около губ, т. е. для получения решения должны быть заданы граничные условия у обоих концов трубы. Со стороны губ граничные условия должны отображать эффект излучения, а со стороны голосовой щели – характер возбуждения.

Кроме граничных условий необходимо задать функцию площади *A*(*x,t*). Для протяжных звуков можно предположить, что *A*(*x,t*) не изменяется во времени. Однако это предположение неверно для непротяжных звуков. Подробные измерения *A*(*x,t*) весьма затруднительны и могут быть выполнены только для протяжных звуков. Одним из методов проведения таких измерений является рентгеновская киносъемка. Фант и Перкелл провели несколько таких экспериментов. Однако подобные измерения могут быть выполнены лишь в ограниченном объеме. Другим методом является вычисление формы голосового тракта по акустическим измерениям. Описав подобный метод, предполагающий возбуждение голосового тракта внешним источником. Оба метода являются полезными для получения сведений о динамике речеобразования. Тем не менее, они не могут быть применены для получения описания речевых сигналов, например, в задачах связи. В работе Атала описаны результаты прямого измерения *A*(*x,t*) по сигналу речи, произнесенной в нормальных условиях.

Точное решение уравнений (1.36) является весьма сложным, даже если значение *A*(*x,t*) точно известно. Вместе с тем для решения поставленной задачи нет необходимости в точном и общем решениях этих уравнении [1,2].

1.4 Обработка речевого сигнала во временной области

В основе большинства методов обработки речи лежит предположение 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. Процедура повторяется так часто, как это требуется. Сегменты, которые иногда называют интервалами, (кадрами) анализа обычно пересекаются. Результатом обработки на каждом интервале является число или совокупность чисел. Следовательно, подобная обработка приводит к новой, зависящей от времени последовательности, которая.может служить характеристикой речевого сигнала.

Большинство методов кратковременного анализа, в том числе и кратковременный Фурье-анализ, могут быть описаны выражением

(1.57)



Речевой сигнал (возможно, после ограничения частотного диапазона в линейном фильтре) подвергается преобразованию Т[·], линейному или нелинейному, которое может зависеть от некоторого управляющего параметра или их совокупности. Результирующая последовательность умножается затем на последовательность значений временного окна (весовой функции), расположенную во времени в соответствии с индексом п. Результаты затем суммируются по всем ненулевым значениям. Обычно, хотя и не всегда, последовательность значений временного окна имеет конечную протяженность. Значение *Qn* представляет собой, таким образом, «взвешенное» среднее значение последовательности *Т* [*х* (*m*)].

Простым примером, иллюстрирующим изложенное, может служить измерение кратковременной энергии сигнала. Полная энергия сигнала в дискретном времени определяется как

(1.58)



Вычисление этой величины не имеет особого смысла при обработке речевых сигналов, поскольку она не содержит информации о свойствах сигнала, изменяющихся во времени. Кратковременная энергия определяется выражением

(1.59)



Таким образом, кратковременная энергия в момент *n* есть просто сумма квадратов *N* отсчетов от *n*-*N*-1 до *n*. Из (1.37) видно, что в (1.39) Т [•] есть просто операция возведения в квадрат, а

(1.60)



Вычисление кратковременной энергии, иллюстрирует рис. 1.6 Окно «скользит» вдоль последовательности квадратов значений сигнала, в общем случае вдоль последовательности *Т*[*х*(*m*)], ограничивая длительность интервала, используемого в вычислениях.

Как отмечалось выше, амплитуда речевого сигнала существенно изменяется во времени.

В частности, амплитуда невокализованных сегментов речевого сигнала значительно меньше амплитуды вокализованных сегментов. Подобные изменения амплитуды хорошо описываются с помощью функции кратковременной энергии сигнала. В общем случае определить функцию энергий можно как

(1.61)



Это выражение может быть переписано в виде

(1.62)



где

(1.63)



Сигнал *х*2(*n*) в этом случае фильтруется с помощью линейной системы с импульсной характеристикой *h*(*n*).

Выбор импульсной характеристики *h*(*n*) или окна составляет основу описания сигнала с помощью функции энергии. Чтобы понять, как влияет выбор окна на функцию кратковременной энергии сигнала, предположим, что *h*(*n*) в (1.35) является достаточно длительной и имеет постоянную амплитуду; значение *Еn* 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, иначе говоря, полосу следует выбрать так, чтобы функция энергии отражала изменения амплитуды речевого сигнала.

Описанная ситуация выражает противоречие, которое нередко возникает при изучении кратковременных характеристик речевых сигналов. Суть его состоит в том, что 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 функций энергии.

Влияние вида окна на вычисление изменяющейся во времени энергии сигнала можно проиллюстрировать на примере использования двух наиболее распространенных окон: прямоугольного и окна Хэмминга.

Прямоугольное окно, как это видно из (1.39), соответствует случаю, когда всем отсчетам на интервале от (*n*-*N*-1) до п приписывается одинаковый вес. Частотная характеристика прямоугольного окна равна

(1.64)



Для окна с шириной 51 отсчет (*N*=51) логарифм амплитудно-частотной характеристики представлен на рис. 1.6а.

Первое нулевое значение амплитудно-частотной характеристики (1.64) соответствует частоте

(1.65)



где частота дискретизации. Это номинальная частота среза фильтра нижних частот, соответствующего прямоугольному окну.



Амплитудно-частотная характеристика окна Хемминга при *N*=51 показана на рис. 1.6б. Полоса пропускания фильтра с окном Хемминга при одинаковой ширине примерно вдвое превосходит полосу фильтра с прямоугольным окном. Очевидно также, что окно Хемминга обеспечивает большее затухание вне полосы пропускания по сравнению с прямоугольным окном. Затухание, вносимое вне полосы, несущественно зависит от ширины каждого из окон. Это означает, что увеличение ширины приведет просто к сужению полосы. Если *N* мало (порядка периода основного тона или менее), то *Еn* будет изменяться очень быстро, в соответствии с тонкой структурой речевого колебания. Если *N* велико (порядка нескольких периодов основного тона), то *Еn* будет изменяться медленно и не будет адекватно описывать изменяющиеся особенности речевого сигнала. Это, к сожалению, означает, что не существует единственного значения *М*, которое в полной мере удовлетворяло бы перечисленным требованиям, так как период основного тона изменяется от 10 отсчетов (при частоте дискретизации 10 кГц) для высоких женских и детских голосов до 250 отсчетов для очень низких мужских голосов. На практике *N* выбирают равным 100—200 отсчетов при частоте дискретизации 10 кГц (т. е. длительность порядка 10-20 мс).

Основное назначение *Еn* состоит в том, что эта величина позволяет отличить вокализованные речевые сегменты от невокализованных. Значения *Еn* для невокализованных сегментов значительно меньше, чем для вокализованных. Функция кратковременной энергии может быть использована для приближенного определения момента перехода от вокализованного сегмента к невокализованному и наоборот, а в случае высококачественного речевого сигнала (с большим отношением сигнала к шуму) функцию энергии можно использовать и для отделения речи от пауз.

Одним из недостатков функции кратковременной энергии, определяемой выражением (1.35), является ее чувствительность к большим уровням сигнала (поскольку в (1.35) каждый отсчет возводится в квадрат). Вследствие этого значительно искажается соотношение между значениями последовательности *х*(*n*). Простым способом устранения этого недостатка является переход к определению функции среднего значения в виде

(1.66)



где вместо суммы квадратов вычисляется взвешенная сумма абсолютных значений. Исключение операции возведения в квадрат упрощает арифметические вычисления.

При вычислении среднего значения по (1.46) динамический диапазон (отношение максимального значения к минимальному) определяется примерно как квадратный корень из динамического диапазона при обычном вычислении энергии. В данном случае различия в уровнях между вокализованной и невокализованной речью выражены не столь ярко, как при использовании функций энергии.

Поскольку полоса частот при определении как функции энергии, так и среднего значения приближенно совпадает с полосой пропускания используемого фильтра нижних частот, то нет необходимости дискретизировать эти функции столь же часто, как исходный речевой сигнал. Например, для окна длительностью 20 мс достаточна частота дискретизации около 100 Гц. Это означает, что значительная часть информации теряется при использовании подобных кратковременных представлений. Очевидно также, что информация, относящаяся к динамике амплитуд речевого сигнала, сохраняется в весьма удобной форме [1,2].

При обработке сигналов в дискретном времени считают, что если два последовательных отсчёта имеют различные знаки, то произошёл переход через ноль. Частота появления нолей в сигнале может служить простейшей характеристикой его спектральных свойств. Это наиболее справедливо для узкополосных сигналов.

Среднее число нулевых переходов можно принять в качестве подходящей оценки частоты синусоидального колебания.

Речевой сигнал является широкополосным и, следовательно, интерпретация среднего числа переходов через нуль менее очевидна. Однако можно получить грубые оценки спектральных свойств сигнала, основанные на использовании функции среднего числа переходов через нуль для речевого сигнала; рассмотрим способ вычисления этой величины. Функция среднего числа переходов через нуль имеет те же общие свойства, что и функции энергии и среднего значения. Все, что в действительности требуется, это проверить пары отсчетов с целью определения нулевых пересечений, а затем вычислить среднее по всем *N* последовательным отсчетам (деление на *N*, конечно, необязательно). Как и ранее, может быть вычислено взвешенное среднее и при использовании симметричных окон конечной длительности задержка может быть скомпенсирована точно. Могут быть получены и рекуррентные уравнения.

Рассмотрим теперь применение функции среднего числа переходов через нуль для обработки речевых сигналов. Модель речеобразоаания предполагает, что энергия вокализованных сегментов речевого сигнала концентрируется на частотах ниже 3 кГц, что обусловлено убывающим спектром сигнала возбуждения, тогда как, для невокализованных сегментов большая часть энергии лежит в области высоких частот. Поскольку высокие частоты приводят к большому числу переходов через нуль, а низкие – к малому, то существует жесткая связь между числом нулевых пересечений и распределением энергии по частотам. Разумно предположить, что большому числу нулевых пересечений соответствуют невокализованные сегменты, а малому числу — вокализованные сегменты речи. Это, однако, очень расплывчатое утверждение, поскольку мы не определили, что означает «много» или «мало», и количественно определить эти понятия в действительности трудно. Гауссовская кривая хорошо согласуется с приведенными гистограммами. Среднее число пересечений составляет 49 для вокализованных и 14 для невокализованных сегментов длительностью 10 мс.

Поскольку оба распределения перекрываются, нельзя вынести однозначное решение о принадлежности сегмента к вокализованным или невокализованным отрезкам только по среднему числу переходов через нуль. Тем не менее, подобное представление весьма полезно при осуществлении такой классификации.

Так же, как и в случае функций энергии и среднего, функцию среднего числа переходов через нуль можно дискретизировать с очень низкой частотой. Хотя среднее число переходов через нуль изменяется значительно, вокализованные и невокализованные сегменты просматриваются очень четко.

При использовании описания сигнала средним числом переходов через нуль следует иметь в виду ряд практических соображений. Хотя в основу алгоритма вычисления нулевых переходов положено сравнение знаков соседних отсчетов, тем не менее, при дискретизации сигнала следует предпринимать специальные меры.

Очевидно, что число нулевых переходов зависит от уровня шума при аналого-цифровом преобразовании, интенсивности фона переменного тока и других шумов, которые могут присутствовать в цифровой системе. Таким образом, с целью уменьшения влияния этих факторов следует проявлять особую осторожность при аналоговой обработке сигнала, предшествующей дискретизации. Например, часто оказывается более целесообразным использовать полосовой фильтр вместо фильтра нижних частот для уменьшения эффекта наложения при аналого-цифровом преобразовании и устранения фона переменного тока из сигнала. Кроме того, при измерении числа переходов через нуль следует учитывать соотношение между периодом дискретизации и интервалом усреднения *N*. Период дискретизации определяет точность выделения нулевых пересечений по времени (и по частоте), т. е. чтобы добиться высокой точности, нужна большая частота дискретизации. Вместе с тем от каждого отсчета требуется информация объемом лишь 1 бит (информация только о знаке сигнала).

Вследствие практической ограниченности этого метода было предложено множество сходных представлений сигнала. В каждом из них содержатся дополнительные особенности, направленные на снижение чувствительности оценок к шуму, но все они имеют и свои собственные ограничения. Наиболее заметным среди них является представление сигнала, исследованное Бейкером. Представление основано на интервалах времени между положительными переходами через нуль (снизу вверх). Бейкер применил это описание для фонетической классификации звуков речи.

Другое применение анализа переходов через нуль состоит в получении промежуточного представления речевого сигнала в частотной области. Метод включает фильтрацию речевого сигнала в нескольких смежных частотных диапазонах. Затем по сигналам на выходе фильтров измеряют кратковременную энергию и среднее число переходов через нуль. Совместное использование этих характеристик дает грубое описание спектральных свойств сигнала. Этот подход, предложенный Рэдди и исследованный Вайсенсом и Эрманом, положен в основу систем распознавания речи.

Задача определения моментов начала и окончания фразы при наличии шума является одной из важных задач в области обработки речи. В частности, при автоматическом распознавании слов важно точно определить моменты начала и окончания слова. Методы обнаружения моментов начала и окончания фразы можно использовать для уменьшения числа арифметических операций, если обрабатывать только те сегменты, в которых имеется речевой сигнал, например, в системах, работающих не в реальном масштабе времени.

Проблема отделения речи от окружающего шума очень сложна, за исключением случаев очень большого отношения сигнал/шум, т. е. в случае высококачественных записей, выполненных в заглушённой камере или звуконепроницаемой комнате. В этих случаях энергия даже наиболее слабых звуков речи (фрикативных согласных) превышает энергию шума и, таким образом, достаточно лишь измерить энергию сигнала. Но подобные условия записи, как правило, не встречаются в реальных ситуациях.

2. Реализация систем распознавания речи

2.1 Гомоморфная обработка речи

Речевой сигнал на коротких интервалах можно рассматривать как отклик системы с медленно меняющимися параметрами на периодическое или шумовое возбуждение. Это означает, что во временной области дискретный сигнал *у*(*n*) представляется результатом свертки функции возбуждения *х*(*n*) с импульсной реакцией голосового тракта п(п). Гомоморфная обработка речи сводится к решению обратной задачи — имея речевой сигнал *у*(*n*)=*х*(*n*)\**h*(n), можно получить параметры сигналов, участвующих в свертке. Эта задача называется иногда задачей обратной свертки или развертки.

Смысл гомоморфной системы анализа становится более понятным, если учесть, что в частотной области речевой сигнал представляется произведением спектра сигнала возбуждения и передаточной функции частотной характеристики голосового тракта, учитывающего спектральные свойства излучателя. Это означает, что в спектре речевого сигнала содержится информация о спектре сигнала возбуждения и передаточной функции голосового тракта. Гомоморфная обработка сигнала — это способ извлечь информацию об основном тоне и формантных частотах на основании преобразований сигнала, которые будут описаны далее.

Если произвести кратковременное дискретное преобразование Фурье (т.е. получить динамический спектр речевого сигнала), а затем прологарифмировать спектральные составляющие динамического спектра, то каждый спектральный отсчет можно рассматривать как сумму логарифмов спектра сигнала возбуждения и частотной характеристики речевого тракта (по свойству логарифмической функции логарифм произведения равен сумме логарифмов сомножителей). Обратное дискретное преобразование Фурье прологарифмированного спектра позволяет вновь перейти к анализу сигнала во временной области. Сигнал, полученный в результате обратного дискретного преобразования Фурье прологарифмированного спектра, называется кепстром входного сигнала, равного сумме кепстров сигналов возбуждения и составляющих, обусловленных особенностями речеобразующего тракта. В результате подобных преобразований дискретный речевой сигнал, представляющий собой свертку сигнала возбуждения и импульсного отклика фильтра, моделирующего голосовой тракт, приближенно преобразуется в сложение кепстров.

Логарифм кратковременного спектра вокализованных звуков содержит медленно меняющуюся составляющую, обусловленную передаточными свойствами голосового тракта, и быстро меняющуюся периодическую составляющую, которая вызывается периодическим сигналом возбуждения (рис. 2.1а). Для невокализованной речи прологарифмированный спектр носит характер, показанный на рис.2.1б. Спектр содержит случайную составляющую с быстрыми изменениями.

Кепстры отрезков вокализованной и невокализованной речи (рис. 2.2) показывают, что медленно меняющаяся часть прологарифмированных значений кратковременного спектра представлена составляющими кепстра в области малых времен. Быстро меняющаяся периодическая составляющая прологарифмированного спектра, соответствующая частоте основного тона, в кепстре вокализованной речи проявляется в виде резкого пика, расположенного от начала координат на расстоянии, равном периоду основного тона. Кепстр невокализованной речи (рис. 2.2б) таких пиков не имеет.

Если кепстр перемножить на подходящую функцию окна, например прямоугольное окно, пропускающее только начальные участки кепстра (которые соответствуют области малых времен и отражают относительно медленно меняющиеся параметры голосового тракта), а затем вычислить дискретное преобразование Фурье результирующего взвешенного кепстра, то получим сглаженный спектр сигнала.

Он отражает резонансные свойства тракта, позволяя оценивать частоты и полосы формант. Наличие или отсутствие ярко выраженного пика в области, соответствующей диапазону изменений периода основного тона, указывает на характер возбуждения, а местоположение пика является хорошим индикатором периода основного тона (рис. 2.2).

Гомоморфные относительно свертки системы удовлетворяют обобщенному принципу суперпозиции. Принцип суперпозиции, если его записать для обычных линейных систем, имеет вид

(2.1a)



(2.1б)



где *L* – линейный оператор. Принцип суперпозиции устанавливает, что если сигнал на входе является линейной комбинацией элементарных сигналов, то и сигнал на выходе будет представлен в виде линейной комбинации соответствующих сигналов.

Прямым следствием принципа суперпозиции является тот факт, что сигнал на выходе линейной системы может быть представлен в виде дискретной свертки

(2.2)



Символ « \* » здесь и далее означает свертку в дискретном времени. По аналогии с принципом суперпозиции для обычных линейных систем определим класс систем, удовлетворяющих обобщенному принципу суперпозиции, в котором сложение заменяется сверткой (легко показать, что свертка обладает такими же алгебраическими свойствами, как и сложение [1]), т. е.

(2.3)



В общем случае возможно сформулировать и уравнение, аналогичное (2.16), в котором выражено свойство скалярного умножения [2]. Системы, обладающие свойством (2.3), названы гомоморфными относительно свертки системами. Эта терминология объясняется тем [3], что данные преобразования оказываются гомоморфными преобразованиями линейного векторного пространства. При изображении таких систем операцию свертки представляют в явном виде на входе и выходе системы. Гомоморфный фильтр является гомоморфной системой, обладающей тем свойством, что одна компонента (выделяемая) проходит через эту систему без изменений, а другая – устраняется. В соотношении (2.3), например, если *x*1(*n*) - нежелательная компонента, то необходимо потребовать, чтобы выход, соответствующий *x*1(*n*), представлял собой единичный отсчет, в то время как выход, соответствующий *х*2(*n*), близко совпадал бы с *х*2(*n*). Это полностью аналогично ситуации в линейных системах, где ставится задача выделения сигнала из смеси его с аддитивным шумом.

Важным аспектом теории гомоморфных систем является то, что любая из них может быть представлена в виде каскадного соединения трех гомоморфных систем. Первый блок преобразует компоненты на входе, представленные в виде свертки, в аддитивную сумму на выходе. Второй блок -обычная линейная система, удовлетворяющая принципам суперпозиции в соответствии с (2.1). Третий блок является обратным первому, т. е. преобразует сигналы, представленные в виде суммы, в сигналы, представленные в виде свертки. Важность такого канонического представления заключается в том, что разработка гомоморфной системы сводится к разработке линейной системы. Блок\*[], называемый характеристическим блоком гомоморфной относительно свертки системы, фиксирован при каноническом представлении. Очевидно, что обратное преобразование также фиксировано. Характеристическая система для гомоморфной обратной свертки подчиняется обобщенному принципу суперпозиции, в котором операция на входе – свертка, а на выходе – обычное сложение. Свойства характеристической системы определяются выражением

(2.4)



Аналогично обратная характеристическая система удовлетворяет соотношению

(2.5)



Математическое описание характеристической системы определяется требованиями к выходному сигналу. Если на входе имеется сигнал свертки, то

(2.6)



и *z*-преобразование входного сигнала имеет вид

(2.7)



Из (2.4) очевидно, что *z*-преобразование сигнала на выходе системы должно представлять собой сумму *z*-преобразований компонент. Таким образом, в частотной области характеристическая система для свертки должна обладать следующим свойством: если на входе имеется произведение компонент, то на выходе должна возникнуть их сумма.

С учетом возможности вычисления комплексного логарифма, обратное преобразование комплексного логарифма преобразования Фурье входного сигнала, являющееся выходом характеристической системы для свертки, имеет вид

(2.8)



Выход характеристической системы назван «комплексным кепстром» Термин «кепстр» используется для величины

(2.9)



Все системы этого класса отличаются только линейной частью. Выбор линейной системы определяется свойствами входного сигнала.

Следовательно, для правильного построения линейной системы необходимо прежде всего определить вид и структуру сигнала на выходе характеристической системы, т. е. рассмотреть свойства комплексного кепстра для типичных входных сигналов.

Для определения свойств комплексного кепстра достаточно рассмотреть случай рационального z-преобразования. Наиболее общая форма преобразования имеет вид

(2.10)



где модули величин *ак*, *bk*, *ck* и *dk* меньше единицы. Таким образом, сомножители (1-*akz-1*) и (1-*ckz-1*) соответствуют нолям и полюсам внутри единичной окружности, a (1-*bkz*) и (1-*dkz*) - нолям и полюсам вне единичной окружности. Параметр *zr* означает соответствующую задержку во временной области. Комплексный логарифм *X*(*z*) имеет вид

. (2.11)



Когда (7.13) вычисляется на единичной окружности, легко видеть, что член вносит вклад только в минимальную часть комплексного логарифма. Поскольку этот член несет информацию только о взаимном расположении во временной области, то при вычислении комплексного кепстра он обычно опускается [2]. Таким образом, при обсуждении свойств комплексного кепстра далее этот член не рассматривается. Используя то обстоятельство, что логарифм можно разложить в степенной ряд, относительно несложно показать, что комплексый кепстр имеет вид



(2.12)



Уравнения (2.12) позволяют выявить ряд важных свойств комплексного кепстра. Прежде всего, комплексный кепстр в общем случае отличен от ноля и бесконечен как для положительных, так и для отрицательных значений *n*, даже если *х*(*n*) удовлетворяет принципу причинности, устойчив и имеет конечную протяженность. Далее видно, что комплексный кепстр является затухающей последовательностью, ограниченной сверху

(2.13)



где *α* - максимальное абсолютное значение величин *а,k* *bk*, *сk* и *dk*, *β* -постоянный сомножитель.

Если *Х*(*z*) не содержит нулей и полюсов вне единичной окружности (т.е. *bk* = *dk*=0),то

(2.14)



Такие сигналы называются минимально-фазовыми [1]. Общий результат для последовательности (2.14) состоит в том, что такая последовательность полностью определяется действительной частью преобразования Фурье. Таким образом, для минимально-фазовых систем комплексный кепстр определяется лишь логарифмом модуля преобразования Фурье. Это можно легко показать, если вспомнить, что действительная часть преобразования Фурье представляет собой преобразование Фурье от четной части последовательности, т. е. если – преобразование Фурье кепстра, то



(2.15)



Используя (2.14) и (2.15) легко показать, что

(2.16)



Таким образом, для минимально-фазовых последовательностей комплексный кепстр можно получить путем вычисления кепстра и последующего использования (2.16). Другой важный результат для минимально-фазовых систем заключается в: том, что комплексный, кепстр можно вычислить рекуррентно по входному сигналу [1, 2, 5]. Рекуррентная формула имеет вид

(2.17)



Аналогичные результаты можно получить и тогда, когда Х(г) не содержит полюсов и нулей, лежащих внутри единичной окружности. Такие сигналы называют максимально-фазовыми. Для этого случая, как это видно из (2.12),

(2.18)



Совместное использование (2.14) и (2.15) даёт

(2.19)



Как и в случае минимально-фазовых последовательностей, здесь также можно получить рекуррентное соотношение для кепстра

(2.20)



2.2 Кодирование речи на основе линейного предсказания

Линейное предсказание является одним из наиболее эффективных методов анализа речевого сигнала. Этот метод - доминирующий при оценке таких основных параметров речевого сигнала как период основного тона, форманты, спектр, функция площади речевого тракта, а также при сокращённом представлении речи с целью её низкоскоростной передачи и экономного хранения. Важность метода обусловлена высокой точностью получаемых оценок и относительной простотой вычислений.

Линейное предсказание - это метод анализа, основанный на цифровой фильтрации оцифрованной речи, при которой текущий отсчет сигнала может быть «предсказан» (например, при автоматическом синтезе речи) линейной комбинацией прошлых значений выходной последовательности и настоящих, а также прошлых значений входной последовательности. Понятие «линейная комбинация» означает сумму произведений известных дискретных отсчетов сигнала (входных и выходных), умноженных на соответствующие коэффициенты линейного предсказания для предсказания (определения) неизвестного выходного отсчета. При линейном предсказании основная задача анализа речи – найти коэффициенты этой линейной комбинации, которые дают минимальную ошибку предсказания на участке анализа сигнала.

Модель сигнала, наиболее часто используемая при линейном предсказании, сводится к получению неизвестного отсчета *х*(*n*) без учета предыдущих входных воздействий на выходе некоторой системы

(2.21)



(2.22)



где *р* – число коэффициентов, используемых в модели; *k* – коэффициенты линейного предсказания; *G* – коэффициент усиления, определяющий вклад в линейную комбинацию входного отсчета; *u*(*n*) – текущий входной отсчет.

Задача анализа оцифрованной речи сводится к определению коэффициентов *k* и *G* этой модели. Метод определения величин, используемых при расчетах, называется методом наименьших квадратов. Чтобы понять его суть, пойдем на некоторые упрощения в представлении текущего выходного отсчета. Будем считать, что входное воздействие на вход системы, моделирующей формирование речевых сигналов, ненаблюдаемо, что справедливо для ряда прикладных задач. Тогда на интервале анализа текущие отсчеты речевого сигнала приближенно опишутся линейной комбинацией предыдущих значений.

*х*(*n*) = *а*1*х*(*n* -1) + *а*2*х*(*n* - 2) +... + *аkх*(*n* - *k*) +... + *арх*(*n* -*р*) = (2.23)



где *х*(*n*-1),...,*х*(*n*-*р*) - предыдущие значения речевого сигнала. Получаемая при этом ошибка предсказания *εр* называется иногда остатком предсказания и равняется

(2.24)



Коэффициенты линейного предсказания *а* вычисляются из условия минимума среднеквадратичного значения ошибки на интервале анализа. На этом интервале полная среднеквадратичная ошибка складывается для каждого отсчета сигнала, представленного линейной комбинацией *р* предыдущих значений сигнала

(2.25)



Здесь *n* – номер предыдущего отсчета сигнала на анализируемом интервале; *k* – номер предыдущего отсчета сигнала при построении линейной комбинации, представляющей текущий отсчет.

(2.26)



Коэффициенты линейного предсказания, минимизирующие полную ошибку предсказания *Е*, находятся после того, как выражение для полной ошибки продифференцировать по всем коэффициентам (полная ошибка предсказания может рассматриваться как функция параметров *аk*) и приравнять нулю все частные производные.

Частными производными называются производные сложной функции по одной из переменных с учетом того, что остальные переменные при таком дифференцировании считаются константами.

Результатом дифференцирования по *а*, является система из линейных уравнений с неизвестными коэффициентами линейного предсказания, минимизирующими ошибку линейного предсказания на отрезке анализа

сигнала, где коэффициенты *k* считаются постоянными.

Основной принцип метода линейного предсказания состоит в том, что текущий отсчет речевого сигнала можно аппроксимировать линейной комбинацией предшествующих отсчетов. Коэффициенты предсказания при этом определяются 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.

Основные положения метода линейного предсказания хорошо согласуются с моделью речеобразования, где показано, что речевой сигнал можно представить в виде сигнала на выходе линейной системы с переменными во времени параметрами, возбуждаемой квазипериодическими импульсами (в пределах вокализованного сегмента) или случайным шумом (на невокализованном сегменте). Метод линейного предсказания позволяет точно и надежно оценить параметры этой линейной системы с переменными коэффициентами.

Идеи и методы линейного предсказания довольно давно обсуждаются в технической литературе. Эти идеи используются в теориях автоматического управления и информации, где их называют методами оценивания систем, или металлами идентификации систем. Под термином «идентификация» понимаются методы линейного предсказания (ЛП), основанные на оценивании параметров, однозначно описывающих систему при условии, что ее передаточная функция является полюсной. Применительно к обработке речевых сигналов методы линейного предсказания означают ряд сходных формулировок задачи моделирования речевого сигнала [1,2]. Эти формулировки часто отличаются в исходных предпосылках. Иногда они сводятся, к различным методам вычисления, используемым для оценки коэффициентов предсказания. Так, применительно к речевым сигналам существуют следующие методы вычисления (часто равноценные); ковариационный [3], автокорреляционный [1, 2, 9], лестничного фильтра [11, 12].

обратной фильтрации [1], оценки спектра [12], максимального правдоподобия [4, 6] и скалярного произведения [1].

Целесообразность использования линейного предсказания обусловлена высокой точностью описания речевого сигнала с помощью модели.

Модель речеобразования в дискретном времени представляется в форме, наиболее удобной для решения задач линейного предсказания. В этом случае общий спектр, обусловленный излучением, речевым трактом и возбуждением, описывается с помощью линейной системы с переменными параметрами и передаточной функцией

(2.27)



Эта система возбуждается импульсной последовательностью для вокализованных звуков речи и шумом для невокализованных. Таким образом, модель имеет, следующие параметры: классификатор вокализованных и невокализованных звуков, период основного тона для вокализованных сегментов, коэффициент усиления *G* и коэффициенты {*аk*} цифрового фильтра. Все эти параметры, разумеется, медленно изменяются во времени.

Определение периода основного тона и классификация тон/шум могут быть осуществлены на основе использования ряда методов, в том числе с помощью рассматриваемых ниже методов линейного предсказания. Для вокализованных звуков хорошо подходит модель, содержащая только полюса в своей передаточной функции (чисто полюсная), но для носовых и фрикативных звуков требуется учитывать и нули. Если порядок *р* модели достаточно велик, то полюсная модель позволяет получить достаточно точное описание почти для всех звуков речи. Главное достоинство этой модели заключается в том, что как параметр *G* так и коэффициенты можно оценить непосредственно с использованием очень эффективных с вычислительной точки зрения алгоритмов.

Отсчет речевого сигнала *s*(*n*) связан е сигналом возбуждения *u*(*n*) простым разностным уравнением

(2.28)



Линейный предсказатель с коэффициентами *аk* определяется как система, на выходе которой имеем

(2.29)



Системная функция предсказателя *р*-го порядка представляет собой полином вида

(2.30)



Погрешность предсказания определяется как

(2.31)



Из (2.31) видно, что погрешность предсказания представляет собой сигнал на выходе системы с передаточной функцией

(2.32)



Сравнение (2.28) и (2.31) показывает, что если сигнал точно удовлетворяет модели (8.2), то *e*(*n*)=*Gu*(*n*). Таким образом, фильтр погрешности предсказания *A* (*z*) является обратным фильтром для системы *H*(*z*), соответствующей уравнению (2.27), т. е.

(2.33)



Основная задача анализа на основе линейного предсказания заключается в непосредственном определении параметров {} по речевому сигналу с целью получения хороших оценок его спектральных свойств путем использования уравнения (2.31). Вследствие изменения свойств речевого сигнала во времени коэффициенты предсказания должны оцениваться на коротких сегментах речи. Основным подходом является определение параметров предсказания таким образом, чтобы минимизировать дисперсию погрешности на коротком сегменте сигнала. При этом предполагается, что полученные параметры являются параметрами системной функции *H*(*z*) в модели речеобразования.



То, что подобный подход приводит к полезным результатам, возможно, не сразу очевидно, но его полезность будет неоднократно подтверждена различными способами. Во-первых, пусть *e*(*n*)=*Gu*(*n*). Для вокализованной речи это означает, что *е*(*n*) будет состоять из последовательности импульсов, т.е. *е*(*n*) будет весьма мало почти все время. Поэтому в данном случае минимизация погрешности предсказания позволит получить требуемые коэффициенты. Другой повод, приводящий к тому же подходу, вытекает из того, что даже если сигнал формируется системой (2.28) с постоянными во\* времени параметрами, которая возбуждается либо единичным импульсом, либо белым шумом, то можно показать, что коэффициенты предсказания, найденные по критерию минимизации среднего квадратического значения погрешности (в каждый момент времени), совпадают с коэффициентами в (2.28). Третьей, весьма важной для практики причиной является то, что подобная минимизация приводит к линейной системе уравнений, решение которых сравнительно легко приводит к получению параметров предсказания. Кроме того, полученные параметры, как это будет ясно из дальнейшего, составляют весьма плодотворную основу для точного описания сигнала. Кратковременная энергия погрешности предсказания

(2.35)



(2.36)



(2.37)



где - сегмент речевого сигнала, выбранный в окрестности отсчета *n*, т. е.



(2.38)



Пределы суммирования справа в (2.35)-(2.37) пока не определены, но поскольку предполагается использовать концепции кратковременного анализа, то эти пределы всегда предполагаются конечными. Кроме того, для получения среднего значения необходимо разделить полученный результат на длину речевого сегмента, Однако эти константы несущественны с точки зрения решения системы линейных уравнений и поэтому далее опускаются. Параметры ак можно получить, минимизируя *Еn* в (2.37) путем вычисления, что приводит к системе уравнений



(2.38)



где - значения *аК*, минимизирующие *Еn*. Если ввести определение



(2.39)



тогда (2.38) можно переписать в более компактном виде

(2.40)



Эта система из *р* уравнений с *р* неизвестными может быть решена достаточно эффективным способом для получения неизвестных коэффициентов предсказания, минимизирующих средний квадрат погрешности предсказания на сегменте . Используя (2.37) и (2.39), можно показать, что средняя квадратическая погрешность предсказания имеет вид



(2.41)



и, используя (2.40), можно выразить *Еn* в виде

(2.42)



Таким образом, общая погрешность предсказания состоит из двух слагаемых, одно из которых является постоянным, а другое - зависит от коэффициентов предсказания.

Для решения системы уравнений относительно коэффициентов предсказания следует первоначально вычислить величины , 1≤i≤*р* и 1≤o≤*р*. Только после этого можно переходить к решению (2.40) и получению оценок Таким образом, принципиально анализ на основе линейного предсказания очень простой. Однако подробности, связанные с вычислением и последующим решением системы уравнений, являются достаточно запутанными и нуждаются в дальнейшем обсуждении.



Хотя пределы суммирования в (2.35)-(2.37) и (2.39) не определены, в (2.39) они совпадают с соответствующими пределами в (2.35)-(2.37). Как было установлено, для кратковременного анализа соответствующие пределы должны охватывать конечный интервал. Имеется два подхода к этому вопросу, и в зависимости от пределов суммирования и выбора сегмента 8п(ш) различают два метода линейного предсказания: автокорреляционный метод и ковариационный метод.

В зависимости от определения сегмента анализируемого сигнала можно получить две различные системы уравнений. Для автокорреляционного метода сигнал взвешивается с использованием *N*-точечного окна и величины получаются на основе кратковременной автокорреляционной функции. Полученная матрица корреляций является теплицевой и приводит к первой системе уравнений для параметров предсказания. При ковариационном методе сигнал предполагается известным на множестве значений -*p*≤*n*≤*N-*1. Никаких предположений о сигнале вне данного интервала не делается, поскольку только этот интервал необходим для вычислений. Полученная матрица корреляций в данном случае симметричная, но не теплицева (симметричная и такая, что элементы на любой диагонали равны между собой). В результате два различных метода вычисления корреляции приводят к двум различным системам уравнений и к двум совокупностям коэффициентов предсказания с различными свойствами.



2.3 Цифровая обработка речи в системах речевого общения человека с машиной

2.3.1 Классификация систем речевого общения человека с машиной

Эта область является чрезвычайно важной, дающей все новые и новые приложения, область, которая только еще развивается и демонстрирует огромные возможности для широкого применения.

Системы речевого обмена между человеком и машиной можно подразделить на три класса: с речевым ответом, распознавания диктора и распознавали речи.

Системы с речевым ответом предназначаются для выдачи информации пользователю в форме речевого сообщения. Таким образом, системы с речевым ответом - это системы односторонней связи, т. е. от машины к человеку. С другой стороны, системы второго и третьего классов - это системы связи от человека к машине. В системах распознавания диктора задача состоит в верификации диктора (т. е. в решении задачи о принадлежности данного диктора к некоторой группе лиц) или идентификации диктора из некоторого известного множества. Таким образом, класс задач распознавания диктора распадается на два подкласса: верификации и идентификации говорящего.

Последний класс задач распознавания речи также можно разделить на подклассы в зависимости от таких факторов, как размер словаря, количество дикторов, условия произнесения слов и т. д. Основная задача распознающей системы сводится либо к точному распознаванию произнесенной на входе фразы (т.е. система фонетической или орфографической печати произнесенного текста), либо к «пониманию» произнесенной фразы (т. е. к правильной реакции на сказанное диктором). Именно задача понимания, а не распознавания наиболее важна для систем с достаточно большим словарем непрерывных речевых сигналов, в то время как задача точного распознавания более важна для систем с ограничением словарем, малым количеством дикторов, систем распознавания изолированных слов.

2.3.2 Системы с речевым ответом

Элементами общей структурной схемы системы с речевым ответом на базе ЭВМ являются блоки памяти для хранения словаря системы с речевым ответом; хранения правил синтеза сообщений по элементам словаря; программ формирования речевого ответа.

На вход системы с речевым ответом поступает сообщение о содержании вопроса, порождаемого либо другой системой обработки информации, либо непосредственно от человека, обратившегося с интересующим его вопросом к информационной системе. Откликом системы на поставленный вопрос служит выходное сообщение в виде речевой фразы. Простым примером такой системы является автоматическая справочная телефонная служба, которая обнаруживает неправильно набранный номер, определяет причину ошибки (например, телефон отключен или ему присвоен новый номер и т. д.) и посылает на выход системы с речевым ответом сообщение, содержащее необходимее абоненту указания. В таких системах словарь обычно состоит из ограниченного набора изолированных слов (например, цифр с различными окончаниями).

В качестве другого примера рассмотрим информационную систему о состоянии курса акций. Здесь абонент должен с помощью кнопочного набора ввести код интересующего его курса. Система декодирует набор, определяет текущий курс акций и затем выдает соответствующую информацию в систему с речевым ответом для составления требуемой фразы. В данном случае словарь должен содержать достаточно широкий набор различных слов и фраз.

Существуют два основных подхода к построению систем с речевым ответом. Один из них заключается в попытке построения системы, речевые возможности которой сравнимы с возможностями человека. В этом случае для синтеза достаточно хранить словарь произношений элементов. Сигналы, необходимые для управления речевым синтезатором, в соответствии с моделью речеобразования формируются на основе правил синтеза. Такие системы представляют интерес в том случае, если требуется словарь весьма большого объема. Реализация подобных систем - это проблема, требующая чрезвычайно трудоемких исследований, и на этапе синтеза сигнала имеются обширные возможности применения рассмотренных выше методов цифровой обработки сигналов.

В системах с речевым ответом второго типа используется ограниченный словарь, и сигнал на выходе таких систем формируется посредством сочленения отдельных элементов реального речевого сигнала, взятых из словаря. Сообщения конструируются в этом.случае путем отыскания требуемых слов и фраз в памяти и воспроизведения их в требуемой последовательности. При разработке систем подобного типа следует учитывать три основных соображения. Во-первых, способ представления и хранения словаря должен быть выбран таким образом, чтобы в разработанной системе имелась возможность свободного доступа к любому элементу словаря. Во-вторых, должен быть выбран способ редактирования речевого материала словаря совместно со способом записи его элементов в память. В-третьих, необходимо обеспечить заданную последовательность выбора и воспроизведения элементов словаря (т.е. способ формирования сообщения).

Поскольку назначение систем с речевым ответом состоит в формировании речевых сообщений, предназначенных для человека, Требование к разборчивости становится определяющим. Не менее важное, значение, однако, имеют и такие параметры речи, как качество восприятия и натуральность. Таким образом, в разрабатываемой системе необходимо с предельной полнотой реализовать все три основных условия с тем, чтобы добиться максимально возможной разборчивости и натуральности речевого сигнала.

Центральным фактором, определяющим сложность систем с речевым ответом, является выбор способа цифрового представления речи при составлении словаря. Выбор способа цифрового представления оказывает большое влияние на объем и тип цифровой памяти, а также на способ синтеза речевого сообщения.

При рассмотрении способа цифрового представления речевого сигнала применительно к системам с речевым ответом полезно остановиться на трех основных моментах:

* скорость передачи информации (в битах в секунду), необходимая для получения приемлемого качества;
* сложность способа кодирования и декодирования;
* гибкость представления, т. е. возможность модификации элементов словаря.

Представление на основе кодирования речевого колебания требует наибольших скоростей передачи и, следовательно, максимального объема памяти для хранения элементов словаря. Эти способы являются простейшими с точки зрения алгоритмов кодирования-декодирования. С другой стороны, способы анализа-синтеза, которые буквально «разбивают речевой сигнал на части», обладают широкими возможностями полезной модификации элементов словаря. Два первых фактора, т.е. скорость передачи и сложность реализации, оказывают существенное влияние на технико-экономические показатели при разработке полностью цифровых систем речевого ответа.

Другой важной задачей, решаемой при построении систем с речевым ответом, являются создание и редактирование словаря. При решении этой задачи, т. е. подготовке элементов словаря и обеспечении высококачественного сигнала на выходе, цифровые методы оказываются чрезвычайно эффективными и гибкими. Обычно слова и фразы, включаемые в словарь, произносятся специально обученным диктором и записываются с высоким качеством. Затем слова или фразы подвергаются аналого-цифровому преобразованию и кодированию. Цифровое представление (которое может быть как описанием формы сигнала, так и основанным, на представлении типа, «анализ–синтез») оперативно хранится в цифровой форме в ЭВМ. Для исключения пауз между фразами используется специальный метод поиска начала и конца фразы. При высококачественной записи начало и конец каждой фразы можно определить с высокой точностью. При этом можно точно сказать, удовлетворяет ли протяженность данной фразы заданной. Фраза, кроме того, может быть воспроизведена для

проверки окончаний слов или фразы на слух. Записи можно легко повторять, пока не будут достигнуты требуемые длительность и окончание вводимой фразы.

Заключительным шагом в создании словаря являются сравнение энергетических уровней всех слов в словаре и соответствующее изменение уровней для получения некоторого единого уровня или такого распределения уровней, которое предопределяется предполагаемым использованием словаря. Это может быть сделано или на основе вычисления максимального значения сигнала, или на основе использования других мер, таких, как кратковременная энергия.

Если слово или фраза записаны с требуемым качеством, то они хранятся в определенном месте памяти словаря. Это достигается простой установкой файлов в речевой системе и указанием адресов, которые используются системой синтеза фраз для определения начала и окончания каждого элемента словаря.

Помимо рассмотренных методов создания словаря система с речевым ответом включает в себя методы синтеза фраз по элементам словаря. В этом случае методы цифрового представления также обладают значительными преимуществами. Если используется метод кодирования формы речевого колебания, то все, что здесь необходимо, - это сочленить речевые сигналы элементов словаря. Если элементом словаря является отдельное слово, то такой метод может привести к некоторой потере-натуральности звучания, но подобный подход обладает важным преимуществом, состоящим в том, что система синтеза фраз оказывается очень простой.

С другой стороны, представление, основанное на преобразовании типа «анализ-синтез», обладает большой гибкостью по отношению к изменяющимся свойствам элементов словаря, например временным соотношениям, окончаниям и т. д. Это свойство является даже более важным, чем малая скорость передачи (объем описания), которую можно достигнуть при использовании описания на основе преобразования «анализ-синтез».

Поскольку элементы словаря представлены в виде набора основных параметров речевого сигнала, можно, например, изменять период основного тона и длительность слов таким образом, чтобы привести их в соответствие с контекстом. Более интересной представляется возможность такого изменения параметров на границах слов, чтобы добиться как можно большего сходства между синтезированными и реальными речевыми сигналами. Достигнуть такого эффекта даже в простейших случаях можно лишь на основе использования правил для определения требуемого периода основного тона и протяжённости во времени, а также алгоритмов изменения параметров в соответствии с изменяющейся протяженностью слов и поглощением их границ в слитной речи [1,2].

2.3.3 Системы распознавания дикторов

При распознавании дикторов цифровая обработка речи является тем первым шагом, с которого начинается решение задачи распознавания образов. Речевой сигнал представляется с использованием таких методов цифровой обработки, которые сохраняют индивидуальные особенности диктора. Полученный образ сравнивается с предварительно подготовленными эталонными образами, а затем применяется соответствующая логика принятия решений для определения голоса заданного диктора среди возможного множества. Системы распознавания, дикторов подразделяются на два вида: идентификация и верификация. При верификации диктора требуется установить его идентичность данному эталону. Устройство верификации принимает одно из двух возможных решений: диктор является тем, за кого он себя выдает, или не является. Для вынесения такого решения используется совокупность параметров, содержащих необходимую информацию об индивидуальности диктора и измеряемых по одной или нескольким фразам. Измеренные значения сравниваются (часто с использованием некоторых существенно нелинейных метрик близости) с аналогичными параметрами эталонных образов подлежащего опознанию диктора.

Таким образом, при верификации диктора требуется однократное сравнение совокупности (совокупностей) измеренных значений со значениями параметров-ионов, на основе которого выносится решение о принятии или отклонении предполагаемой идентичности. В общем случае вычисляется расстояние между измеренными значениями и распределением эталонов. На основе распределения потерь между возможными типами ошибок (т. е. верификации «самозванца» и отклонении «подлинного» диктора) устанавливается соответствующий порог различимости (расстояния). Вероятность перечисленных выше ошибок практически не зависит от *N* (числа эталонов, хранимых в системе), поскольку все эталоны голосов других дикторов используются для формирования устойчивого распределения, характеризующего всех дикторов. Записывая сказанное выше в математической форме, обозначим распределение вероятности измеренных значении вектора *х* для диктора как *рi*(*х*), что приводит к простому решающему правилу вида

Верифицировать диктора *i*, если *рi*(*х*)>*ci* *раv* (*х*);

Отклонить диктора *i*, если *рi*(*х*)<*ci* *раv* (*х*); (2.43)

где *ci* - константа для *i*-го диктора, определяющая вероятности ошибок *i*-го диктора, а *раv* (*х*)–среднее (по всему ансамблю дикторов) распределение вероятности измеренных значений вектора *х*. Изменяя порог *ci* можно изменять вероятность ошибки, определяемую вероятностями ошибок обоих типов. Задача идентификации диктора существенно отличается от задачи верификации. В этом случае система должна точно указать одного из дикторов среди N дикторов данного множества. Таким образом, вместо однократного сравнения измеряемых параметров с хранимым в системе эталоном необходимо провести N сравнений, Решающее правило в этом случае сводится к выбору такого диктора *I*, для которого

(2.44)



т.е. выбирается диктор с минимальной абсолютной вероятностью ошибки. С увеличением количества дикторов в ансамбле возрастает и вероятность ошибки. поскольку большое число вероятностных распределений в ограниченном пространстве параметров не может не пересекаться. Все более вероятным становится то, что два или более дикторов в общем ансамбле будут иметь распределения вероятностей, которые близки друг к другу. При таких условиях приемлемая идентификация дикторов становится практически невозможной. Приведенный выше анализ позволяет сделать вывод, что между задачами идентификации и верификации имеется много общего и много различий. В каждом случае диктор должен произнести одну или несколько тестовых фраз. По этим фразам проводятся некоторые измерения, и затем вычисляются одна или несколько мер различимости («расстояния») между предъявленным и эталонным векторами. Таким образом, с позиции методов цифровой обработки обе эти задачи сходны. Основное различие возникает на этапе вынесения решений [1,2].

2.3.4 Системы распознавания речи

Как и при распознавании диктора, методы цифровой обработки применяются при распознавании речевого сигнала для получения описания распознаваемого образа, которое затем сравнивается с хранимыми в памяти эталонами. Задача распознавания речевого сигнала состоит в определении того, какое слово, фраза или предложение были произнесены.

В отличие от областей машинного речевого ответа и распознавания диктора, где задача в общем случае достаточно определена, область распознавания слов является одной из тех, где, прежде чем поставить задачу, требуется ввести большое число предположений например:

* тип речевого сигнала (изолированные слова, непрерывная речь и т.д.);
* число дикторов (система для одного диктора, нескольких дикторов, неограниченного числа дикторов);
* тип диктора (определенный, случайный, мужчина, женщина, ребенок);
* условия произнесения фраз (звукоизолированное помещение, машинный зал, общественное место);

- система передачи (высококачественный микрофон, узконаправленный микрофон, телефон);

* тип и число циклов обучения (без обучения, с ограниченным числом циклов обучения, с неограниченным числом циклов обучения);
* размер словаря (малый объем 80—20 слов, средний объем 20-100 слов и большой объем - более 100 слов);
* формат произносимых фраз (ограниченный по длительности текст, свободный речевой формат).

Из приведенного перечня условий следует, что при создании систем распознавания речи реализация некоторых из условий может оказаться более предпочтительной.

Существует много способов представления сигнала, которые можно использовать в системах распознавания речи, предоставления, применяемые в системах, инвариантных к диктору, должны быть достаточно устойчивыми. Измерения параметров должны быть простыми и однозначными, а их измеренные значения должны наиболее полно отражать различия в звуках речи. Кроме того, измерения должны допускать достаточно простую интерпретацию с позиций систем, инвариантных к диктору. Во многих таких системах использованы следующие параметры: среднее число переходов через нуль, энергия, коэффициенты линейного предсказания с использованием двухполюсной модели и погрешность предсказания [1,2].

2.3.5 Обзор существующих систем распознавания речи

В настоящее время отсутствуют дикторонезависимые системы распознавания слитной речи как с неограниченным словарем, так и ограниченным, а имеющиеся системы (такие как Dragon Dictate - программа для печати текста с голоса) требуют очень много времени и терпения для того, чтобы обучить их удовлетворительно распознавать раздельно произносимые слова одного диктора. Среди других распознающих систем можно назвать Lotus Word Pro, MedSpeak, Voice Type Simplify Speaking, ViaVoice, Kurzweil Voice. He требующие обучения системы распознают обычно от нескольких десятков до сотен слов и используются для подачи команд голосом. Однако они также являются дикторозависимыми. В этой же области остается реализация амбициозных планов, наподобии принятого в 1986 в Японском национальном проекте АТК (Advanced Telecommunication Research), который состоял в том, чтобы получать речь на одном языке и одновременно синтезировать ее на другом или утверждения о реализации идеи человеко-машинного общения. Поэтому исследования в этой области являются весьма актуальными.

3. Разработка программного обеспечения для распознавания команд управления промышленным роботом

3.1 Реализация интерфейса записи и воспроизведения звукового сигнала в операционной системе Microsoft Windows

3.1.1 Основные сведения

Звуковые данные хранятся в компьютере с помощью метода импульсно-кодовой модуляции – РСМ. Расшифровывается РСМ как Pulse-Code Modulation. При этом методе аналоговый звук квантуется по времени и амплитуде. При выводе звука на колонки происходит обратное преобразование.

При прямом преобразовании кривая звука превращается в точки. Все, что между точками, теряется, поэтому чем чаще происходит замер амплитуды, тем выше качество звука и меньше потерь. Значение, обратное времени замера (*Т3*), называется частотой дискретизации - *Тд*. Для качественного звука частота дискретизации должна быть вдвое больше, чем высшая частота в обрабатываемом сигнале звука.

В Windows есть хорошо продуманная и хорошо документированная библиотека для работы со звуком - mmsystem.

Для работы со звуковой картой используется следующий алгоритм работы:

* инициализировать звуковой драйвер;
* установить параметры;
* воспроизвести или записать звуковые данные;
* закрыть драйвер.

Для вывода звука данные о звуке в цифровом виде просто отправляются звуковой карте. Она преобразует их в аналоговый сигнал [14].

3.1.2 Основные функции, необходимые для записи звука

Все основные функции содержатся в модуле rnmsystem, поэтому необходимо подключить этот модуль, а также добавить в проект файл WINMM.LIB.

Прежде всего необходимо заполнить структуру WAVEHDR. Она содержит следующие поля:

typedef struct wavehdr\_tag {LPSTR lpData;

DWORD dwBufferLength;

DWORD dwBytesRecorded;

DWORD dwUser;

DWORD dwFlags;

DWORD dwLoops;

struct wavehdr\_tag далеко \* lpNext;

reserved DWORD;} WAVEHDR;

Структура WAVEHDR определяет заголовок, используемый, чтобы идентифицировать буфер звуковых данных

Составляющие структуры:

lpData - указатель на звуковые данные;

dwBufferLength - размер звуковых данных;

dwBytesRecorded - количество записанных байт (при воспроизведении этот параметр не используется);

dwUser - сюда можно записать все, что угодно;

dwFlags - здесь происходит описание заголовка. Конкретизирует флаги, предоставляющие информацию о буфере данных. Может принимать значения, приведенные в табл. 3.1

dwLoops - указывает количество циклов. Этот параметр используется только с буферами выходных данных.

lpNext - зарезервирован для использования драйвером устройства, чтобы указать на следующую структуру WAVEHDR в очереди.

reserved - зарезервирован для использования драйвером устройства.

Это поле может использоваться, чтобы запомнить указатель на структуру, содержащую информацию о клиенте.

Таблица 3.1- Флаги и их значения

|  |  |
| --- | --- |
| Флаг | Значение |
| WHDR\_DONE | устанавливается драйвером устройства, чтобы указать, что работа с буфером данных закончена и возвращается к приложению |
| WHDR\_BEGINLOOP | показывает, что этот буфер - первый буфер в цикле. Этот флаг используется только с буферами выходных данных. |
| WHDR\_ENDLOOP | показывает, что этот буфер - последний буфер в цикле. Этот флаг используется только с буферами выходных данных |
| WHDR\_PREPARED | устанавливается драйвером устройства, чтобы указать, что буфер данных был подготовлен с помощью waveInPrepareHeader или waveOutPrepareHeader |
| WHDR\_JNQUEUE | устанавливается драйвером устройства, чтобы указать, что буфер данных находится в очереди для воспроизведения |

Используйте WHDR\_BEGINLOOP и флаг WHDR\_ENDLOOP в поле dwFlags, чтобы указать начала и концы блоков данных для использования их в цикле. Чтобы прокручивать в цикле один блок, необходимо установить оба флага для этого блока. Необходимо использовать поле dwLoops в структуре WAVEHDR для первого блока в цикле, чтобы установить количество циклов.

Первая функция используемая для записи звука - wavelnOpen. Эта функция устройство ввода для звукового потока. В модуле она объявлена следующим образом:

MMRESULT waveInOpen(

LPHWAVEIN phwi,

UINT\_PTR uDevicero,

LPWAVEFORMATEX pwfx,

DWORD\_PTR dwCallback,

DWORD\_PTR dwCallbacklnstance,

DWORD fdwOpen);

В функции определен ряд параметров. Рассмотрим их основное назначение

phwi - адрес, по которому будет записан указатель на устройство воспроизведения. Этот параметр может быть NULL, если WAVE\_FORMAT\_QUERY указан для fdwOpen.

UDevicelD – идентификатор устройства, которое необходимо открыть, если поставить сюда константу WAVE\_MAPPER, то откроется устройство по умолчанию.

IpFormat – указатель на структуру типа WAVEFORMATEX, в которой описан формат записываемых звуковых данных.

dwCallback – указатель на функцию «семафор». Эта функция вызывается для сообщения о текущих событиях.

fdwOpen – параметры открываемого устройства. Может принимать значения, приведенные в табл. 3.2

Необходимо использовать функцию wavelnGetNumDevs, чтобы определить число устройств ввода, присутствующих в системе. Идентификатор устройства, указанный uDeviceID, изменяется в пределах от ноля до на один меньше, чем количество присутствующих устройств. Константа WAVE\_MAPPER может также использоваться в качестве идентификатора устройства.

Если выбирать, чтобы окно или поток получал информацию отзыва, следующие сообщения отправляются процедуре окна или потока, чтобы указать изменения в процессе ввода звуковой информации: MM\_WIM\_OPEN, MM\_WIM\_CLOSE, и MM\_WIM\_DATA.

Если вы выбираете, чтобы функция получала информацию отзыва, функции отправляются следующие сообщения, чтобы указать изменения в процессе ввода звуковой информации: WIM\_OPEN, WIM\_CLOSE, и WIM\_DATA.

Таблица 3.2 – Параметры открываемого устройства

|  |  |
| --- | --- |
| CALLBACK\_EVENT | в параметре dwCallback находится событие THandle, через которое происходит информирование о ходе воспроизведения; |
| CALLBACK\_FUNCTION | в dwCallback находится указатель на функцию; |
| CALLBACK\_THREAD | в dwCallback находится идентификатор потока |
| CALLBACK\_WINDOW | в dwCallback находится указатель на окно, которому посылается сообщение |
| CALLBACK\_NULL | в dwCallback ничего нет |
| WAVE\_ALLOWSYNC | можно открыть устройство в синхронном режиме |
| WAVE\_FORMAT\_DIRECT | запрещается преобразование данных с помощью АСМ-драйвера (ACM - Audio Compression Manager) |
| WAVE\_FORMAT\_QUERY | если установить этот параметр, то реального открытия звуковой карты не происходит, функция проверяет возможность открытия с заданными параметрами, и если всё нормально, то возвращает MMSYSERRNOERROR (если параметры недопустимы, то возвращается код ошибки) |
| MMS YSERR\_ALLOCATED | указанный ресурс уже занят |
| MMSYSERR\_BADDEVICEID | указанный идентификатор устройства не существует |
| MMSYSERR\_NODRIVER | нет драйвера для устройства |
| MMSYSERR\_NOMEM | невозможно использовать память |
| WAVERR\_BADFORMAT | попытка открытия в неподдерживаемом звуковом формате |

Необходимо заполнить структуру WAVEFORMATEX. Она имеет следующий вид:

typedef struct WAVEFORMATEX { WORD wFormatTag;

WORD nChannels;

DWORD nSamplesPerSec;

DWORD nAvgBytesPerSec;

WORD nBlockAlign;

WORD wBitsPerSample;

WORD cbSize;} WAVEFORMATEX;

В этой структуре:

wFormatTag – формат звуковых данных (чаще всего используется WAVE\_FORMAT\_PCM);

nChannels – количество каналов (1 - монО, 2 - стерео);

nSamplesPerSec – частота дискретизации (возможны значения 8000, 11025, 22050 и 44100);

nAvgBytesPerSec – количество байт в секунду для WAVEFORMATPCM это является результатом nSamplesPerSec\* nBlockAlign;

nBlockAlign – выравнивание блока (для WAVEFORMATPCM равен wBitsPerSample /8\* nChannels);

wBitsPerSample – количество бит в одной выборке (для WAVE\_FORMAT\_PCM может быть 8 или 16);

cbSize - размер дополнительной информации, которая располагается после структуры (если ничего нет, то должен быть 0).

Теперь следует подготовить заголовки, которые будут отправляться драйверу. Для этого существует функция waveInPrepareHeader

MMRESULT waveInPrepareHeader(

HWAVEOUT hwo,

LPWAVEHDR pwh,

U1NT cbwh);

Внутренняя структура:

hwo – идентификатор устройства записи (полученный после вызова функции wavelnOpen).

pwh – указатель на структуру wavehdr\_tag;

cbwh – размер структуры wavehdr\_tag.

Функция возвращает значения, приведенные в табл. 3.3.

При этом lpData, dwBufferLength, и dwFlags – члены структуры

WAVEHDR - должны быть установлены перед запросом этой функции (dwFlags должен быть нулевым).

Таблица 3.3 – Возвращаемые значения

|  |  |
| --- | --- |
| MMSYSERR\_NOERROR | успех |
| MMSYSERR\_INVALHANDLE | указанный дескриптор устройства недействителен |
| MMSYSERR\_INVALPARAM | базовый адрес буфера не выровнен с объемом выборки |
| MMSYSERR\_NODRIVER | нет драйвера для устройства |
| MMSYSERR\_NOMEM | невозможно использовать память |

Для того, чтобы освободить драйвер необходимо использовать функцию waveInUnprepareHeader. Эта функция должна быть вызвана после того, как драйвер устройства заполняет буфер и возвращает его приложению. Необходимо вызвать эту функцию перед освобождением буфера.

MMRESULT waveInUnprepareHeader(

HWAVEIN hwi,

LPWAVEHDR pwh,

U1NT cbwh);

Параметры:

*hwi* - указатель на записывающее устройство;

pwh - указатель на структуру WAVEHDR, идентифицирующую буфер, который необходимо очистить.

cbwh - размер, в байтах, структуры WAVEHDR.

Функция возвращает MMSYSERR\_NOERROR, если всё проходит успешно, если есть ошибки, возвращается одно из значений, приведенных в табл. 3.4.

Эта функция дополняет waveInPrepareHeader функцию. Необходимо вызвать эту функцию перед освобождением буфера. После передачи буфера в драйвер устройства функцией waveInAddBuffer, необходимо ждать, пока драйвер не закончит работу с буфером перед запросом *waveInUnprepareHeader*. Очистка буфера, который не был занят, не имеет никакого эффекта, и функция возвращает ноль.

Таблица 3.4 – Значения, выдаваемые при ошибках

|  |  |
| --- | --- |
| Ошибка | Значение |
| MMSYSERR\_INVALHANDLE | указанный дескриптор устройства недействителен |
| MMSYSERR\_NODRIVER | нет драйвера для устройства |
| MMSYSERR\_NOMEM | невозможно использовать память |
| MMSYSERR\_STILLPLAYING | буфер, указанный pwh, занят |

Для передачи буфера приёма данных предоставленному звуковому устройству используется функция waveInAddBuffer. Когда буфер заполнен, приложение получает соответствующее сообщение. Функция имеет следующий вид:

MMRESULT waveInAddBuffer(HWAVEIN hwi,

LPWAVEHDR pwh,

UINT cbwh);

Описание параметров:

hwi- указатель на входное устройство;

pwh - указатель на структуру WAVEHDR, которая идентифицирует буфер.

*cbwh* - размер, в байтах, структуры WAVEHDR.

Функция возвращает MMSYSERR\_NOERROR, если всё проходит успешно, если есть ошибки, возвращает одно из значений, приведенных в табл. 3.5.

Таблица 3.5 - Возвращаемые значения

|  |  |
| --- | --- |
| Ошибка | Значение |
| MMSYSERR\_INVALHANDLE | указанный дескриптор устройства недействителен |
| MMSYSERR\_NODRIVER | нет драйвера для устройства |
| MMSYSERR\_NOMEM | невозможно использовать память |
| WAVERR\_UNPREPARED | буфер, указанный pwh, не был готов |

Когда буфер заполнен, бит WHDR\_DONE установлен в dwFlags -члене структуры WAVEHDR.

Буфер должен быть подготовлен функцией waveInPrepareHeader перед тем, как это будет передаваться функции waveInAddBuffer.

Далее вызывается функция waveInStart. Она начинает ввод данных через предоставленное устройство. Эта функция выглядит следующим образом:

MMRESULT waveInStart(HWAVEIN hwi);

Параметр hwi указывает на устройство, принимающее информацию.

Функция возвращает MMSYSERR\_NOERROR, если всё проходит успешно, если есть ошибки, возвращается одно значений, приведенных в табл. 3.6

Таблица 3.6 - Возвращаемые значения

|  |  |
| --- | --- |
| Ошибка | Значение |
| MMSYSERR\_INVALHANDLE | указанный дескриптор устройства недействителен |
| MMSYSERR\_NODRIVER | нет драйвера для устройства |
| MMSYSERR\_NOMEM | невозможно использовать память |

Буферы возвращаются приложению, когда заполнены или, когда вызвана функция waveInReset(dwBytesRecorded) член в заголовке будет содержать длину данных). Если нет никаких буферов в очереди, данные проходят мимо приложения, и ввод продолжается.

Вызов этой функции, когда она уже была вызвана, не имеет эффекта, и функция возвращает ноль [15].

3.1.3 Основные функции, необходимые для воспроизведения звука.

Для воспроизведения звуковой информации первая функция waveOutOpen. Она открывает устройство вывода для звукового потока. В модуле она объявлена следующим образом:

MMRESULT waveOutOpen (LPHWAVEIN phwi,

UINT\_PTR uDeviceID, LPWAVEFORMATEX pwfx, DWORD\_PTR dwCallback, DWORD\_PTR dwCallbacklnstance, DWORD fdwOpen);

В функции определен ряд параметров. Рассмотрим их основное назначение

phwi - адрес, по которому будет записан указатель на устройство воспроизведения;

UDeviceID - идентификатор устройства, которое необходимо открыть, если поставить сюда константу WAVE\_MAPPER, то откроется устройство по умолчанию.

lpFormat - указатель на структуру типа WAVEFORMATEX, в которой описан формат воспроизводимых звуковых данных.

dwCallback - указатель на функцию «семафор». Эта функция вызывается для сообщения о текущих событиях.

dwFlags - параметры открываемого устройства. Может принимать следующие значения, совпадающие со значениями fdwOpen для функции wavelnOpen

Структура WAVEFORMATEX для воспроизведения звуковой информации имеет такой же вид, как и для записи. Однако при воспроизведении звука, она заполняется автоматически при выборе воспроизводимого файла.

Теперь предстоит подготовить заголовки, которые будут отправляться драйверу. Для этого есть функция waveOutPrepareHeader

MMRESULT waveOutPrepareHeader(HWAVEOUT hwo,

LPWAVEHDR pwh,

UINT cbwh);

Возвращаемые параметры те же. Внутренняя структура:

hwo - идентификатор устройства воспроизведения (полученный

после вызова функции waveOutOpen).

pwh - указатель на структуру wavehdr\_tag; cbwh - размер структуры wavehdr\_tag.

Теперь структура wavehdr\_tag. Она выглядит так же, как и в случае записи.

После окончания работы с этим заголовком нужно освободить его с помощью функции *waveOutUnprepareHeader*.

Теперь нужно отправить подготовленный заголовок драйверу. Для этого используется функция waveOutWrite, которая выглядит следующим образом:

MMRESULT waveOutWrite (HWAVEOUT hwo,

LPWAVEHDR pwh,

UINT cbwh);

В ней:

hwo - идентификатор устройства воспроизведения;

pwh - это указатель на структуру, сформированную с помощью waveOutPrepareHeader;

cbwh - размер структуры wavehdr. Возвращаемые значения те же, что и при открытии звуковой карты.

После вывода звука нужно вызвать waveOutUnprepareHeader, чтобы очистить заголовки и закрыть устройство. Теперь необходимо закрыть устройство с помощью функции waveOutClose. Она имеет следующий вид:

MMRESULT waveOutClose (HWAVEOUT hwo);

Параметр *hwo* указывает на устройство вывода. Если функция завершается успешно, дескриптор не больше действителен после этого вызова. Функция возвращает MMSYSERR\_NOERROR, если всё проходит успешно, если есть ошибки, возвращается одно из значений, приведенных в табл. 3.7.

Если устройство ещё воспроизводит звуковой файл, функция его не закрывает. Необходимо использовать функцию waveOutReset, чтобы закончить воспроизведение перед запросом waveOutClose.

Таблица 3.7 - Возвращаемые значения

|  |  |
| --- | --- |
| Ошибка | Значение |
| MMSYSERR\_INVALHANDLE | указанный дескриптор устройства недействителен |
| MMSYSERR\_NODRIVER | нет драйвера для устройства |
| MMSYSERR\_NOMEM | невозможно использовать память |
| WAVERR\_STILLPLAYING | буфер, указанный pwh, занят |

waveOutReset останавливает воспроизведение на указанном вывода и сбрасывает текущую координату в нолю. Воспроизведение из буферов отмечается как выполнено и управление передаётся приложению. Функция имеет следующий вид:

MMRESULT waveOutReset (HWAVEOUT hwo);

Параметр hwo указывает на устройство вывода.

Функция возвращает MMSYSERR\_NOERROR, если всё проходит успешно, если есть ошибки, возвращается одно из значений, приведенных в табл. 3.8 [14,15].

Таблица 3.8 - Возвращаемые значения

|  |  |
| --- | --- |
| Ошибка | Значение |
| MMSYSERR\_INVALHANDLE | указанный дескриптор устройства недействителен |
| MMSYSERR\_NODRIVER | нет драйвера для устройства |
| MMSYSERR\_NOMEM | невозможно использовать память |
| MMSYSERR\_NOTSUPPORTED | указанное устройство является синхронным и не поддерживает приостановку |

3.2 Реализация программного обеспечения для записи, воспроизведения и анализа звукового сигнала

Для реализации программного обеспечения для записи. воспроизведения и анализа звукового сигнала были созданы следующие окна: главное окно программы, окно добавления слова в словарь, окно записи и обработки звука и окна, обеспечивающие просмотр рассчитанных характеристик в виде графиков.

В главном окне обеспечивается просмотр и редактирование существующих словарей, а также их сохранение. Это окно содержит главное меню, панель управления и рабочую область приложения. Главное меню состоит из следующих пунктов: File, Edit, View, Help. Нажатие на пункт File позволяет создать новый файл, открыть существующий файл с расширением \*.spl, открыть недавно использованный файл, сохранить новый файл и сохранить существующий файл под другим именем, выйти из программы. При нажатии на пункт Edit можно выбрать одно из предлагаемых действий: добавить слово, удалить слово, воспроизводить звук выделенного слова в рабочей области приложения. Пункт View отвечает за вид главного окна: наличие или отсутствие панели управления и строки состояния.

В панели управления присутствуют следующие кнопки: создание нового файла, открытие существующего файла, сохранение файла, удаление и добавление слова, воспроизведение слова,

Для воспроизведения выбранного слова требуется лишь нажать на соответствующую кнопку в панели управления либо выбрать Edit->Play Sound.

Для записи нового слова требуется нажать на соответствующую кнопку в панели управления либо выбрать Edit->Add Word. При этом появляется новое окно, основанное на диалоговой форме, в котором есть поле для ввода буквенного обозначения нового слова. После ввода слова в поле Word, необходимо нажать кнопку Record. После этого появляется новое окно, также основанное на диалоговом окне. Это окно обеспечивает запись и анализ нового слова. Для записи нового слова, прежде всего, необходимо выбрать в поле Source присутствующую в данном компьютере звуковую карту. Далее в поле Format необходимо выбрать формат записываемых данных. Предлагаются форматы данных, приведенные в табл. 3.9.

Выбор формата записываемых данных необходимо осуществлять исходя из следуюпщх предпосылок: с одной стороны, чем выше частота дискретизации (количество элементов в секунду), чем выше количество каналов и чем выше количество бит в элементе, тем выше качество записи, а потом и воспроизведения звука; с другой стороны, чем выше эти параметры, тем сложнее проводить анализ звука, тем дольше обрабатываются данные. Поэтому для решения поставленной задачи выбирается частота дискретизации 11025 Гц, выбирается моно-режим, т.е. количество каналов=1, и количество бит в элементе=8.

После этого всё уже готово для записи. Для непосредственного осуществления записи звукового аналога слова необходимо нажать на кнопку Record.

Таблица 3.9 - Форматы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Название | Количество | Частота | Количество |
|  | каналов | дискретизации | бит |
| WAVE\_FORMAT\_1M08 | 1 | 11025 | 8 |
| WAVE\_FORMAT\_1M16 | 1 | 11025 | 16 |
| WAVE\_FORMAT\_J S08 | 2 | 11025 | 8 |
| WAVE\_FORMAT\_lS16 | 2 | 11025 | 16 |
| WAVE\_FORMAT\_2M08 | 1 | 22050 | 8 |
| WAVE\_FORMAT\_2M16 | 1 | 22050 | 16 |
| WAVE\_FORMAT\_2S08 | 2 | 22050 | 8 |
| W AVE\_FORMAT\_2 S16 | 2 | 22050 | 16 |
| WAVE\_FORMAT\_4M08 | 1 | 44100 | 8 |
| WA VE\_FORMAT\_4Ml 6 | 1 | 44100 | 16 |
| WAVE\_FORMAT\_4S08 | 2 | 44100 | 8 |
| WAVE\_FORMAT\_4S16 | 2 | 44100 | 16 |

После нажатия на кнопку Record, в программе устанавливается режим работы программы - запись. При этом вызываются функции, которые

останавливают воспроизведение, если оно было включено. Кнопка Record становится неактивной, а кнопка Stop - активной. Далее определяются параметры выбранной звуковой карты и формат записываемого звука, Очищаются буферы, в которые будет производиться запись.

Ниже приведена функция, обеспечивающая запись звуковой информации.

BOOL CWaveIriDevice::Record(CWave\* pWave, int iBlockSize) {if (IsOpenQ) {if (! CanDoFormat(pWave->GetFormat())) {TRACE("Imput device is already open and doesn't support format"); return FALSE;}

} else {if (!Open(WAVE\_MAPPER, pWave->GetFormat())) { TRACE("No input device supports format"); return FALSE;}} miBlockSize = iBlockSize;

WAVEHDR\* phdr = (WAVEHDR\*)malloc(sizeof(WAVEHDR)); ASSERT(phdr);

memset(phdr, 0, sizeof(WAVEHDR));

phdr->lpData = (char\*)malloc(iBlockSize);

ASSERT(phdr->lpData);

phdr->dwBufferLength = iBlockSize;

phdr->dwUser = (D WORD)(void\*)p Wave;

MMRESULT mmr = waveInPrepareHeader(m\_hInDev, phdr,

sizeof(WAVEHDR)); if (mmr) (MMERR(mmr); return FALSE;}

mmr = waveInAdabuffer(m\_hInDev, phdr, sizeof(WAVEHDR)); if (mmr) {MMERR(rnmr);

return FALSE;} (3.1)

phdr = (WAVEHDR\*)malloc(sizeof(WAVEHDR));

ASSERT(phdr);

memset(phdr, 0, sizeof(WAVEHDR)); phdr->lpData = (char\*)malloc(iBlockSize); ASSERT(phdr->lpData); phdr->dwBufferLength = iBlockSize; phdr->dwUser = (DWORD)(void\*)pWave;

mmr = waveInPrepareHeader(m\_hInDev, phdr, sizeof(WAVEHDR)); if (mmr) {MMERR(mmr); return FALSE;}

mmr = waveInAddBufTer(m\_hInDev, phdr, sizeof(WAVEHDR)); if (mmr) (MMERR(mmr); return FALSE;}

mmr = wavelnS tart(m\_h InDe v); if (mmr) {MMERR(mmr); return FALSE;} return TRUE;}

Прежде всего, проверяется, открыто ли устройство. Если открыто, то проверяется, поддерживает ли оно выбранный формат. Если не поддерживает, то выводится сообщение, что устройство уже открыто, но такой формат оно не поддерживает. Если устройство закрыто, то предпринимается попытка открыть какое-либо устройство, поддерживающее данный формат. Если не удалось открыть устройство, поддерживающее данный формат, то выводится сообщение, что никакое устройство не поддерживает данный формат. Устанавливаем два буфера и отправляем их в драйвер устройства. Устанавливаем заголовок. Опустошаем заголовок. Подготавливаем заголовок. Начинаем запись

До тех пор, пока не будет нажата кнопка Stop, будет производиться запись. После этого при нажатии на кнопку Play можно прослушать весь записанный сигнал. При наличии претензий к записанному сигналу, его

можно переписать, повторно нажав на кнопку Record. Все ранее записанные данные при этом теряются

Для обработки полученных данных необходимо нажать на кнопку OpenData, которая вызывает функцию обработки данных. После нажатия на кнопку OpenData, в программе устанавливается режим работы программы -обработка данных, находящихся в буфере.

Ни же приведена функция, обеспечивающая обработку данных, находящихся в буфере.

BOOL CWaveOutDevice::OpenData(CWave \*pWave,CClientDC \*curDC) (int count=0;

CWaveBlockList\* pBL = pWave->GetBlockList(); if(!pBL) return FALSE; POSITION pos = pBL->GetHeadPositionO; if(!pos) return FALSE;

if (pWave->m\_bAutoDestruct) pWave->AddRef(); while (pos)

{CWaveBlock\* pBlock = pBL->GetNext(pos); ASSERT(pBlock);

WAVEHDR\* phdr = (WAVEHDR\*)malloc(sizeof(WAVEHDR)); ASSERT(phdr);

memset(phdr, 0, sizeof(WAVEHDR)); phdr->lpData = (char\*) pBlock->GetSamples(); phdr->dwBufferLength = pBlock->GetSize(); phdr->dwUser = (DWORD)(void\*)pWave; CString ss("");

ss.Format(\_T("%s"),pBlock->m\_pSamples); int iw = ss.GetLengthO;

intnum=0; (3.2)

for(long h=0;h<4096;h++)

{num=count\*4096+h; bunker[num]=(int)ss. GetAt(O); CString s("");

s.Format(\_T("%f "),bunker[num]);

ss.Delete(0,l);}

count-H-;

len=4096\*count;}

Noise(bunker,len,0);

Filter(bun,l);

Hamming(fнl,l);

Furje(ham,l);

СИр(пЦ);

AfxMessageBox("Data are..."); return TRUE;}

Прежде всего, обнуляем номер буфера. Далее берём список блоков данных из функции GetBlockList(). Если это невозможно, выходим из функции. Определяем позицию первого элемента в блоке данных. При невозможности выполнения действия, выходим из функции. Наращиваем счётчик ref, если мы его используем. До тех пор, пока существует pos, выполняем следующий цикл. Определяем позицию следующего элемента. Устанавливаем заголовок. Очищаем заголовок. Передаём в параметр IpData значения элементов. Определяем длину буфера. Объявляем строку. В эту строку записываем данные о звуке из pBlock. Определяем длину строки. Объявляем и обнуляем счётчик для массива, в который будем записывать данные о звуке. Т.к. буфер ограничен числом 4096, в цикле от 0 до 4096 записываем данные из строки в массив. Номер элемента массива определяется как номер буфера умноженный на его длину плюс номер элемента в буфере. Непосредственно сама запись из строки в массив. Переопределяем тип данных массива. Увеличиваем номер массива.

Рассчитываем длину массива. Вызываем функцию избавления от шумов. Вызываем функцию обработки с помощью фильтра. Вызываем функцию обработки в окне Хэмминга. Вызываем функцию выполнения преобразования Фурье. Вызываем функцию проведения клиппирования. Выводим на экран сообщение, что данные успешно обработаны. Выходим из функции.

После появления сообщения о том, что данные успешно обработаны, можно воспроизвести звуковую информацию, находящуюся в буфере, просмотреть графики функций, записать новую информацию (при этом теряется обработанная) или выйти из программы с сохранением звуковой информации, нажав Ok, или без него, нажав Cancel.

Для просмотра графиков, представляющих результаты обработки данных, в этом окне расположены кнопки, при нажатии на каждую из которых появляется окно, в котором есть кнопка, нажав на которую, получаем график интересующей нас зависимости.

3.3 Реализация функции распознавания голосовых команд голосового управления промышленным роботом

Для реализации функции распознавания голосовых команд голосового управления промышленным роботом была создана функция OpenData. После того, как данные о записанной информации занесены в массив, функция вызывает функции, отвечающие за преобразование этого массива. Прежде всего, вызывается функция Noise, которая на основании разделения вокализованного звука и невокализованного переписывает массив с данными о звуке в новый массив, содержащий лишь участки вокализованного звука.

После этого проводится обработка в окне Хэмминга. График полученный в результате этого преобразования приведен на рис 3.3.

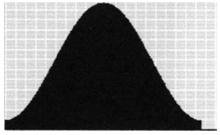


Рисунок 3.3 – График функции, после обработки в окне Хэмминга

Далее выполняется преобразование Фурье для коэффициентов, взвешенных окном Хэмминга После этого производится фильтрация. Затем проводится обратное преобразование Фурье. Далее следует операция клигширования, на основании графика которой делается вывод о том, какое слово было произнесено. На рис. 3.4 приведены графики сигнала для слова «вперёд».

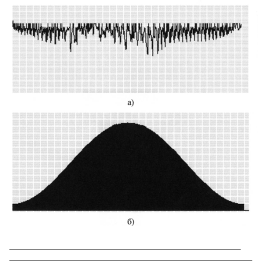


Рисунок 3.5 - Графики сигнала для слова «назад» (а - график вокализованной части сигнала; б - график сигнала после обработки в окне Хэмминга; в -график сигнала после клиппирования)

На рис. 3.6 приведены графики сигнала для слова «стоп».

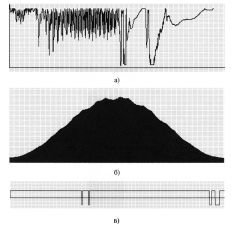


Рисунок 3.6 – Графики сигнала для слова «стоп» (а - график вокализованной части сигнала; б - график сигнала после обработки в окне Хэмминга; в -график сигнала после клиппирования)

На рис. 3.11 приведены графики сигнала для слова «пять».

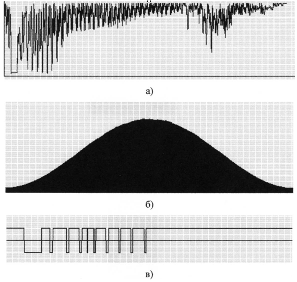


Рисунок 3.11 – Графики сигнала для слова «пять» (а - график вокализованной части сигнала; б - график сигнала после обработки в окне Хэмминга; в - график сигнала после клиппирования)

Недостатком предложенного метода распознавания слова является необучаемость программы, отнесение всех произнесенных слов к какому-то из заложенных в программе, а также невозможность настройки в конкретном помещении без специалиста. К достоинствам программы можно отнести независимость программы от диктора, кроме случаев дефекта дикции.

3.4 Реализация голосового управления трёхмерными моделями промышленного робота

Библиотека OpenGL представляет собой интерфейс программирования трехмерной графики - это её основное предназначение. Библиотека представляет собой программный интерфейс для аппаратного обеспечения машинной графики. Этот интерфейс состоит приблизительно из 250 отдельных команд (почти 200 команд в ядре OpenGL и еще 50 команд в библиотеке утилит OpenGL), которые используются для того, чтобы определить объекты и операции, необходимые для создания интерактивных трехмерных прикладных программ.

Единицей информации является вершина, из них состоят более сложные объекты. При создании заданной модели необходимо создать вершины, указать, как их соединять (линиями или многоугольниками), установить координаты и параметры камеры и ламп, а библиотека OpenGL создает изображения на экране [16].

Исходя из вышеуказанных причин было принято решение реализовать моделирование функционирования промышленного робота с помощью средств библиотеки OpenGL.

Для создания требуемой модели робота используются различные трехмерные объекты. Каждая трехмерная модель имеет две разновидности: каркас (wireframe) без поверхностных нормалей и объемная модель (solid) с закраской и поверхностными нормалями. При создании программного обеспечения использовались каркасные модели.

Преобразование вида и моделирование сложно связаны в OpenGL и в действительности объединены в одну матрицу видового преобразования (modelview matrix). Существуют три подпрограммы OpenGL для преобразований модели: glTranslate(), glRotate(), glScale(). Эти подпрограммы выполняют преобразование объекта (или системы координат) с помощью его параллельного переноса (moving), поворота (rotating), растягивания (stretching), сжатия (shrinking) или зеркального отражения (reflecting). Все три команды эквивалентны созданию соответствующей матрицы параллельного переноса (translation), поворота (rotation) или

масштабирования {scaling), а затем вызову команды glMultMatrixQ с этой матрицей в качестве аргумента. Однако использование этих трех подпрограмм должно быть быстрее использования команды glMultMatrixQ. OpenGL автоматически вычисляет эти матрицы для пользователя [17].

Все преобразования в программе осуществлены с помощью указанных подпрограмм.

Ниже приведена функция, обеспечивающая моделирование функционирования промышленного робота.

void CMain::OnOpenGLFirst()

{glPushMatrixQ;

glTranslatef(k\*per,-3.0,0.0);

glRotatef(360.0\* vspos/100,0,1,0);

glRotatef(l 80,0,0,1);

glLineWidth(2.0);

glPushMatrix();

glRotatef(90.0,1.0,0.0,0.0);

glColor3f(0.0,0.0,1.0);

auxWireCone(1.5,1.0);

glPopMatrix();

glTranslatef(0.0,-l.0,0.0);

glColor3f(0.0,0.5,0.5);

auxWireCylinder(0.6,4.0);

glPushMatrix();

glTranslatef(0.0,-3.0,0.0);

glRotatef(90.0,1.0,0.0,0.0);

glColor3f(l.0,0.0,0.0);

auxWireCylinder(0.6,2.0);

glRotatef(180.0\*hspos/100,0.0,1.0,0.0);

glTranslatef(l.0,0.5,0.0); (3.3)

Прежде всего, перемещаем сцену. Далее задаём вращение вокруг оси ъ. Задаём переменную, которая будет обеспечивать перемещение робота согласно голосовым командам. Запоминаем положение основы робота. Строим его основу – конус, поднимаемся к его вершине. Строим колонну робота и перемещаемся к её вершине, поворачиваемся на 90 градусов относительно оси *х*. Строим цилиндр – крепление плеча. Задаём движение

плеча горизонтальной прокруткой. Перемещаемся в конец крепления плеча и создаём плечо –- строим призму. Перемещаемся к её концу, поворачиваемся на 90 градусов и создаём цилиндр – локоть. Далее после перемещения к концу этого цилиндра строим призму – основание схвата, поворачиваемся на 90 градусов и строим первый и второй пальцы.

Результат моделирования робота приведен на рис. 3.1

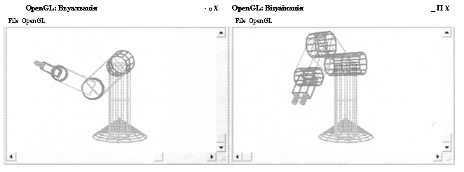


Рисунок 3.1 Результат моделирования функционирования промышленного робота

Выводы

Организация управления промышленными роботами (ПР) продолжает оставаться актуальной задачей робототехники. Управление роботами реализуется различными способами. К ним относятся: управление при помощи пультов ручного управления, командное управление от управляющих компьютеров, дистанционное управление при помощи внешних вычислительных систем. С точки зрения повышения эффективности обучения ПР одним из наиболее перспективных способов является голосовое управление.

Во время написания магистерской аттестационной работы были проанализированы методы цифровой обработки звуковых сигналов и их использование в системах распознавания речи, выполнен анализ цифровой фильтрации и особенностей акустической фонетики и её учёт при обработке цифровых сигналов.

В работе рассмотрены гомоморфная обработка речи, кодирование речи на основе линейного предсказания и цифровая её обработка в системах речевого общения человека с машиной, особенности различных систем распознавания речи.

Первоначальной задачей являлось создание интерфейса записи и воспроизведения звукового сигнала. Это было выполнено с использованием встроенной библиотеки mmsystem операционной системы Windows. Далее были добавлены функции, обеспечивающие обработку записанного сигнала: выделение вокализованных участков речи, взвешивание коэффициентов дискретного сигнала с помощью окна Хэмминга, преобразование Фурье, фильтрация, обратное преобразование Фурье, а также клиппирование и вычисление кепстра.

На основании проведенного анализа сигнала создана функция распознавания слова-команды. При помощи графической библиотеки OpenGL была обеспечена визуализация выполнения трёхмерной моделью ПР

распознанной голосовой команды. Применение библиотеки OpenGL позволяет моделировать любой из как уже существующих, так и разрабатываемых видов ПР.

Сложность решаемой задачи обуславливалась тем, что даже один и тот же человек не может два раза произнести одно и то же слово абсолютно одинаково и возникает разница в скорости произнесения слова и амплитуде звукового смгнала. При этом возникает необходимость выделения зависимости фонем (сочетания элементарных звуков), присущих одному слову, а потом описания этих зависимостей в виде, понятном для распознавания. Выделение этих зависимостей - является достаточно сложно реализуемой задачей.

Основным достоинством программы является её независимость от оператора: при условиях достаточно чёткого произношения команды, пол и возраст диктора не имеет влияния на распознавание слова. Её применение целесообразно для адаптации промышленного робота к условиям производственной среды, а также для создания информационного потока управления роботом. Возможными перспективами улучшения программы являются увеличение словаря, добавление возможности обучения и улучшение гибкости цифрового представления звукового сигнала.

Перечень ссылок

1. Рабинер Л. Р., Шафер Р. В. Цифровая обработка речевых сигналов: Пер. с англ./Под ред. М. В. Назарова и Ю. Н. Прохорова. - М.: Радио и связь, 1981.-496 с.
2. Рабинер Л. Р., Гоулд Б. Теория и применение цифровой обработки сигналов: Пер. с англ./Под ред. Александрова Ю. Н. - М.: Мир, 1978. -848 с.
3. А. V. Oppenhehn and R. W. Schafer, Digital Signal Processing,, Prentice-Hall, Inc., Englewood Cliffs, N.J., 1975. - 436 p.
4. A. Peled and B. Liu, Digital Signal Processing. Theory, Design and Implementation, John Wiley and Sons.New York, 1976. - 675 p.
5. J. W. Cooley and J. W. Tukey, "An Algorithm for the Machine Computation of Complex Fourier.Series," Math Computation, Vol. 19, 1965. - 452 p.
6. Бондарев В. H., Трёстер Г., Чернега В. С. Цифровая обработка сигналов: методы и средства. Учеб. Пособие для вузов. 2-е изд. - X.: Конус, 2001. -398 с.
7. Марпл С. Л. Цифровой спектральный анализ и его приложения. - М.: Мир, 1990.-584 с.
8. Бендат Дж., Пирсол А. Применения корреляционного и спектрального анализа. -М.: Мир, 1983. -312 с.
9. Гольденберг Л. М., Матюшкин Б. Д., Поляк М. Н. Цифровая обработка сигналов: Справочник. -М.: Радио и связь, 1985. - 312 с.
10. Гутников В. С. Фильтрация измерительных сигналов. - Л.: Энергоатомиздат, Ленингр. отд-ние, 1990. - 192 с.
11. J. Н. McClellan, Т. W. Parks, and L. R. Rabiner, "A Computer Program for Designing Optimum FIR Linear Phase Digital Filters," IEEE Trans. Audio and Electro acoustics, Vol. AU-21, 1973. - 347 p.
12. Цифровые фильтры в электросвязи и радиотехнике/ А. В. Брунченко, Ю.Г. Бутыльский, Л. М. Гольденберг и др.: под ред. Л. М. Гольденберга.

- M.: радио и связь, 1982. - 224 с.

1. J. L. Ftanagan, С. H. Coker, L. R. Rabiner, R. W. Schafer, and N. Umeda, "Synthetic Voices for Computers," IEEE Spectrum, Vol. 7, No. 10, October 1970.-536 p.
2. MFC и Visual С++. Энциклопедия программиста: Пер. с англ./ Юджин Олафсен, Кенн Скрайбнер, К. Дэвид Уайт и др. - СПб.: ООО «ДиаСофтЮП», 2004. - 992с.
3. Шилдт Герберт. MFC: основы программирования. - К.: Издательская группа BHV, 1997. - 560 с.
4. Тихомиров Ю.В. OpenGL. Программирование трехмерной графики. -СПб.: БХВ-Петербург, 2002.-304 с.
5. OpenGL. Официальное руководство программиста: Пер. с англ. / Мейсон By, Джеки Нейдер, Том Девис, Дейв Шрайнер. - СПб: ООО «ДиаСофтЮП», 2002. - 592 с.

Приложение А. Элементы текстов программного кода

// recorddlh: header file class CRecordDlg;

class CRecDlgNotifyObj: public CWaveNotifyObj {public:

CRecDlgNotifyObjO; -CRecDlgNotifyObjO; void Attach(CRecordDlg\* pDlg) {m\_pDlg = pDlg;}

virtual void NewData(CWave \*pWave, CWaveBIock\* pBlock);

virtual void EndPlayback(CWave \*pWave);

CRecordDlg\* m\_pDlg;};

class CFilterDlg. public CDialog

{public. CWaveOutDevice\* m\_pOutDev;

CFilterDlg(CWnd\* pParent = NULL);

enum { IDD = IDD\_FILTERDLG };

protected:

virtual void DoDataExchange(CDataExchange\* pDX); protected:

afxmsg void OnFilter(); DECLAREMESSAGEMAPOJ; class CRecordDlg: public CDialog {public:

CRecordDlg(CWnd\* pParent = NULL); ~CRecordDlg();

void NewData(CWave \*pWave, CWaveBIock\* pBlock);

void EndPlayback(CWave \*pWave);

enum { IDD = IDD\_RECORDDLG };

CButtonmbtnOpenData;

CComboBox mcbSource;

CComboBox mcbFormat;

CStatic mwndFrame;

CButton mbtnPause;

CButton mbtnPlay;

CButton mbtnRecord;

CButton mbtnOK;

CButton mbtnStop;

CString mstrFormat;

CString m\_strSource;

public:

void UDdatdTJClientDC \*curDC>:

CFilterDIg\* m\_pFUter; CWave\* m\_pWave; private:

enum MODE {IDLE, SAMPLING,PLAYTNG,RECORDING,OPEN };

CVUMeter m\_VU;

int mJNumDevs;

WAVEINCAPS\* m\_pDevCaps;

CWavelnDevice mlnDev;

CWave mSampleWave;

CRecDlgNotifyObj m\_NotifyObj;

int mJMode;

void FillDeviceList();

void FillFormatListQ;

void SetMode(MODE m);

void BuildFormat(PCMWAVEFORMAT& fmt, DWORD dwFormat); protected:

virtual void DoDataExchange(CDataExchange\* pDX);

virtual BOOL OnlnitDialogO;

afxmsg void OnSelchangeFormat();

afxmsg void OnClickedPlay();

afxjmsg void OnClickedRecordO;

afxmsg void OnClickedStop();

afxmsg void OnSelchangeSource();

virtual void OnCancel();

virtual void OnOK();

afxmsg void OnDrawItem(int nlDCtl, LPDRAW1TEMSTRUCT IpDrawItemStruct); afxmsg void OnMeasureItem(int rJDCtl, LPMEASUREITEMSTRUCT IpMeasureltemStruct); afxmsg void OnClickedOpenData(); afxmsg void OnUpdate();

afxmsg void OnHScroll(UTNT nSBCode, UINT nPos, CScrollBar\* pScrollBar);

afx\_msg void OnGraphicQ;

afxmsg void OnHamming();

afxmsg void OnFilterQ;

afxmsg void OnFurjeQ;

afx\_msg void OnObrfurje();

afxmsg void OnClip();

DECLARE\_MESSAGE\_MAP()};

class CGraphicDlg: public CDialog

{public:

CWaveOutDevice\* mjOutDev; CGraphicDlg(CWnd\* pParent = NULL); enum { IDD = IDD GRAPHIC };

protected:

virtual void DoDataExchange(CDataExchange\* pDX); protected:

afxmsg void OnGraphicQ; DECLAREMESS AGE\_MAP()}; class CHammingDlg: public CDialog {public:

CWaveOutDevice\* m\_pOutDev; CHammingDlg(CWnd\* pParent = NULL); enum {IDD = IDD HAMMING }; protected:

virtual void DoDataExchange(CDataExchange\* pDX); protected:

afxmsg void OnHammingO; DECLAREMESSAGEMAPO); class CFurjeDlg. public CDialog {public:

CWaveOutDevice\* m\_pOutDev; CFurjeDlg(CWnd\* pParent = NULL); enum {IDD = IDDFURJE }; protected:

virtual void DoDataExchange(CDataExchange\* pDX); protected:

afxmsg void OnFurjeO; DECLAREMESSAGEMAPO}; class CObrFurjeDlg: public CDialog {public:

CWaveOutDevice\* m\_pOutDev; CObrFurjeDlg(CWnd\* pParent = NULL); enum { IDD = IDD OBRFURJE }; protected:

virtual void DoDataExchange(CDataExchange\* pDX); protected:

afxmsg void OnObrfurje(); DECLAREMESSAGEMAPO}; class CClipDlg. public CDialog {public:

CWaveOutDevice\* m\_pOutDev; CClipDlg(CWnd\* pParent = NULL); enum { IDD = EDDCLIP }; protected:

virtual void DoDataExchange(CDataExchange\* pDX);

ON\_BN\_CLICKED(IDC\_GRAPHIC, OnGraphic) ON\_BN\_CLICKED(IDC\_HAMMrNG, OnHamming) ON\_BN\_CLICKED(IDC\_FILTER OnFilter) ON\_BN\_CLICKED(IDC\_FURJE, OnFurje) ON\_BN\_CLICKED(IDC\_OBRFURJE, OnObrfurje) ON\_BN\_CLICKED(IDC\_CLIP, OnClip) END\_MESSAGE\_MAP() BOOL CRecordDlg::OnInitDialog() {CDialog::OnInitDialog(); ASSERT(mjpWave = NULL); CRect rcVU;

m\_wndFrame.GetWindowRect(&rcVU); ScreenToClient(&rcVU);

m\_VU.Create(nVU",WS\_CHILD | WS\_VISIBLE,rcVU,this,l);

m\_VU.SetValue(0> 0);

FillDeviceListQ;

FillFormatListO;

SetMode(SAMPLLNG);

return TRUE; }

void CRecordDlg::OnSelchangeFormatO

{OnClickedStopO;}

void CRecordDlg::OnClickedPIayO

{SetMode(PLAYING);

CClientDCdc(this);}

void CRecordDlg. OnClickedRecordO

{ SetMode(RECORDLNG);}

void CRecordDlg. OnClickedStopO

{SetMode(S AMPLING);}

void CRecordDlg::OnSelchangeSource()

{OnClickedStopO;

FillFormatListO;}

void CRecordDlg::OnCancelO

{ OnClickedStopO;

SetMode(IDLE);

mVU.DestroyWindowO;

if (m\_pWave) {delete m\_pWave;

m\_pWave = NULL;}

CDialog::OnCancel();}

void CRecordDlg. OnOKO

{ OnClickedStopO;

SetMode(IDLE);

mVU.DestroyWindowO;

CDialog::OnOK();}

void CRecordDlg::FillDeviceList()

{miNumDevs = wavelnGetNumDevsO;

if (miNumDevs == 0) {

AfxMessageBox("There are no suitable input devices");

EndDialog(IDCANCEL);

return;}

if (m\_pDevCaps) delete m\_pDevCaps;

m\_pDevCaps = new WAVEINCAPSfmiNumDevs];

m\_cbSource.ResetContent();

for (int i=0; KmiNumDevs; i++) {

waveInGetDevCaps(i,&m\_pDevCaps[i],sizeof(WAVEINCAPS)); m\_pDevCaps[i].wMid = i;

m\_cbSource.AddString((LPCSTR)&m\_pDevCaps[i]);}

mcbSource. SetCurSel(0);}

void CRecordDlg.:FillFormatList()

{ mcbFormat.ResetContentO;

int iSel = m\_cbSource.GetCurSel();

if(iSel = CBERR) return;

WAVEINCAPS\* pCaps = (WAVEINCAPS\*) m\_cbSource.GetItemData(iSel);

ASSERT(pCaps);

DWORD dwMask = 0x00000001;

for(inti=0; i<12; i++) {

if (pCaps->dwFormats & dwMask) {

m\_cbFormat.AddString((LPCSTR)dwMask);}

dwMask = dwMask « 1; }

mcbFormat. SetCurSel(O);}

void CRecordDlg.OnDrawItem(int nlDCtl, LPDRAWITEMSTRUCT pDI) {char\* pszText = NULL; switch (nlDCtl) { case IDC\_SOURCE: {

WAVEINCAPS\* pCaps = (WAVEINCAPS\*)(pDI->itemData); pszText = pCaps->szPname; } break;

case IDC FORMAT: { switch(pDI->itemData) { case WAVE\_FORMAT\_1M08: pszText = "11.025 kHz, 8 bit, mono"; break;

case WAVE\_FORMAT\_1S08: pszText = "11.025 kHz, 8 bit, stereo"; break;

case WAVE\_FORMAT\_lM16: pszText = "11.025 kHz, 16 bit, mono"; break;

case WAVE\_F0RMAT\_1S16. pszText = "22.05 kHz, 16 bit, stereo"; break;

case WAVE\_FORMAT\_2M08: pszText = "22.05 kHz, 8 bit, mono"; break;

case WAVE\_FORMAT\_2S08: pszText = "22.05 kHz, 8 bit, stereo"; break;

case WAVE\_FORMAT\_2M16: pszText = "22.05 kHz, 16 bit, mono"; break;

case WAVEJFORMATJ2S16. pszText = "22.05 kHz, 16 bit, stereo"; break;

case WAVEJ?ORMAT\_4M08: pszText = "44.1 kHz, 8 bit, mono"; break;

case WAVE\_FORMAT\_4S08: pszText = "44.1 kHz, 8 bit, stereo"; break;

case WAVE\_FORMAT\_4M16: pszText = "44.1 kHz, 16 bit, mono"; break;

case WAVE\_FORMAT\_4S16:

pszText = "44.1 kHz, 16 bit, stereo";

break; default:

pszText = "Unknown";

break;} } break; default:

break;} if (IpszText) return;

: ^rawTextCpDI^hDC^szText.-L&CpDI^rcItem), DTLEFT | DTVCENTER); if (pDI->itemState & ODSSELECTED) {

::InvertRect(pDI->hDC, &(pDI->rcItem));} if (PDI->itemState & ODS\_FOCUS) {

::DrawFocusRect(pDI->hDC, &(pDI->rdtem)); }}

void CRecordDlg::OnMeasureItem(int ruDCtl, LPMEASUREITEMSTRUCT lpMeasureltemStruct) { CClientDC dc (this);

TEXTMETRIC tm;

dc. GetTextMetrics(&tm);

lpMeasureItemStruct->itemHeight = tm.tmHeight;} void CRecordDlg::SetMode(MODE m) { CClientDC dc(this); if (m = miMode) return; if (m\_pWave) m\_pWave->Stop(); switch (miMode) {case SAMPLING: case RECORDING: mlnDev.ResetO; m\_InDev.Close(); break; case PLAYING: break;

case OPEN:

break;}

miMode = IDLE;

switch (m) {case SAMPLING:

f case RECORDING.

{int iSel = m\_cbSource.GetCurSel(); if(iSel = CBERR) return;

WAVEINCAPS\* pCaps = (WAVEINCAPS\*) m\_cbSource.GetItemData(iSeI);

ASSERT(pCaps);

UINT uiBD = pCaps->wMid;

iSel = m\_cbFormat.GetCurSel();

if(iSel == CBERR) return;

DWORD dwFormat = mcbFormat.GetltemData(iSel);

ASSERT(dwFormat);

PCMWAVEFORMAT fmt;

BuildFormat(fmt, dwFormat);

if (!m\_InDev.Open(uiID, &fmt)) return;

if (m = SAMPLING) {

mSampleWave.DeleteAllO;

mSampleWave. Create(&frnt);

m\_SampleWave.Record(&m\_InDev, 1024,&m\_NotifyObj); } else if (m — RECORDING) { if (!m\_pWave) m\_pWave = new CWave; ASSERT(m\_pWave); mjpWave->Create(&fmt);

m\_pWave->Record(&mJnDev,4096,&m\_NotifyObj);}

case PLAYING, if (m\_pWave) m\_pWave->Play(NULL, &m\_NotifyObj); break;

case OPEN.

if (m\_pWave) m\_pWave->OpenData(NULL, &m\_NotifyObj,&dc); break;} miMode = m,

if ((mjMode = PLAYING) || (mjMode = RECORDING)! |(m\_iMode btnStop.EnableWindow(TRUE);

} else {m\_btnStop.EnableWindow(FALSE);}

if (mjMode = PLAYING) { m\_btnPlay.EnableWindow(FALSE); } else {mbtnPlay.EnableWindow(TRUE);}

if (m\_iMode = OPEN) {mbtnOpenData.EnableWindow(FALSE); } else {mbtnOpenData.EnableWindow(TRUE);} if (mjMode = RECORDING) {m\_btnRecord.EnableWindow(FALSE); } else {m\_btnRecord.EnableWindow(TRUE);}} void CRecordDlg::NewData(CWave \*pWave, CWaveBlock\* pBlock) { ASSERT(pWave); ASSERT(pBlock);

PCMWAVEFORMAT\* pwf = (PCMWAVEFORMAT\*) pWave->GetFormat(); ASSERT(pwf->wf.wFormatTag = WAVEFORMATPCM); int iCount = pBlock->GefNumSamples(); if (pwf->wBitsPerSample = 8) {

BYTE\* pData = (BYTE\*)pBlock->GetSamplesO; BYTE bMax - 0; while (iCount—) {

if (\*pData > bMax) bMax = \*pData; pData++;}

if (bMax < 128) { bMax = 0;} else {bMax -= 128;} m\_VU.SetValue(bMax « 8, bMax « 8); } else {ASSERT(sizeof(short int) = 2); short int\* pData = (short int\*) pBlock->GetSamples(); int iMax = 0;

while (iCount—) {if (\*pData > iMax) iMax = \*pData;

pData++;} m\_VU.SetValue(iMax, iMax);} if(m\_iMode!= RECORDING) { pWave->GetBlockList()->FreeAll();}} void CRecordDlg::EndPlayback(CWave \*pWave) { ASSERT(pWave); SetMode(SAMPLjrNG);}

void CRecordDlg::BuildFormat(PCMWAVEFORMAT& fmt, DWORD dwFormat) { ASSERT(dwFormat); fmt.wf.wFormatTag = WAVEFORMATPCM; switch (dwFormat) { case WAVE\_FORMAT\_1M08: fmt.wf.nChannels = 1; rmtwf.nSamplesPerSec = 11025; fmtwBitsPerSample = 8; break;

case WAVE\_F0RMAT\_1M16: fmt.wf.nChannels = 1; fmt.wf.nSamplesPerSec = 11025; fmt.wBitsPerSample = 16; break;

case WAVE\_FORMAT\_1S08: fmt.wf.nChannels = 2; fmt.wf.nSamplesPerSec = 11025; fmt.wBitsPerSample = 8; break;

case WAVE\_F0RMAT\_1S16: fmt.wf.nChannels = 2; fmtwf.nSamplesPerSec = 11025; fmt.wBitsPerSample = 16; break;

case WAVE\_FORMAT\_2M08: fmt.wf.nChannels = 1; fmt.wf.nSamplesPerSec = 22050; fmt.wBitsPerSample = 8; break;

case WAVE FORMAT 2M16 fmt.wf.nChannels = 1; fmt.wf.nSamplesPerSec = 22050; fmt.wBitsPerSample = 16; break;

case WAVE\_FORMAT\_2S08: fmt.wf.nChannels = 2; fmt.wf.nSamplesPerSec = 22050; fmt.wBitsPerSample = 8; break;

case WAVE\_FORMAT\_2S16: fmt.wfnChannels = 2; fmt.wf.nSamplesPerSec = 22050;

fmt.wBitsPerSatnple = 16; break;

case WAVE\_FORMAT\_4M08: fmt.wf.nChannels = 1; fmt.wf. nSamplesPerSec = 44100; fmt.wBitsPerSampIe = 8; break;

case WAVE\_FORMAT\_4M16: fmt.wf.nChannels = 1; fmt.wf.nSamplesPerSec = 44100; fmt.wBitsPerSampIe = 16; break;

case WAVE\_FORMAT\_4S08. fmt.wf.nChannels = 2; fmt.wf.nSamplesPerSec = 44100; fmt.wBitsPerSampIe = 8; break;

case WAVE\_FORMAT\_4S 16:

fmt.wf.nChannels = 2;

fmt.wf.nSamplesPerSec = 44100;

fmtwBitsPerSample = 16;

break; default:

ASSERT(O);

return;}

fmt.wf.nAvgBytesPerSec = fmt.wf.nSamplesPerSec;

fmt.wf.nBlockAlign = fmt.wBitsPerSampIe /8;} CRecDlgNotifyObj::CRecDlgNotifyObj() { m\_pDlg = NULL;} CRecDlgNotifyObj.:~CRecDlgNotifyObj() {}

void CRecDlgNotifyObj::NewData(CWave \*pWave,CWaveBlock\* pBlock) { ASSERT(m\_pDlg);

m\_pDlg->NewData(pWave, pBlock);} void CRecDlgNotifyObj::EndPlayback(CWave \*pWave) { ASSERT(m\_pDlg);

m\_pDlg->EndPlayback(pWave);}

void CRecordDlg.:OnClickedOpenData0

{ SetMode(OPEN);

CClientDC dc(this);} CFilterDlg::CFilterDlg(CWnd\* pParent /\*=NULL\*/): CDialog(CFilterDlg::IDD, pParent)

void CFilterDlg::DoDataExchange(CDataExchange\* pDX)

{ CDialog::DoDataExchange(pDX);}

BEGIN\_MESSAGE\_MAP(CFilterDlg, CDialog)

ONJBN\_CLICKED(IDC\_FILTER, OnFilter) ENDMES S AGEM AP() void CFiIterDlg::OnFilterO (CCIientDC dc(this);

dc.SelectObject(GetStockObject(WHITE\_PEN));

for(inti=0; i<110;i++)

{ dc.MoveTo(lO\*i,0);

dc.LineTo(10\*i,800);

dc.MoveTo(0,10\*i);

dc.LineTo(1100,10\*i);} dc.SelectObject(GetStockObject(BLACK\_PEN)); dc.MoveTo(0,200); dc.LineTo(l 100,200); m\_pOutDev->ShowFilter(&dc);}

void CRecordDlg::OnUpdate{) { CCIientDC dc(this); Update(&dc);}

void CRecordDlg::Update(CClientDC \*curDC) (CWnd \*win= curDC->GetWindowO; win->Invalidate(TRUE);}

int hspos=0; int curpos=0;

void CRecordDlg::OnHScroll(UTNT nSBCode, UINT nPos, CScrollBar\* pScrollBar) { curpos=hspos;

switch (nSBCode) { case SBJLEFT:.

if(hspos>0)

hspos--;

break;

case SBRIGHT: hspos++;

break;

case SBLINELEFT: if(hspos>0) hspos--;

case SBLINEPJGHT: hspos++;

break;

case SBTHUMBPOSITION: hspos = nPos;

break;

case SBTFTUMBTRACK: hspos = nPos; break;

case SBPAGELEFT: if(hspos>0) hspos—;

break;

case SBPAGERIGHT.

hspos++;

break;}

CREATESTRUCT cs;

cs.cx =::GetSystemMetrics(SM\_CXSCREEN);

SetScrollPos(SB\_HORZ,hspos,TRUE);

curpos=hspos-curpos;

Scroll Window(-curpos\* 10,0,NULL,NULL); CDialog::OnHScroll(nSBCode, nPos, pScrollBar);} CGraphicDlg::CGraphicDlg(CWnd\* pParent /\*=NULL\*/): CDialog(CGraphicDlg::IDD, pParent)

(}

void CGraphicDlg..DoDataExchange(CDataExchange\* pDX) { CDialog::DoDataExchange(pDX);} BEGLN\_MESSAGE\_MAP(CGraphicDlg, CDialog)

ON\_BN\_CLICKED(IDC\_GRAPHIC, OnGraphic) END MESSAGEJVIAP() void CRecordDlg::OnGraphicO (CGraphicDlg\* a; a = new CGraphicDlgO; if(a!= NULL)

{ BOOL ret = a->Create(IDD\_GRAPHIC,this); if(!ret)

AfxMessageBox("Error creating Dialog"); a->ShowWindow(S W\_SHOW);} else AfxMessageBox("Error Creating Dialog Object");} void CGraphicDlg..OnGraphicQ

{ CClientDC dc(this);

dc.SelectObject(GetStockObject(WHITE\_PEN));

for(int i=0; i<110;i++)

{ dc.MoveTo(10\*i,0);

dc.LineTo(10\*i,800);

dc.MoveTo(0,10\*i);

dc.LineTo(1100,10\*i);} dc.SelectObject(GetStockObject(BLACK\_PEN)); dc.MoveTo(0,200); dc.LineTo(1100,200); m\_pOutDev->Graphic(&dc);} CHammingDlg:.CHammingDlg(CWnd\* pParent /\*=NULL\*/): CDialog(CHammingDlg::IDD, pParent)

0

void CHammingDlg.:DoDataExchange(CDataExchange\* pDX)

{ CDialog.:DoDataExchange(pDX);}

BEGIN\_MESSAGE\_MAP(CHarnmingDlg, CDialog)

ON\_BN\_CLICKED(IDC\_HAMMING, OnHamming) END\_MESSAGE\_MAP() void CRecordDlg::OnHamming() {CHammingDlg\* a; a = new CHammingDlgO; if(a!=NULL)

{ BOOL ret = a->Create(roDJHAMMING,this);

ifTJret) ArxMessageBox("Error creating Dialog");

a->ShowWindow(SW\_SHOW); } else AfxMessageBox("Error Creating Dialog Object");}

void CHammingDlg::OnHamming() (CClientDC dc(this);

dc.SelectObject(GetStockObject(WHITE\_PEN));

for(inti=0; i<110;i-H-)

{ dc.MoveTo(10\*i,0);

dc.LineTo(10\*i,800); dc.MoveTo(0,10\*i); dc.LineTo(l 100,10\*i); } dc.SelectObject(GetStockObject(BLACK\_PEN)); dc.MoveTo(0,250); dc.LineTo(l 100,250); m\_pOutDev->ShowHamming(&dc);} void CRecordDlg::OnFilter() {CFilterDlg\* a;

a = new CFilterDlg(); if(a!=NULL)

{ BOOL ret = a->Create(TDDJFILTER,this);

if(!ret) AfxMessageBox("Error creating Dialog");

a->ShowWindow(SW\_SHOW);} else

AfxMessageBox("Error Creating Dialog Object");} CFurjeDlg::CFurjeDlg(CWnd\* pParent /\*=NULL\*/): CDialog(CFurjeDlg::TDD, pParent)

{}

void CFurjeDlg::DoDataExchange(CDataExchange\* pDX) { CDialog::DoDataExchange(pDX);} BEGIN\_MESSAGE\_MAP(CFurjeDlg, CDialog)

ON\_BN\_CLICJQED(IDC\_FURJE, OnFurje) ENDMESSAGEMAPQ

void CRecordDlg::OnFurje() {CFurjeDlg\* a; a = new CFurjeDlgQ; if(a!=NULL)

{ BOOL ret = a->Create(lDD\_FURJE,this); if(!ret) AfxMessageBox("Error creating Dialog");

a->ShowWindow(SW\_SHOW); } else AfxMessageBox("Error Creating Dialog Object");} void CFurjeDlg::OnFurje() {CClientDC dc(this);

dc.SelectObject(GetStockObjectOVHITE\_PEN));

for(inti=0; i<110;i++)

{ dc.MoveTo(10\*i,0);

dc.LineTo(10\*i,800);

dc.MoveTo(0,10\*i);

dc.LineTo(1100,10\*i); } dc.SelectObject(GetStockObject(BLACK\_PEN)); dc.MoveTo(0,250); dc.LineTo(l 100,250); m jpOutDev->ShowFurje(&dc);} CObrFurjeDlg::CObrFurjeDlg(CWnd\* pParent /\*=NULL\*/): CDialog(CObrFurjeDlg::IDD, pParent)

{}

void CObrFurjeDlg.:DoDataExchange(CDataExchange\* pDX)

{ CDialog::DoDataExchange(pDX);}

BEGINMESS AGEM AP(CObrFurj eDlg, CDialog)

ON\_BN\_CLICBCED(IDC\_OBRJFURJE, OnObrfurje) END\_MESSAGE\_MAP() void CRecordT>lg::OnObrfurje() {CObrFurjeDlg\* a; a = new CObrFurjeDlg(); if(a!=NULL)

{ BOOL ret = a->Create(IDD\_OBRFURJE,this);

if(!ret) AfxMessageBox("Error creating Dialog");

a->ShowWindow(SW\_SHOW); } else

AfkMessageBox("Error Creating Dialog Object");} void CObrFurjeDlg::OnObrfurje() {CClientDC dc(this);

dc.SelectObject(GetStockObject(WHITE\_PEN));

for(int i=0; i<l 10;i++)

{ dc.MoveTo(10\*i,0);

dc.LineTo(10\*i,800);

dc.MoveTo(0,10\*i);

dc.LineTo(1100,10\*i); } dc.SelectObject(GetStockObject(BLACK\_PEN)); dc.MoveTo(0,250); dc.LineTo(l 100,250); m\_pOutDev->ShowObrFurje(&dc);} CClipDlg::CClipDlg(CWnd\* pParent /\*=NULL\*/): CDialog(CClipDlg::IDD, pParent)

{}

void CClipDlg..DoDataExchange(CDataExchange\* pDX) { CDialog..DoDataExchange(pDX);} BEGIN\_MESSAGE\_MAP(CClipDlg, CDialog)

ON\_BN\_CLICKED(IDC\_CLIP, OnClip) ENDMESSAGEMAPQ void CRecordDlg::OnClip() {CClipDlg\* a; a = new CClipDlgQ; if(a!=NULL)

{ BOOL ret = a->CreateODD\_CLIP,this); itXIret) AfxMessageBox("Error creating Dialog"); a->ShowWindow(SW\_SHOW); } else AfxMessageBox("Error Creating Dialog Object");} void CClipDlg::OnClip() {CClientDC dc(this);

dc.SelectObject(GetStockObject(WHITE\_PEN));

for(inti=0;i<llO;i++)

{ dc.MoveTo(10\*L0);

dc.LineTo(10\*i,800);

dc.MoveTo(0,10\*i);

dc.LineTo(1100,10\*i); } dc.SelectObject(GetStockObject(BLACK\_PEN)); dc.MoveTo(0,100); dc.LineTo(l 100,100); m\_\_pOutDev->ShowClip(&dc);}

// animate.h: header file #include <mmsystem.h> #include "OutDialog.h" /Mnclude "recorddl.h" #include "resource.h" #include <complex> using namespace std; class CDIB. public CObject { DECLARESERIAL(CDIB) public: CD1B0; ~CDIB();

BITMAPFNFO \*GetBitmapInfoAddress()

{return m\_pBMI;} void \*GetBitsAddress()

{return m\_pBits;} RGBQUAD \*GetClrTabAddress()

{return (LPRGBQUAD)(((BYTE \*)(m\_pBMI)) + sizeof(BITMAPINFOHEADER));} int GetNumClrEntries();

BOOL Create(int width, int height, CPalette\* pPal = NULL); BOOL Create(int width, int height,

HP ALETTE hPal);

BOOL Create(BITMAPrNFO\* pBMI, BYTE\* pBits, BOOL bOwnBits = FALSE); BOOL Create(BITMAPINFO\* pPackedDTB); void \*GetPixelAddress(int x, int y); int GetBitsPerPixelO

{return m\_pBMI->bmiHeader.biBitCount;} virtual BOOL Load(const char\* pszFileName = NULL); virtual BOOL Load(CFile\* fp); virtual BOOL Load(WORD wResid); virtual BOOL Save(const char\* pszFileName = NULL); virtual BOOL Save(CFile \*fp);

virtual void Serialize(CArchive& ar); virtual void Draw(CDC \*pDC, int x, int y); virtual void DrawfHDC hDC, int x, int y); virtual int GetWidthO (return DibWidth();} virtual int GetStorageWidth() (return StorageWidthQ;} virtual int GetHeight() (return DibHeight();} virtual BOOL MapColorsToPalette(CPalette \*pPal); virtual BOOL MapColorsToPalette(HPALETTE hPal); virtual void GetRect(RECT\* pRect);

virtual void CopyBits(CDIB\* pDIB, int xd, int yd,int w, int h, int xs, int ys,COLORREF

clrTrans = OxFFFFFFFF);

virtual void StretchBits(CDIB\* pDIB, int xd, int yd.int wd, int hd, int xs, int ys, int ws,

int hs,COLORREF clrTrans = OxFFFFFFFF);

virtual void DoubleBits(CDIB\* pDIB, int xd, int yd,int xs, int ys, int ws, int

hs,COLORREF clrTrans = OxFFFFFFFF);

BOOL SetColorTable(CPalette\* pPal); static BOOL IsWinDffi(BITMAPrNFOHEADER\* pBIH); static int NumDIBColorEntries(BITMAPINFO\* pBmpInfo); protected:

BITMAPINFO \*m\_pBMI; BYTE \*m\_pBits; BOOL mbMyBits; private:

int DibHeightO

(return m\_pBMI->bmiHeader.biHeight;} int DibWidthQ

(return m\_pBMI->bmiHeader.biWidth;} int StorageWidthO

(return (((m\_pBMI->bmiHeader.biWidth \* m\_pBMI->bmiHeader.biBitCount)}}; class CDIBPal. public CPalette (public:

CDIBPalO;

~CDIBPal();

BOOL Create(CDIB \*pDIB);

BOOL Create(BITMAPINFO\* pBMI);

BOOL Create(RGBQUAD\* pRGB, int iClrs);

void Draw(CDC \*pDC, CRect \*pRect, BOOL bBkgnd = FALSE);

void Draw(HDC hDC, RECT\* pRect, BOOL bBkgnd = FALSE);

int GetNumColorsQ;

BOOL SetSysPalColors();

BOOL Load(const char\* pszFileName = NULL);

BOOL Load(CFile\* fp);

BOOL Load(UINT hFile);

BOOL Load(HMMIO hmmio);

BOOL Save(const char\* pszFileName = NULL);

BOOL Save(CFile\* fp);

BOOL SavefUTNT hFile);

BOOL Save(HMMIO hmmio);

BOOL CreateWash();

BOOL CreateSystemColorPaletteO;}; typedef HDC (WLNGAPI WinGCreateDCProc)(void); typedef HBITMAP (WLNGAPI WinGCreateBitmapProc)(HDC WinGDC,

const BITMAPINFO\* pHeader, void\*\* ppBits);

typedef UINT (WLNGAPI WinGSetDIBColorTableProc)(HDC WinGDC,

UINT Startlndex,

UINT NumberOfEntries,

const RGBQUAD\* pCoIors);

typedef BOOL (WLNGAPI WinGBitBlfProc)(HDC hdcDest, int nXOriginDest,int

nYOriginDest, int nWidthDest,int nHeightDest,HDC hdcSrc, int nXOriginSrc,

int nYOriginSrc);

typedef BOOL (WINGAPI WinGStretchBltProcXHDC hdcDest, int

nXOriginDest,int nYOriginDest,int nWidthDest,int nHeightDest, HDC hdcSrc,irt

nXOriginSrc, int nYOriginSrc, int nWidthSrc, int nHeightSrc);

class CDIBSurface: public CDIB { DECLARE\_SERIAL(CDrBSurface) public: CDIBSurfaceQ; -CDIBSurfaceO; public:

CDC\* GetDCO {return &m\_dcBuffer;} HDC GetHDCQ {return (HDC) m\_dcBuffer.GetSafeHdc();} public:

BOOL Create(int cx, int cy, CPalette\* pPal = NULL); BOOL Create(int width, int height, HP ALETTE hPal); void BitBlt(CDC\* pDC, int xd, int yd, int w, int h,

int xs, int ys); void BitBlt(HDC hDC, int xd, int yd, int w, int h,

int xs, int ys);

void StretchBlt(CDC\* pDC, int xd, int yd, int wd, int hd,

int xs, int ys, int ws, int hs); void Draw(CDC\* pDC); void Draw(HDC hDC, int x, int y); void SefPalettelCPalette\* pPal);

BOOL Load(const char\* pszFileName = NULL);

BOOL MapColorsToPalette(CPalette \*pPal);

BOOL MapColorsToPalette(HPALETTE hPal); protected:

CPalette\* m\_pPal;

HP ALETTE m\_hPal;

CDC m\_dcBuffer; private:

static BOOL mbUseCreateDIBSection;

static BOOL mbUseWinG;

static WinGCreateDCProc\* m\_pWinGCDCProc;

static WinGCreateBitmapProc\* m\_pWinGCBProc;

static WinGSetDIBColorTableProc\* m\_pWinGSDCTProc;

static WinGBitBltProc\* mjpWinGBBProe;

static WinGStretchBltProc\* m\_pWinGSBProc;

HBITMAP mhbmBufFer;

HBITMAP mhbmOld;

static BOOL CDIBSurface::TestPlatform();}; class COSBView: public CScrollView {protected:

COSBViewQ;

DECLAREDYNCREATE(COSBView) public:

CDocument\* GetDocumentO; CDIB\* GetDffiO {return m\_pDIBSurf;} CDIBPal\* GetPaletteQ {return m\_pPal;} CDC\* GetBufferDCO; public:

BOOL Create(CDIB\* pDIB); BOOL Create(int w, int h, CPalette\* pPal); void Draw(CRect\* pClipRect = NULL); virtual void RenderfCRect\* pClipRect = NULL) {return;} void AddDirtyRegion(CRect\* pRect); void RenderAndDrawDirtyListO; public: virtual -COSBViewQ; virtual void OnDraw(CDC\* pDC); virtual void OnInitialUpdate(); virtual void OnUpdate{CView\* pSender,

LP ARAM Hint,

CObject\* pHint);

virtual void AssertValid() const;

virtual void Dump(CDumpContext& dc) const; protected:

CDIBSurface\* mjpDIBSurf;

CDffiPal\* m\_pPal; private:

CObList mDirtyList;

void EmptyDirtyList(); protected:

afxmsg void OnPaletteChanged(CWnd\* pFocusWnd);

afxmsg BOOL OnQueryNewPalette();

afxmsg BOOL OnEraseBkgnd(CDC\* pDC);

DECLAREMESSAGEMAPO}; inline CDocument\* COSBView::GetDocument()

{ return (CDocument\*) m\_pDocument; } class CSprite;

class CSpriteNotifyObj: public CObject {public:

enum CHANGETYPE { ZORDER =0x0001, POSITION =0x0002, IMAGE 0x0004 };

public:

virtual void Change(CSprite \*pSprite,CHANGETYPE change, CRect\*

pRectl = NULL, CRect\* pRect2 = NULL) = 0;};

class CSprite: public CDBB { DECLARE\_SERIAL(CSprite) public:

CSprite();

~CSprite();

virtual int GetXQ { return m\_x;} // get x

virtual int GetY() { return m\_y;} // get y

virtual int GetZ() { return m\_z;} // get z order

virtual void Render(CDIB\* pDIB, CRect\* pClipRect = NULL);

virtual BOOL HitTest(CPoint point);

virtual void SetPosition(int x, int y);

virtual void SetZ(int z);

virtual void SetNotificationObject(CSpriteNotifyObj\* pNO)

{m\_pNotifyObj = pNO;} virtual void Serialize(CArchive& ar); virtual BOOL Load(CFile \*fp); virtual BOOL Load(char \*pszFileName = NULL); virtual BOOL Load(WORD wResid);

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Формат | Зона | Поз. | | Позначення | | | | Назва | | | | | Кільк. | Примітка | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  | 1 | | ГЮИК. 505330.002 ПЗ | | | | Текстові документи | | | | |  |  | |
|  |  |  | |  | | | | Пояснювальна записка | | | | | 1 | \_\_\_ | |
|  |  |  | |  | | | |  | | | | |  | арк. ф.А.4 | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | | Графічні документи | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | | Елементи текстів програмного коду | | | | |  |  | |
|  |  |  | |  | | | | файл формату \*.doc | | | | | 1 | \_\_ арк. ф. | |
|  |  |  | |  | | | |  | | | | |  | А4 | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | | Демонстраційний графічний матеріал | | | | |  |  | |
|  |  |  | |  | | | | у вигляді презентації, файл формату | | | | | 1 | \_\_ арк. ф. | |
|  |  |  | |  | | | | \*.ppt | | | | |  | А4 | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  |  | |  | | | |  | | | | |  |  | |
|  |  | |  | |  |  | ГЮИК. 505330.002 ВД | | | | | | | | |
|  |  | |  | |  |  |
|  |  | | № докум | | Підп. | Дата |
| Виконав | | |  | |  |  | Голосове керування тривимірними моделями функціонування ПР  відомість дипломної роботи | | Літера | | | Аркушш | | | Аркушів |
| Перевір. | | |  | |  |  |  |  |  |  | | | 1 |
| Н.контр. | | |  | |  |  | ХНУРЕ  КАФЕДРА ТАВР | | | | | | |
|  | | |  | |  |  |
| Затв. | | | Невлюдов | |  |  |