МИНИСТЕРСТВО ОБЩЕГО И ПРОФЕССИОНАЛЬНОГО

ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

КЕМЕРОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Кафедра теоретической физики

ПОПОВ РУСЛАН АНАТОЛЬЕВИЧ

СОЗДАНИЕ СИСТЕМЫ РАСПОЗНАВАНИЯ РЕЧИ НАЧАЛЬНОГО УРОВНЯ ГОЛОСОВОЙ ТЕЛЕФОННЫЙ СПРАВОЧНИК

/ Дипломная работа /

Научный руководитель:
к.ф.-м.н., доцент
_____ М. Л. Золотарев

Работа допущена к защите
«_____» _____ 2001 г.
Зав. кафедрой: д.ф.-м.н.
профессор
_____ А. С. Поплавной

Работа защищена
«_____» _____ 2001 г.
с оценкой _____
Председатель ГАК _____
Члены ГАК _____
_____
_____
_____

Кемерово 2001

Содержание

Введение

Внедрение информационных компьютерных технологий в разнообразные сферы человеческой деятельности требует создания новых интерфейсов, поддерживающих речевой диалог "пользователь-компьютер". Широкомасштабные и дорогостоящие работы, проведенные в 70-х - 90-х годах в США (проект ARPA), СССР (проект института Автоматики и телемеханики), Японии, странах Западной Европы привели к созданию компьютерных систем, работающих с ограниченным количеством слов или набором определенных фраз. Проведенный библиографический поиск и анализ информации в INTERNET показал, что в настоящее время многие ведущие компании усиливают работу в этом направлении.

В настоящее время обучение десятипальцевому слепому методу набора текста на клавиатуре является редким явлением в нашей стране, в отличие от США. Внедрение систем распознавания речи позволит упростить взаимодействие пользователя с компьютером. К сожалению, большинство систем распознавания речи ориентированы на англоязычного пользователя. Разработка такой системы для русского языка является трудоемкой задачей из-за специфичных особенностей русской речи.

Целью данной дипломной работы является изучение текущего состояния области распознавания речи; создание компонента распознавания речевых команд, который можно будет применять при создании приложений, управляемых с помощью голоса. Для практической демонстрации созданного компонента необходимо создать демонстрационное приложение.

В разделе "История распознавания речи" проведен краткий обзор программных средств распознавания речи, методов цифровой обработки сигналов и методов динамического программирования.

В разделе "Теория распознавания речи" даны теоретические аспекты алгоритмов, применяемых в данной работе: преобразование Фурье, перекрывающий анализ с использованием оконной функции, метод динамического искажения времени.

В разделе "Разработка компонента распознавания речи" дано краткое описание системы быстрой разработки приложений Borland C++ Builder 4. Приведено описание свойств, методов и событий компонента распознавания речи.

В разделе "Описание программ" дано описание демонстрационных программ, которые были созданы в период разработки компонента. Первая программа предназначена для изучения работы компонента, вторая — является демонстрационным однопользовательским телефонным справочником, управляемым голосом.

В разделе "Заключение" подводятся итоги проведенной работы и намечаются последующие шаги, необходимые для улучшения точности и возможностей системы распознавания речи.

Разработанные программы и их исходные тексты находятся в приложении на дискете.

История распознавания речи

Введение

На первый взгляд, идея распознавания речи проста: говорите со своим компьютером, который преобразует ваши слова в текст или команды. Но представьте себе встретившиеся трудности, когда исследователи впервые задумались о проектировании компьютерной системы, которая могла бы точно понимать человеческую речь. Как объяснить компьютеру, когда одно слово уже закончилось, а другое началось в произносимом предложении? Возьмите простые слова одинаково произносимые на английском "two", "too" и "to" или фразы на русском "он же ребенок", "он жеребенок". Человеческое ухо не может различить их. Как может компьютер сообразить, что конкретно вы сказали? Это были первые проблемы, встреченные исследователями с тех пор как они начали работать над распознаванием речи с конца 1950-х.

Окончательной целью большинства исследований в области распознавания речи являются независимые от диктора системы распознавания слитной речи, т.е. системы, которые бы могли понять любого человека и могли бы распознать каждое слово обычной речи. Некоторые исследователи в области распознавания речи изучали и продолжают изучать принципы обработки речи мозгом. Они изучали как сила различных звуковых частот в слове или фразе меняются со временем. Оттуда, они надеются найти правила, следуя которым, можно было бы сымитировать процесс, посредством которого человеческий мозг распознает речь.

Обзор программных средств

В настоящее время мы по-прежнему далеки от возможности ведения разумного диалога с компьютером. Но уже появились средства распознавания речи для персональных компьютеров (далее ПК), которые достигли уровня развития, достаточного для изменения способа взаимодействия человека и машины.

Уже почти пятнадцать лет в продаже имеются пакеты программ распознавания речи для ПК, однако до последнего времени они оставались лишь забавной игрушкой, требующей больших затрат на аппаратуру. За последние пять лет коммерческие изделия претерпели бурное развитие, и теперь такие пакеты реализованы на стандартных платформах без привлечения дополнительных дорогостоящих аппаратных средств.

Простейшие из современных изделий, способных реагировать на слова, позволяют управлять выполнением стандартных прикладных программ в системах MS Windows и Linux посредством устных команд (используемых вместо мыши и клавиатуры). Более сложные системы, имеющие словари большого объема, позволяют осуществлять устный ввод текста или служить основой для ваших собственных прикладных систем распознавания речи.

Три основных типа.

Все программные изделия для распознавания речи подразделяются на три основных типа: речевые навигаторы, средства речевого ввода текста и средства разработки. Среди изделий этих трех типов программы-навигаторы наиболее многочисленны.

Программы-навигаторы для ПК, которые поставляются уже более десяти лет, позволяют, отдавая устные команды, управлять работой системы, например, запуском и исполнением прикладных про-

грамм. Так и в компьютерах, работающих под управлением операционной системы Windows, можно использовать управление голосом для работы с отдельными программами и группами программ, для копирования или перемещения файлов и т. д. Ярким примером служат программы IBM ViaVoice[1] и Microsoft Voice[2]. Для Linux применяют программу CVoiceControl[3], которая распространяется в исходных текстах.

Средства речевого ввода, появившиеся на рынке в 1993 г., позволяют с помощью голоса создавать текстовые документы, вводить числа в электронные таблицы и многое другое. Поскольку системы речевого ввода текста должны "помнить" намного больше слов, чем навигаторы, которым требуется распознать лишь команды меню, продвижение этих систем на рынок происходит медленнее. Действительно, задача распознавания речи при диктовке настолько сложна, что предъявляет серьезные аппаратные требования к компьютеру. На маломощных компьютерах, разработчики таких пакетов требуют обязательного использования плат специализированных сопроцессоров. Примером этой группы систем диктовки является комплекс Dragon Dictate[4] компании Dragon Systems [3] для MS Windows и SPHINX[5] для Linux. Имеется локализованный вариант программы Dragon Dictate — Горыныч.

Средства разработки позволяют создавать как универсальные, так и специализированные прикладные программы, в которых используются методы распознавания речи. В основном эти системы

---

[1] Дополнительная информация доступна на сайте http://www.ibm.com

[2] Дополнительная информация доступна на сайте http://www.microsoft.com

[3] Дополнительная информация доступна на сайте http://www.freshmeat.net

[4] Дополнительная информация доступна на сайте http://www.dragon.com

[5] Дополнительная информация доступна на сайте http://www.speech.cs.cmu.edu/sphinx/

ориентируются на программистов, создающих программы на языках C++ и MS Visual Basic, и содержат базовые программы распознавания, а также интерфейсы прикладных программ (API) и необходимые для их использования библиотеки языков программирования. Пример — Microsoft Speech API[6]. К сожалению, бесплатных средств создания приложений с использованием распознавания русской речи пока не существует.

В настоящее время на рынок программных продуктов известными компаниями, как Microsoft, IBM, Dragon Systems, Lernout&Hauspie, Digalo, поставляются движки распознавания и синтеза речи. Под термином "движок" понимается набор программных средств, специально разработанных и оптимизированных для выполнения определенной операции. Данные движки могут работать со всеми стандартными европейскими языками: английский, немецкий, французский, итальянский, испанский и т.д. Поддержка русского языка пока еще довольно редкое явление.

[6] Дополнительная информация доступна на сайте http://www.microsoft.com/speech/

Методы обработки сигнала

В настоящее время для цифровой обработки сигнала применяются методы цифровых фильтров, линейного предсказания, преобразования Фурье и их комбинации (Рис. 1).

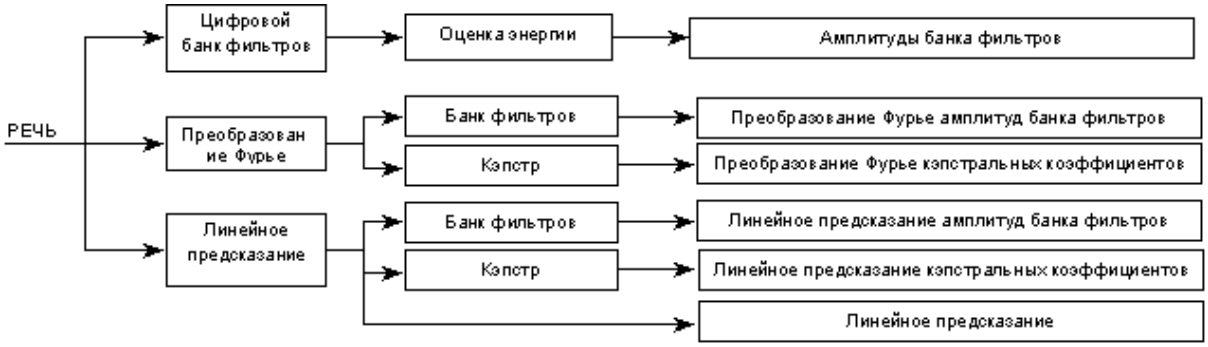

Рис. 1. Методы цифровой обработки.

Линейное предсказание

Линейная модель речеобразования была разработана Фантом в конце 1950-х, ее математическое обоснование и подробное изучение проведено Фантом и Фланганом на основе тщательно поставленных экспериментальных исследований. Голосовой тракт (Рис. 2) в этой модели представляет собой полюсный фильтр, входной сигнал e(t) которого - либо импульсная последовательность с периодом P для вокализованных звуков, либо случайный шум с равномерным спектром для невокализованных звуков.

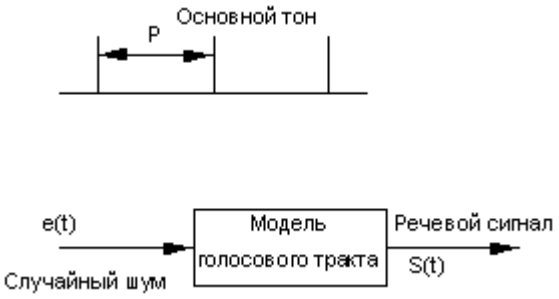

Рис. 2. Модель голосового тракта.

Сущность метода линейного предсказания заключается в том, что n-я выборка речевого сигнала может быть приблизительно предсказана линейной комбинацией предшествующих отсчетов сигнала:

$$S_n' = \sum_{i=1}^{m} a_i S_{n-i}$$

Здесь S_n' — предсказанное значение речевого сигнала; a_i — весовой коэффициент или коэффициент линейного предсказания; m — число коэффициентов линейного предсказания или порядок предсказания.

Описание теории линейного предсказания не входит в цель этой работы, подробности теории можно найти в [11,17,19,33].

Преобразование Фурье

Целью применения преобразования Фурье является перевод речевого сигнала из амплитудного представления в частотное. Необходимость перевода сигнала из одной формы в другую вытекает из того, что алгоритмы цифровой обработки сигналов работают с частотным представлением сигнала, а сам сигнал хранится и передается в амплитудном представлении.

Более подробно теория Фурье рассмотрена в разделе "Теория распознавания речи" и в [6,8,9,10,14,22,24].

Кэпстральный анализ

Целью кэпстрального анализа является выделение характеристик речевого сигнала для получения отклика вокального тракта после удаления частотных пульсаций из высокочастотной части спектра речевого сигнала. Для этого применяется фильтрование логарифма энергии сигнала с помощью инверсного преобразования Фурье; округление коэффициентов на определенных интервалах; прямого преобразования Фурье.

Описание теории кэпстрального анализа не входит в цель этой работы, подробности теории можно найти [24].

Методы динамического программирования

Динамическое искажение времени

Алгоритм динамического искажения времени является методикой эластичного сравнения вектора наблюдений с хранящимся шаблоном. Вектор наблюдений и шаблон лежат на соответствующих осях сетки (Рис. 7). Для каждой ячейки сетки высчитывается разность между соответствующими кадрами вектора наблюдений и шаблона.

Оптимальное выравнивание между вектором наблюдений и шаблоном показано маршрутом, проходящим по сетке.

Более подробно метод рассмотрен в разделе "Теория распознавания речи" и в [13,15,16,25].

Скрытые Марковские модели

Одно из основных направлений в распознавании речи состоит в представлении каждого слова одним или несколькими эталонами в пространстве измерений и вычислении расстояния от эталонов до неизвестной реализации речевого сигнала. В простейшем случае, когда отклонения реализаций некоторого слова от эталона порождаются случайным процессом с нормальным распределением, оптимальным является вычисление расстояний в евклидовой метрике, реализуемое, в частности, в виде коэффициента корреляции. Очень скоро выяснилось, что наряду со случайными отклонениями в речевых сигналах присутствуют и детерминированные компоненты искажений. Одним из таких неслучайных искажений является различие в длительности произнесения любого слова разными дикторами, причем это различие невозможно компенсировать линейным сжатием или растяжением оси времени. В конце 60-х годов для измерения степени сходства между реализацией и эталоном слова снова было предложено использовать динамическое программиро-

вание, которое позволяло отыскать минимум расстояния в заданной метрике с учетом нелинейных деформаций оси времени. Впоследствии метод динамического программирования был распространен на полунепрерывное по времени представление речевого сигнала в виде последовательности символов сегментов из конечного множества.

Применение динамического программирования позволило достичь весьма высокой (до 97-99%) надежности распознавания изолированных слов с настройкой на диктора для словарей объемом до 100-200 слов. Но этот метод обладает рядом принципиальных недостатков, затрудняющих или исключающих его применение для большинства практических задач. К числу этих недостатков относится зависимость надежности распознавания от типа микрофона и расстояния до него, необходимость обновления эталонов каждые несколько месяцев, большие вычислительные затраты, фактическая невозможность распознавания слитной речи и больших словарей, а также зависимость от диктора. Слабость метода динамического программирования заключается в его универсальности, т. е. в очень малом использовании особенностей речевого сигнала. В частности, для успешного распознавания нужно назначить меру сходства между сигналами, а она в методе динамического программирования не определяется, а задается извне.

Необходимость в распознавании слитной речи независимо от диктора, потребность в повышении надежности путем перехода к помехоустойчивому параметрическому представлению речевого сигнала заставили искать другие способы сравнения реализаций и эталонов. Тот факт, что речь предназначена для передачи, а следовательно, и для защиты информации, позволяет рассматривать ее как некоторый код, а речевой поток - как последовательность некоторых кодо-

вых посылок. Что является элементом этого кода - слоги, фонемы, артикуляторные движения или характерные сегменты, в данном случае безразлично. Значение имеет лишь то, что вероятность появления любого элемента кода зависит от некоторого числа предшествующих элементов. Таким образом, речь порождается Марковским источником, а речевой код есть случайный код.

Такая точка зрения весьма конструктивна для поиска способов распознавания речи. Прежде всего, она указывает на способ декодирования - для случайных кодов в настоящее время применяется только последовательное декодирование. В математическом смысле оно является специфической разновидностью динамического программирования, позволяя найти максимум сходства между эталонным кодовым словом и искаженной последовательностью элементов кода (символов). Например, алгоритм Витерби [21] есть просто вероятностный вариант динамического программирования. При таком подходе также выясняется характер действий и необходимая информация для исправления искажений в принятой последовательности символов. Очевидно, что возможно лишь три типа искажений: выпадение, вставка и замена символов. Соответственно нужна информация о вероятностях этих событий. Наконец, более определенным становится и понятие меры сходства - это функция правдоподобия, вычисленная тем или иным способом. Применение такого подхода к распознаванию речи, хотя и с настройкой на диктора, но для больших словарей в слитном произнесении, дало хорошие результаты: ошибка в распознавании слов составляет 0.1-0.9% для словаря в 1000 слов [2] и около 6% для словаря в 5000 слов, но для изолированного произнесения [1].

Параллельно с разработкой методов декодирования развивался метод скрытых Марковских моделей, позволяющий автоматизиро-

вать поиск вероятностных характеристик Марковских моделей в речи [3,18,21]. Скрытой Марковской моделью называется Марковский процесс, который не наблюдается непосредственно. Результаты действия этого процесса искажаются некоторым случайным процессом и лишь после этого становятся доступными наблюдению. Параметрами скрытой Марковской модели являются:

- возможные состояния процесса;
- вероятность перехода из одного состояния в другое;
- вероятность искажения наблюдаемого состояния.

Интерес к скрытым Марковским моделям возник после того, как была доказана сходимость простого итеративного алгоритма, позволяющего оценить параметры модели [4]. В настоящее время в этом методе достигается надежность распознавания изолированных слов независимо от диктора, сравнимая с надежностью распознавания при настройке на диктора [18,21]. Таким образом, последовательное декодирование при распознавании речи и приемы метода скрытых Марковских моделей при обучении адекватны природе речевого сигнала и создают основу для решения задачи распознавания слитной речи независимо от диктора.

Теория распознавания речи

Первичный речевой сигнал

Речь с физической точки зрения состоит из последовательности звуков речь с паузами между их группами [29,35,38]. При нормальном темпе речи паузы появляются между отрывками фраз, так как при этом слова произносятся слитно (хотя слух, как правило, воспринимает слова по отдельности). При замедленном темпе речи, например при диктовке, паузы могут делаться между словами и даже их частями. Предлоги, союзы звучат всегда слитно с последующим словом.

Один и тот же звук речи разные люди произносят по-разному, каждому человеку свойственна своя манера произнесения звуков речи. Произношение звуков речи зависит от ударения, соседних звуков и т. п. Но при всем многообразии в их произношении они являются физическими реализациями (произнесением) ограниченного числа обобщенных звуков речи, называемых фонемами. Фонема — это то, что человек хочет произнести, а звук речи — это то, что человек фактически произносит. Фонема по отношению к звуку речи играет ту же роль, что и образцовая буква по отношению к ее рукописной форме в конкретном написании.

В русском языке насчитывается 42 основные и 3 неопределенные фонемы.

Звуки речи делятся на звонкие и глухие. Звонкие звуки образуются с участием голосовых связок, в этом случае находящихся в напряженном состоянии. Под напором воздуха, идущего из легких, они периодически раздвигаются, в результате чего создается прерывистый поток воздуха. Импульсы потока воздуха, создаваемые голосовыми связками, с достаточной точностью могут считаться перио-

дическими. Соответствующий период повторения импульсов называют периодом основного тона голоса T_0. Обратную величину T_0, т. е. 1/ T_0, называют частотой основного тона. Если связки тонкие и сильно напряжены, то период получается коротким и частота основного тона высокой; для толстых, слабо напряженных связок частота основного тона получается низкой. Частота основного тона для всех голосов лежит в пределах 70..450 Гц. При произнесении речи частота основного тона непрерывно изменяется в соответствии с ударением и подчеркиванием звуков и слов, а также для проявления эмоций (вопрос, восклицание, удивление и т. д.). Изменение частоты основного тона называется интонацией. У каждого человека свой диапазон изменения основного тона (обычно он бывает немногим более октавы) и своя интонация. Последняя имеет большое значение для узнаваемости говорящего. Основной тон, интонация, устный почерк и тембр голоса служат для опознавания человека, и степень достоверности опознавания выше, чем по отпечаткам пальцев. Импульсы основного тона имеют пилообразную форму, и поэтому при их периодическом повторении получается дискретный спектр с большим числом гармоник (до 40), частоты которых кратны частоте основного тона. Огибающая спектра основного тона имеет спад в сторону высоких частот с крутизной около 6 дБ/окт, поэтому для мужского голоса уровень составляющих около 3000 Гц ниже их уровня около 100 Гц примерно на 30 дБ.

При произнесении глухих звуков, связки находятся в расслабленном состоянии, поток воздуха из легких свободно проходит в полость рта. Встречая на своем пути различные преграды в виде языка, зубов, губ, он образует завихрения, создающие шум со сплошным спектром.

Согласные, по способу образования, делятся на сонорные (**л**, **ль**, **р**, **рь**, **м**, **мь**, **н**, **нь**, **й**), щелевые (**ж**, **з**, **зь**, **в**, **вь**, **ш**, **с**, **сь**, **ф**, **фь**, **х**, **хь**), взрывные (**б**, **бь**, **д**, **дь**, **г**, **гь**, **п**, **пь**, **т**, **ть**, **к**, **кь**) и аффрикаты (**ц**, **ч** — комбинация глухих взрывных и щелевых). Гласных фонем всего шесть: **а**, **о**, **у**, **э**, **и**, **ы** (гласные **е**, **я**, **ё**, **ю** — составные из **й** или мягкого знака и гласных **э**, **а**, **о**, **у**).

По месту образования фонемы делятся на губные, зубные, небные, гортанные, передние и задние.

При произнесении звуков речи язык, губы, зубы, нижняя челюсть, голосовые связки должны находится для каждой фонемы в строго определенном положении или движении. Эти движения называют артикуляцией органов речи. При этом в речеобразующем тракте создаются резонансные полости, определенные для данной фонемы, а для слитного звучания фонем в речи — определенные переходы от одной формы тракта к другой.

При произнесении звуков речи через речевой тракт проходит или тональный импульсный сигнал, или шумовой, или тот и другой вместе. Речевой тракт представляет собой сложный акустический фильтр с рядом резонансов, создаваемых полостями рта, носа и носоглотки, т. е. с помощью артикуляционных органов речи. Вследствие этого равномерный тональный или шумовой спектр превращается в спектр с рядом максимумов и минимумов. Максимумы спектра называют формантами, а нулевые провалы — антиформантами. Для каждой фонемы огибающая спектра имеет индивидуальную и вполне определенную форму. При произнесении речи спектр ее непрерывно изменяется и образуются формантные переходы. Частотный диапазон речи находится в пределах 70..7000 Гц.

Звонкие звуки речи, особенно гласные, имеют высокий уровень интенсивности, глухие — самый низкий. При произнесении речи

громкость ее непрерывно изменяется. Особенно резко она изменяется при произнесении взрывных звуков речи. Динамический диапазон уровней речи находится в пределах 35..45 дБ. Гласные звуки речи имеют в среднем длительность около 0.15 с, согласные — около 0.08 (звук **п** — около 30 мс).

Звуки речи неодинаково информативны. Так, гласные звуки содержат малую информацию о смысле речи, а глухие согласные наиболее информативны (например, в слове "посылка" последовательность "о, ы, а" ничего не говорит, а "п, с, лк" дает почти однозначный ответ о смысле. Поэтому разборчивость речи снижается при действии шумов, в первую очередь из-за маскировки глухих звуков.

Известно, что для передачи одного и того же сообщения по телеграфу и по речевому тракту требуется различная пропускная способность тракта. Для телеграфного сообщения достаточна пропускная способность не более 100 бит/с, а для речевого — около 100000 бит/с (полоса равна 7000 Гц, динамический диапазон 42 дБ, т.е. требуется семизначных код, откуда имеем 2·7000·7=98000 бит/с), т. е. в 100 раз большая.

Образование звуков речи происходит путем подачи команд к мускулам артикуляционных органов речи от речевого центра мозга. Общий поток сообщений от него составляет в среднем не более 100 бит/с. Вся остальная информация в речевом сигнале называется сопутствующей.

Речевой сигнал представляет собой своего рода модулированную несущую. Его спектр $p(\omega) = E(\omega) \cdot F(\omega)$, где $E(\omega)$ — спектр генераторной функции, т. е. импульсов основного тона или шума; $F(\omega)$ — фильтровая функция речевого тракта — модулирующая кривая. Эта модуляция особая — спектральная. При ней несущая имеет широкопо-

лосный спектр, а в результате модуляции изменяется соотношение между частотными составляющими, т. е. изменяется форма огибающей спектра. Почти вся информация о звуках речи заключена в спектральной огибающей речи и ее временном изменении (частично информация о звуках речи заключена в переходах от тонального спектра к шумовому и обратно — по этим переходам узнают о смене звонких звуков на глухие и обратно). Все эти изменения происходят медленно (в темпе речи).

Для передачи смысла речи достаточно передавать сведения о форме огибающей спектра речи и ее временном изменении в темпе смены звуков речи, а также изменение основного тона речи и переходов тон-шум.

Рассмотренные закономерности построения речи формируют сложный многочастотный сигнал, который нужно должным образом обработать для выделения информационной части. Для этого применяют различные преобразования, например, Фурье. Кратко теория такого преобразования изложена в следующем разделе.

Преобразование Фурье

Введение

Линейные преобразования, особенно Фурье и Лапласа, широко используются для решения научных и инженерных задач. Преобразование Фурье используется в системах линейного анализа, при настройке антенн, в оптике, в моделировании случайных процессов, в теории вероятности, в квантовой физике и было успешно применено для распознавания речи. Преобразование Фурье, повсеместно применяемое разностороннее средство, используется во многих областях науки как математическое или физическое средство преобразования задачи для упрощения ее решения.

В сущности, преобразование Фурье разлагает на составные части или выделяет сигнал или функцию на синусоиды другой частоты, сумма которых составит оригинальный сигнал или функцию. Оно идентифицирует или различает синусоиды различных частот и их соответствующие амплитуды. Преобразование Фурье для функции f(x) задано как:

$$F(s) = \int_{-\infty}^{\infty} f(x)\exp(-i2\pi xs)dx$$

Применение такого же преобразования к функции F(s) дает такой результат:

$$f(w) = \int_{-\infty}^{\infty} F(s)\exp(-i2\pi ws)ds$$

Если функция f(x) является четной, т.е. f(x)=f(-x), тогда f(w)=f(x). Если функция f(x) является нечетной, т.е. f(x)=-f(-x), тогда f(w)=f(-x). Если же функция f(x) не является ни четной, ни нечетной, то почти всегда ее можно разделить на четную и нечетную части.

Для избежания неточностей, принято писать прямое преобразование Фурье и обратное, чтобы показать обратимость:

$$F(s) = \int_{-\infty}^{\infty} f(x)\exp(-i2\pi xs)dx$$

$$f(x) = \int_{-\infty}^{\infty} F(s)\exp(-i2\pi xs)ds$$

то есть

$$f(x) = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(x)\exp(-i2\pi xs)\right]\exp(-i2\pi xs)ds$$

до тех пор пока интеграл существует и любые пределы, обычно представленные умножением интегралов формы $\frac{1}{2}[f(x+)+f(x-)]$, конечны. Количество преобразований F(s) часто записывается, как $\tilde{f}(s)$, а преобразование Фурье описывается оператором $\hat{F}$.

Существуют функции, для которых преобразование Фурье не задано; тем не менее, большинство физических функций имеют преобразование Фурье. Другие функции могут работать с теорией Фурье с некоторыми ограничениями. Многие из общих теоретических функций действительно являются ограниченными случаями в теории Фурье.

Обычно функции и сигналы могут быть разделены на четные и нечетные части как показано далее:

$$f(x) = E(x) + O(x)$$

где

$$E(x) = \tfrac{1}{2}[f(x) + f(-x)]$$

$$O(x) = \tfrac{1}{2}[f(x) - f(-x)]$$

причем E(x) и O(x) в общем, являются комплексными функциями. В этом представлении преобразование Фурье для функции f(x) уменьшается до:

$$2\int_0^\infty E(x)\cos(2\pi xs)dx - 2i\int_0^\infty O(x)\sin(2\pi xs)dx$$

Это следует из того, что четная функция имеет четное преобразование, а нечетная функция — нечетное. Дополнительная симметрия характеристик показана на следующей таблице:

Таб. 1. Симметрия характеристик преобразования Фурье.

Функция	Преобразование
вещественная и четная	вещественная и четная
вещественная и нечетная	мнимая и нечетная
мнимая и четная	мнимая и четная
комплексная и четная	комплексная и четная
комплексная и нечетная	комплексная и нечетная
вещественная и асимметричная	комплексная и асимметричная
мнимая и асимметричная	комплексная и асимметричная
вещественная четная и мнимая нечетная	вещественная
вещественная нечетная и мнимая четная	мнимая
четная	четная
нечетная	нечетная

Важный вывод из Таб. 1 — это эрмитова функция, смысл которой в том, что вещественная часть четная, а мнимая — нечетная, т.е. f(x)=f*(-x). Преобразование Фурье для эрмитовой функции четное.

Преобразование косинуса для функции f(x) задано как:

$$F_c(s) = 2\int_0^\infty f(x)\cos(2\pi sx)dx$$

Эта запись эквивалентна преобразованию Фурье при условии, что функция f(x) четная. В общем случае, четная часть преобразования Фурье для функции f(x) является преобразованием косинуса для четной части f(x). Преобразование косинуса имеет обратную форму заданную как:

$$f(x) = 2\int_0^{\infty} F_c(s)\cos(2\pi sx)ds$$

Подобно этому, преобразование синуса задано как:

$$F_S(s) = 2\int_0^{\infty} f(x)\sin(2\pi sx)dx$$

Объединяя преобразования синуса и косинуса четной и нечетной частей функции f(x), получаем преобразование Фурье для все функции f(x):

$$\hat{F}f(x) = \hat{F}_C E(x) - i\hat{F}_S O(x)$$

где $\hat{F}$, $\hat{F}_C$ и $\hat{F}_S$ применяются для *i*-го элемента преобразования Фурье, преобразования синуса или косинуса соответственно, или:

$$F(s) = \tfrac{1}{2}F_C(s) - \tfrac{1}{2}F_S(s)$$

Так как преобразование Фурье F(s) является частотной областью представления функции f(x), s характеризует частоту выделенных косинусоид и синусоид и равно количеству циклов по величине x. Если функция или сигнал непериодические, то преобразование Фурье функции будет бесконечной функцией частоты.

Две области представления

Часто бывает полезно рассматривать функцию и ее преобразование как представления в двух областях. Также их называют представлениями функции и преобразования, но в большинстве статей их называют временным и частотным представлениями соответст-

венно. Операторы, выполненные в одном представлении, имеют соответствующие операторы в другом. Например, как будет показано ниже, операция свертки во временном представлении становится операцией умножения в частотном представлении, т.е. $f(x)\otimes g(x)\leftrightarrow F(s)\cdot G(s)$, есть и обратная функция: $F(s)\otimes G(s)\leftrightarrow f(x)\cdot g(x)$. Эти теоремы позволяют переходить от одного представления к другому для упрощенного выполнения операций и для получения других преимуществ.

Свойства преобразования Фурье

Масштабирование

Если $\hat{F}\{f(x)\}=F(s)$ и *a* является реальным числом (не нуль), то:

$$\hat{F}\{f(ax)\}=\int_{-\infty}^{\infty}f(ax)\exp(i2\pi sx)dx=|a|^{-1}\int_{-\infty}^{\infty}f(\beta)\exp\left(i2\pi s\frac{\beta}{a}\right)d\beta=|a|^{-1}F\left(\frac{s}{a}\right)$$

Из масштабирования по времени следует, что если ширина функции уменьшается в то время как высота остается постоянной, то преобразование Фурье этой функции будет шире и ниже. Если же ширина функции увеличивается, ее преобразование станет уже и выше.

Подобно, масштабирование по частоте задано как:

$$\hat{F}\left\{|a|^{-1}f\left(\frac{x}{a}\right)\right\}=F(as)$$

Сдвиг

Если $\hat{F}\{f(x)\}=F(s)$ и x_0 является вещественной константой, то:

$$\hat{F}\{f(x-x_0)\}=\int_{-\infty}^{\infty}f(x-x_0)\exp(i2\pi sx)dx=\int_{-\infty}^{\infty}f(\beta)\exp(i2\pi s(\beta+x_0))d\beta=$$

$$\exp(i2\pi x_0 s)\int_{-\infty}^{\infty}f(\beta)\exp(i2\pi s\beta)d\beta=F(s)\exp(i2\pi x_0 s)$$

Временной сдвиг указывает на то, что преобразование Фурье сдвигаемой функции — это исходная функция, умноженная на экспоненциальный показатель имеющий линейную фазу.

Подобно этому, частотный сдвиг указывает на то, что если функция F(s) сдвинута на константу s_0, то ее обратное преобразование умножено на $\exp(i2\pi x s_0)$.

$$\hat{F}\{f(x)\exp(i2\pi x s_0)\} = F(s - s_0)$$

Теорема свертки

Определим вышеупомянутую теорему временной свертки. Предположим, что $g(x) = f(x) \otimes h(x)$. Тогда получим, что $\hat{F}\{g(x)\} = G(s)$, $\hat{F}\{f(x)\} = F(s)$ и $\hat{F}\{h(x)\} = H(s)$,

$$G(s) = \hat{F}\{f(x) \otimes h(x)\} = \hat{F}\left\{\int_{-\infty}^{\infty} f(\beta)h(x-\beta)d\beta\right\} = \int_{-\infty}^{\infty}\left[\int_{-\infty}^{\infty} f(\beta)h(x-\beta)d\beta\right]\exp(-i2\pi sx)dx =$$

$$\int_{-\infty}^{\infty} f(\beta)\left[\int_{-\infty}^{\infty} h(x-\beta)\exp(-i2\pi sx)dx\right]d\beta = H(s)\int_{-\infty}^{\infty} f(\beta)\exp(-i2\pi s\beta)d\beta = F(s)H(s)$$

Этот чрезвычайно мощный результат показывает, что преобразование Фурье свертки задано как умножение индивидуальных преобразований, т.е.:

$$\hat{F}\{f(x) \otimes h(x)\} = F(s) \cdot H(s)$$

Используя тот же вывод, можно показать, что преобразование Фурье для умножения задано как свертка индивидуальных преобразований, т.е.:

$$\hat{F}\{f(x) \cdot h(x)\} = F(s) \otimes H(s)$$

Последняя формула — теорема частотной свертки.

Теорема корреляции

Корреляционный интеграл, как и интеграл свертки, важен для теоретических и практических применений. Корреляционный интеграл задан следующим образом:

$$h(x) = \int_{-\infty}^{\infty} f(u)g(x+u)du$$

и как для интеграла свертки он формирует пару преобразования Фурье, заданную как:

$$\hat{F}\{h(x)\} = F(s)G^*(s)$$

Последняя формула — теорема корреляции. Если f(x) и g(x) — одинаковые функции, вышеприведенный интеграл обычно называется автокорреляционной функцией или кросскорреляционной в противном случае. Пара преобразования Фурье задана как:

$$\hat{F}\left[\int_{-\infty}^{\infty} f(u)f(x+u)du\right] = |F|^2$$

Теорема Парсеваля

Теорема Парсеваля означает, что энергия сигнала представленная функцией h(t) не зависит от того, в каком представлении идет подсчет, т.е.:

$$\int_{-\infty}^{\infty} h^2(t)dt = \int_{-\infty}^{\infty} |H(f)|^2 df$$

Энергия спектра задана как:

$$P(f) = |H(f)|^2, \quad -\infty \le f \le +\infty$$

Теорема дискретизации

Узкополосный сигнал — это сигнал f(t), который не имеет спектральных компонентов за частотой B Гц, т.е., F(s)=0 для $|s| > 2\pi B$.

Теорема дискретизации означает, что узкополосный сигнал f(t) ограниченный на частоте B Гц может быть восстановлен без ошибок из отсчетов взятых на частоте R>2B отсчетов в секунду. Эта минимальная частота $F_s = 2B$ Гц, называется частотой Котельникова-Найквиста. Соответствующий шаг дискретизации, $T = \frac{1}{2}B$ (где t=nT) называется интервалом Найквиста. Если узкополосный сигнал f(t), ограниченный на частоте B Гц, был взят на частоте меньшей чем частота Котельникова-Найквиста, то такой сигнал называется испорченным.

Эффект наложения

Многие практические трудности встречаются во время восстановления сигнала из его значений. Теорема дискретизации предполагает, что сигнал узкополосный. На практике сигналы таковыми не являются. В результате, определение адекватной частоты дискретизации, которое не приведет к потери некоторой информации является довольно сложной задачей. Когда сигнал взят на недостаточной частоте, в его спектре есть перекрывающиеся концы; т.е. F(s) больше не имеет полной информации о спектре и больше не может восстановить f(t) из оцифрованного сигнала. В этом случае концы спектра не достигают нуля, но свертывается на сам спектр. Такая инверсия концов спектра называется спектральной сверткой или эффектом наложения.

Дискретное преобразование Фурье

Дискретное преобразование Фурье[7] [6,7,8,9,10,12,14,20,24] представляет собой вариант преобразования Фурье, реализованный для работы на компьютерах. Так как компьютеры работают с дискретными данными, числовое вычисление преобразования Фурье функции f(t) требует набор дискретных значений f_k. К тому же компьютер может рассчитать преобразование F(s) только для дискретного числа s, т.е. для дискретного числа значений F(r). Если f(kT) и $f(rs_0)$ являются k-ым и r-ым значениями f(t) и F(s) соответственно, а N_0 является числом значений сигнала на период T_0, то:

$$F_k = Tf(kT) = T_0 N_0^{-1} f(kT)$$

$$F_r = F(rs_0)$$

где

$$s_0 = 2\pi\hat{F}_0 = 2\pi T_0^{-1}$$

Дискретное преобразование Фурье задано как:

$$F_r = \sum_{k=0}^{N_0-1} f_k \exp(-ir\Omega_0 k)$$

где $\Omega_0 = 2\pi N_0^{-1}$. Обратная операция задана как:

$$f_k = N_0^{-1} \sum_{r=0}^{N_0-1} F_r \exp(ir\Omega_0 k)$$

Эти уравнения могут быть использованы для вычисления преобразования и обратной операции.

Более подробное описание и реализацию данного метода можно найти на сайте FFTW[7].

---

[7] Дополнительная информация доступна на сайте http://www.fftw.org/

Быстрое преобразование Фурье

Быстрое преобразование Фурье[8] [6,7,8,9,10,12,14,20,24] разработано в 1965 году, как вариант дискретного преобразования Фурье. Его применение уменьшает количество вычислений с порядка N_0^2 до порядка $N_0 \log N_0$. Существуют два основных варианта этого алгоритма: разрешение по времени и разрешение по частоте. Алгоритм упрощается, если N_0 выбирается как степень двойки.

Реализацию алгоритма можно посмотреть в приложении на дискете (файлы RealFFT.h и RealFFT.cpp).

БПФ на практике

Допустим, у нас есть сигнал, взятый с частотой 11025 Гц, и мы берем из него восемь отсчетов для получения частотного разложения для этого участка сигнала. В результате БПФ у нас получится 5 пар коэффициентов — четыре синусоиды с частотами 5512.5 Гц, 4134.375 Гц, 2756.25 Гц, 1378.125 Гц и синусоида с частотой 0 Гц, т.е. константа. Таким образом, взяв восемь отсчетов сигнала, мы можем разложить эту часть на четыре гармоники (синусоиды, частоты). Взяв 16 значений, получим восемь частот и т.д.

Как указывалось выше, для быстрой работы алгоритма необходимо, чтобы количество отсчетов сигнала совпадало со степенью двойки. Перевод сигнала из амплитудного представления в частотное представление осуществляется блоками одинакового размера, так называемый размер преобразования (FFT size).

Важно понимать, что БПФ раскладывает сигнал не на его гармоники, а на свои! Допустим, что в исходной функции была частота

---

[8] Дополнительная информация доступна на сайте http://www.fftw.org/

ровно в 100 Гц и больше ничего. А наше преобразование разложило функцию на частоты 96, 99, 102, 105 Гц. В результате ничего полезного мы не получим. БПФ сможет восстановить функцию в исходном виде, но полученный спектр будет бесполезен.

Таб. 2. Пример результатов БПФ.

Удачный БПФ		Неудачный БПФ	
Частоты	Коэффициенты	Частоты	Коэффициенты
95	0	91	0.04
96	0	93	0.11
97	0	96	0.14
98	0	99	0.85
99	0	102	0.19
100	1	105	0.15
101	0	108	0.09
102	0	111	0.02
103	0	114	0.001

Как видно из таблицы, удачный БПФ, который «попал» в исходную частоту, дал правильный, с нашей точки зрения, результат. Неудачный БПФ — «неправильный» спектр. По такому спектру мы можем только догадываться, что в сигнале была частота примерно в 100 Гц. Важно понимать, что при обратном преобразовании обоих спектров мы получим оригинальный сигнал, несмотря на то, что второе разложение «бестолково».

Для решения подобной проблемы используются сглаживающие оконные функции. Простейшая оконная функция (окно Хэмминга) имеет следующий вид:

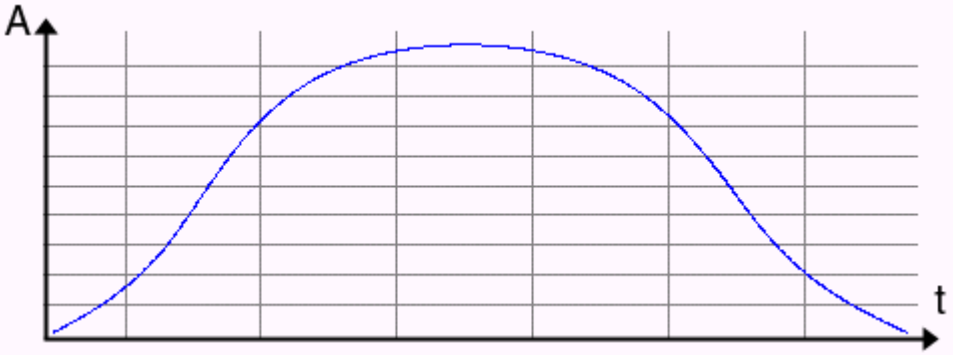

Рис. 3. Окно Хэмминга.

Ниже показан блок до и после применения оконной функции:

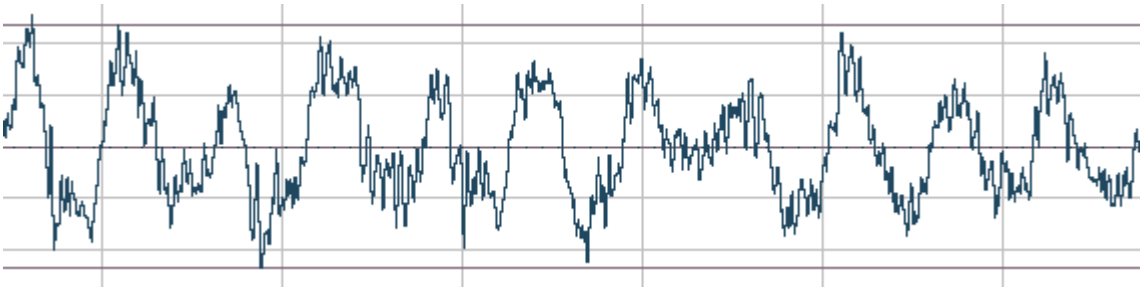

Рис. 4. Сигнал до наложения окна.

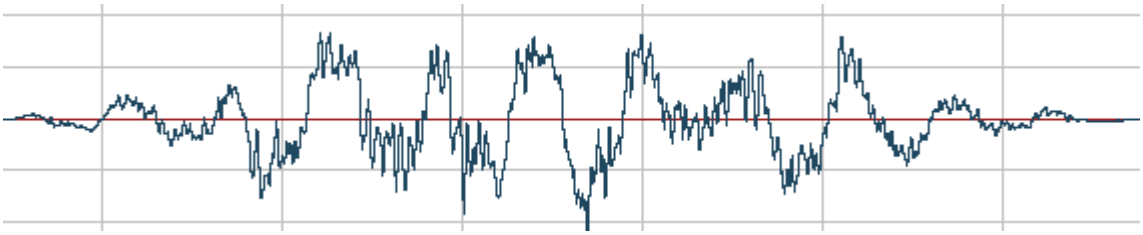

Рис. 5. Сигнал после наложения окна.

Сглаживающая оконная функция сильно стягивает эффект размазывания непопадающих в разложение частот по всему спектру:

Таб. 3. Сравнение результатов после применения сглаживающего окна.

Неудачный БПФ		Неудачный БПФ с оконной функцией	
Частоты	Коэффициенты	Частоты	Коэффициенты
91	0.04	91	0.0011
93	0.11	93	0.013
96	0.14	96	0.021
99	0.85	99	0.93
102	0.19	102	0.14
105	0.15	105	0.011
108	0.09	108	0.003
111	0.02	111	0.0015
114	0.001	114	0.0001

К сожалению у сглаживающей оконной функции имеет побочный эффект — теперь и удачный БПФ стал немного размазанным.

Теория окна была раньше активным направлением исследований в области цифровой обработки сигнала [19,22]. Существует очень много типов оконных функций, включая прямоугольную, Хэмминга (Hamming), Хэннинга (Hanning), Блэкмана (Blackman), Бартлетта (Bartlett) и Кайзера (Kaiser). В настоящее время в распознавании речи используется исключительно окно Хэмминга, которое является особым случаем окна Хэннинга. В общем случае окно Хэмминга определяется следующим образом:

$$w(n) = \frac{1}{\beta_w}\left(\alpha_w - (1-\alpha_w)\cos\left(\frac{2\pi n}{N_s - 1}\right)\right)$$

для $0 \le n \le N_s$ и w(n)=0 во всех других случаях; α_w — константа в диапазоне 0..1, на практике $\alpha_w = 0.54$; N_s — размер окна в отсчетах; β_w — нормализационная константа заданная следующим образом:

$$\beta_w = \sqrt{\frac{1}{N_s}\sum_{n=0}^{N_s-1} w^2(n)}$$

На практике желательно нормализовать окно таким образом, чтобы энергия сигнала до наложения окна была бы приблизительно равна энергии сигнала после наложения окна.

После применения оконной функции уже невозможно восстановить исходную функцию, т.к. информация с концов блока практически не используется. Эта проблема решается, если обработку сигнала вести перекрывающимися блоками, на Рис. 6 они выделены зеленым цветом.

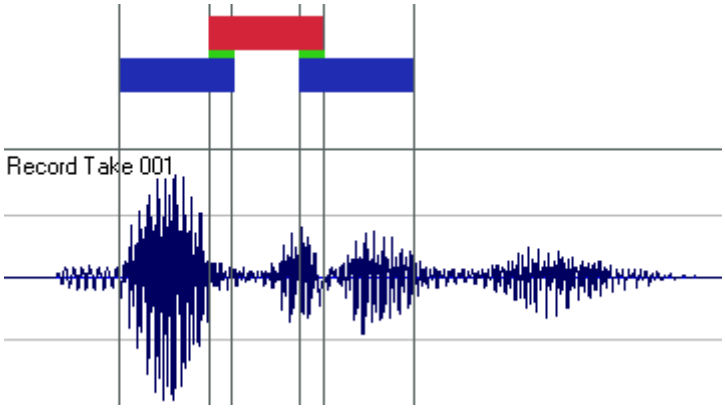

Рис. 6. Перекрывающий анализ.

Рассмотрим размер блока. Если его выбрать равным 2^{14}=16384, то для частоты оцифровки 11025 Гц это будет блок размером около 1.5 секунды, т.е. мы получим 8192 частот. Это очень хорошее частотное разрешение, но информация о частотах усреднена более чем за секунду. Если же взять блок меньшего размера (повысить разрешение по времени), то мы проигрываем по частотному разрешению.

Существует оригинальное решение этой проблемы. Нам ничего не мешает взять 512 значений сигнала и добавить к ним 15872 нулей и провести обработку блока длиной в 16384 значения. Таким образом, мы получим большое разрешение, как по времени, так и по частоте. Побочными эффектами будут: уменьшение амплитуды спектра и изменение фазы. Но фаза нам и не нужна, мы строим

спектр, а изменение амплитуды вычисляется. Таким образом, мы можем получить какое угодно разрешение по времени и частоте.

Таким образом, используя быстрое преобразование Фурье и перекрывающий анализ, можно представить речевой сигнал в виде набора коэффициентов.

Динамическое программирование

Введение

Распознавание речи является важной областью исследования, имеющей широкое применение на практике. Хорошо изучен подход к распознаванию речи базирующийся на хранении одного или нескольких шаблонов для каждого слова словаре распознавания, который также называют моделью пользователя. Таким образом, процесс распознавания состоит из сравнения входящей речи с хранящимися шаблонами. Шаблон с минимальным расхождением от полученного сигнала и будет распознанным словом. Алгоритм, используемый для поиска минимального расхождения, базируется на динамическом программировании.

Динамическое искажение времени

Алгоритм динамического искажения времени является методикой эластичного сравнения вектора наблюдений с хранящимся шаблоном. Вектор наблюдений и шаблон лежат на соответствующих осях сетки (Рис. 7). Для каждой ячейки сетки высчитывается разность между соответствующими кадрами вектора наблюдений и шаблона.

Оптимальное выравнивание между вектором наблюдений и шаблоном показано маршрутом, проходящим по сетке.

Основой алгоритма динамического искажения времени[9] (ДИВ) являются:

- **Признаки**. Информация в каждом сигнале должна иметь одно и то же представление.

[9] Динамическое искажение времени – Dynamic Time Warping

- **Оценки**. При анализе должны использоваться одинаковые виды измерения. Оценки в свою очередь делятся на:
  - **Локальные**. Вычисление разницы между кадром одного сигнала и кадром другого.
  - **Глобальные**. Вычисление полной разницы между входящим сигналом и другим сигналом, возможно другой длины.

Алгоритм ДИВ работает с кадрами, т.е. анализ признаков состоит из обработки вектора признаков в регулярных интервалах. Так как в этой работе выполняется анализ сигнала с помощью преобразования Фурье и банка цифровых фильтров, то наш вектор признаков будет состоять из энергий на каждой полосе в 20 мс.

Так как вектор признаков может иметь множество кадров, то требуются средства расчета локальной оценки расстояния. Оценка расстояния между двумя векторами признаков рассчитывается с помощью Евклидовой нормы:

$$d(x,y) = \sqrt{\sum_i (x_i - y_i)^2}$$

Хотя вычисление Евклидовой нормы в вычислительном отношении не выгодно по сравнению с любой другой нормой, она дает наилучшие результаты для этого алгоритма.

Речь является процессом, изменяющимся во времени. Различные произношения одного и того же слова обычно имеют разную длительность, а произношения одного и того же слова одинаковой длительности отличаются в середине из-за различных частей слова, произносимых с разной скоростью. Чтобы получить глобальную оценку расхождения между двумя речевыми образцами, представленными как вектора, должно быть выполнено выравнивание по времени, как показано на Рис. 7. На рисунке шаблон показан верти-

кально, а входящий сигнал — горизонтально. На этом рисунке, входящий сигнал "SsPEEhH" — это "зашумленная" версия шаблона "SPEECH". Идея алгоритма ДИВ заключается в том, что "h" — это ближайшее совпадение с "H" по сравнению с чем-нибудь еще в шаблоне. Входящий сигнал "SsPEEhH" сравнивается со всеми шаблонами хранящимися в модели пользователя. Результатом сравнения будет шаблон, для которого было найдено минимальное расхождение между входящим сигналом и шаблоном. Глобальная оценка расхождения для маршрута — это просто сумма локальных расстояний между кадрами сигнала и шаблона.

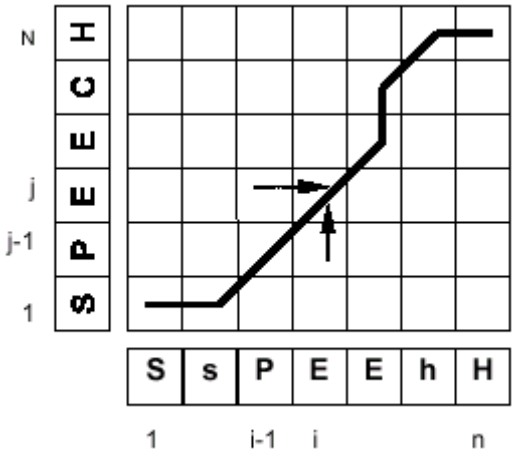

Рис. 7. Пример маршрута ДИВ.

На процесс сравнения накладываются следующие ограничения:

- Маршрут анализа не может идти назад во времени.
- Каждый кадр входящего сигнала и шаблона должен быть использован при сравнении.
- Локальные оценки совпадения объединяются добавлением к текущему глобальному расхождению.

Более подробно алгоритм ДИВ рассмотрен [13,15,16,25].

Виды алгоритма

Примеры алгоритмов даны в псевдокоде, который представляет собой С-подобный язык.

Симметричный алгоритм

Формула для вычисления минимальной глобальной оценки:

$$D(i,j) = \min\begin{bmatrix} D(i-1,j-1) \\ D(i-1,j) \\ D(i,j-1) \end{bmatrix} + d(i,j)$$

где *D(i, j)* является глобальной оценкой для точки (i, j), а *d(i, j)* — локальной.

Алгоритм поиска глобального наименьшего маршрута:

1. Вычислить нулевой столбец, начиная с самой нижней ячейки. Глобальная оценка для этой ячейки равна локальной. Тогда глобальная оценка для каждой последующей ячейки равна локальной оценке для этой ячейки плюс глобальная оценка до ячейки под ней.
2. Вычислить глобальную оценку для первой ячейки следующего столбца, сложив локальную оценку с глобальной оценкой для самой нижней ячейки предыдущего столбца.
3. Вычислить глобальную оценку для оставшихся ячеек текущего столбца. Например, для точки (i, j) — локальная оценка для точки (i, j) плюс минимальная глобальная оценка из (i-1, j), (i, j-1) или (i-1, j-1).
4. Текущий столбец становится предыдущим и все повторяется со второго шага, до тех пор, пока не будут просчитаны все столбцы.

5. Глобальная оценка — это значение, сохраненное в самой верхней ячейке последнего столбца.

Графическое представление алгоритма:

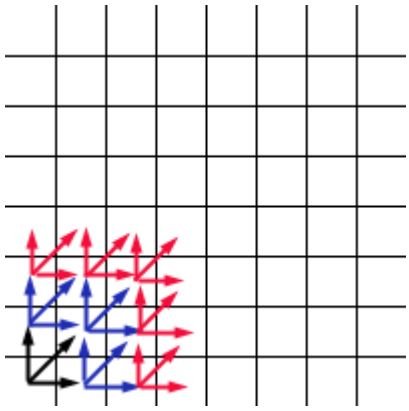

Рис. 8. Симметричный алгоритм.

Ниже показан псевдокод описанного процесса:

```
calculate first column (predCol)
for i=1 to number of input feature vector frames
{
   curCol[0] = local cost at (i,0) + global cost at (i-1,0);
   for j=1 to number of template feature vector frames
   {
      curCol[j] = local cost at (i,j) + minimum of global cost
                  at (i-1,j), (i,j-1) or (i-1,j-1);
   }
   predCol = curCol;
}
minimum global cost is value in curCol[number of template fea-
ture vector frames]
```

Асимметричный алгоритм

Формула для вычисления минимальной глобальной оценки:

$$D(i,j) = \min \begin{bmatrix} D(i-1,j-1) + 2d(i,j) \\ D(i-1,j) + d(i,j) + d_h \\ D(i,j-1) + d(i,j) + d_v \end{bmatrix}$$

где *D(i, j)* является глобальной оценкой для точки (i, j), а *d(i, j)* — локальной; пригодные величины для d_v и d_h могут быть найдены экспериментально.

Графическое представление алгоритма:

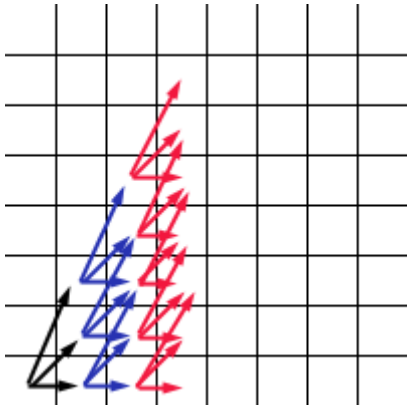

Рис. 9. Асимметричный алгоритм.

Алгоритм поиска минимальной глобальной оценки стал более сложным, чем в симметричной версии. Следовательно, будет легче объяснить процесс в псевдокоде, чем описать его словами:

```
predCol[0] = local cost at (0,0);
for i=1 to number of input feature vector frames
{
   curCol[0] = local cost at (i,1) + global cost at (i-1,0);
   for j=1 to (minimum of number of template feature vector
               frames and 2i+1)
   {
      store j in hoghestJ;
      if the cell (i,j) is a special case
      {
         if it is row 1
         {
            curCol[j] = local cost at (i,j) + global cost at
                        (i-1,j-1);
         }
         else
         {
            if the cell (i,j) is lower of the special case
                                    pair
            {
```

```
                curCol[j] = local cost at (i,j) + minimum of
                            global cost at (i-1,j-1) or
                            (i-1,j-2);
             }
             else
             {
                curCol[j] = local cost at (i,j) + global cost
                            at (i-1,j-2);
             }
          }
       }
       else
       {
          if it is row 1
          {
             curCol[j] = local cost at (i,j) + minimum of
                         global cost at (i-1,j) or (i-1,j-1);
          }
          else
          {
             curCol[j] = local cost at (i,j) +
                        minimum of global cost at (i-1,j),
                        (i-1,j-1) or (i-1,j-2);
          }
       }
    }
    predCol = curCol;
}
minimum global cost is value in curCol[highestJ];
```

Использующийся алгоритм

Формула для вычисления минимальной глобальной оценки:

$$D(i,j) = \min\begin{bmatrix} D(i-2,j-1)+d(i-1,j)+d(i,j) \\ D(i-1,j-1)+d(i,j) \\ D(i-1,j-2)+d(i,j-1)+d(i,j) \end{bmatrix}$$

где *D(i, j)* является глобальной оценкой для точки (i, j), а *d(i, j)* — локальной.

Графическое представление алгоритма:

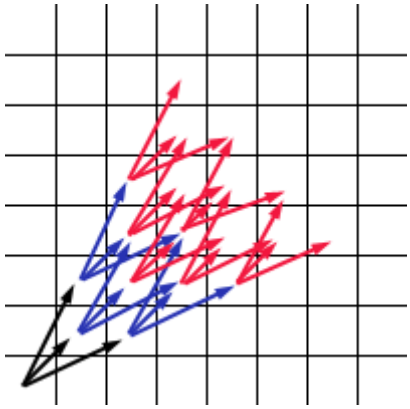

Рис. 10. Используемый алгоритм.

Реализацию алгоритма можно посмотреть в приложении на дискете (файл Model.cpp, функция Model::DTW).

Разработка компонента распознавания речи

Для практической реализации описанных алгоритмов необходимо использовать современную среду быстрой разработки приложений для Windows. В настоящий момент самыми распространенными средами являются:

Среда разработки	Цена, $	Адрес в Интернет
Borland C++Builder 5	$2250	http://www.borland.com/bcppbuilder/
Borland Delphi 5	$2250	http://www.borland.com/delphi/
Microsoft Visual C++ 6	$1630	http://www.microsoft.com/vstudio/
Microsoft Visual Basic	$389	http://www.microsoft.com/vbasic/

В связи с высокой стоимостью, возможно использование только демонстрационных версий данных продуктов. Демонстрационная версия отличается от полнофункциональной ограниченным временем действия (60 дней) и отсутствием некоторых компонентов (помощь, исходные коды и т.д.).

При разработке компонента распознавания речи и соответствующих приложений была использована демонстрационная версия среды быстрой разработки приложений Borland C++Builder 4.

Borland C++Builder

C++Builder (BCB) начал свою историю в 1997 году в качестве нового средства быстрой разработки корпоративных информационных систем. Он сочетает в себе удобства визуальной среды разработки, объектно-ориентированный подход, разнообразные возможности повторного использования исходного кода, открытую архитектуру и высокопроизводительный компилятор языка C++, являющегося на сегодняшний день одним из самых популярных языков программирования.

C++Builder очень похож на Delphi — средство разработки, ставшее одним из самых популярных на сегодняшний день инструментов для создания как настольных, так и корпоративных информационных систем благодаря уникальному сочетанию удобства разработки пользовательских интерфейсов, компонентной архитектуры, однотипности доступа к разнообразным базам данных.

Сходство C++Builder и Delphi не является чисто внешним. C++Builder обладает компонентной архитектурой и создан на основе библиотеки визуальных компонентов Delphi, ставшей в последнее время весьма популярной среди разработчиков. По этой причине этот продукт имеет общую с Delphi библиотеку классов, часть из которых осталась написанной на Object Pascal. Однако совместимость с Delphi на этом не исчерпывается. В проектах C++Builder можно использовать не только библиотеку компонентов Delphi, но и код, написанный на Object Pascal, а также формы и модули Delphi. Поддерживается визуальное наследование форм и модулей данных, в том числе и созданных в Delphi. Эти возможности появились благодаря включению в C++Builder обоих компиляторов — C++ и Object Pascal.

Также C++Builder предоставляет программисту широкие возможности повторного использования кода не только за счет наличия библиотеки компонентов, но и за счет поддержки стандарта ActiveX, что позволяет встраивать в приложение ActiveX-компоненты сторонних производителей, например, движок синтеза речи компании Microsoft.

В C++Builder используется компилятор Borland C++ 5.0. Следует отметить, что данная версия компилятора полностью поддерживает стандарт ANSI/ISO C++.

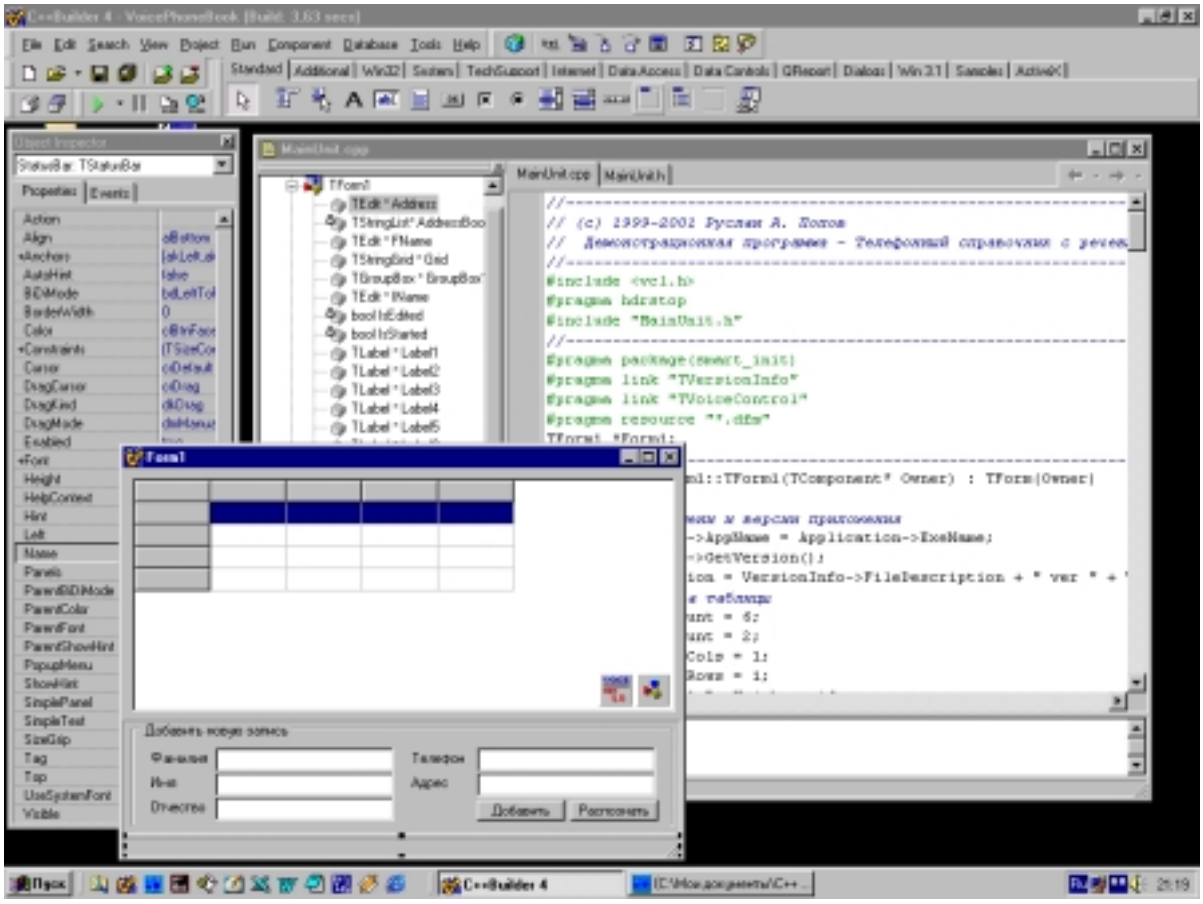

Рис. 11. Интерфейс C++Builder 4.

Необходимую информацию о работе в Borland C++Builder можно получить из хорошо проработанной системы помощи, дополнительно, в специализированных магазинах компьютерной литературы можно приобрести книги по разнообразным вопросам программирования в этой среде разработки приложений.

Аналого-цифровое преобразование речевого сигнала

Для анализа речи её необходимо преобразовать в форму, понятную вычислительной системе. Это может быть цифровая форма, спектральное представление, представление аналоговыми электрическими сигналами и т.д. Так как в работе затрагивается только моделирования систем анализа речи на персональном компьютере, то рассматривается только один вид представления звука — в цифровой форме. Для представления акустического сигнала в цифровой

форме практически во всех системах, имеющих дело со звуком, используется дискретное амплитудное представление. Как известно, звук представляет собой продольные волны разрежения-сжатия, распространяющиеся в акустически проводящей среде. Посредством звукозаписывающих устройств (микрофон) он преобразуется в электрический сигнал, колебания которого повторяют звуковые колебания (Рис. 12).

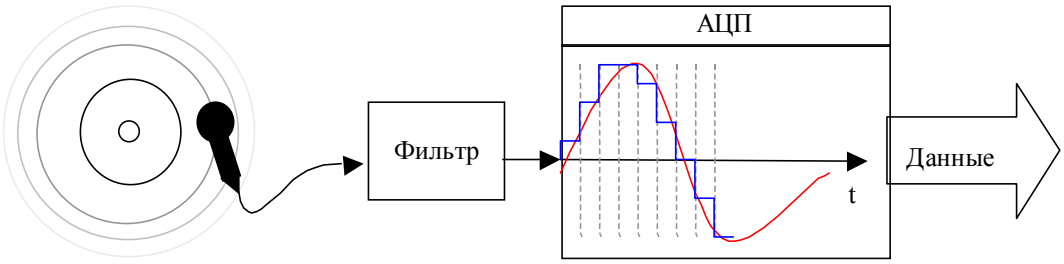

Рис. 12. Ввод звука в компьютер.

Затем этот сигнал фильтруется с целью отсечения частот, превышающих некоторую частоту f_{max}. После этого он подается на аналого-цифровой преобразователь, который с некоторой частотой f_d, называемой *частотой дискретизации*, записывает текущий уровень сигнала в цифровой форме. Из теоремы Котельникова-Найквиста следует:

$$f_{max} < \frac{f_d}{2}$$

При вводе звука определяющими являются *частота дискретизации* (f_d) и *разрядность преобразования* (сколько единиц информации кодирует один отсчет). Частота дискретизации определяет максимальную частоту сигнала, которую можно закодировать. Типичные значения — 11025, 22050, 44100 Гц.

Разрядность определяет потерю информации при аналого-цифровом преобразовании. Типичные значения — 4 бита, 8 бит, 16 бит на отсчет. Естественно, чем больше разрядность и частота дис-

кретизации, тем точнее записывается звук, но и тем больше поток информации и тем сложнее его записать или обработать.

Вопросами оцифровки и воспроизведения звука в персональном компьютере занимается специальное устройство, которое называется звуковой картой. Для воспроизведения звука к специальному разъему карты подключаются колонки или наушники. Запись звука может производиться как с микрофона, компакт-диска, так и через линейный вход.

Звуковые карты применяют для хранения данных метод импульсно-кодовой модуляции, в котором звук — исходный аналоговый сигнал — для хранения в компьютере квантуется по времени и амплитуде. Очевидно, что при таком способе хранения данных, их часть тоже теряется, но в значительно меньшей степени. Чем меньше Δt и ΔA, тем более качественный сигнал, а в идеале при $\Delta t \Rightarrow 0$ и $\Delta A \Rightarrow 0$ получается полноценный аналоговый сигнал.

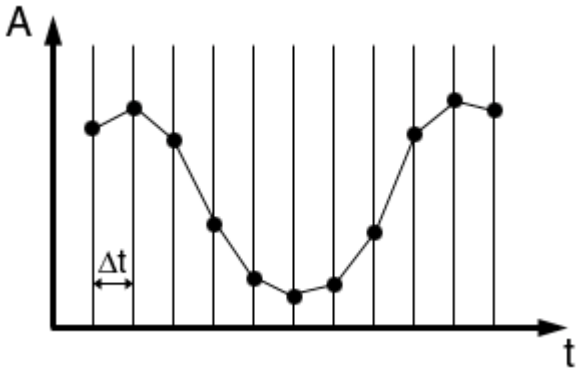

Рис. 13. Квантование по времени.

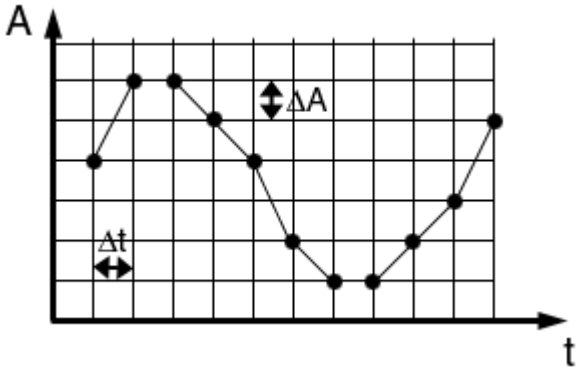

Рис. 14. Квантование по амплитуде.

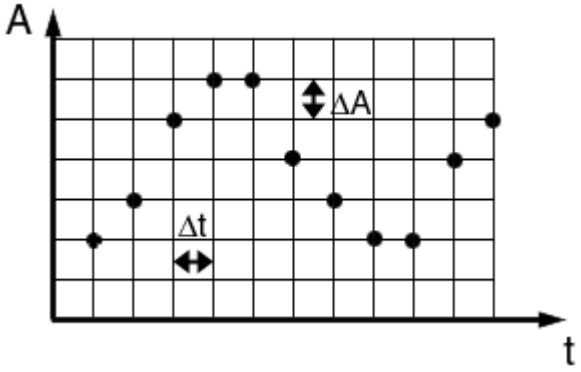

Рис. 15. Хранение звука.

Величина $1/\Delta t$ называется частотой дискретизации. В настоящее время практически все звуковые карты поддерживают частоты дискретизации до 44100 Гц. Такая частота позволяет с высоким качеством воспроизводить весь диапазон звуков, которые воспринимает человеческое ухо, и дальнейшее увеличение частоты не приводит к соответствующему улучшению качества звучания.

Весь диапазон восприятия уровня сигнала разбивается обычно на 256 или 65536 участков. Важно, понимать, что указанные числа выбраны не просто так, они являются степенями двойка (2^8 и 2^{16}), определяя диапазон изменения значений сигнала, который обеспечивают элементарные единицы хранения данных в компьютере — байт (8 бит) и слово (16 бит).

Описание компонента TVoiceControl

Компонент TVoiceControl для Borland C++Builder 4 предназначен для распознавания речевых команд. Распознавание команд выполнено с применением вышеописанных алгоритмов преобразования Фурье и динамического искажения времени.

Инсталляция компонента производится стандартным способом, после которой появляется новая вкладка "TechSupport" в палитре компонентов. На новой вкладке появится компонент, показанный на Рис. 16.

Рис. 16. Палитра с компонентом.

После инсталляции компонент можно использовать при разработке любых приложений. После помещения компонента на форму приложения, последнее станет обладать всеми свойствами компонента.

Рис. 17. Приложение с компонентом.

Исходный код компонента находится в приложении на дискете.

Свойства

Свойство *Name* используется для изменения названия компонента для отражения его назначения в текущем приложении. По умолчанию BCB назначает имена, основываясь на типе, т.е. VoiceCon-

trol1, VoiceControl2 и т.д. Данное свойство используется для указания на компонент в исходном коде приложения.

```
__property AnsiString Name
```

Свойство *Tag* не имеет предопределенного значения. Оно поддерживается для удобства разработчиков и может использоваться для хранения какого-нибудь целого значения или указателя на информацию о компоненте.

```
__property int Tag
```

Методы

Конструктор класса.

```
__fastcall TVoiceControl(TComponent* Owner);
```

Деструктор класса.

```
__fastcall ~TVoiceControl(void);
```

Метод для запуска процесса распознавания голосовых команд, остановить который можно только командой Stop.

```
void __fastcall Start(void);
```

Метод для останова процесса распознавания голосовых команд, инициализированный методом Start.

```
void __fastcall Stop(void);
```

Метод для обучения новой команде пользователя. Во время работы данного метода проверяется наличие в модели пользователя команды с подобной меткой. Если такая команда найдена, то происходит ее замещение, в противном случае в модель пользователя добавляется новая команда.

```
void __fastcall Train(AnsiString Command);
```

Метод для создания новой модели пользователя. Заключается в отчистке существующей модели пользователя. Необходимо контролировать необходимость сохранения текущей модели.

```
void __fastcall NewModel(void);
```

Метод для загрузки модели пользователя из файла. Перед загрузкой модели происходит сброс старой. Необходимо контролировать необходимость сохранения текущей модели.

```
void __fastcall LoadModel(AnsiString FileName);
```

Метод для сохранения текущей модели в файл.

```
void __fastcall SaveModel(AnsiString FileName);
```

Метод для определения наличия модели пользователя. Данный метод полезен для предотвращения работы компоненты в случае отсутствия модели пользователя, которую необходимо создать или загрузить из файла.

```
bool __fastcall IsModelReady(void)
```

Метод для определения наличия внесенных изменений в модель пользователя, это необходимо для обеспечения своевременного сохранения измененных данных.

```
bool __fastcall IsModelModified(void);
```

Метод для озвучивания последней произнесенной команды. Работает только в процессе обучения.

```
void __fastcall PlayVector(void);
```

События

Событие *OnRecognized* возникает при получении результата процесса распознавания голосовой команды.

```
OnRecognized(System::TObject* Sender, const AnsiString
             Result, float Distance);
```

где Sender — указатель на вызывающий объект, Result — метка распознанной команды, Distance — дистанция между шаблонами

Событие *OnStatus* возникает при изменении состояния компонента.

```
OnStatus(System::TObject* Sender, const AnsiString Status);
```

где Sender — указатель на вызывающий объект, Status — состояние, текст.

Событие *OnProcess* возникает при изменении состояния внутренних процессов компонента.

```
OnProcess(System::TObject* Sender, const AnsiString
          Process);
```

где Sender — указатель на вызывающий объект, Process — состояние, текст.

Событие *OnLevel* возникает при изменении уровня сигнала с микрофона.

```
OnLevel(System::TObject* Sender, const DWORD Level);
```

где Sender — указатель на вызывающий объект, Level — уровень сигнала

Событие *OnRecognitionBegin* возникает при начале процесса распознавания полученной голосовой команды.

```
OnRecognitionBegin(System::TObject* Sender, const float*
                   Vector, const DWORD Size);
```

где Sender — указатель на вызывающий объект, Vector — указатель на параметрический вектор, Size — размер параметрического вектора в кадрах.

Событие *OnRecognitionFinish* возникает при завершении процесса распознавания полученной голосовой команды.

```
OnRecognitionFinish(System::TObject* Sender);
```

где Sender — указатель на вызывающий объект.

Событие *OnTrainingBegin* возникает при начале процесса обучения полученной голосовой команды.

```
OnTrainingBegin(System::TObject* Sender, const float*
                Vector, const DWORD Size);
```

где Sender — указатель на вызывающий объект, Vector — указатель на параметрический вектор, Size — размер параметрического вектора в кадрах.

Событие *OnTrainingFinish* возникает при завершении процесса обучения полученной голосовой команды.

```
OnTrainingFinish(System::TObject* Sender);
```

где Sender — указатель на вызывающий объект.

Алгоритм работы

На Рис. 18 показан алгоритм работы компонента в режиме распознавания речевых команд. В режиме обучения происходит упрощение алгоритма работы, в после обработки полученного сигнала с помощью БПФ, он помещается в модель пользователя.

Рис. 18. Алгоритм работы компонента.

Описание программ

Введение

В качестве практического применения разработанного компонента распознавания речевых команд предлагается приложение, выполняющее функции телефонного справочника, которое управляется голосом пользователя. Приложение написано на Borland C++Builder4 для платформы Win9x/ME. Для хранения данных используется форматированный текстовый файл, который далее будет называться базой данных.

Система распознавания речевых команд выполнена в виде компонента для среды ускоренной разработки приложений Borland C++Builder 4 (подробное описание компонента можно найти в предыдущем разделе).

Для тестирования системы распознавания было разработано приложение, показанное на Рис. 19.

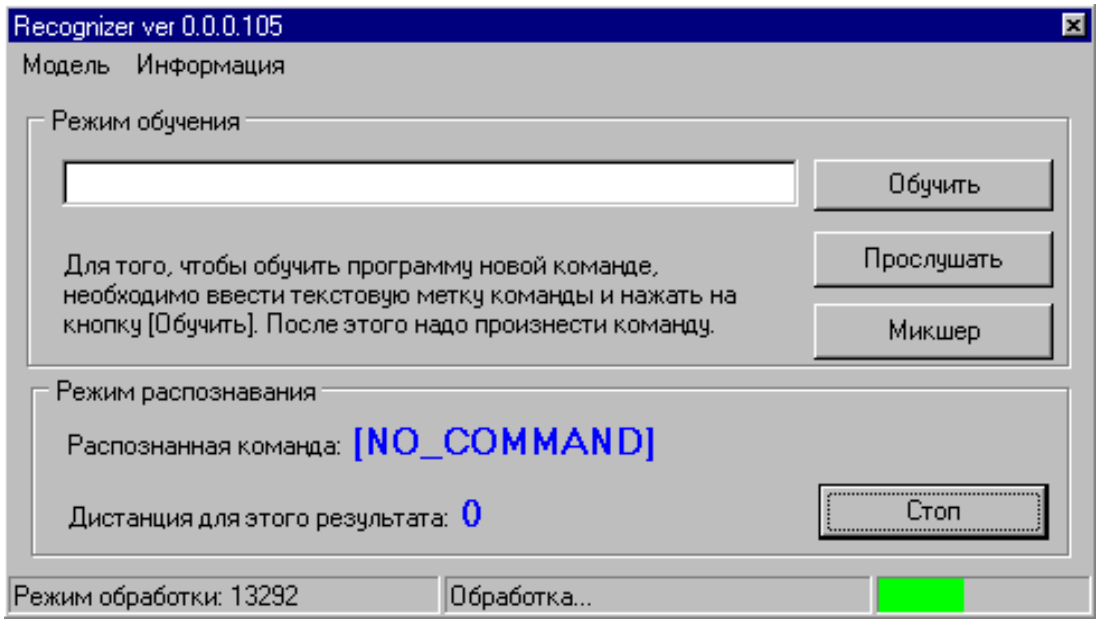

Рис. 19. Приложение для отладки компонента.

Отладочное приложение позволяет создавать, загружать и сохранять модель пользователя. Для создания новой модели пользователя необходимо написать команду в ячейке, нажать на кнопку **[Обучить]** и произнести написанную команду. В результате этого в модель пользователя добавится вектор характеристик для этой команды. Теперь, при необходимости, можно прослушать отданную команду, для этого достаточно нажать на кнопку **[Прослушать]**. Этот процесс надо повторить для всех остальных команд. После завершения процесса обучения, модель пользователя необходимо сохранить.

В нижней части приложения есть кнопка **[Старт]/[Стоп]**, предназначенная для управления режимом системы распознавания речи. После запуска распознавания речи, программа будет воспринимать команды пользователя через микрофон и отображать результаты распознавания.

Кнопка **[Микшер]** предназначена для быстрого вызова системного микшера.

Интерфейс

Интерфейс телефонного справочника состоит из одного окна, которое разделено на две части (Рис. 20). Верхняя часть окна содержит таблицу, в ячейках которой отображаются данные, хранящиеся в базе данных. При необходимости на таблице появляются соответствующие полоски прокрутки. Нижняя часть окна содержит форму для добавления или изменения данных. На форме также расположена кнопка [Распознать]/[Остановить] (надпись зависит от текущего режима работы приложения), которая предназначена для смены режима работы приложения.

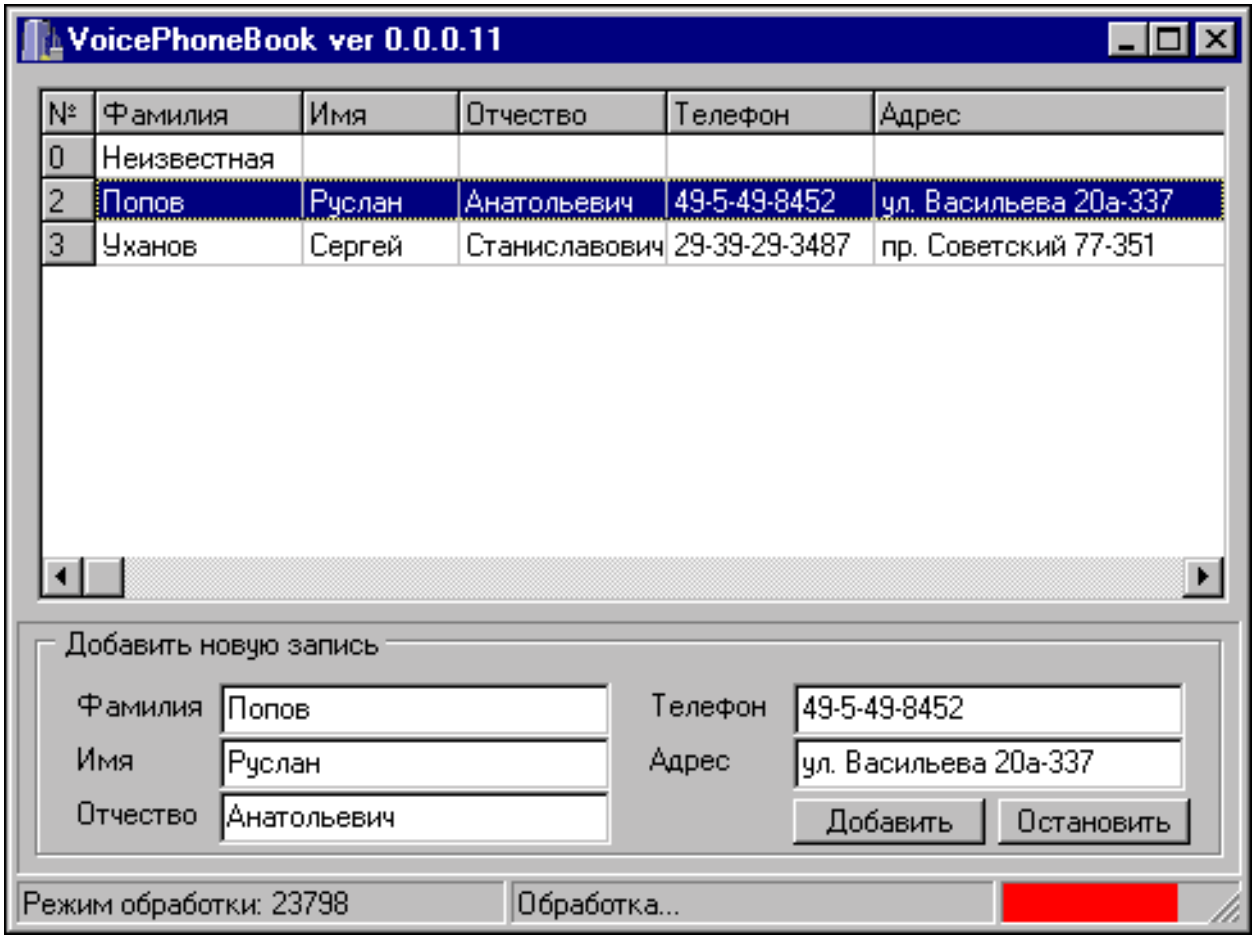

Рис. 20. Интерфейс приложения.

Режимы работы

Приложение имеет два основных режима работы:

- Обучение. В этом режиме происходит обучение приложения пользователем новым фамилиям.
- Распознавания. В этом режиме приложение ожидает фамилию от пользователя, получив которую, пытается найти ее в своей базе данных. В результате поиска, строка с необходимыми данными отмечается синим цветом.

Заключение

Таким образом, в данной дипломной работе:

- Проанализированы особенности распознавания русской речи, которые необходимо учитывать при дальнейшем совершенствовании созданной системы распознавания речевых команд.
- Рассмотрено математическое и алгоритмическое описание задачи распознавания речи и собрано описание доступных для использования программных средств распознавания речи.
- Создана система распознавания речевых команд в виде компонента для среды быстрой разработки приложений Borland C++Builder4.
- Создано демонстрационное приложение, которое представляет собой телефонный справочник, управляемый голосом. При разработке этого приложения использовался компонент распознавания речевых команд.

Опыт работы показал, что для разработки более качественной системы распознавания речи, которая сможет работать с непрерывной русской речью, необходимо использовать более совершенные методы цифровой обработки сигнала и статистического моделирования: кэпстральный анализ [24], скрытые Марковские модели [18] и нейросети [34].

Приложение

Приложение на дискете содержит :

Diplom.pdf	Электронная версия диплома в формате PDF (необходим Acrobat Reader 4).
Programs.zip	Архив с программами и исходными текстами компонента распознавания речи и приложений. Содержание архива:
Recognizer.exe	Приложение, используемое для тестирования компонента распознавания речи.
VoicePhoneBook.exe	Приложение "Голосовой телефонный справочник".
RecognizerWAV	Папка с исходными текстами приложения Recognizer.
TVoiceControl	Папка с исходными текстами компонента распознавания речи.
VoicePhoneBook	Папка с исходными текстами "Голосового телефонного справочника".

Литература

1. Averbuch A., Bahl R. L., Baris R et al. A real time, isolated-word, speech recognition system for dictation transcription // IEEE Intern. conf. on acoustics, speech and signal processing (ICASSP), Vol. 2. P. 858-861, 1985.
2. Bahl L. R., Jelinek F., Mercer R. L. A maximum likelyhood approach to continuous speech recognition // IEEE Trans. Pattern Anal. and Math. Intelligence, Vol. 5, N 2. P. 179-190, 1983.
3. Baker J. K. The Dragon system - an overview // IEEE Trans. Acoust., Speech, and Signal Process, Vol. 23, N 1. P. 24-29, 1975.
4. Baum L. E. An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov process // Inequalities, Vol. 3. P. 1-8, 1972.
5. Blass, William E. and Halsey, George W., Deconvolution of Absorption Spectra, New York: Academic Press, 158 pp, 1981.
6. Bracewell, Ron N., The Fourier Transform and Its Applications, New York: McGraw-Hill Book Company, 381 pp, 1965.
7. Brault, J. W. and White, O. R., The analysis and restoration of astronomical data via the fast Fourier transform, Astron. & Astrophys., 13, pp. 169-189, 1971.
8. Brigham, E, The Fast Fourier Transform, Prentice-Hall, Englewood Cliffs, New Jersey ,USA, 1974.
9. Brigham, E. Oren, The Fast Fourier Transform and Its Applications, Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1988.

10. Cooley, J. W. and Tukey, J. W., An algorithm for the machine calculation of complex Fourier series, Mathematics of Computation, 19, 90, pp. 297-301, 1965.

11. Gabel, Robert A. and Roberts, Richard A., Signals and Linear Systems, New York: John Wiley & Sons, 415 pp, 1973.

12. Gaskill, Jack D., Linear Systems, Fourier Transforms, and Optics, New York: John Wiley & Sons, 554 pp, 1978.

13. Goldenstein Siome, Time Warping of Audio Signals, University of Pensilvania, VAST Lab. http://www.graphics.cis.upenn.edu/

14. Hoffman Forrest, An Introduction To Fourier Theory, Интернет, http://aurora.phys.utk.edu/~forrest/papers/fourier/

15. Keogh E. J., Pazzani M. J., Derivative Dynamic Time Warping, Department of Information and Computer Science University of California, Irvine, http://www.ics.uci.edu/

16. Keogh E. J., Pazzani M. J., Scaling up Dynamic Time Warping to Massive Datasets, Department of Information and Computer Science University of California, Irvine, http://www.ics.uci.edu/

17. Lathi, B. P., Linear Systems and Signals, Carmichael, Calif: Berkeley-Cambridge Press, 656 pp, 1992.

18. Mari J. F., Rdicos S. Speaker independent digit recognition using hidden Markov models // Speech Technol. Conf, pp. 127-132, 1985.

19. Markel J. and Gray A. H., Jr., Linear Prediction of Speech, Springer-Verlag, New York, NY, USA, 1980.

20. Picone J., Signal Modeling Techiques In Speech Recognition, 1993. http://www.isip.msstate.edu/

21. Rabiner L. R., Juang B.-H., Levinson S. E., Sondhi M. M. Recognition of isolated digits using hidden Markov models with continuous mixture densities // ATT Techn. J, Vol. 64, N 6, pt 1. pp. 1211-1234, 1985.

22. Rabiner L.R. and Schafer R.W., Digital Processing of Speech Signals, Prentice-Hall, Englewood Cliffs, New Jersey, USA, 1978.

23. Ravishankar Mosur, Efficient Algorithms for Speech Recognition, School of Computer Science, Carnegie Mellon University, 1996.

24. Robinson Tony, Speech Analysis, 1995, Интернет, http://svr-www.eng.cam.ac.uk/

25. Wrigley Stuart, Dynamic Time Warping, Интернет, http://www.dcs.shef.ac.uk/~stu/com326

26. Аракин В. Д., Выгодская З. С., Ильина Н. Н., Англо-русский словарь, Изд. 13, Русский язык, Москва, 1992.

27. Березин Б. И., Березин С. Б., Начальный курс С и С++, Диалог-МИФИ, Москва, 1996.

28. Гордеев О., Программирование звука в Windows, BHV, Санкт-Петербург, 1999.

29. Гробман М. З., Тумаркин В. И., Выделение скрытых периодичностей и формантный анализ речи. Распознавание образов: теория и приложения. Наука, Москва, 1977.

30. Дарахвелидзе П., Марков Е., Программирование в Delphi 4, BHV, Санкт-Петербург, 1999.

31. Елманова Н. З., Кошель С. П., Введение в Borland C++ Builder, Диалог-МИФИ, Москва, 1997.

32. Жешке Рекс, Толковый словарь стандарта языка С, Питер, Санкт-Петербург, 1994.

33. Любимов А., Евсиков М., Линейное предсказание речи — это просто. Монитор, сс. 30-35, №4, 1995

34. Москаленко А. Система анализа речи с помощью нейросетей. Диплом, 2000, Интернет, http://chat.ru/~alexmoshp/

35. Потапова Р. К., О типологических особенностях слога. Распознавание образов: теория и приложения. Наука, Москва, 1977.

36. Прохоров Ю. Н. Рекуррентное оценивание параметров речевых сигналов. Распознавание образов: теория и приложения. Наука, Москва, 1977.

37. Сапожникова М. А., Акустика, Справочник. Изд. 2-е. Радио и Связь, Москва, 1989, стр. 45-47.

38. Сорокин В. Н., Элементы кодовой структуры речи. Распознавание образов: теория и приложения. Наука, Москва, 1977.

39. Стрыгин В. В., Щарев Л. С., Основы вычислительной микропроцессорной техники и программирования. Высшая школа, Москва, 1989.

40. Уэйн Раш, мл., Говорите со своим компьютером. PC Magazine, Russian Edition, Апрель 1995, с. 46-62.