Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий

Кафедра компьютерных систем и программных технологий

ВЫПУСКНАЯ РАБОТА БАКАЛАВРА

Исследование методов обработки речи для передачи по каналу связи

направление: 09.03.01 - Информатика и вычислительная техника

Выполнил:
Алексеев Даниил Михайлович
Подпись_____

Руководитель: доцент, к.т.н.,
Богач Наталья Владимировна
Подпись_____

Санкт-Петербург

2016

Реферат

РЕЧЕВОЙ КОДЕР, ПОМЕХОЗАЩИЩЁННОЕ КОДИРОВАНИЕ, КАНАЛЬНЫЕ КОДЕКИ, ПОДКЛЮЧАЕМАЯ БИБЛИОТЕКА, МЕТОДОЛОГИЯ ВЫБОРА СПОСОБОВ ОБРАБОТКИ РЕЧИ.

В выпускной работе рассматривается разработка библиотек для канального и речевого кодирования и методы обработки речевого сигнала.

Содержание

Список рисунков

Введение

Не существует настолько надёжных каналов связи, которые могут обеспечить полное отсутствие помех, воздействующих на передаваемую речь. Поэтому были изобретены различные методы защиты от помех. Из них можно выделить три наиболее популярных: аппаратное улучшение аппаратуры и канала связи, повышение отношения сигнал-шум, обнаружение и исправление ошибок в принятых сообщениях. Первый способ защиты позволяет улучшить качество передаваемой речи только до определённых пределов, второй подразумевает работу на физическом уровне (что также имеет свои пределы). Поэтому мы рассмотрим методы обнаружения и исправления ошибок в речевом сигнале.

Кодирование речи можно разбить на два основных этапа: непосредственно речевое кодирование и канальное кодирование (которое, как правило, делится на блочное и свёрточное кодирования, а также деинтерливинг (перемежение)). На Рис. 1 приведена поясняющая схема последовательности кодирования.

Рис. 1.1. Последовательность кодирования речи

Речевое кодирование не является помехоустойчивым, поэтому данные с речевого кодера поступают на канальный. Его задача – сделать передаваемую информацию помехоустойчивой, т.е. дать возможность приёмнику обнаружить (и, в некоторых случаях, исправить) ошибки, которые возникают при передаче информации. Помимо этого, канальное кодирование может выполнять такие функции, как добавление управляющей информации и шифрование.

При блочном кодировании информация делится на блоки определенной длины, и каждый блок кодируется отдельно. Простейшим примером блочного кодирования является дополнение до четности. В этом случае каждый блок делится на группы, которые дополняются одним битом со значением единица или ноль, в зависимости от четного или нечетного количество единиц в исходной группе.

При свёрточном кодировании работа ведётся сразу со всей поступившей информацией (без разделения её на части). Самый простой способ свёрточного кодирования – это добавление между информационных бит проверочных, которые зависят от рядом стоящих информационных бит.

Оба описанных выше метода кодирования позволяют не только определить наличие ошибки в передаваемом коде, но и устранить её. Последнее не всегда является обязательным требованием: бывают ситуации, когда намного проще повторно отправить данные, или же оставить всё, как есть (в таком случае мы получим небольшие помехи во время разговора). К минусам данных методов можно отнести то, что они не могут восстановить правильную последовательность, если ошибкам подверглось слишком много подряд идущих бит. Решением данной проблемы является интерливинг.

Интерливинг нужен для того, чтобы переставить местами закодированную последовательность: если при передаче информация подверглась пачечным ошибкам, то на приёмной стороне после сборки последовательности для декодирования повреждённые биты окажутся на значительном расстоянии друг от друга. Для надёжности можно выполнить процедуру интерливинга несколько раз подряд.

Постановка задачи

Исходя из вышесказанного, мы можем сформулировать задачи, которые требуется решить при выполнении данной работы.

Необходимо исследовать существующие методы обработки речи и предложить свою реализацию библиотек, предназначенных для обработки речи, а также методологию выбора более удобного варианта.

К канальным кодекам выдвигаются следующие требования:

- простота реализации;
- допустимая избыточность;
- возможность коррекции ошибок.

К речевому кодеру выдвигают следующие требования:

- минимальное снижение качества сигнала;
- максимальный процент сжатия.

Стоит отметить, что не всегда удаётся соответствовать всем требованиям. Чаще всего они даже противоречат друг другу.

1. Исследование методов обработки речевого сигнала

1.1. Канальный уровень: блочные кодеки

Блочному кодированию подвергается не весь речевой сигнал, а только его часть. Это связано с тем, что блочными кодами, как правило, не могут кодировать длинные последовательности бит, или при повышении числа информационных бит избыточность становится неприемлемой.

По ходу выполнения работы было выделено три наиболее популярных вида блочных кодов, а также предложены конкретные варианты их применения (по числу информационных и контрольных бит).

1.1.1. Проверка на чётность

Данный вид кодирования выделяется следующими свойствами:

- простота реализации;
- возможность находить и восстанавливать ошибки;
- не защищён от ошибок в контрольных битах.

Нами предлагается проверка на чётность через матрицу; при этом можно выделить два основных вида матриц: двумерные и трёхмерные.

В ходе исследований было установлено, что для первого вида наиболее оптимальным является размер матрицы 2x2 информационных бит: в таком случае мы передаём 4 информационных и 4 контрольных бита, и можем исправить 1 ошибку. На Рис. 2 изложена основная идея алгоритма.

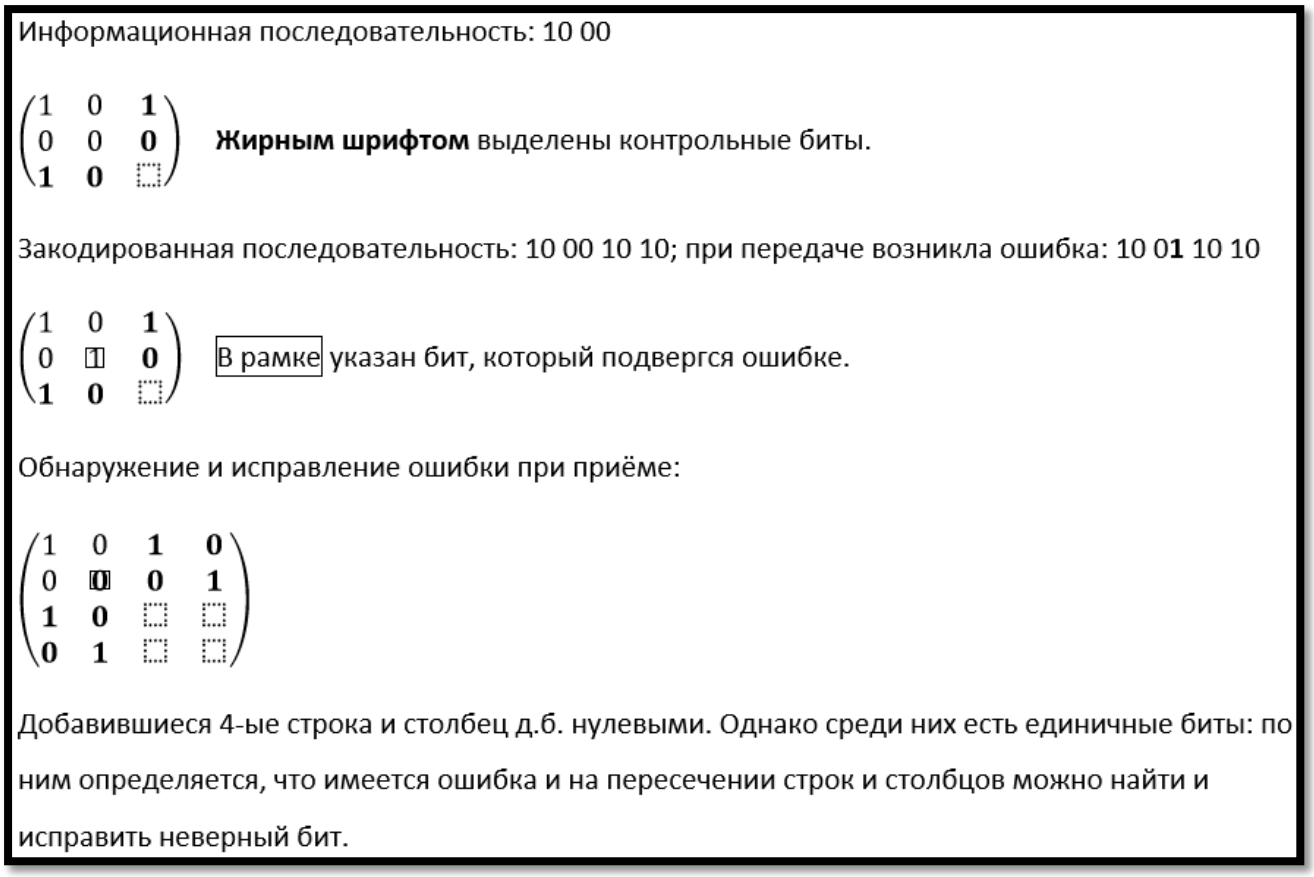

Рис. 1.2. Алгоритм дополнения до чётности, двумерный вариант

Для трёхмерного представления более удобно использовать кодирование бит в виде трёх матриц 3x3. Помимо чётности по столбам и строкам, добавляется матрица проверки «по глубине» - см. Рис. 3. Пример работы с такой матрицей рассмотрен в [1]. На каждые 27 информационных бит мы получаем 27 контрольных, и можем исправить 3 ошибки.

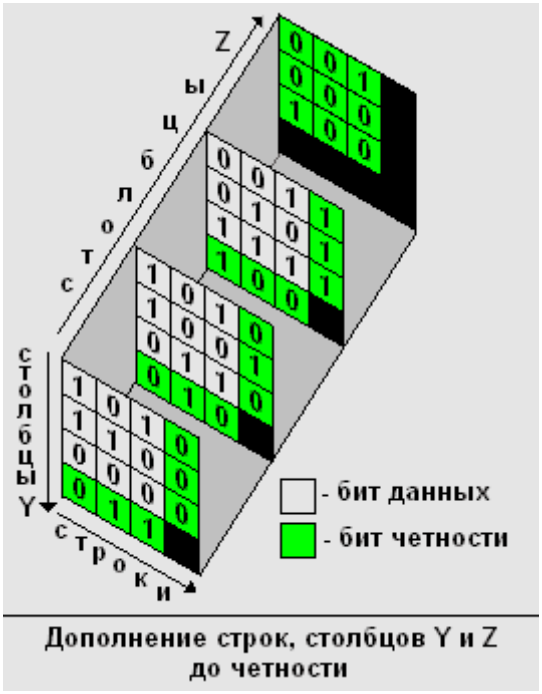

Рис. 1.3. Представление закодированных бит в трёхмерном варианте контроля на чётность

Следует напомнить, что данный вид кодирования не защищает от ошибок в контрольных битах. Однако по сравнению с одномерным вариантом при чётном числе ошибок мы можем узнать, что принятая последовательность – битая.

1.1.2. Код Хэмминга

Данный алгоритм весьма популярен. Он позволяет обнаружить и исправить однократные ошибки. В результате исследований было определено, что наиболее оптимальным вариантом будет код Хэмминга (4, 7): 4 информационных бита, 3 контрольных (7 всего). Пример работы с кодом Хэмминга приведён в [2].

Следует отметить, что кодирующий полином м.б. задан разработчиком: как следствие мы получим дополнительную защиту от прослушки. Кроме того, данный алгоритм не ломается при наличии ошибки в контрольном бите.

К минусам данного алгоритма можно отнести сложность алгоритма декодирования: он требует работы с матрицами (что часто бывает ресурсоёмкой

процедурой). Однако при использовании частного варианта кода Хэмминга, при котором код ошибки совпадает с позицией сломанного бита, работа с матрицами не требуется (однако при этом мы теряем дополнительную защиту от прослушки).

1.1.3. Циклические коды

Данный вид кодов является наиболее сложным. Однако он позволяет при вводе большой избыточности исправлять более одной ошибки. Тем не менее, в результате исследования было решено, что наиболее оптимальным вариантом будет применением циклического кода (4, 7). Примеры работы с данным видом кодов можно найти в [3].

Несомненный плюс предложенного варианта алгоритма – ввод своей таблицы кодирования. Также при вводе большой избыточности мы получаем возможность исправлять более чем одну ошибку.

К минусам данного алгоритма стоит отнести декодирование: он требует многократные циклические сдвиги и деление полиномов.

1.2. Канальный уровень: свёрточные кодеки

Канальное кодирование применяется сразу ко всему передаваемому пакету, без разделения его на блоки. По результатам исследований было выделено два наиболее популярных свёрточных кодека, а также предложено улучшение к одному из них.

1.2.1. Ввод контрольных бит

Данный метод является наиболее простым. Пример работы с данным методом описан в [1], однако сам предложенный там алгоритм требует доработки.

При кодировании между информационными битами вставляются проверочные (они представляют собой сумму по модуль два соседних бит).

Декодирование выглядит следующим образом: суммируются соседние биты исходных данных и сравниваются с их проверочным битом.

- Если для двух соседних проверочных битов была зафиксирована ошибка, то общий информационный бит для этих двух проверочных битов - неверен. Для исправления ошибки необходимо заменить его на противоположный.
- Если для одного проверочного символа была зафиксирована ошибка, а два соседних проверочных символа ошибку не показали, это означает, что сбой произошел в проверочном символе, а информационные биты корректны.

Очевидная проблема данного алгоритма – ошибка в крайних битах. Для её устранения предложен следующий подход: добавлять в начало и конец последовательности по два дополнительных бита, значение которых нам всегда известно. Независимо от принятых данных, мы будем декодировать код исходя из того, что первые и последние три бита – это наши константы и проверочный бит, образованный ими. Таким образом, даже при наличии ошибки в первых/последних двух переданных битах мы сможем без проблем декодировать последовательность.

К плюсам данного алгоритма можно отнести следующее: простота реализации, ошибка в контрольных битах не приводит к краху алгоритма.

Серьёзным минусом данного алгоритма можно считать следующее: если расстояние между ошибочными битами меньше 2 (они стоят рядом или через один), то алгоритм породит дополнительные ошибки.

1.2.2. Кодирование полиномом с задержкой

Данный метод подразумевает использование полинома с задержкой. Кодеры такого типа имеют скорость ½ или ¼ (т.е. на каждый входной бит приходится 2 или 4 закодированных бита). Алгоритм кодирования описан в [4] и [5], алгоритм декодирования – в [6].

К плюсам данного метода можно отнести высокую надёжность: по результатам опытов декодер справлялся примерно с 40% ошибок. Кроме того, несколько подряд идущих ошибок (не более 3) не ломали алгоритм декодирования.

К минусам данного метода можно отнести большую избыточность, высокую сложность алгоритма декодирования и его ресурсоёмкость.

Однако несмотря на указанные минусы, именно этот метод является самым распространённым и используемым в сетях связи.

1.3. Канальный уровень: перемежение (interleaving)

Ранее отмечалось, что не все кодеки могут справиться с несколькими ошибками в переданном пакете/блоке информации. Такие ошибки называются пачечными: они возникают из-за того, что длительность воздействующего вредного сигнала достаточно высока для того, чтобы повредить несколько рядом стоящих бит.

Для борьбы с пачечными ошибками было введено перемежение (интерливинг – interleaving). Перед отправкой в канал связи биты перестраиваются таким образом, чтобы соседние биты оказались как можно дальше друг от друга. При приёме происходит деинтерливинг: мы получаем исходную последовательность; в результате повреждённые биты оказываются на довольно большом расстоянии друг от друга. Примеры интерливинга можно найти в [4].

1.4. Речевые кодеки

Можно выделить два основных вида речевых кодеров, которые в свою очередь порождают гибридный вариант: это вокодеры и липридеры. Нередко бывает так, что одно устройство поддерживает сразу оба варианта.

1.4.1. Вокодер

Принцип действия вокодеров основан на том, что они сопоставляют принятый голосовой сигнал с имеющимся фиксированным словарём, достают из него номер фонемы, который соответствует сигналу, и передают его. Такой подход значительно снижает количество передаваемой информации, но сам голосовой сигнал теряет характерное для человека звучание. Более подробно с принципом действия вокодеров можно ознакомиться в [7].

Сейчас почти не используется в чистом виде, однако применяется в сочетании с липридером.

1.4.2. Липридер

Данный вид речевых кодеров является наиболее распространённым в наши дни. Липридеры снимают характеристики принятого голосового сигнала и передают их. Данный подход обеспечивает более человеческое звучание, но количество передаваемой информации заметно возрастает. Более подробно с принципом действия липридеров можно ознакомиться в [7] и [8].

Следует заметить: из-за использования линейного предсказания в липридерах речь собеседника не всегда удаётся передать более точно (возможно искажение звуков).

1.4.3. Гибрид из вокодера и липридера

При сочетании вокодера и липридера удаётся избежать минусов устройств, в которых данные подходы реализованы отдельно (однако в таком случае повышается количество передаваемой информации: помимо номера фонемы передаются параметры речевого сигнала).

Основой аналитической части таких устройств служат фиксированный и адаптивный кодовые словари, которые позволяют более точно подобрать параметры фильтра для воспроизводящего устройства. Более подробно можно ознакомиться с гибридным вариантом в [8] и [9].

2. Проектирование библиотек

С учётом того, что обработка речевого сигнала м.б. разделена на два основных этапа, мы можем выделить библиотеки для работы с канальным уровнем и работы с самим речевым сигналом.

Библиотека для канального уровня будет содержать четыре модуля: кодирования, декодирования, интерливинга и деинтерливинга.

Для обработки речевого сигнала требуется четыре модуля: два кодирования и два декодирования (по одному на каждый вид кодека).

Было решено написать библиотеки на языке Си. Принятию данного решения способствовали следующие факторы:

- согласно индексу TIOBE [10] язык программирования Си довольно долгое время занимает верхние позиции (1-2 место) по популярности;
- язык Си наравне с ассемблерами используется для программирования микроконтроллеров – таким образом область применения написанных библиотек значительно расширяется;
- написанное на классическом Си приложение будет кроссплатформенным (на уровне исходных кодов): данные библиотеки получат возможность встраиваться в приложения, написанные на разных ОС.

Данные библиотеки разрабатывались по такой технологии проектирования ПО, как прототипирование. В пользу данной технологии были следующие факторы:

- нам известны не все требования (формат, в котором мы будем передавать данные в канал связи; скорость передачи данных);
- данная технология позволила нам быстро увидеть некоторые свойства продукта (применимость, удобство);

Подробнее с данной моделью можно ознакомиться в [11].

Также было решено определить такой тип данных, как Voice_type: он описывает речевой сигнал. В данном типе предлагается хранить принятый речевой сигнал, а также содержимое словарей (фиксированного и адаптивного).

2.1. Модули для обработки речевого сигнала

Данные модули можно использовать отдельно от предлагаемых модулей для работы на канальном уровне. В таком случае, если взаимодействие предлагаемых кодеков для обработки речи с канальными кодеками (отличными от предложенных в данной работе) будет невозможно из-за различающихся форматов представления данных, предлагается использовать такой шаблон проектирования, как адаптер (adapter/wrapper).

Все модули для обработки речевого сигнала представлены в виде интерфейсов. Это было сделано по следующим причинам:

- существует множество способов реализации речевых кодеков, каждый из которых требует довольно большой теоретической базы (математическое описание, особенности речи). Ввод хотя бы в один способ реализации займёт крайне много времени.
- используя интерфейс, мы можем без проблем рассказать, какие действия должен уметь выполнять данный алгоритм и, следовательно, что в данной функции должно быть реализовано.

2.1.1. Модуль для кодирования речевого сигнала

Предполагается, что речевой сигнал уже получен, и мы имеем его в цифровом виде. Пользователю требуется указать, откуда считать полученный сигнал и вид кодера.

Создав объект типа Voice_type, в него следует считать полученный речевой сигнал. Далее идёт вызов функций, которые переводят речевой сигнал в формат кодека. Это функции Phoneme_number_generator(2), Set_phoneme_num(1) (если используется вокодер) или LPC(2) и

Set_filter_parametres(1) (если используется липридер; для использования гибрида рекомендуется последовательно применить указанные функции последовательно, или модифицировать Set_filter_parametres(1) – в таком случае можно будет обойтись вызовом только функций липридера).

После, используя функцию Code_generator(3), пользователь подвергает полученный речевой сигнал помехоустойчивому кодированию, и результат кладётся в глобальную переменную CONV_CODE[500] (определена в Synthetic.c).

2.1.2. Модуль для декодирования речевого сигнала

При декодировании пользователь вызывает ф-цию made_voise(4), в которую передаёт следующие параметры: тип кодека и массив, в который нужно поместить полученный с выхода канального декодера результат (номер фонемы, число нулей интенсивности и прочие параметры речевого сигнала), а также ID свёрточного и блочного кодера.

Далее к указанному массиву обращается модуль, который воспроизводит полученную информацию (в данной работе этот модуль не рассматривается).

2.2. Модули для работы на канальном уровне

Часть модулей для работы на канальном уровне представлены в виде интерфейсов. В данном случае это сделано по следующим причинам:

- алгоритм хорошо известен и/или его написание не составляет особого труда;
- пользователю предлагается самому подобрать нужный кодирующий полином/образующую матрицу.

2.2.1. Модуль для канальных кодеков

Данный модуль служит для помехоустойчивого кодирования и декодирования речевого сигнала. По-умолчанию он предоставляет

пользователю 3 описанных в данной работе кодека блочного кодирования и 2 свёрточных кодека (см. 1.1 и 1.2).

За вызов функций данного модуля отвечает функция Code_generator(3), которая принимает на вход тип речевого кодека, ID блочного кодека и ID свёрточного кодека.

Ниже приведены таблицы 2.1 и 2.2, в которых указано соответствие между ID кодеков и реализуемых ими алгоритмами.

Таблица 2.1.

Соответствие между ID блочного кодека и его алгоритмом

ID кодека	Реализованный алгоритм
1	Проверка на чётность (двумерная)
2	Код Хэмминга (4, 7)
3	Циклический код (4, 7)

Таблица 2.2.

Соответствие между ID свёрточного кодека и его алгоритмом

ID кодека	Реализованный алгоритм
1	Ввод контрольных бит
2	Кодирование полиномом с задержкой

При желании пользователь может добавить в библиотеку свои кодеки.

2.2.2. Модуль интерливинга и деинтерливинга

Данный модуль предназначен для перемежения речевого сигнала, который уже подвергся помехоустойчивому кодированию.

Функции данного модуля вызываются из модулей канальноых кодеков: применение интерливинга без помехоустойчивого кодирования не имеет смысла! Данный модуль содержит две функции: Interleaving(1) (на вход подаются биты, подвергнутые помехоустойчивому кодированию), и DE_interleaving(1) (на вход подаются принятые из канала передачи данных биты).

После интерливинга данные отправляются в канал передачи. Из него они поступают в деинтерливер.

3. Разработка библиотек

Перед описанием самого процесса разработки пару слов о том, как подключать библиотеки.

3.1. Структура библиотек и описание

3.1.1 Библиотека для канальной обработки

3.1.2 Библиотека для речевого сигнала

3.2. Разработка канальной библиотеки

Разработка библиотеки проходит по следующему плану:

- разработка программ для реализации описанных выше методов;
- тестирование.

Стоит отметить, что в работе приведено описание процесса разработки только одной панели показателей и только для одного типа данных – недвижимого имущества. Однако процесс практически не отличается для других типов данных.

3.2.1. Создание метода 1

Сказать про использованный язык. Листинг программы на языке С, реализующей метод 1 приведен в приложении 1.

3.2.2. Создание метода 2

В таком же духе продолжить со всеми последующими методами. Возможно, где-то придётся обойтись 2 словами (или сделать всё одним пунктом: процесс разработки канальных методов весьма похож). Можно представить тестирование списком:

1. Подаём на вход последовательность бит xxx.
2. Кодированные биты подаются на декодер.
3. Сравниваем декодированные биты с исходной последовательностью.

Листинг программы на языке С, реализующей метод 2 приведен в приложении 2.

3.3. Разработка библиотеки для речевого сигнала

Напомнить про постановку задачи. А далее – как в пункте выше. Сделать акцент на том, что непосредственно речевое кодирование в разы сложнее канального.

Заключение

В результате выполнения работы были исследованы методы обработки речи для передачи по каналу связи.

На основании исследования были спроектированы и разработаны требуемые библиотеки:

1. для канального кодирования с кодеками:
    - блочными,
    - свёрточными.
2. для речевого кодирования.

Стоит отметить, что данная библиотека может считаться уникальной, т.к. аналогов в открытом доступе нет/мало.

Список использованных источников

1. Избыточное кодирование информации: [Электронный документ]. (http://all-ht.ru/inf/systems/p_0_11.html). Проверено: 13.05.2016.
2. Код Хэмминга: [Электронный документ]. (http://informkod.narod.ru/5_3item.htm). Проверено: 13.05.2016.
3. Исправление ошибок с помощью циклических кодов: [Электронный документ]. (http://ido.tsu.ru/iop_res1/kodi/index.php-mod=article&id=196.htm). Проверено: 13.05.2016.
4. GEO-Mobile Radio Interface Specifications; Part 5: Radio interface physical layer specifications; Sub-part 3: Channel Coding: [Электронный документ]. (http://www.etsi.org/deliver/etsi_ts/101300_101399/1013770503/01.01.01_60/ts_1013770503v010101p.pdf). Проверено: 13.05.2016.
5. Способы задания сверточного кода: [Электронный документ]. (http://sernam.ru/book_tec.php?id=95). Проверено: 13.05.2016.
6. Алгоритм декодирования Витерби: [Электронный документ]. (http://sernam.ru/book_tec.php?id=96). Проверено: 13.05.2016.
7. Что такое вокодер и липридер?: [Электронный документ]. (http://www.bnti.ru/showart.asp?aid=720&lvl=01.02.09.). Проверено: 13.05.2016.
8. Кодирование и декодирование речевого сигнала: [Электронный документ]. (http://www.sbi-telecom.ru/kodirovanie-i-dekodirovanie-rechevogo-signala.html). Проверено 13.05.2016.
9. Речевой кодер с линейным предсказанием и использованием анализа через синтез: [Электронный документ]. (http://www.freepatent.ru/patents/2163399). Проверено 13.05.2016.
10. TIOBE Index: [Электронный документ]. (http://www.tiobe.com/tiobe_index). Проверено 16.05.2016.
11. Основы программной инженерии. Жизненный цикл ПО. 2015: [Электронный документ].

(http://kspt.icc.spbstu.ru/media/files/2015/course/se/SE2015_01_LifeCycle.pdf). Проверено 16.05.2016

12. Кунегин С.В., Особенности сотовой системы подвижной связи стандарта GSM: [Электронный документ]. (http://www.aboutphone.info/js/kunegin/gsm/index.html). Проверено 12.05.2016.

Приложение 1

Листинг 1. Канальный метод 1

```
void bm_1(int to_encode[], int out[])
{
     //Шрифт – Courier New, 12
     //Каждое приложение должно идти с НОВОЙ СТРАНИЦЫ!
}
```

Приложение 2

Листинг 2. Файл Voice_type.h

```
/*
Данный файл содержит тип данных, который описывает человеческую
речь.
*/

#ifndef VOICE_TYPE
#define VOICE_TYPE

typedef struct
{
     int amplitude;
     int frequency;
     int phoneme;
} Voice_type;

#endif
```

Приложение 3

Листинг 3. Метод xx1 класса xxx

```
//Если мы хотим разместить 2 метода одного класса, то можно
//использовать одно приложение; отделение страницей при этом не
//происходит!
```

Листинг 4. Метод xx2 класса xxx

```
//Класс – как и в листинге 3, но уже с другим методом!
```

Приложение 4

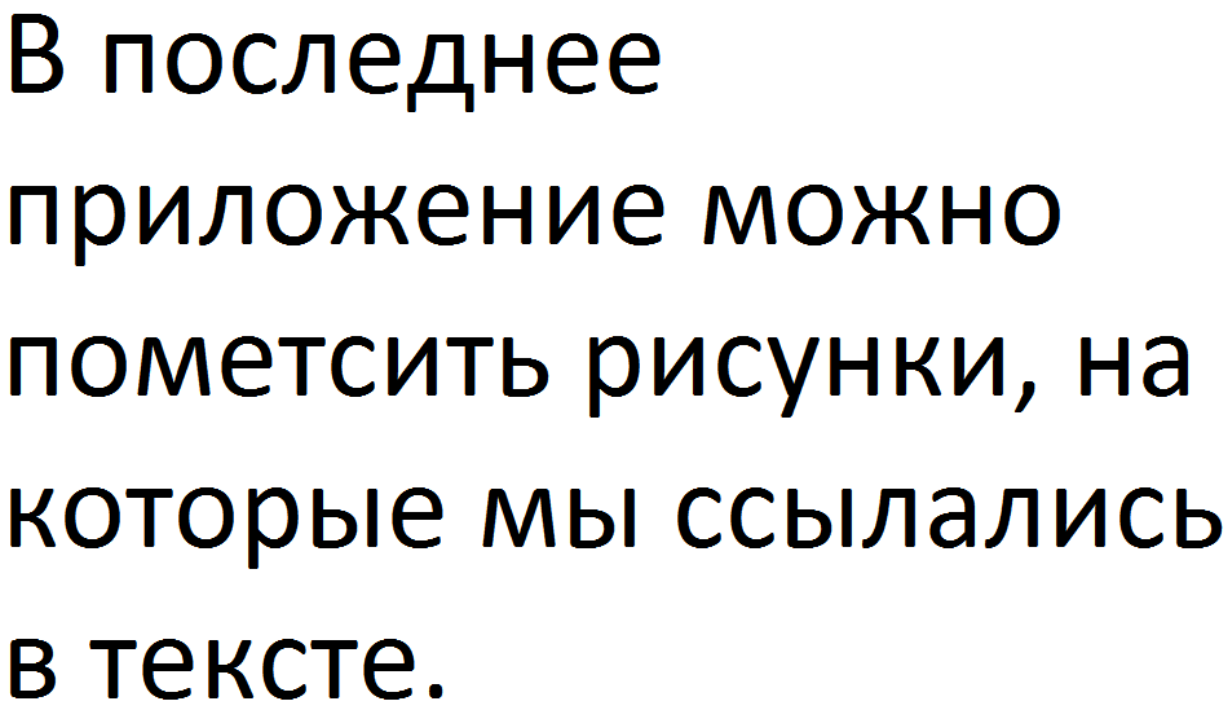

Рис. 4. Рисунок на который ссылаемся из текста

Заключительный лист работы

Выпускная работа выполнена мною самостоятельно. Использованные в работе материалы и концепции из опубликованной научной литературы и других источников имеют ссылки на них.

Список использованных источников содержит _____ наименований.

Работа выполнена на_____ страницах.

Приложения к работе на _____ страницах.

Один экземпляр сдан в директорат.

Подпись_____/_____/

(фамилия, инициалы)

Дата «_____» _____ 20_____г.