МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ФАКУЛЬТЕТ ПРИКЛАДНОЙ МАТЕМАТИКИ И ИНФОРМАТИКИ
Кафедра многопроцессорных систем и сетей

РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЙ ПОД ПЛАТФОРМУ ANDROID СИНХРОННОГО ПЕРЕВОДА РЕЧИ В АНИМАЦИЮ ЯЗЫКА ЖЕСТОВ

Курсовая работа

Ульяницкого Владимира
Александровича
студента 3 курса,
специальность «Информатика»

Научный руководитель:
старший преподаватель кафедры
МСС
А. С. Гусейнова

Минск, 2020

РЕФЕРАТ

Курсовая работа, 23 стр., 6 рис., 10 ист.

РАСПОЗНАВАНИЕ РЕЧИ, СУРДОПЕРЕВОД, MOCAP, АНИМАЦИЯ, ANDROID, LEAP MOTION

Объект исследования – изучение вопроса использования мобильных приложений в качестве удобного инструмента для осуществления перевода голосовых команд в анимацию языка жестов.

Цель работы – усовершенствовать средства коммуникации для людей с особенностями слуха и речи.

Результат работы – изучены средства распознавания речи и захвата движения и применены для написания демо-приложения.

Область применения – социальная сфера, создание комфортной и безопасной среды для людей с особенностями речи и слуха и облегчение для них интеграции в общество.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Речь — один из наиболее важных инструментов взаимодействия между людьми. Мы постоянно пользуемся ей в нашей повседневной жизни для общения друг с другом, выражения своих мыслей и эмоций, зачастую забывая про то, что существуют люди, которые не имеют возможности использовать этот способ коммуникации. Мы привыкли к тому, что можем с лёгкостью воспринимать звуки речи и издавать их. Однако для людей, которые страдают нарушениями слуха или речи, это гораздо более сложная, а то и вовсе невыполнимая, задача. Общение с ними для обычного человека сильно затруднено, так как большая часть людей не умеет пользоваться языком жестов.

Решение данной проблемы имеет высокую социальную значимость. Это может являться частью программы создания комфортной и безопасной среды для людей с особенностями речи и слуха и облегчить для них интеграцию в общество. Это весьма актуально, ведь на учете Белорусского общества глухих на данный момент стоит около 10 тысяч инвалидов по слуху. Приложение-сурдопереводчик могло бы значительно улучшить жизнь этих людей.

Идея о необходимости реализации данного приложения поступила от автошколы “Нужный шаг”, в которой ведется обучение глухонемых и слабослышащих людей. По словам руководства и преподавателей автошколы, они нуждаются в приложении, которое бы могло быстро переводить голосовые указания инструкторов по вождению в анимированные жесты.

ГЛАВА 1 ПОСТАНОВКА ЗАДАЧИ, ЦЕЛИ И ПРОБЛЕМЫ ПРОЕКТА

1.1 Постановка задачи

Задачей курсового проекта является реализация приложения под платформу Android для синхронного перевода устной речи в анимацию языка жестов. Данная задача была сформулирована исходя из нескольких требований к приложению: способность быстро реагировать на команды пользователя, возможность работы оффлайн, возможность использования без специального оборудования и удобство восприятия результата глухонемыми и слабослышащими пользователями.

Так как данное приложение предполагается использовать непосредственно на дороге в процессе поездки, очень важно, чтобы перевод происходил именно на язык жестов. Как удалось выяснить, язык жестов воспринимается глухонемыми проще, чем обычный текст, в связи с этим пользователи будут более комфортно себя чувствовать в процессе движения, своевременно реагировать на указания, которые будут поступать от инструкторов, и, соответственно, их обучение будет более удобным и качественным, что в дальнейшем позволит избежать аварийных ситуаций на дороге.

Исходя из подобного сценария использования, кажутся очевидными выбор платформы Android, как наиболее распространённой среди целевой аудитории мобильной операционной системы, а также необходимость обеспечить работу приложения без подключения к сети Интернет.

1.2 Цели проекта

Проект ставит целью усовершенствовать средства коммуникации для людей с особенностями слуха и речи. В частности, приложение должно облегчить для таких людей обучение вождению.

Для достижения данной цели планируется предпринять следующие шаги:

1. Изучение проблематики для точного понимания задачи, которую собираемся решать;
2. Общение с целевой группой для понимания их потребностей и обсуждения возможностей будущего приложения;
3. Изучение доступных технологий, которые уже решают данную проблему. Анализ приложений-конкурентов с целью понять, что уже решено, и что может быть улучшено;
4. Определение перечня технических средств: платформы, доступных библиотек и решений, которые собираемся использовать для реализации прототипа;

5. Составление технического задания на основе требований целевой аудитории к приложению и возможностей, выбранных нами, технических средств;
6. Реализация приложения;
7. Тестирование приложения в реальной жизни.

1.3 Проблемы проекта

Основная проблема проекта заключается в том, чтобы использовать технологию распознавания речи без какой-либо серверной составляющей, вместо этого используя исключительно возможности мобильного устройства. Это накладывает определённые ограничения на качество распознавания, однако вместе с этим значительно повышает его скорость.

Существует несколько способов решения проблемы недостаточной точности распознавания речи, основные из них это использование небольшого по объёму (200 – 300 слов) целевого словаря и адаптация акустической модели на входных данных, максимально приближенных к реальным [1].

1.4 Выводы

Таким образом, в данной главе поставлены задачи и цели приложения, требования к разработке, сформулированы необходимые характеристики продукта, проанализированы проблемы и способы их решения. Был сделан выбор целевой платформы для реализации приложения.

ГЛАВА 2 ОПИСАНИЕ ПРИЛОЖЕНИЯ И ЕГО ИНТЕРФЕЙСА

2.1 Описание приложения

В физическом смысле приложение после сборки представляет собой файл с расширением *.apk для установки основной части приложения на устройство пользователя. Скачать приложение можно будет напрямую из Google Play Market — официального магазина приложений операционной системы Android. Целевые словари планируется сделать доступными для загрузки отдельно, непосредственно из приложения.

Приложение позволяет пользователю осуществлять перевод устной речи в её жестовую интерпретацию, с дополнительным выводом тестовой версии распознанного текста для контроля возможных неточностей распознавания, а также просматривать все доступные жесты из предварительно скачанных целевых словарей.

2.2 Интерфейс приложения

Весь интерфейс приложения можно условно разделить на три части:

- интерфейс главного экрана;
- интерфейс просмотра словаря;
- интерфейс настроек приложения.

Внешний вид интерфейса главного экрана и просмотра словаря приведён на рисунке 2.1.

Интерфейс главного экрана — интерфейс, предоставляющий пользователю возможность использования основной функции приложения – перевода. Служит для осуществления голосового ввода при нажатии соответствующей кнопки или при голосовой активации и для просмотра жестового перевода.

Интерфейс просмотра словаря служит для просмотра текстовой версии слов и фраз, содержащихся в словаре, а также их анимированных жестовых версий.

Интерфейс настроек приложения предоставляет пользователю возможность скачать требующиеся целевые словари либо удалить ненужные, установить ключевую фразу для активации функции распознавания и уровень чувствительности, с которым она будет распознаваться.

Навигация между элементами интерфейса осуществляется с помощью кнопок меню, расположенного в верхней части экрана. Меню создано с помощью средств разработки, встроенных в среду Android Studio. Разметка интерфейса и стилизация объектов реализованы с помощью правил, описанных в структуре xml [3].

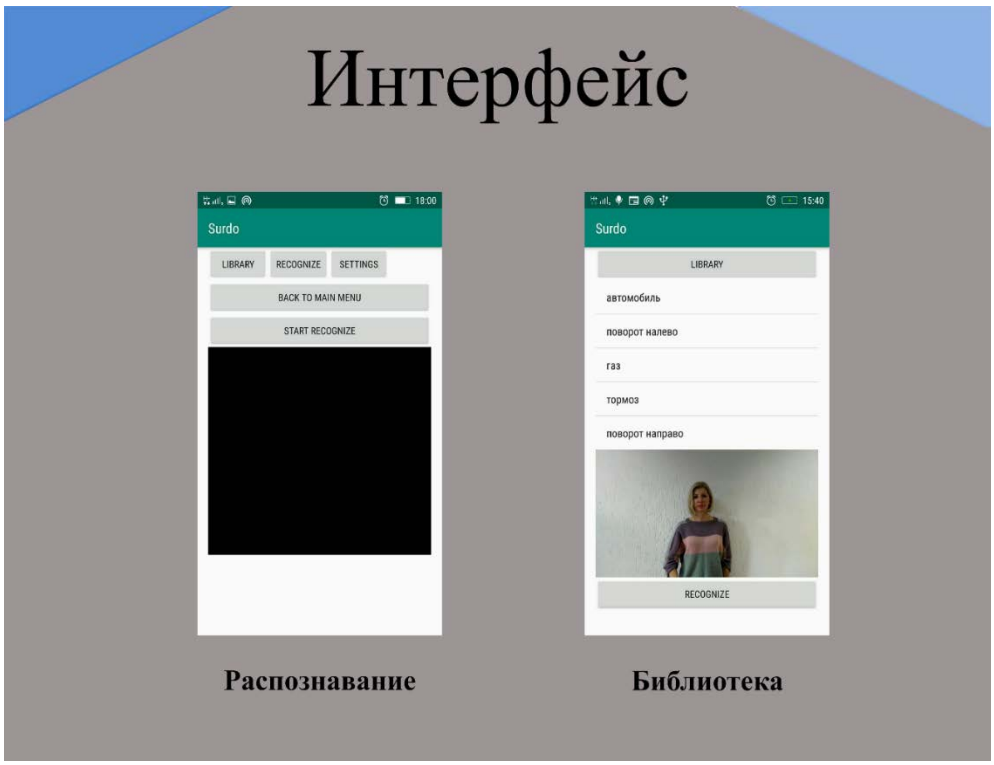

Рисунок 2.1 – Интерфейс приложения

Интерфейс не создаётся в коде приложения, меню и его подпункты описываются в файлах ресурсов, подключаемых в приложении, а в коде приложения выполняется управление уже созданными элементами интерфейса, например скрытие и показ кнопок и пунктов меню в зависимости от логики работы приложения. Такой подход позволяет отделить реализацию логики приложения и интерфейса, использовать различные варианты интерфейса в зависимости от экрана простым выбором из нескольких доступных ресурсов. Также это позволяет использовать визуальные редакторы интерфейса, такие как встроенный в среду Android Studio Layout Editor.

ГЛАВА 3 АРХИТЕКТУРА ПРИЛОЖЕНИЯ

3.1 Основные компоненты приложения

В качестве платформы, которая будет использована для разработки приложения (рисунок 3.1), выбор был сделан в пользу Android, так как он является одним из самых распространенных и доступных. К тому же Android является более простым для начинающих разработчиков.

В структуре приложения можно выделить несколько основных компонент: модуль распознавания речи, база данных и клиентское приложение. Клиентское приложение при этом является прослойкой между базой данных и модулем распознавания речи. Такой подход позволяет упростить добавление и удаление целевых словарей без необходимости модификации всего приложения.

Рисунок 3.1 – Архитектура приложения

3.2 Модуль распознавания речи

Модуль распознавания речи используется для перевода устной речи пользователя в текст, который далее передаётся в клиентское приложение. Подавляющее большинство подобных систем работают онлайн: речь передаётся на удалённый сервер, где и происходит распознавание. Такой вариант не подходит для данного приложения, поэтому был сделан обзор среди систем, имеющих возможность работы офлайн. Также, для большего удобства интеграции в приложение, выбор производился среди систем, распространяемых с open-source лицензиями, т. к. это избавляет от ограничений на распространение итогового продукта и даёт больше возможностей для

модификации, в том числе адаптации существующей акустической модели либо создания своей собственной.

Таким образом были рассмотрены следующие системы: CMU Sphinx, HTK, iAtros, Julius, Kaldi и RWTH ASR. Выбор системы зависел от таких критериев, как точность и скорость распознавания, удобство использования в программном коде и внутренняя структура. По точности системы оценивались исходя из наиболее популярных метрик: коэффициент распознавания слов (WRR), коэффициент ошибок в словах (WER). Скорость работы системы оценивалась с помощью коэффициента Speed Factor (SF) — показателя отношения времени распознавания к длительности распознаваемого сигнала, иногда называемого также Real Time Factor [4], результаты сравнения приведены в таблице 3.1.

Таблица 3.1 — Результаты сравнения по точности и скорости

Система	WER, %	WRR, %	SF
HTK	19,8	80,2	1,4
CMU Sphinx (pocketsphinx/sphinx4)	21.4/22.7	78.6/77.3	0,5/1
Kaldi	6,5	93,5	0,6
Julius	23,1	76,9	1,3
iAtros	16,1	83,9	2,1
RWTH ASR	15,5	84,5	3,8

Также было произведено сравнение по алгоритмам их работы. Первая стадия распознавания речи — извлечение признаков, значимых для распознавания. На данном этапе необходимо отделить признаки, имеющие лингвистическую значимость, от искажений, вносимых шумом, эмоциями, индивидуальными различиями дикторов. Все системы используют для этих целей мел-кепстральные коэффициенты (MFCC), а CMU Sphinx, Kaldi и RWTH ASR также более шумоустойчивые коэффициенты линейного перцепционного предсказания (PLP). Для акустического моделирования все рассмотренные системы используют статистические методы: скрытые марковские модели (HMM) — статистическая модель, имитирующая работу процесса, похожего на марковский процесс с неизвестными параметрами, задачей ставится разгадывание неизвестных параметров на основе наблюдаемых, — Kaldi, iAtros и RWTH ASR также могут использовать обобщённый метод моментов (GMM). Языковое моделирование выполняется во всех системах с помощью N-грамм и конечного автомата-преобразователя (кроме HTK). Непосредственно для распознавания используется алгоритм Витерби (HTK, CMU Sphinx, Julius,

iAtros и RWTH ASR) и двухпроходной алгоритм прямого-обратного хода (Kaldi) [5, 6].

Важным требованием также является наличие API. Этому требованию соответствуют только системы HTK, CMU Sphinx и Julius.

Исходя из вышеизложенного, система CMU Sphinx была выбрана для использования в проекте. CMU Sphinx включает в себя два декодера - pocketsphinx, реализованный на C, и sphinx4, реализованный на Java. Это позволяет использовать эту систему на многих платформах, включая те, на которых установлена операционная система Android, что значительно облегчает ее интеграцию в проекты, написанные на Java. Модульность реализации системы CMU Sphinx значительно упрощает работу с ней, позволяя легко вносить в неё изменения и исправлять ошибки. Что касается простоты использования в стороннем приложении, то наличие в системе API в дополнение к интерфейсу консоли значительно упрощает этот процесс. CMU Sphinx также содержит подробную, тщательно разработанную документацию, которую может использовать даже начинающий разработчик программного обеспечения, что значительно упрощает процесс знакомства с системой и работы с ней. Кроме того, преимуществом системы CMU Sphinx является возможность работы с множеством различных языков, для которых в открытом доступе присутствуют их языковые и акустические модели. В списке таких языков присутствуют, например, английский, русский, итальянский, испанский. Кроме того, в случае необходимости не составит труда создать необходимые модели с помощью встроенных средств системы и данных из корпуса Voxforge. Лицензия, под которой распространяется CMU Sphinx, разрешает его использование в коммерческих проектах без раскрытия их исходного кода, что несомненно также является преимуществом.

В процессе разработки приложения выявилась необходимость увеличения шумоустойчивости модуля распознавания речи. Рассмотрим более подробно, как реализовано распознавание речи в системе CMU Sphinx. CMU Sphinx использует для извлечения частотных признаков мел-частотные кепстральные коэффициенты (mel-frequency cepstral coefficients), точнее их модификацию – нормированные по мощности частотные кепстральные коэффициенты (power-normalized cepstral coefficients), которые достаточно шумоустойчивы. Соответственно нет принципиальных препятствий для создания шумоустойчивой акустической модели. Вносить специфические шумы в данные корпуса также не имеет смысла, как минимум в случае, если они имеют иные частотные характеристики.

Мел-частотные кепстральные коэффициенты широко используются для решения задачи извлечения признаков из человеческой речи. Шкала мел – так называемая психофизическая шкала высоты звука. Используется она потому,

что восприятие изменения высоты звука человеком отличается от реального изменения, выраженного в герцах. Человеческое ухо гораздо чувствительнее различает незначительную разницу в высоте на низких частотах, чем на высоких. Именно поэтому низкие частоты более важны для распознавания речи. Шкала мел обеспечивает это – зависимость высоты звука в мел от его частоты в герцах (см. рисунок 3.2) нелинейная и выражается формулой:

$$M(f) = 1127{,}01ln\left(1 + \frac{f}{700}\right)$$

Эта формула отражает восприятие высоты звука человеком достаточно точно, хоть и не идеально, и при этом достаточно проста и не требует ресурсоёмких вычислений.

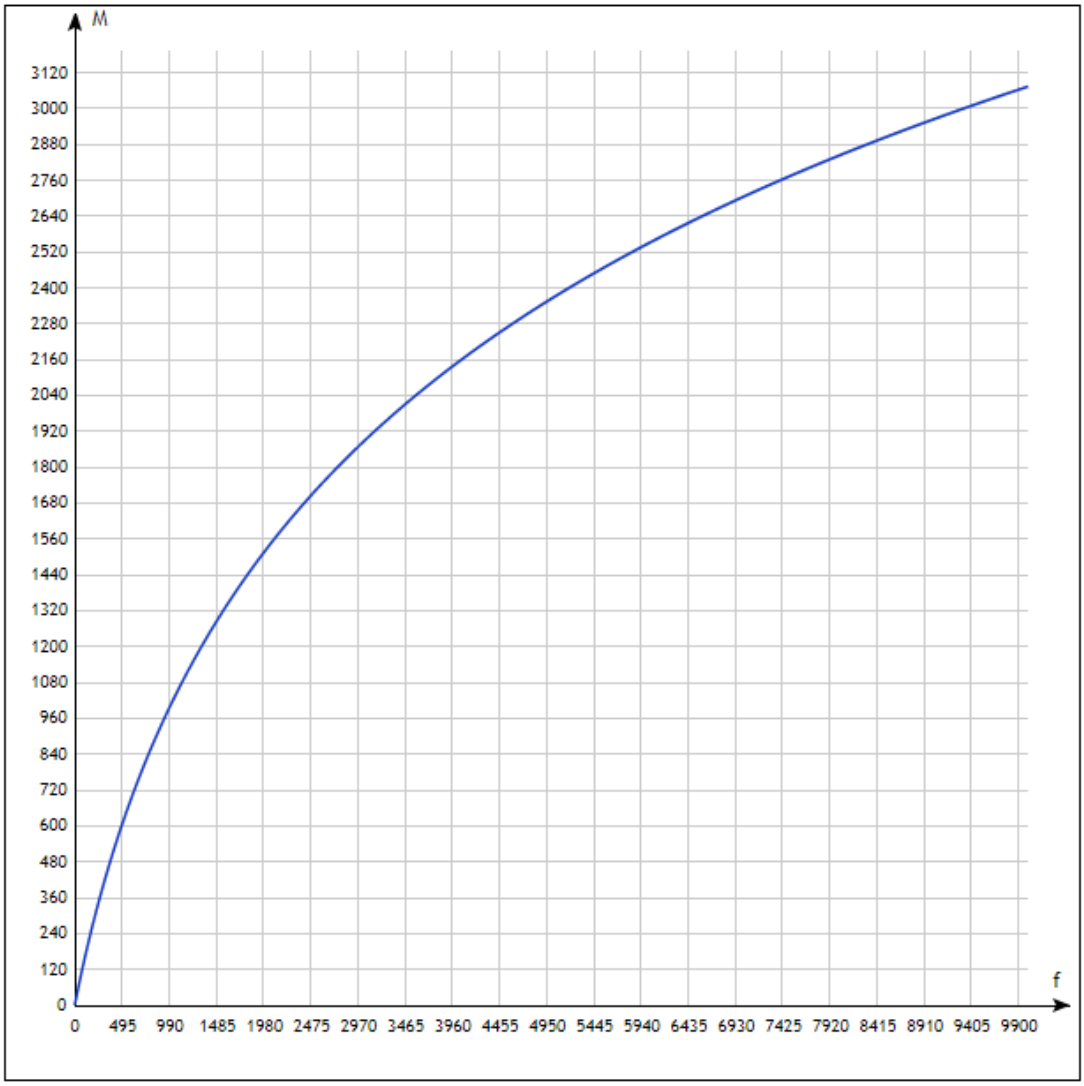

Рисунок 3.2 – График зависимости высоты звука (мел) от частоты (Гц)

Синтез речи можно разделить на две стадии: сначала акустическая волна генерируется в дыхательной системе (лёгкие, бронхи, трахея), а затем преобразуется в речевом тракте. Таким образом процесс можно представить как

последовательную генерацию основного тона и его фильтрацию. При этом основной тон является дикторозависимым, а коэффициенты фильтра зависят от параметров голосового тракта, при этом именно эти параметры определяют произносимые фонемы и, соответственно, важны для распознавания. Итоговый сигнал можно представить в виде:

$$F(\omega) = S(\omega)H(\omega)$$

Для выделения характеристик фильтра, не пересекающихся с характеристиками основного фона, выполняются преобразования. Сначала прологарифмируем равенство:

$$ln(F^2(\omega)) = ln(H^2(\omega)) + ln(S^2(\omega))$$

Это обусловлено и особенностями человеческого восприятия: зависимость субъективного восприятия громкости звука от его мощности близка к логарифмической, т. е. чтобы удвоить воспринимаемую человеком громкость, необходимо повысить мощность источника звука примерно в 8 раз.

Затем для получения непересекающихся характеристик основного тона и параметров голосового тракта выполняется обратное преобразование Фурье:

$$C_s(q) = \frac{1}{2\pi} \int_{-\infty}^{+\infty} ln|S^2(\omega)|e^{i\omega q} d\omega$$

Итоговая функция – кепстр – функция обратного преобразования Фурье от логарифма спектра мощности сигнала. Для вычисления коэффициентов сигнал разбивается на отрывки-фреймы длиной 23 мс.

На следующем этапе – акустическом моделировании – используются скрытые марковские модели (рисунок 3.3). При этом вычисленные на предыдущем шаге коэффициенты сопоставляются с фонемами языка. Скрытые марковские модели описывают двухэтапный стохастический процесс. Первый этап – последовательность фонем рассматривают как марковскую цепь – цепь событий, в которой вероятность наступления последующего события зависит только от настоящего и не зависит от предыдущих:

$$P(X_i|X_1, X_2, \dots, X_{i-1}) = P(X_i|X_{i-1}) \ (1)$$

На следующем этапе для каждого фрейма рассматривается наблюдаемый результат Y_1, которым в нашем случае является последовательность коэффициентов, вычисленных для фреймов. Предполагается, что для наблюдаемого результата справедливо утверждение о том, что вероятность наступления определённого результата наблюдения зависит только о текущего скрытого состояния и не зависит от предыдущих состояний и результатов наблюдения:

$$P(Y_i|X_1, X_2, \dots, X_{i-1}, X_i, Y_1, Y_2, \dots, Y_{i-1}) = P(Y_i|X_i) \ (2)$$

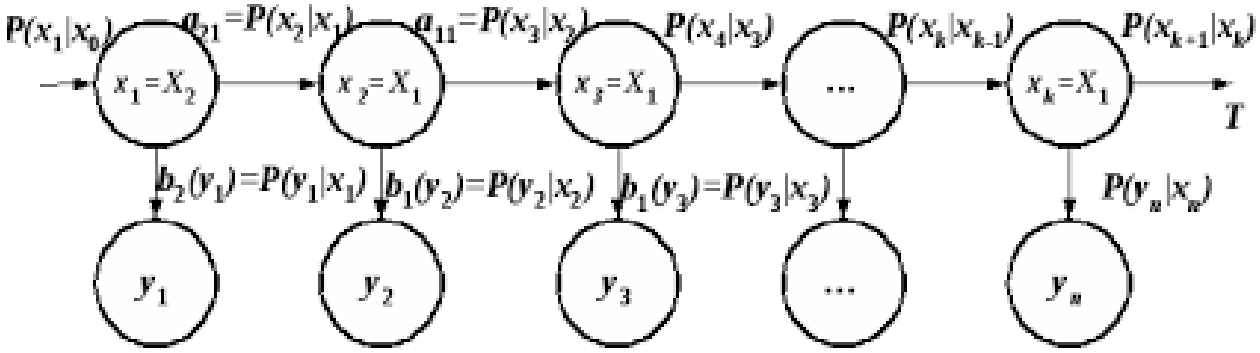

Рисунок 3.3 – Скрытая марковская модель

Для разгадывания неизвестных параметров необходимо также знать вероятности переходов между скрытыми состояниями (1) и вероятности наблюдаемых результатов (2). Эта информация генерируется заранее при «обучении» модели на большом наборе размеченных аудиоданных.

Для вычисления наиболее вероятного значения скрытой последовательности в CMU Sphinx используется алгоритм Витерби – алгоритм динамического программирования для поиска наиболее вероятной скрытой цепи Маркова в скрытой марковской модели; этот список состояний называется путём Витерби. Обозначим множество скрытых состояний $H = \{h_1, h_2, \dots, h_n\}$, множество наблюдаемых результатов – $V = \{v_1, v_2, \dots, v_k\}$, Y_t – наблюдаемое в момент времени t значение, X_t – соответствующее ему скрытое состояние. Через s_i обозначим вероятность нахождения скрытой цепи в состоянии h_i в начальный момент времени, матрица A_{n*n} – матрица вероятностей перехода, где a_{ij} – вероятность перехода из состояния h_i в состояние h_j. Если имеется последовательность наблюдаемых результатов $\{Y_1, Y_2, \dots, Y_K\}$, то наиболее вероятная последовательность скрытых состояний может быть вычислена с помощью рекуррентных формул:

$$P_{1i} = P(Y_1|X_i) * s_i; \quad (3)$$

$$P_{ji} = max_{h \in H}(P(Y_j|X_i) * a_{arg(h),i} * P_{j-1,arg(h)}); \quad (4)$$

Здесь P_{ji} – вероятность наиболее вероятной цепи скрытых состояний длины t, конечное состояние которой равно $h_i, arg(h)$ – позиция элемента h в множестве скрытых состояний H. Для восстановления пути Витерби необходимо запоминать, какое состояние $h \in H$ использовалось в уравнении (4). Определим функцию *St(i, j)*:

$$St(i,j) = \begin{cases} arg\ max_{h \in H}(P(Y_j|X_i) * a_{arg(h),i} * P_{j-1,arg(h)}), j > 1, \\ i, j = 1 \end{cases}$$

Тогда наиболее вероятная скрытая цепь:

$$X_K = arg\ max_{h \in H}(P_{T,arg(h)}),$$

$$X_{i-1} = St(arg(X_i), i).$$

На этапе языкового моделирования необходимо на основе фонем определить произнесённые слова. Задачей языковой модели является определение вероятности произнесения слов и различных словосочетаний. Данные вероятности значительно отличаются в зависимости от тематики текста. Именно поэтому использование для узкоспециализированных задач специального словаря даёт значительный прирост точности распознавания. Для решения задачи в CMU Sphinx используются N-граммы и конечный автомат-преобразователь. N-граммы – это последовательность из N элементов, в нашем случае слов. При генерации статистической языковой модели используются тексты по тематике, для которой планируется использовать модель. На их основе рассчитываются вероятности появления для всех встреченных N-граммы. CMU Sphinx использует униграммы, биграммы и триграммы (рисунок 3.4) как компромисс между точностью и скоростью: на вероятность появления слова в конкретной позиции в основном влияют предыдущее и последующее слова, влияние отстоящих дальше менее значимо, в то время как количество 4-грамм уже значительно больше и их использование требует гораздо больше вычислений.

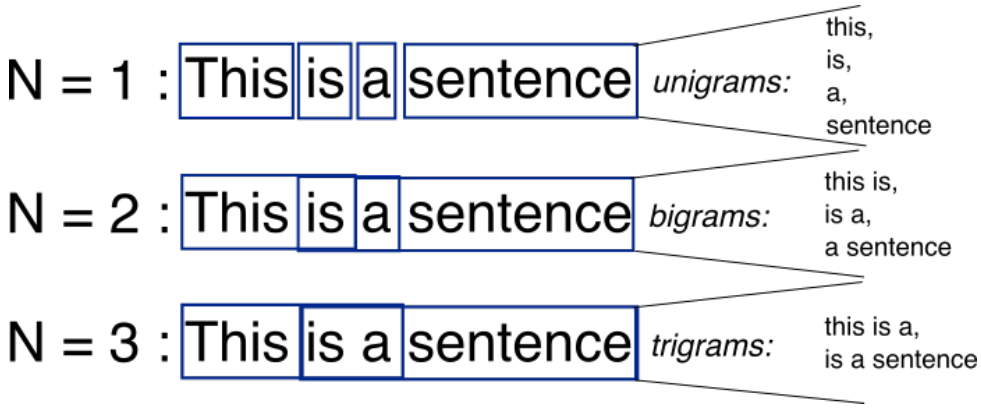

Рисунок 3.4 – Уни-, би- и триграммы

Конечный автомат-преобразователь в качестве входного алфавита содержит фонемы, выходного – слова, его начальное состояние это последовательность полученных на предыдущем этапе фонем, из которого генерируется наиболее вероятная последовательность слов. Такая грамматика является достаточно гибкой и наиболее подходит для распознавания текстов на естественном языке. Это означает, что если модель обучена на данных, состоящих из двух фраз «сто двадцать четыре» и «триста сорок четыре», то сочетание «сто сорок четыре» также будет считаться возможным, пусть и с меньшей вероятностью. Однако такое поведение может быть нежелательным и снижать точность распознавания в случае, если набор команд строго

формализован. В такой ситуации CMU Sphinx позволяет использовать жёстко заданные грамматические модели в формате грамматики речи Java (JSGF):

```
#JSGF V1.0;

grammar numbers;

public <numbers> = (сто двадцать | триста сорок) (четыре);
```

Грамматике выше будут соответствовать только два словосочетания: «сто двадцать четыре» и «триста сорок четыре». Это увеличивает точность и повышает скорость распознавания, однако может приводить к неудаче при ошибках диктора. В нашей работе было принято решение использовать грамматику речи Java ввиду того, что ценность для распознавания представляют только фразы, для которых существует представление в виде анимации, остальные всё равно будут проигнорированы, а быстродействие, наоборот, весьма значимо.

3.3 Реализация базы данных

Для хранения словарей необходима база данных – набор таблиц, связанных определёнными отношениями. В качестве базы данных для проекта было решено использовать SQLite — компактную встраиваемую СУБД. Данный вариант лучше всего походит для не высоконагруженной системы, хранящей достаточно малый объём данных, в не предназначенном для масштабирования приложении.

База данных приложения состоит из набора команд, который был составлен инструктором автошколы и анимированных изображений. Для хранения слов, доступных пользователям, понадобится таблица, которая имеет поля: ID слова или выражения – сгенерированный искусственный номер лексической единицы, непосредственно сама лексическая единица и путь к анимации, которая хранится в локальных ресурсах приложения. Список словарей сохраняется в другую таблицу, каждая запись которой также содержит уникальный идентификатор словаря и его название. Ещё одна таблица используется для связи словарей и соответствующих им слов по их уникальным идентификаторам.

SQLite является файловой базой данных. Физически база данных представляет из себя один файл, хранящийся в ресурсах приложения. При использовании SQLite в приложении их связь производится с помощью функциональных и прямых вызовов файлов, содержащих данные, без использования какого-либо интерфейса. Это повышает скорость выполнения запросов.

3.4 Клиентское приложение

Клиентское приложение, разработанное для операционной системы Android, используется пользователем для осуществления перевода и удобного доступа к словарям. Кроме того, пользователь использует клиентское приложение для изменения своего настроек приложения и скачивания необходимых ему целевых словарей.

При запуске приложения перед пользователем появляется главный экран, в связи с чем он может быстро и беспрепятственно начать работу.

Разделы приложения служат для группировки информации, предоставляемой пользователю, и включают в себя:

- раздел «Перевод» позволяет осуществлять основную работу по переводу и просмотру его результатов;
- раздел «Настройки» позволяет скачивать дополнительные словари, настраивать чувствительность голосового распознавания и активационной фразы, указывать эту фразу;
- раздел «Словарь» позволяет ознакомиться с целевым словарем, загруженным на устройство, просматривать соответствующую словам анимацию жестов.

ГЛАВА 4 СРЕДСТВА, ИСПОЛЬЗУЕМЫЕ ДЛЯ РАЗРАБОТКИ ПРИЛОЖЕНИЯ

4.1 Интегрированная среда разработки

Для реализации клиентского приложения и работы с базами данных была выбрана интегрированная среда разработки Android Studio от Google. Основные её преимущества: бесплатность для любого использования, в том числе в коммерческих проектах, и наилучшая поддержка разработки под систему Android, в том числе функции:

- сборка приложений, основанная на Gradle;
- рефакторинг кода;
- статический анализатор кода Lint, выявляющий такие проблемы, как несовместимость версий и недостатки производительности;
- редактор макетов, позволяющий легко переключаться между редактированием xml-файла и представлением WYSIWYG;
- возможность предпросмотра макетов на различных конфигурациях экрана;
- отладка приложения как на реальном устройстве, так и с помощью встроенного эмулятора Android Virtual Device, в котором можно настроить любую комбинацию разрешения, размера экрана и версии операционной системы [7].

4.2 Захват движения. Технология Leap Motion

Для создания анимации было решено использовать технологию захвата движения (MOCAP), которая позволяет перенести жесты реального человека на цифровую 3-D модель путём распознавания и синхронизации определённых ключевых точек на теле человека.

Для захвата движения рук предполагается использование Leap Motion. Это технология, основанная на захвате движения, осуществляющая человеко-компьютерное взаимодействие [8]. Ее разработала компания OcuSpec, основанная в 2010 г. Устройство представляет собой небольшой USB-адаптер, верхняя часть которого создает невидимую 3D-область взаимодействия достаточно большого объема, которую можно себе представить как куб с ребром в 61 см. Внутри данной области Leap Motion может отслеживать движение не только пальцев и рук пользователя, но также и других предметов: карандашей, ручек, указок и т.п. Удается это с помощью установленных в устройстве двух камер и трех ИК-светодиодов. Стоит отметить высокую скорость и точность захвата, она достигает 200 кадров в секунду. Для разработчиков предлагается абсолютно бесплатный SDK, который работает с 14 различных платформ и библиотек. Среди них есть такие известные

платформы, как Unity 3D и Unreal Engine, также реализована поддержка различных языков программирования, например Processing (в том числе специально написанная на его базе среда разработки LeapMotionP5), C++, Objective-C, Java, JavaScript, AS3 и другие. Работа с предложенным SDK не составляет большого труда благодаря тому, что к нему прилагается подробная документация и множество примеров. Среди минусов можно отметить небольшое количество предопределенных движений, что вынуждает программиста самому пытаться математически описывать более сложные жесты (например, полет птицы, волну и т.д.). Некоторые разработчики решили самостоятельно начать работу над универсальным плагином или инструментом, который мог бы просто записать показанное движение, а потом его использовать, что могло бы сильно облегчить работу множеству программистов. Принцип работы устройства прост – инфракрасные (ИК) диоды освещают руки, а инфракрасные камеры их фиксируют, передавая изображения на программный обработчик Leap Motion. На уровне программного обеспечения используются математические алгоритмы, которые различают контуры рук и отслеживают координаты пальцев. Начиная с версии SDK 2.0 Leap Motion научился выделять составные части руки, проще говоря, алгоритм определяет кости рук и запястье, отслеживает их перемещение в пространстве. Таким образом, открываются новые горизонты для расширения базы распознаваемых жестов.

Распознавание рук происходит достаточно быстро, но скорость зависит от мощности компьютера, на котором, собственно, и происходит обработка данных, полученных с двух камер.

К недостаткам можно отнести невозможность распознавать жесты, которые требуют повернуть руку ребром к устройству. Также Leap Motion некорректно распознает жесты, в которых две руки соединяются вместе. Примеры таких жестов приведены на рисунке 4.1.

Leap Motion — это принципиально новый контроллер, который не только может заменить сенсорные экраны, но и может использоваться для взаимодействия с пользователем в устройствах виртуальной и дополненной реальности.

Если смотреть с точки зрения разработчика программного обеспечения, использования контроллера весьма удобно за счёт удобного интерфейса для управления Leap Motion. Инструменты для разработки приложений с использованием контроллера доступны на множестве языков программирования, таких как C++, Java, Python, JavaScript. Программный интерфейс контроллера передаёт в использующую его программу распознанные в кадре объекты, такие как руки, пальцы, указки. Снятие кадра инфракрасными камерами и распознавание в нем объектов происходит без

участия прикладной программы. Для настройки и калибровки контроллера можно использовать специальную программу, которая выводит на экран кадр, снятый камерами контроллера, и накладывает на него скелетное изображение распознанных объектов. Так значительно проще обнаружить ошибки распознавания и бороться с ними.

Рисунок 4.1 – Пример некорректно распознаваемых жестов

4.3 Программное обеспечение для 3-D анимации

iClone — разработка компании Reallusion, программное обеспечение для создания 3D-анимации в режиме реального времени. Работа в этом режиме обеспечивается использованием игровых движков для отрисовки анимации на экране. Также iClone предоставляет возможность создания полной анимации лица и применение захвата движения для переноса их на анимированных персонажей. Эти функции необходимы в нашем случае и их наличие обуславливает выбор ПО. Также плюсом iClone является возможность бесплатного использования без каких-либо последующих ограничений на работу с созданной с его помощью анимации.

Самой сложной задачей является автоматизация создания анимации лица с помощью захвата движения. Обычно мимика у анимированных персонажей прорисовывается в значительной степени вручную, при этом художники не

обязательно в точности воспроизводят реальные движения мышц лица. Однако с учётом наших ограниченных ресурсов и специфики требований было необходимо реализовать другой подход, который бы позволил создавать анимацию достаточно быстро и в то же время качественно, чтобы мимика в точности соответствовала реальной.

Для решения этой задачи было использовано приложение Blender, распространяемое под свободной лицензией. Оно позволяет считывать мимические жесты с помощью нанесённых на лицо маркеров. Для этого сначала записывается видео, которое затем обрабатывается в Blender, получая на выходе анимацию, которую можно использовать в Unity.

Предполагается обрабатывать данные, полученные с Leap Motion, в iClone, а затем экспортировать полученную анимацию в среду разработки Unity 3D. Преимуществами данной среды можно считать кроссплатформенность, популярность, доступность. Платформа поддерживается и постоянно обновляется. Найти необходимую информацию, дополнительное программное обеспечение или расширить возможности стандартной платформы не составит труда.

ГЛАВА 5 ОБЗОР СУЩЕСТВУЮЩИХ АНАЛОГОВ

Большинство существующих систем машинного сурдоперевода создано для работы на английском языке, примером таких систем может послужить проект ViSiCAST. Наиболее близко к текущему проекту приложение HandTalk, Приложение работает с виртуальным переводчиком Хьюго, который отвечает на голосовые и текстовые команды, преобразуя речь в жесты в режиме реального времени. Это также позволяет слушателям учиться общаться на языке жестов. Однако HandTalk может работать только на португальском языке, на данный момент также реализуется поддержка английского [9].

Примером русскоязычной разработки является российская система «Сурдофон». «Сурдофон» – компания-создатель революционного приложения для облегчения коммуникации для людей с нарушениями слуха, использующими русский жестовый язык, который распространён в России, Беларуси, Казахстане. Приложение «Сурдофон» — коммуникатор, который распознает речь говорящего собеседника и переводит ее на русский жестовый язык – это делает анимированное изображение человека. Со своей стороны человек с ограничениями по слуху набирает текст, который через эту же анимацию озвучивается компьютерным синтезатором речи. Таким образом, каждый получает информацию наиболее удобным для себя способом: глухой человек – на жестовом языке, слышащий – на обычном [10].

К достоинствам программы «Сурдофон» можно отнести большой объём жестового словаря — около 10000 жестов. Однако есть и недостатки: программа требует подключения к серверам компании и, соответсвенно, работает только при наличии интернета. Также «Сурдофон» не может переводить речь синхронно.

ЗАКЛЮЧЕНИЕ

В ходе работы над проектом была изучена проблематика задачи и обоснована её актуальность. В процессе общения с целевой группой был составлен список необходимых функций и возможностей приложения.

Далее были изучены уже существующие решения с целью анализа их преимуществ и недостатков, определения степени соответствия предъявленным требованиям и необходимости реализации собственного проекта.

После определения того, что все имеющиеся решения в той или иной степени не соответствуют требованиям, было принято решение реализовать приложение и изучены технологии, применимые для решения проблемы.

Была выбрана платформа для реализации приложения и определена его архитектура, определён перечень необходимых технических и программных средств. Был сделан выбор в пользу таких инструментов:

1. Распознавание речи — библиотека CMU Sphinx;
2. Захват движений рук — Leap Motion;
3. Создание анимации — iClone 7 и Unity 3D.

После выбора данных инструментов было проведено обучение работе с ними, рассмотрены примеры использования, и они были успешно применены в прототипе приложения.

Был создан тестовый прототип приложения. На данный момент в нем доступны следующие функции: ввод голосовой команды и получение видеофрагмента, который демонстрирует команду на языке жестов.

Также были определены цели на ближайшее будущее, а именно:

1. Реализация полной версии приложения;
2. Создание словаря, включающего все необходимые фразы и жесты;
3. Тестирование приложения в реальных условиях, исправление возможных недостатков.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. CMU Sphinx [Электронный ресурс] / Carnegie Mellon University. – 2019. – Режим доступа: https://cmusphinx.github.io/. – Дата доступа: 25.01.2020.
2. Меню // Android Developers [Электронный ресурс]. – 2019. – Режим доступа: https://developer.android.com/guide/topics/ui/menus. – Дата доступа: 15.04.2020.
3. Android – Программирование для профессионалов / Б. Харди [и др.]; под общ. ред. Б. Харди. — 2-е изд. — СПб.: Питер, 2016. – 640 с.
4. Беленко, М. В. Сравнительный анализ систем распознавания речи с открытым исходным кодом / М. В. Беленко, П. В. Балакшин // Международный научно-исследовательский журнал [Электронный ресурс]. – 2017. – Режим доступа: https://research-journal.org/technical/sravnitelnyj-analiz-sistem-raspoznavaniya-rechi-s-otkrytym-kodom/. – Дата доступа: 25.01.2020.
5. Kaldi [Электронный ресурс] / Daniel Povey. – 2013. – Режим доступа: http://kaldi-asr.org/doc. – Дата доступа: 03.02.2020.
6. CMU Sphinx Wiki [Электронный ресурс] / Carnegie Mellon University. – 2019. – Режим доступа: http://cmusphinx.sourceforge.net/wiki/. – Дата доступа: 25.03.2020.
7. Android Studio [Электронный ресурс] / Google LLC. – 2020. – Режим доступа: https://developer.android.com/studio/. – Дата доступа: 15.04.2020.
8. Попов, Д. И. Технология Leap Motion и ее применение в образовательных процессах / Д. И. Попов, Д. Г. Григорьевич, М. В. Алпатова // CYBERLENINKA [Электронный ресурс]. – 2015. – Режим доступа: https://cyberleninka.ru/article/n/tehnologiya-leap-motion-i-ee-primenenie-v-obrazovatelnyh-protsessah. – Дата доступа: 14.03.2020.
9. HandTalk [Электронный ресурс] / Hand Talk. – Maceió, 2019. – Режим доступа: https://www.handtalk.me/en. – Дата доступа: 27.02.2020.
10. Сурдофон [Электронный ресурс] / ООО «Сурдофон». – 2019. – Режим доступа: https://www.xn--d1ascahfol.xn--p1ai/. – Дата доступа: 27.02.2020.