РАЗРАБОТКА КЛИЕНТ-СЕРВЕРНОГО ПРИЛОЖЕНИЯ ПО РАСПОЗНАВАНИЮ ПАРТИТУР МУЗЫКАЛЬНЫХ ПРОИЗВЕДЕНИЙ

В. Ю. Дементей, А. А. Егоров

Белорусский государственный университет, Минск, Беларусь
fpm.dementey@bsu.by, fpm.egorovaa1@bsu.by,
науч. рук. – С. В. Марков, доц, зав. кафедрой

Важной частью разработки приложения, которое работает с музыкальными партитурами, является первичная обработка входных данных, а именно нотных листов. В статье описывается применение интеллектуальной обрезки изображения. Приведен алгоритм нахождения линеек нотного стана. Для удобной работы с клиентской частью рассмотрены и проанализированы варианты веб и мобильного приложений на платформе Android. Так как важной частью работы с нотами является наглядность на стороне клиента, осуществлена работа с музыкальной нотацией, рассмотрены преимущества ее использования. Описан прототип приложения конвертации музыкальных партитур в аудио формат с возможностью выбора или фотографирования изображения, выгрузки результирующего файла и дальнейшим его проигрыванием.

Ключевые слова: интеллектуальная обрезка изображения, клиент-серверное приложение, музыкальная нотация, мобильное приложение, одноканальное изображение.

ВВЕДЕНИЕ

Не мало важным процессом 21 века является оцифровка данных с целью получения электронных копий пригодных для более удобной работы, например, электронные книги, цифровые фото и видео, mp3 и т.д. Но само наличие цифровой копии, хранящейся в памяти компьютера не всегда решает проблему удобной работы с ней. При необходимости правки ошибок или копировании части текста нам необходим не просто цифровой скан носителя, но и представление его в виде понятном компьютеру. Ведь само изображение это всего лишь набор упорядоченных пикселей и невозможно создать универсальное программное обеспечение способное воспринимать их также как это делает человеческий мозг. Исходя из этого перед этапом распознавания нот необходимо улучшить изображение и привести его в наиболее пригодную форму, которая позволит снизить число ошибок и сбоев в работе других алгоритмов. Также в возможности серверной части распознавания партитур должна входить работа с изображением, в ходе которой четко определяется расположение горизонтальных полос нотного стана.

ИНТЕЛЛЕКТУАЛЬНАЯ ОБРЕЗКА ИЗОБРАЖЕНИЯ

Предположим, что изображение, полученное от пользователя, имеет небольшой наклон относительно горизонтали, необходимо это исправить. В начале применим двусторонний фильтр, позволяющий очистить изображение от нежелательных шумов, при этом оставляя контуры всех фигур неизменными.

После этого найдем границы всех объектов на изображении, воспользовавшись алгоритмом, разработанным Джоном Канни [1]. Следующим этапом необходимо получить координаты найденных контуров, чтобы определить какой из них является контуром нотной партитуры и произвести обрезку изображения по его периметру. Необходимо аппроксимировать все найденные контуры до простых фигур.

На последнем шаге необходимо выбрать именно тот аппроксимированный контур, который является искомым. Для этого отсортируем их всех по площади. После этого будем перебирать отсортированный массив начиная от самого большого в поисках того четырехугольника, который содержит внутри себя прямоугольники меньших размеров и имеющих углы приближенно равные прямым. При таком подходе будет найден самый большой прямоугольник, содержащий внутри себя все неверно найденные контуры фигур. При отсутствии контура, удовлетворяющего всем условиям, считаем, что попытка интеллектуальной обрезки не удалась и завершаем работу алгоритма.

АЛГОРИТМ НАХОЖДЕНИЯ ЛИНЕЕК НОТНОГО СТАНА

Необходимыми входными данными для этого алгоритма являются верхняя и нижняя граница нотного ключа, а также горизонтальное расположение ноты.

Далее приведено описание алгоритма. Для поиска координат будем устанавливать горизонтальный курсор в определенную позицию стана и перебирать все пиксели сверху вниз начиная от верхней границы стана, заканчивая нижней границей вдоль этого курсора.

Сперва необходимо разделить все пиксели по цветовому признаку – белые или черные. Так возможно будет определить какие пиксели принадлежат черной линейке, а какие являются промежутками между ними.

Переведем изображение в градации серого, тем самым закодируем каждый пиксель однобайтным значением характеризующем его прозрачность. После получения одноканального изображения избавимся от возможных шумов. Для этого применяется механизм усреднения значения пикселей. При рассмотрении конкретного пикселя вычисляются значения

всех его соседей и определяется математическое ожидание в доверительном интервале. Вычисленное значение заменяет исходный пиксель. Последним шагом получим само черно-белое изображение.

Вторым этапом необходимо найти горизонтальную позицию для курсора где нет ни одной ноты, чтобы при определении цвета пикселя, если он черный мы точно могли понимать, что это линейка стана, а не часть ноты.

Стартовую позицию курсора установим по горизонтали на центр выбранной ноты, по вертикали в верхнюю границу стана и начнем перебор всех пикселей сверху вниз пока не дойдем до нижней границы стана.

Если очередной пиксель классифицируется как белый, то пропускаем его, и смещаем курсор на один пиксель вниз.

Если же курсор встречает пиксель, классифицируемый как черный, то запоминается вертикальная позиция курсора (точка подозрительная на верхнюю границу линии) и запускается внутренний цикл, пропускающий все черные пиксели, пока не встретится белый. После выхода из внутреннего цикла получаем точку подозрительную на нижнюю границу линейки.

Добравшись до самой последней позиции курсора, которой является нижняя граница рассматриваемого стана, получаем массив, содержащий верхние и нижние позиции черных линий подозрительных на искомые данные. Подозрительными они являются, потому что курсор мог стать в горизонтальную позицию где по вертикали есть ноты или попасть на границу такта.

В результате работы алгоритма получаем массив из пяти элементов, каждый из которых содержит верхнюю и нижнюю границ соответствующей линейки на нотном стане для конкретной ноты.

РЕАЛИЗАЦИЯ ПРИЛОЖЕНИЯ

Процесс взаимодействия клиента и сервера строится по следующему сценарию: клиент отправляет запрос, содержащий некоторые входные данные, по сетевому соединению, которые затем обрабатываются. Принцип работы данного клиент-сервера заключается в следующем: на вход поступают данные (в нашем случае изображение нотной партитуры). Эти данные сохраняются в базе данных. Затем алгоритм распознавания находит все ноты в отправленном изображении и создает в соответствии для каждой ноты объект с параметрами длительности и тональности [2].

После этого, полученные объекты переводятся в MIDI – формат по соответствующим правилам. Когда файл будет получен он отправляется обратно на сторону клиента, где может быть воспроизведён.

Для удобства реализовано два клиента. Плюсы клиентской части на веб платформе заключаются в отсутствии большого объема занимаемой

памяти, возможности трансляции приложения на разных устройствах. А преимуществами Android приложения являются легкодоступность, возможность возвращаться к предыдущим результатам, возможность музыкальных плееров телефона поддерживать MIDI формат [3].

На клиентах также использовалась музыкальная нотация – формат для указания нот, использующий только строку символов. То есть для каждого действия и объекта существует свое соответствие при письме. Что помогает решать проблемы с графическим и функциональным представлением, например, осуществлять обрисовку нот, быстро редактировать погрешности распознавания, добавлять анимационные эффекты, синхронизировать мелодию и изображение.

ЗАКЛЮЧЕНИЕ

Как итог, с использованием вышеприведенных алгоритмов, разработано мобильное приложение на платформе Android, а также веб реализация, позволяющие конвертировать нотные партитуры музыкальных произведений в аудио формат. В ходе реализации можно выделить плюсы клиент-серверной архитектуры: простота написания различных клиентов, быстрое внесение правок, низкие технические требования, предъявляемые к клиентам, меньшее время, затрачиваемое на получение результата.

Библиографические ссылки

1. Canny Edge Detection Step by Step in Python — Computer Vision // Towards Data Science [Электронный ресурс]. – 2019. - Режим доступа: https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d812
2. Кизенков В. С., Полетаева Т. С. Распознавание партитур музыкальных произведений и преобразование их в MIDI файлы // 77-я научная конференция студентов и аспирантов Белорусского государственного университета [Электронный ресурс]: материалы конф. В 3 ч. Ч. 1, Минск, 14–22 мая 2020 г. / Белорус. гос. ун-т; редкол.: В. Г. Сафонов (пред.) [и др.]. – Минск : БГУ, 202
3. Архитектура Android // Fandroid [Электронный ресурс]. – 2010. Режим доступа: https://www.fandroid.info/category/arhitektura-klient-servernyh-android-prilozhenij/.