# Введение

В рамках данной работы будет предложено реализовать алгоритм [1], позволяющий идентифицировать музыкальную композицию по её фрагменту, который лежит в основе известного сервиса Shazam [2]. Алгоритм будет реализован с использованием средств MATLAB.

Коротко алгоритм можно изложить следующим образом:

1. Создание базы, содержащей информацию об особенностях некоторого набора музыкальных композиций;
2. Запись фрагмента композиции (содержащейся в базе) и поиск его особенностей;
3. Сопоставление особенностей фрагмента и особенностей композиции в базе.

Под особенностями композиции будет пониматься набор её пиков в определенный момент времени. Под пиком понимается точка на частотно-временной плоскости, которая имеет наибольшую магнитуду среди соседних точек.

База строится из отпечатков (fingerprint) или карты созвездий (constellation map) музыкальной композиции. Каждый отпечаток состоит из набора пар пиков (f1, t1) и (f2, t2) с их расстоянием друг от друга в частотно-временной области (f1, f2, t1, t2 - t1). Таким образом, отпечаток композиции представляет собой таблицу вида:

|  |  |  |  |
| --- | --- | --- | --- |
| f1 | f2 | t1 | t2 - t1 |
| … | | | |
| fi | fj | ti | tj – ti |
| … | | | |
| fm | fn | tm | tn – tm |

После того как база построена, получаем и подготавливаем фрагмент этой композиции. Он может быть записан через микрофон, либо вырезан из оригинала и зашумлен. Для сэмпла строится отпечаток, после чего в базе ищется соответствие композиция с наибольшим соответствием.

# Содержание работы

Задачу можно разбить на 2 крупные части: построение базы и поиск отпечатка в базе.

## Построение базы

В данном разделе необходимо будет написать скрипт, который будет создавать базу отпечатков для набора музыкальных композиций. Скрипт должен делать следующее:

1. [Прочитать файл и получить среднее значение двух каналов аудио и объединить в один](#_Чтение_файла_и);
2. [Получить спектрограмму](#_Получение_спектрограммы);
3. [Найти пики на спектрограмме](#_Находим_пики_на);
4. [Получить адаптивный порог количества пиков в секунду](#_Получаем_адаптивный_порог);
5. [Для каждого пика получить пары в целевом окне и добавить их в таблицу](#_Получаем_пары_пиков);
6. [Подготовить базу композиций](#_Подготовка_базы).

Желательно разделять логику скрипта на отдельные логические блоки и размещать их в отдельные функции или даже отдельные файлы.

Теперь рассмотрим каждый пункт подробнее.

### Чтение файла и объединение стереоканалов

Чтобы получить список аудиофайлов из папки, можно воспользоваться готовым скриптом GetFilesWithExtensions, который возвращает все файлы с заданным расширением из папки.

Дальше для каждой композиции необходимо получить отпечаток. Функцию получения отпечатка рекомендуется поместить в отдельный файл с соответствующим названием.

Получить данные из любого аудиофайла средствами MATLAB можно с помощью функции

[fileData, sampleRate] = [audioread (filePath);](https://www.mathworks.com/help/matlab/ref/audioread.html).

В fileData будет лежать 2 массива данных, соответствующих 2-м стереоканалам. Но для обработки нам необходимо объединить их в один. В рамках данной задачи достаточно просто получить среднее значение по стереоканалам. Этом можно сделать с помощью функции

meanChannels = [mean(fileData, 2)](https://www.mathworks.com/help/matlab/ref/mean.html);.

Возможно еще нужен resample.

### Получение спектрограммы

Получить спектрограмму сигнала можно вызвав функцию

[S, F, T, P] = [spectrogram(data,window,noverlap,nfft,sampleRate)](https://www.mathworks.com/help/signal/ref/spectrogram.html);, где S – спектрограмма в виде матрицы, F – вектор частот в герцах для оси ординат, T – вектор временных отсчетов для оси абсцисс, P – матрица спектральной плотности мощности (PSD).

Если необходимо вывести спектрограмму на экран, то можно воспользоваться следующим кодом:

surf(T, FkHz, PdB, 'edgecolor', 'none');

axis tight;

xlabel('Time (seconds)');

ylabel('Frequences (kHz)');

view(0, 90);

Для вывода магнитуды мы преобразуем спектральную мощность в децибелы (PdB = 10 \* log10(P)), а частоту выводим в килогерцах (FkHz).

### Получение пиков на спектрограмме

Далее необходимо найти локальные пики на спектрограмме. Один из способов сделать это – перебрать каждую точку спектрограммы и сравнить её с соседними в некотором квадрате.

Строим логическую матрицу, совпадающую по размеру с матрицей спектральной плотности мощности (P). Назовем её peaks.

1. Используя метод

shiftedM = [circshift(M,[i, j])](https://www.mathworks.com/help/matlab/ref/circshift.html);,

где i – сдвиг матрицы M по оси ординат, j – сдвиг матрицы M по оси абсцисс,

сдвигаем матрицу в нужном направлении.

1. Считаем разницу между исходной и сдвинутой матрицами и отсекаем элементы меньше или равные нулю:

peaks = (peaks & (M - shiftedM) > 0);.

1. Повторяем пункты 2 и 3 для нужных направлений с нужным сдвигом.

Например, при сдвиге [0, -1] и дальнейшем выполнении пункта 3, мы получим логическую матрицу, в которой элементы большие чем их сосед слева будут иметь значение 1 (true).

При решении данной задачи может пригодиться функция

[rowIds, colIds] = [find(peaks)](https://www.mathworks.com/help/matlab/ref/find.html);,

которая позволяет получить индексы ненулевых элементов матрицы.

Получение пиков можно вынести в отдельный файл.

Попробуйте поэкспериментировать с размером квадрата. Для этого можно воспользоваться вспомогательной функций ViewMatrix, которая находится в файле с соответствующим названием. Она позволит наглядно увидеть каждый шаг и результат эксперимента.

Для вывода пиков поверх спектрограммы можно воспользоваться функцией

[scatter3(T, FkHz, PdB, '\*r'](https://www.mathworks.com/help/matlab/ref/scatter3.html));,

где векторы T, FkHz и PdB – координаты пиков, а '\*r' означает, что эти координаты будут отображены красными знаками '\*'.

### Получение адаптивного порога количества пиков в секунду

В виду того, что качество записи сэмпла музыкальной композиции для её детектирования непредсказуемо, должны быть выбраны только наибольшие пики из найденных. Но их количество ограничено максимальным количеством пиков в секунду. Самый простой путь – отфильтровать пики по некоторому порогу.

Подумайте, какой оптимальный порог выбрать, чтобы он соответствовал некоторому требованию количества пиков в секунду (например, 30 пиков в секунду). Напишите функцию, которая будет выбирать порог автоматически.

Отфильтруйте пики по выбранному порогу и выведите их в отдельный график, сравните результаты. Насколько близко расположены пики друг от друга? Равномерно ли они распределены?

Попробуйте разделить пики на части длительностью примерно в секунду и получить порог для каждой части отдельно. Отфильтруйте каждую часть с найденным для неё порогом и выведите результат. Какие преимущества имеет эта фильтрация в сравнении с предыдущей?

### Получение пар пиков в целевом окне и добавление их в таблицу

Мы хотим выбрать пары пиков и получить матрицу, содержащую частоту каждого и них, время начала первого, разницу по времени. Иными словами, набор кортежей (fi, fj, ti, tj – ti). Таких пар может получиться слишком много, поэтому введем дополнительное ограничение на их взаимное расположение и количество.

Например, выбрав пик (f1, t1), обозначим для него целевое окно, которое будет лежать от него на расстоянии ∆l, иметь длину по времени ∆w и высоту по частоте 2∆h. Тремя ограничим количество возможных пар для одного пика (далее будем называть эти пики веером (fan out). Таким образом, для пика (f1, t1) каждый пик из веера должен лежать в целевом окне f1 - ∆h < f2 ≤ f1 + ∆h и t1 + ∆l < t2 ≤ t1 + ∆w. После того, как пики найдены, составляем кортеж (f1, f2, t1, t2 - t1) и добавляем в таблицу.

Основная задача на этом этапе – найти оптимальные параметры ∆l, ∆w, ∆h и веер, после чего найти пики, удовлетворяющие данным параметрам.

Отобразить результаты поверх всего, что было отображено выше можно использовав функцию [line](https://www.mathworks.com/help/matlab/ref/line.html).

Попробуйте поэкспериментировать, меняя параметры ∆l, ∆w, ∆h и веер. Также попробуйте менять при этом размер окна для спектрограммы.

### Подготовка базы

База будет представлять собой хэш-таблицу, в которой для каждого кортежа (f1, f2, t1, t2 - t1) будет храниться информация о начальном времени (t1) и о том, какая это композиция (songId).

Хэш будет получаться с помощью простой хэш-функции

h(f1, f2, t2 – t1) = t2 – t1) \* 216 + f1 \* 28 + f2,

где каждая частота должна быть представлена 8 битами, то есть иметь значение 0-255, а временной интервал, в зависимости от размера выбранного целевого окна, может быть представлен некоторым числом бит.

При заполнении таблицы неизбежны повторения хэш-значений. Необходимо подумать, как можно уменьшить количество таких повторений ([separate chaining](https://ru.coursera.org/learn/algorithms-part1/lecture/qFJf7/separate-chaining), [linear probing](https://ru.coursera.org/learn/algorithms-part1/lecture/He696/linear-probing) и. т. д.). Также подумайте, как метод выбора пар пиков в целевом окне может повлиять на количество повторений. Каким должен быть размер базы данных?

Напишите функцию получения хэша и поместите ее в файл с соответствующим названием. В отдельном файле напишите скрипт добавления отпечатка композиции в базу.

MATLAB позволяет создавать глобальные переменные ([global](https://www.mathworks.com/help/matlab/ref/global.html)), которые можно сохранять в файл и загружать в рабочую область. Создайте в главном скрипте создания базы глобальную переменную HashTable, в которую будут добавляться композиции. После добавления всех композиций эту переменную с помощью функции [save](https://www.mathworks.com/help/matlab/ref/save.html) сохраните в файл.

## Поиск музыкального фрагмента в базе

После того как база построена и сохранена, можно перейти непосредственно к поиску фрагментов музыкальных композиций.

Этот этап можно разделить на два пункта:

1. [Подготовка фрагмента для дальнейшего поиска](#_Подготовка_фрагмента_для);
2. [Поиск и принятие решения о соответствии фрагмента определенной композиции в базе](#_Поиск_и_принятие).

### Подготовка фрагмента для дальнейшего поиска

Прежде чем приступить к поиску, сэмпл музыкальной композиции необходимо получить и подготовить. Можно записать его через микрофон, либо вырезать часть исходной композиции и добавить шумы. Рассмотрим оба варианта.

Чтобы записать фрагмент через микрофон, необходимо выполнить несколько действий:

% Создаем объект записи сэмпла

recorder = [audiorecorder(sampleRate, sampleSize, channelsNum)](https://www.mathworks.com/help/matlab/ref/audiorecorder.html);

% Модальное окно, блокирующее работу во время записи

hMsgBox = msgbox('Recording…');

% Запись блока данных с устройства ввода длительностью duration (секунды)

[recordblocking(recorder, duration)](https://www.mathworks.com/help/matlab/ref/audiorecorder.recordblocking.html);

delete(hMsgBox);

% Получаем данные в виде массива

sample = [getaudiodata(recorder)](https://www.mathworks.com/help/matlab/ref/audiorecorder.getaudiodata.html);

Чтобы получить фрагмент музыкальной композиции можно воспользоваться уже известной функцией, передав ей диапазон:

[fileData, sampleRate] = [audioread(filePath, samplesRange);](https://www.mathworks.com/help/matlab/ref/audioread.html).

Сгенерировать вектор, содержащий аддитивный белый гауссовский шум можно следующим образом:

samplePower = mean(sample .^ 2);

noise = [randn(size(sample))](https://www.mathworks.com/help/matlab/ref/randn.html) \* sqrt(samplePower / 10^(SNRdB/10));

sample = sample + noise;

В дальнейшем мы вернемся к этому моменту, чтобы оценить качество работы алгоритма при разных отношениях сигнал/шум.

Полученный фрагмент можно сохранить в файл с помощью функции

[audiowrite(filePath, sample, sampleRate)](https://www.mathworks.com/help/matlab/ref/audiowrite.html);,

или можно прослушать через объект аудиоплеера:

player = [audioplayer(sample, sampleRate)](https://www.mathworks.com/help/matlab/ref/audioplayer.html);

[play(player)](https://www.mathworks.com/help/matlab/ref/audioplayer.play.html);

### Поиск и принятие решения о соответствии фрагмента определенной композиции в базе

Отпечаток сэмпла получается тем же алгоритмом, который использовался при построении базы. Далее для каждого кортежа из полученного отпечатка с помощью описанной выше хеш-функции hash(fc1, fc2, tc2 – tc1) получаем индекс в базе. По этому индексу получаем пары (ts1, songId), количество которых зависит от коллизии. Необходимо запомнить разницу начала пика в исходной композиции и в сэмпле (ts1 – tc1, songId). После прохождения всего отпечатка сэмпла, получается коллекция значений разниц (tc2 – tc1) для каждой композиции в базе.

Соответственно та композиция, которая имеет наибольшее количество совпадений, звучит в записанном сэмпле. Получить идентификатор этой композиции можно с помощью функции

songId = [mode(songId2TimeOffset)](https://www.mathworks.com/help/matlab/ref/mode.html);.

Чтобы отобразить на графике гистограмму воспользуйтесь функцией [histogram](https://www.mathworks.com/help/matlab/ref/histogram.html).

Подумайте, какие дополнительные проверки можно сделать, чтобы поиск соответствующей композиции более точным.

## Оценка устойчивости алгоритма к шумам

После того, как алгоритм готов, добавьте в базу несколько композиций. Чем больше добавите, тем будет интереснее. Также выше упоминалось как можно сгенерировать и добавить шум к сэмплу.

Чтобы оценить устойчивость алгоритма к шумам, в оригинальной статье [1] было предложено воспользоваться отношениями сигнал/шум, лежащим в диапазоне SNRdB = -15:3:15. Для каждого SNR зафиксируйте процент корректно найденных композиций. Отобразите все на графике.

Попробуйте варьировать длину сэмпла. Как это отобразится на корректности поиска?

Попробуйте протестировать алгоритм на сэмплах, зашумленных речью или другими шумами. Как влияют разные виды шумов?

# Ссылки

1. **Wang Avery Li-Chun** An Industrial-Strength Audio Search Algorithm.
2. **Shazam Entertainment Limited**. - www.shazam.com.