Automatic Detection of Musical Sampling using Scale-Invariant Feature Transform

Colin Fahy

Submitted in partial fulfillment of the requirements for the
Master of Music in Music Technology
in the Department of Music and Performing Arts Professions
Steinhardt School
New York University

Advisor: Dr. Juan Bello

January 19, 2018

Abstract

This paper describes a modified audio fingerprinting approach to detect when a particular audio track has employed musical sampling and identify the source of the sample. The system uses a technique from the field of Computer Vision, the Scale-Invariant Feature Transform for detecting keypoints and extracting keypoint descriptors from an audio spectrogram to create the fingerprints. An evaluation was carried out to measure the performance of the system using examples of popular music recordings and ground truth labels provided by the website whosampled.com. On querying 500 instances of sampling, it was able to achieve a maximum F_1-score of 0.350.

Contents

1 Introduction

Musical sampling is the act of taking a part of one sound recording and reusing it in a different song or piece. Not to be confused with the other term sampling used in digital signal processing, sampling has been used as a creative tool in composition and music production. The technique began in the 1940s with the advent of tape music which spawned the genre of musique concrète. Musical sampling was popularized by the genre of hip-hop in the 1970s. The technique has since developed and spread and is now commonly used in many popular music genres.

Detecting musical sampling is a task related to music similarity, but more specifically audio fingerprinting. While many music similarity tasks focus on higher order features such as tempo, melody, rhythm, or structure, audio fingerprinting is concerned with finding exact matches of a specific recording. Detecting sampling can be likened to "over the air" audio fingerprinting techniques which aim to identify audio being played over speakers and recorded with a microphone, often in a noisy environment. The difference being that instead of the audio fingerprint being obscured by noise, it is embedded within other musical content. There is also an issue that often times the artist may have modified the source audio in several ways such as pitch-shifting, time-stretching, and filtering. Samples appear in many forms and can be categorized into four main types: short, isolated fragments, loops and phrases, larger elements, and transformed material [1]. Short fragments may be difficult to identify, but granularity, identifying audio from as little content as possible, has been a main concern for audio fingerprinting since its inception. Identifying transformed material is a bigger hurdle.

These modifications have already been addressed by audio fingerprinting methods since they are common techniques to attempt to bypass copyright detection. Most notable is a recent method derived from the field of computer vision which uses the scale-invariant feature transform (SIFT) [2]. SIFT keypoints and descriptors taken from a spectrogram image have been shown to be more robust to pitch-shifting and time-stretching modifications when used for audio finger-

printing.
By extracting SIFT descriptors from a large corpus of audio tracks a reference of possible sampling sources can be formed. By extracting similar features from a target track and comparing against the reference to find similarly related keypoints, an instance of sampling can be detected. However, since SIFT descriptors are larger than other fingerprinting feature vectors, a brute-force comparison of every keypoint would be prohibitively slow with a sufficiently large database. This requires the implementation of an approximate nearest neighbor algorithm. This is a technique of quickly finding closely related elements with high probability of finding the closest match, but with no guarantee of finding the absolute nearest neighbor. This may slightly reduce performance compared to a brute-force comparison, but can decrease search time by orders of magnitude. This makes possible finding matches between large feature vectors within a very large dataset.

1.1 Motivation

Knowing when an artist has sampled from another artist can be useful for a number of situations. There is of course the issue of copyright and royalties. Audio fingerprinting has transformed the way copyright infringement is handled in broadcast and user media content hosting applications.
From a musicological perspective sampling can be used to study the influence across different genres, demonstrated by Bryan and Wang who created a system of measuring high-level influence of artists and genres based on data from Whosampled [3]. Influence across music is an interesting topic, but difficult to measure. Sample identification is one objective measure that could potentially give light on how musical influence has spread throughout the years.
It is also useful metadata that could be included in any music database. Sampling information itself can be of interest to end users as websites such as whosampled.com have shown [4]. Whosampled is a website that provides information on which popular music tracks have sampled from others. It has built

it's database over the past 8 years from user crowd-sourced labels and expert analysis. Whosampled is a major inspiration for this project and provides the ground-truth labels used in evaluation.

1.1.1 Goals

This project aims to develop a system that can detect when a particular track has employed musical sampling and identify the source of the sample. Given a database of existing audio files, this system should be able to recognize if an input audio file has sampled from any of the sources in the database. This method must be efficient and scalable to a substantially large database of source music. It should also be able to detect samples, even if the artist has applied signal processing techniques such as pitch-shifting, time-stretching, or filtering.

2 Literature Review

Automatic sample detection is a fairly recent area of music information retrieval that has only been directly addressed in a few previous papers. However, the concept is merely a new application of techniques that have been well researched for music identification, classification, and comparison. This literature review will discuss the work that has been done in sample detection as well as relevant techniques gathered from the areas of music similarity, cover song recognition, remix recognition, and audio fingerprinting.

2.1 Sample Detection

The task of sample detection has been approached in two other papers. In 2013, Whitney and Leider [5] proposed a method for sample detection in hip-hop music using non-negative matrix factorization (NMF). Their process involves taking the log-spectrogram of each track. Then two tracks can be compared to each other by comparing their frequency content frame by frame. For each frame pair a two-step NMF is computed to compare the similarity across 32 frequency

bins. Their results showed up to 78.8% accuracy, and was able to detect different type of samples including vocals, drums, and multiple simultaneous instrument samples. However, this was only evaluated with a single album against a corpus of only 33 sampled tracks. The authors acknowledge that this approach is not feasibly scalable to a large corpus, but rather could be useful as a method of validation to confirm a proposed sampling between two tracks.

In 2012 van Balen et al. [6] proposed a modified audio fingerprinting approach to sample identification in hip-hop music. They used the method proposed by Wang [7] modified to use a constant Q transform (CQT) instead of the Fourier transform. CQT logarithmically spaces the bins in the frequency domain bringing more importance to the lower frequencies which van Balen found to contribute most to the performance. Evaluation was performed using 68 candidate tracks against a corpus of 396 reference tracks containing a total of 104 ground truth sample relations. To improve performance, they addressed the problem of pitch-shifting by time-scaling the query track several times by a set increment and computing the fingerprints for each rescaled track for lookup against the database. This gave performance of up to 0.39 mean average precision, which is arguably low, but scalable and on par with some of the early audio fingerprinting methods. However, this does not address the possibility of pitch-shifting and time-stretching occurring independent of each other. It is also a brute force approach that could only find samples that have been scaled by one of the discrete factors chosen. The evaluation was done on a dataset of hip-hop query tracks which likely only employed time-scaling and not independent pitch shifting or time stretching. It is reasonable to assume this method would not perform as well on a dataset containing a more general use of sampling in modern popular music.

2.2 Related Processes

2.2.1 Music Similarity

In 2011, Fenet et al. [8] proposed a method using onset detection to determine the fingerprint landmarks. At each landmark the mean chroma vector to the right

and left is computed. The binary chroma vector is used as the hash value for that landmark. Notably, this approach is effective at matching similar music, namely alternate recordings of the same track. However, it is not robust to changes in structure or pitch-shifting.

2.2.2 Cover Song Recognition

In [9], Serra et al. present a technique for audio cover song identification. Harmonic Pitch Class Profiles are used as chroma features for computing the dissimilarity measure between two tracks. These pitch class profiles represent a larger scale structure that is not as relevant in short audio sampling. Bertin et al. [10] propose an audio fingerprint inspired approach to cover song recognition that computes the chromagram averaged over beat times to create a beat-synchronous chroma feature representation. Their approach is related to Wang's fingerprinting algorithm, but identifies landmarks as peaks in the chroma instead of in the full spectrum. Additionally the time frames are segmented by beats. This loss of time resolution could be detrimental in the task of general sampling, as there is no guarantee that a sample may be used within the same beat context as the original source.

2.2.3 Remix Recognition

Remix recognition is highly related to sampling in that both cases involve using audio from another recording to create new musical content. In 2007, Casey and Slaney [11] presented an algorithm for the detection of remixes in popular music. They propose a new technique of dividing the audio into chunks they call "audio shingles." The pairwise distance between two log chromagram shingles separate remixed from non-remixed content. The same could not be said for cepstral coefficient shingles. However, a remix is a specific case of sampling that involves many processes that are not present in general sampling. Features that represent a remix will contain global structure elements such as form and melody that are not relevant in sample recognition. By using chromagram features, Casey and Slaney have limited themselves to finding remixes that remain in the same key

as the original source. This may commonly be the case in remixed content, but for the more general case of sample detcetion it is important to be robust to larger pitch shifts. In the realm of music similarity tasks, Casey and Slaney place remix recognition somewhere between the specific audio fingerprinting and the more generic genre recognition. I would posit that the general case of musical sampling detection is actually an extension of audio fingerprinting. Identifying a sample within an audio track is the same as identifying a specific recording, while in addition to all the robustness requirements of audio fingerprinting it has the added challenge that the sample may be masked by other musical content.

2.3 Audio Fingerprinting

Audio fingerprinting appears to be a suitable approach to sample detection. Fingerprinting techniques have been well researched for over a decade and are in use in many real world applications. Fingerprinting transfers easily to sample detection since it addresses identification of a specific recording and has been developed to be robust to noise, distortion, filtering, pitch-shifting and time-stretching. Much of the work in audio fingerprinting also focuses on hashing techniques for quick and scalable lookup. In 2002 Haitsma et al. [12] presented one of the first fingerprinting algorithms to be both highly robust and efficient by introducing a hashing method for fingerprints. In 2003, Wang et al. [7] presented the algorithm used by the popular mobile application Shazam. This technique computes fingerprints by calculating the peaks in the spectrogram. These peaks are highly robust to both noise and most kinds of spectral filtering. Additionally, they were able to correctly identify each of several individual tracks that had been mixed together. This is a promising result for the application of sample detection. However, these spectral peaks are sensitive to even slight changes in pitch. And due to the way hashes were created from pairs of peaks from two different time frames, they are also sensitive to even very small amounts of time-stretching.

2.3.1 Pitch Shifting

In 2011, Fenet et al. [13] proposed a method of audio fingerprinting robust to pitch-shifting. They modified the work of Wang [7] to use a constant-Q transform and encode pairs of peaks at each landmark. The use of CQT allows for a pitch shift to be a simple a translation in the frequency domain. Their results showed improved performance compared to the previous method of Wang, at the cost of only a slight increase in processing time.

2.3.2 Time Stretching

In 2015, George et al. [14] proposed another variant of method proposed in [7], this time robust to time-stretching. Wang et al. extracted fingerprints by computing the time difference between two spectral peaks. By including the time difference within the fingerprint this method is naturally weak against time-shifting. George et al. compute each fingerprint based on the spectral peaks alone, but compute the top three peaks rather than only one to retain robustness to noise and other distortions. If this technique were used with a log-based instead of a normal Fourier spectrogram, it could be robust to both pitch-shifting and time-stretching.

However, while generating fingerprints from spectral peaks may be highly robust to noise, it may not be suitable for detecting musical samples. Since samples are contained within other musical content, the salient spectral peaks may be quite different. To compensate for this, it could be possible to extract a greater number of peaks from the query track. But then some method of comparing this large number of peaks to the smaller number of peaks in the dataset would be required. And it quickly increasing the amount of processing required.

2.3.3 Computer Vision

Zhang et al. [2] propose a novel method of audio fingerprinting using a technique from the discipline of computer vision known as scale-invariant feature transform (SIFT). SIFT is an algorithm for detecting features in images, and is here applied

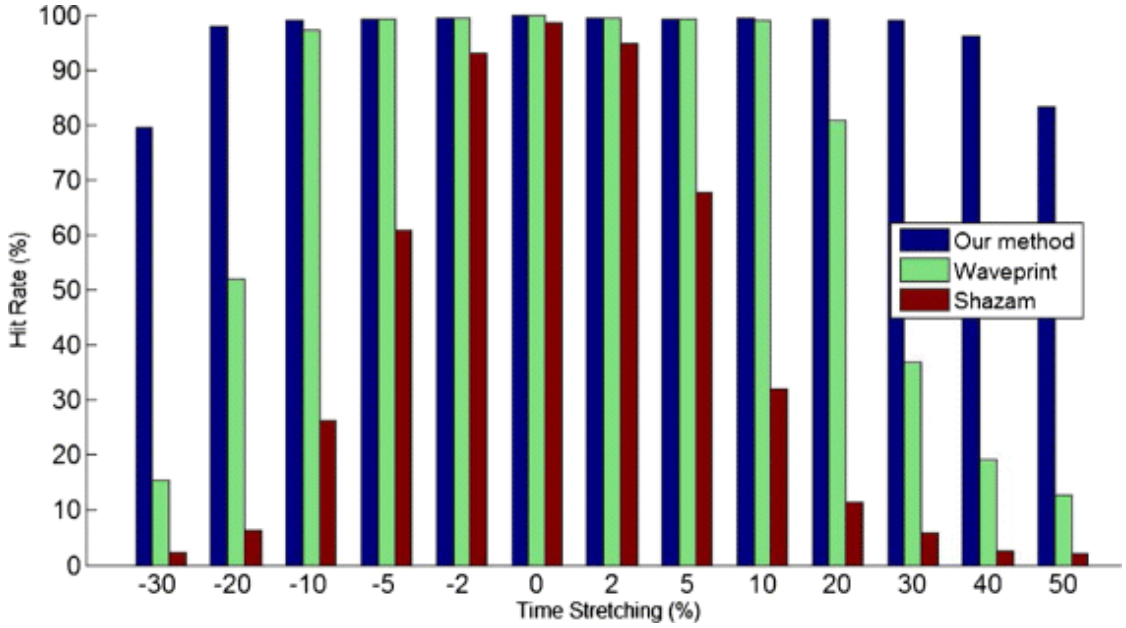

Figure 1: Performance of the SIFT-based audio fingerprinting system proposed by Zhang et al. for identifying time-stretched audio compared to the Shazam system proposed by Wang and the Waveprint system proposed by Baluja et al. [15].

to the spectrogram. Since pitch-shifting and time-stretching can be represented as translations and stretches of the spectrogram image, fingerprints computed from the SIFT key-points are able to achieve strong robustness to pitch-shifting, time-stretching, and time-scaling. The improved performance for identifying time-stretched audio can be seen in Figure 1.

2.4 Scale-Invariant Feature Transform

Scale-Invariant Feature Transform (SIFT) is a local feature extraction algorithm which finds extrema points in scale space, and extracts a position, scale, rotation invariant feature vector for each extrema point. The SIFT descriptor was first proposed by David Lowe in 1999 [16] and further developed in 2004 [17]. SIFT has since been widely used as one of the most popular features for object recognition tasks. Several improvements to the SIFT descriptor or keypoint detection algorithm have since been proposed (FAST, SURF, ORB). These improvements

aim to reduce computational time and feature dimensional, while maintaining comparable performance. Generally, SIFT still has the best performance out of these alternatives for object recognition tasks [18]. The SIFT algorithm finds extrema points in scale space, and extracts position, scale, rotation invariant feature vectors. Sift extraction has four major steps: keypoint identification, keypoint localization, orientation assignment, keypoint descriptor computation.

2.4.1 Scale-Space Extrema Detection

The first stage of calculation is to search over all scales and image locations. The difference-of-Gaussian function is used to detect stable keypoint locations in scale space. This stage attempts to find those locations and scales that are identifiable from different views of the same object. The scale space of a two-dimensional image is defined by equation 1

$$L(x, y) = G(x, y, \sigma) * I(x, y) \tag{1}$$

where $*$ is the convolution operator, G is a variable-scale Gaussian, and I is the input grey-scale image. The parameter σ is the scale of the keypoint. The difference of Gaussians function,D, is used to detect stable keypoint locations in scale space. D is computed by using the difference between two images, one, with scale k times the other. D is given by Equation 2. This process can be seen in Figure 2.

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) \tag{2}$$

To detect the local maxima and minima of D, each point is compared with its 26 adjacent neighbors, 8 from the same scale and 9 from one scale up and down, as seen in Figure 3. If this value is the minimum or maximum of all these points then this point is a scale-space extrema. This extrema is used as a SIFT keypoint.

2.4.2 Keypoint Localization

In order to improve keypoint stability and reduce sensitivity to noise, low-contrast and unstable keypoints are removed. Low-contrast and edge keypoints are removed by calculating the Laplacian value for each keypoint found in stage one. Keypoints above a given contrast threshold are removed.

2.4.3 Orientation Assignment

A key feature of SIFT is its robustness to changes in orientation. This step aims to assign a consistent orientation to the keypoints based on local image properties. The keypoint descriptor can then be represented relative to this orientation, achieving invariance to rotation. Rotation of spectrogram image has no meaningful relation to audio. However this step may still be useful during sample detection since keypoint matches with differing orientations can be filtered out as clearly being false matches.

2.4.4 Keypoint Descriptor

At each keypoint, using its selected scale, the local image gradients are measured for the region around the keypoint. The gradient information is rotated to line up with the orientation of the keypoint and then used to create a set of histograms over a window centered on the keypoint. Keypoint descriptors typically use a set of 16 histograms, aligned in a 4x4 grid, each with 8 orientation bins, representing the cardinal and ordinal directions, and one for each of the mid-points of these directions. This results in a feature vector containing 128 elements.
In addition to invariance to the affine transformations of translation and scaling, SIFT features have been shown to be robust to partial occlusion and changes in illumination which suggest they may be effective at identifying samples within other musical content and those with spectral filtering applied. For these reasons, this technique is the main basis for the sample detection system that will be outlined below.

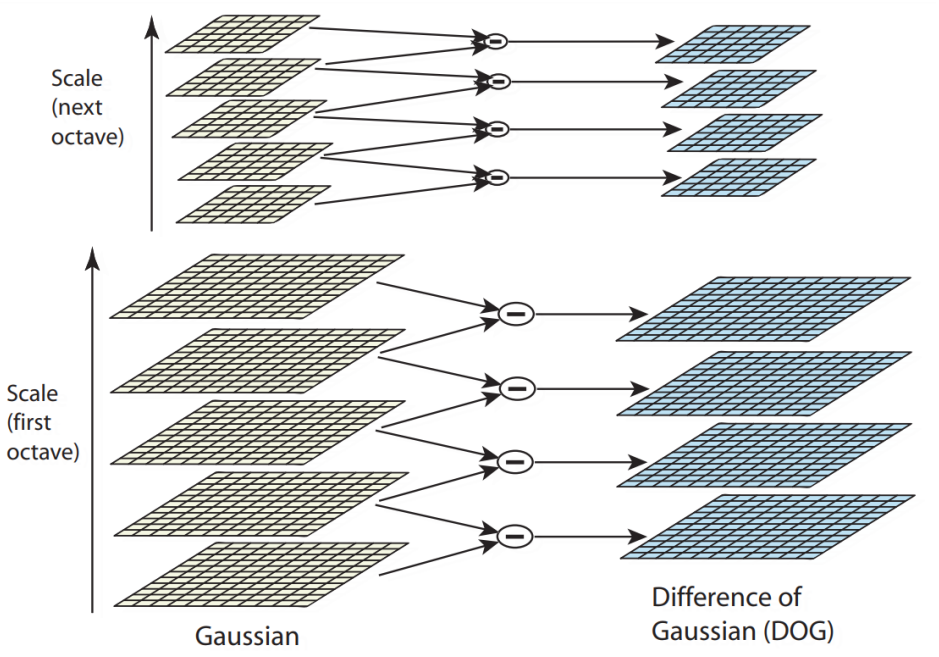

Figure 2: For each octave of scale space, the original image is convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right.

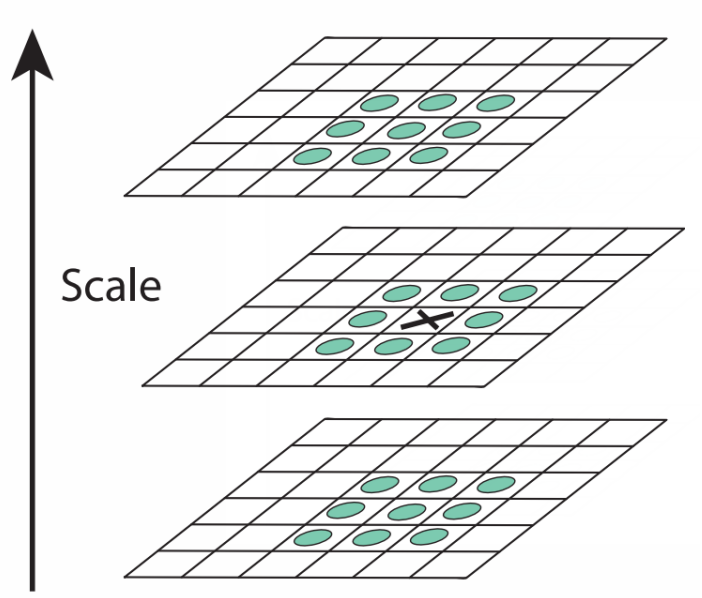

Figure 3: The 26 adjacent neighbors of an image pixel used to detect scale-space extrema.

3 Methodology

3.1 Overview

The aim of a sample detection system is to recognize when a particular track has employed musical sampling and identify the source of the sample. Given a database of existing audio files, this system should be able to recognize if an input audio file has sampled from any of the sources in the database. If keypoints can be repeatedly detected and similar feature vectors extracted to describe those keypoints in different audio contexts the process becomes as simple as finding a nearest neighbor by euclidean distance in a database of keypoint descriptors.

The key components of an automatic sample detection system include: keypoint detection, feature extraction, feature hashing, and nearest neighbor lookup. Keypoint detection and feature extraction are both acheived using the Scale-Invariant Feature Transform (SIFT) method proposed by Lowe [17]. The Spotify Annoy (Approximate Nearest Neighbors Oh Yeah) library is used for feature hashing and approximate nearest neighbor matching based on euclidean distance [19]. For each match the ratio thresholding technique used by Lowe is applied to remove likely invalid matches. The remaining matches are clustered using the generalized Hough transform to group matches that roughly agree on a shape. Clusters of a certain size are determined to be an instance of sampling.

3.2 Keypoint Detection and Feature Extraction

Both keypoint detection and feature extraction are performed using SIFT. This technique was designed for extracting discriminant features from a grey-scale image. A grey-scale image is similar to an audio spectrogram in that both are a matrix of intensity values. The main difference being that in an image both axes represent similar dimensions, coordinate position, while in a spectrogram one axis is time and the other is frequency.

To extract SIFT features from an audio track, the first step is to produce it's spectrogram representation.

3.2.1 Spectrogram Image Construction

Brian McFee's librosa python library is utilized to convert an audio file into a constant-q transform (CQT) based spectrogram matrix [20]. This implementation is based on the recursive sub-sampling method described by Schoerkhuber et al. [21]. CQT is used since it is vital that the frequency axis be logarithmically spaced for SIFT to be robust to pitch-shifting. If a normal Fourier transform were used, a pitch-shift would cause an SIFT descriptor to be disproportionately stretched along the vertical axis. With a CQT-based spectrogram a pitch shift is represented by a simple translation along the vertical axis.

3.2.2 Scale-Invariant Feature Transform

The features will come from scale-invariant feature transform (SIFT). This is a method taken from the field of Computer Vision that has been used for object recognition. SIFT is a technique to detect keypoints within an image by computing a difference of Gaussian functions applied to a series of the image at varying scale, and identifying points of high contrast within this "scale-space". Each keypoint can then be described with an orientation and a descriptor vector containing histogram values of the gradients from the neighboring bins. SIFT descriptors have been shown to be robust to changes in scaling, orientation, affine transformations, noise and illumination. SIFT keypoints and descriptors extracted from an audio spectrogram can be used to define an audio fingerprint. On a spectrogram these image distortions can be related to audio as time-stretching is scaling in the horizontal axis and, if the spectrogram is on a logarithmic scale, pitch-shifting is translation in the vertical axis. [2] Changes in illumination can be seen as related to filtering and equalization. SIFT descriptors have also been shown to be robust to partial occlusion when used for object recognition. This is another reason why SIFT may be a more viable feature for sample detection as opposed to other audio fingerprinting features since the sample is embedded within other musical content. The commonly used method of extracting spectral peaks first employed by Wang et al. and other features that summarize the entire frequency spectrum of a frame may have difficulty when

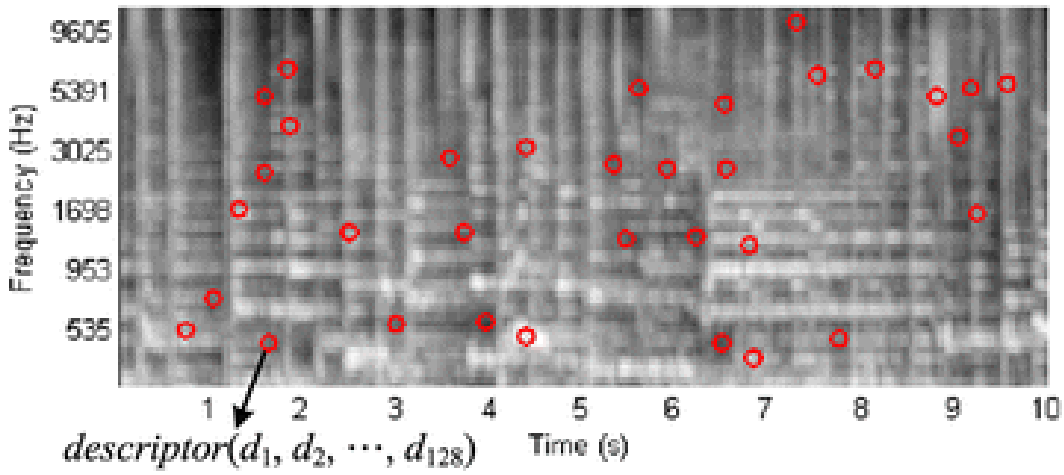

Figure 4: An spectrogram of a 10 second audio clip. The red circles represent a subset of the keypoints where SIFT descriptors are extracted.

the sample may be less prominent than the other musical content [7]. Since a SIFT keypoint descriptor only describes a small portion of the frequency spectrum there is likely to be some portion of the sample unobstructed by other content in the query track.

The open source computer vision library VLFeat implementation of SIFT is used to extract keypoints and descriptors from the spectrogram image [22]. The matrix values (first computed in dB) are scaled to the range 0-255 to be in the format of a standard grey-scale digital image. High contrast keypoints are determined within the spectrogram image according to a contrast threshold, σ. At each keypoint, a 128 dimensional feature vector is extracted describing the direction of gradients around the keypoint at the Gaussian scale determined in the detection stage.

3.3 Build Source Descriptor Database

For each track in the source dataset, keypoints and descriptors are extracted using the above method. The descriptors are used to fit an Annoy approximate nearest neighbor matcher using the python bindings provided by Spotify [19]. Along with each descriptor is stored a reference to the track it was extracted from, the scale of the keypoint, and it's x and y coordinates (time and fre-

quency value) in the spectrogram representation. The keypoint orientation is also stored. Since there is no meaningful concept of rotation of a spectrogram image, the orientation is used to filter out keypoint matches that have differing orientations.

3.4 Query

To query a track, extract the same keypoint descriptors for the provided track by the method described in 3.2. For each descriptor, find it's two (approximate) nearest neighbors in the source database. Ratio thresholding is used to discard keypoints that have no correct match in the source database, or if it's correct match was missed due to the use of ANN.
Cluster the remaining matches in time such that all keypoints in a cluster are within a certain time-frame of each other in both the query and reference track. Any cluster larger than the threshold size is returned and the time of the first match in the cluster is considered the start of the sample.

3.4.1 Approximate Nearest Neighbor

Keypoint descriptors are hashed and stored using the Spotify Annoy library. This is a method of dimensionality reduction used for approximate nearest neighbor (ANN) matching. Annoy hashes feature vectors using random projections to build a tree. At every node in the tree, a random hyperplane is chosen, which divides the space into two subspaces. This hyperplane is chosen by sampling two points from the subset and taking the hyperplane equidistant from them.
Query keypoint descriptors will be compared to the keypoints in the Annoy tree structure. The query descriptor does not need to be compared with every point in the database since it is likely to look at similar neighbors first. Close neighbors will be identified, but there is no guarantee that the exact nearest neighbor will be found. However, the processing time is greatly reduced by not needing to use brute force comparisons. [1]

[1]There are many different algorithms for ANN which may be worth exploring. Informally, kd-tree, ball-tree [23], and Locality Sensitive Hashing forest [24] algorithms were tested on

3.4.2 Thresholding

For each keypoint in the query track we now have a possible nearest neighbor in the source database. However, it is highly likely that a keypoint does not have a true match within the database. For this reason a threshold can be applied to narrow down matches to only closely related keypoints. Lowe has shown that an absolute threshold on the Euclidean distance between two SIFT descriptors does not perform well, since some descriptors are more discriminative than others [17]. Lowe proposed a ratio threshold by comparing the distance of a keypoint to it's two nearest neighbors. For each descriptor we must find it's two nearest neighbors. The second neighbor is defined as the closest neighbor that comes from a different source track than the first neighbor. Divide the Euclidean distance from the query descriptor to the first match by the distance to the second and if it is above the ratio threshold, θ discard that match. This measure performs well because correct matches ought to be significantly closer than the closest incorrect match to achieve reliable matching. Since repetition is common in music, it is important that the second neighbor does not come from the same track as the first. [2] Figure 7 shows an example of a bassline sample that is repeated throughout in the derivative track "Ice Ice Baby" by Vanilla Ice. The bassline is also repeated throughout the original source track, and while it may not be played identically throughout, each repeated will from a very similar spectrogram. This can be seen by the fact that often the nearest neighbor of a keypoint from audio sampled from the first few seconds of "Under Pressure" actually matches to keypoints of the later repetitions of this bassline.

---

the same dataset. All gave comparable performance while Annoy was an order of magnitude faster than the others, at the cost of slightly larger storage size. Annoy has the downside that new feature vectors cannot be added once the structure is created. This may not be ideal for production system where new source tracks would be continually added to the dataset, but worked well for this study.

[2] This may still cause a problem if a track in the source dataset samples from another track in the source dataset. In this evaluation it was made sure this never occurs (according to the labels provided by whosampled.com). In a production system, tracks could be added to the source dataset iteratively by release date. Before being added a track is queried against the database. Any keypoints with close matches found would not be added to the source database, deferring to their original source already in the database.

3.4.3 Clustering

While the ratio threshold discards many false matches, a single match is still likely to be a false positive, and so clustering is performed to limit results to when multiple matches agree on the same sample. The Generalized Hough Transform is used to group matches that agree on roughly the same shape in both the source and query spectrograms. When clusters of keypoints are found to vote for the same pose, the probability of the interpretation being correct is much higher than for any single keypoint match. For each keypoint, a Hough transform entry is created predicting the model location (x and y coordinate) and keypoint scale. Since these parameters are only approximations broad bin sizes of a factor of 2 for scale, and 0.25 times the maximum projected source track dimension are used. Each keypoint votes for the two closes bins for each dimension to avoid boundary effects, for a total of 6 entries for each keypoint. Additionally, clusters are grouped such that for all it's keypoints both the source and query keypoints are within some time threshold, δ. Only clusters of at least a certain size, ϕ, are considered an instance of sampling.

4 Factor estimation of time stretching and pitch shifting

In addition to detecting whether a sample exists in a query track, it is possible to determine by how much the sample may have been pitch-shifted or time-stretched. As stated, in a logarithmic spectrogram pitch-shifting and time-stretching are represented respectively as a translation in the frequency-axis and a stretch in the time-axis. When a cluster of keypoints have been matched between a derivative and original track, the changes in the location of these keypoints can be used to estimate the how the sample has been altered between the two versions.

Figure 8 shows an example of time-stretching factor estimation. In this figure, A and $A0$ represent spectrogram images of a derivative track and its original,

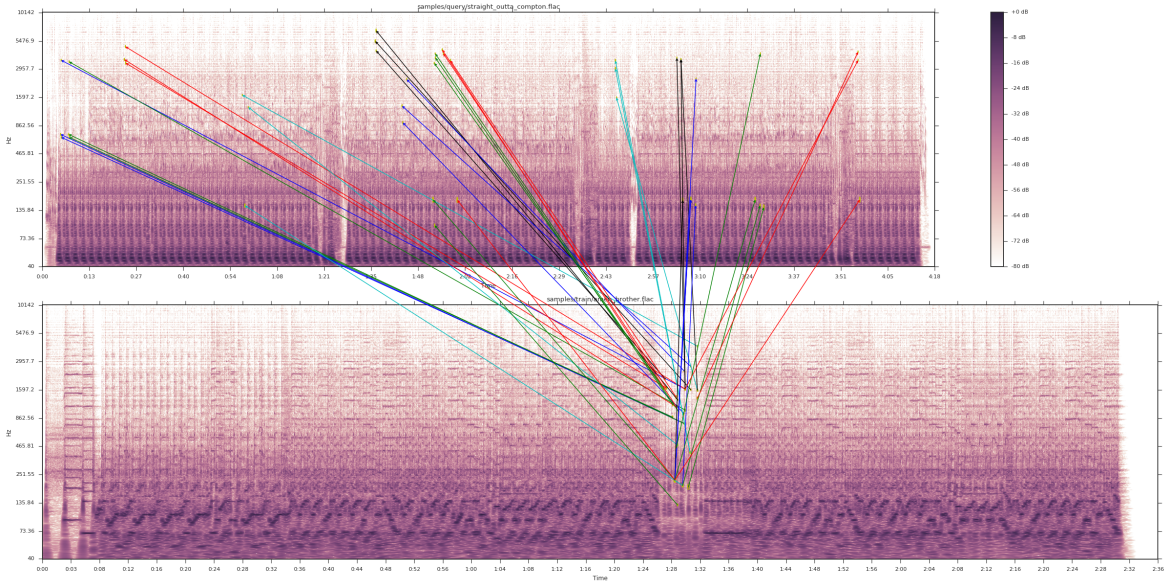

Figure 5: The remaining keypoint matches after ratio thresholding and match clustering. The top is a spectrogram of Straight Outta Compton by N.W.A. Below is Amen Brother by The Winstons. Lines are drawn from each keypoint to its nearest neighbor. Matches are colored by cluster.

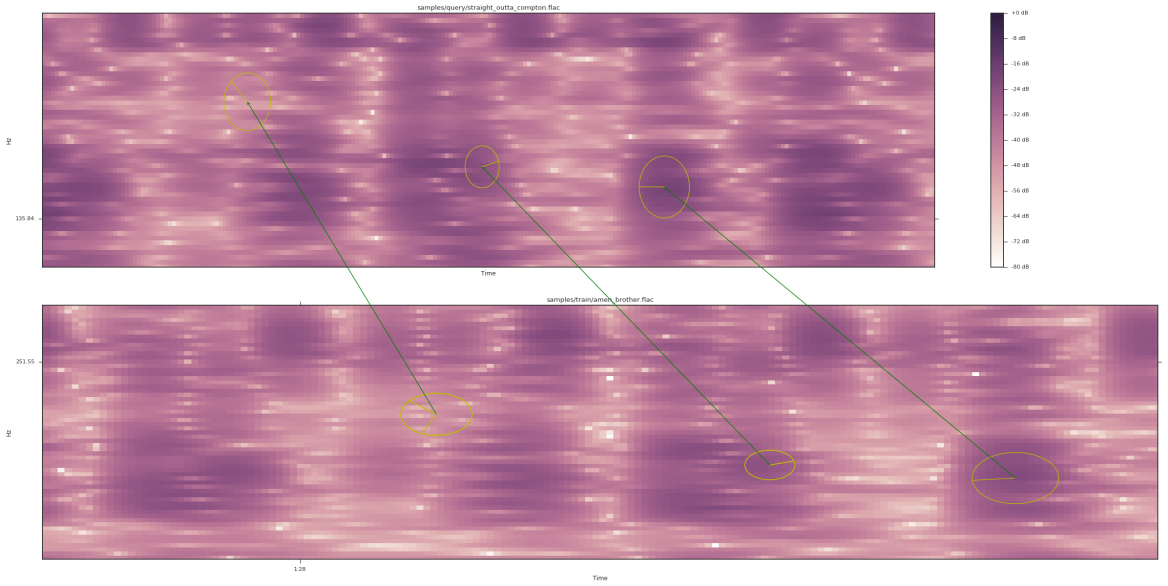

Figure 6: A zoomed in image of Figure 5 showing three keypoint matches from a cluster.

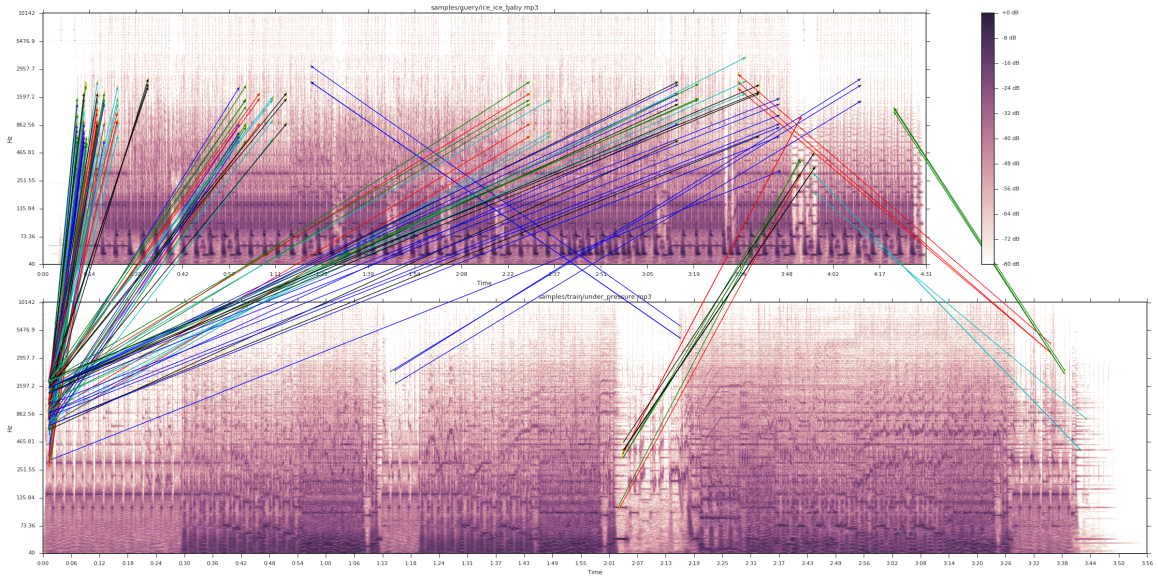

Figure 7: Clustered Keypoint matches between Vanilla Ice's "Ice Ice Baby" and Queen's "Under Pressure." While the sample bassline from the beginning of "Under Pressure" is recognized, often SIFT descriptors from repetitions of this bassline throughout the song are matched instead.

respectively. a_1 and a_2 represent the location of two keypoints in a cluster of nearest neighbor matches. a_{01} and a_{02} are the matched keypoints of a_1 and a_2, respectively. Given the four keypoints a possible time-axis stretch factor, k_t, between A and $A0$ can be estimated according to Equation 3 where d_t is the time-axis distance between a_1 and a_2, and d_{t0} is the time-axis distance between a_{01} and a_{02}. A factor candidate can be computed for each keypoint pair in the cluster of matches. The median of all these candidates is chosen as the estimation of time-stretch factor applied in the derivative track to the sample represented by this cluster.

$$k_t = \frac{d_t}{d_{t0}} \tag{3}$$

Figure 9 shows an example of pitch-shift estimation. From a single keypoint match, the amount of pitch-shift Δy can be computed in terms of a twelve-note musical scale semitones by Equation 4, where y_f and y_{f0} are the frequency-axis coordinate values of a pair of matched keypoints and β is the number of bins

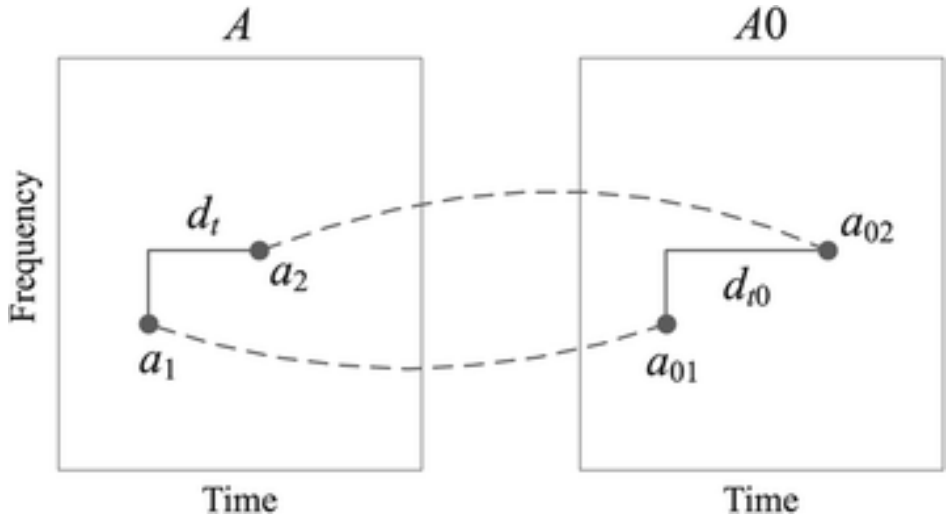

Figure 8: Example of time-stretching factor estimation.

per octave used to create the CQT spectrogram. Similarly, there exists a Δy candidate for each derivative-original pair in a cluster of matches, and the median of these values is selected as the final result.

$$\Delta y = (y_f - y_{f0}) * \frac{12}{\beta} \tag{4}$$

In both cases, the median is used rather than the mean since each candidate estimation is expected to be fairly accurate save for a few possible outliers due to the likelihood that an incorrect match exists in any cluster of matches.

5 Evaluation

An evaluation was carried out to measure the performance of the system using examples of actual popular music recordings. Ground truth labels were provided by the website whosampled.com [4]. The "most influential" and "most popular" tracks are chosen since these tracks likely represent stereotypical uses of sampling, and because Whosampled relies on crowd-sourced data the most popular tracks are more likely to have accurate labels.

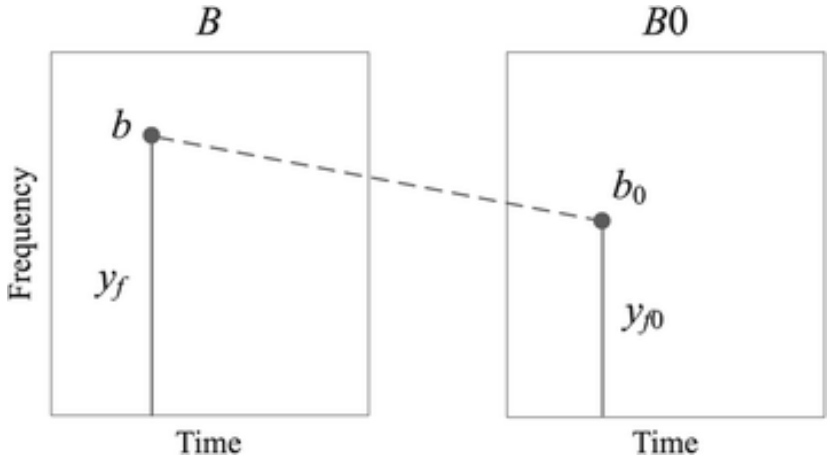

Figure 9: Example of pitch-shifting factor estimation.

5.1 Dataset

The dataset used for evaluation comprises two parts: a set of possible original works to build the database and a set of derivative track to query the system. The set of original works is made up of the 50 "most influential tracks" according to the website whosampled.com. These are the tracks that Whosampled has determined have been sampled the most times within their database. These tracks were originally released between the years 1966-1994, and span the genres "Soul / Funk / Disco," "Hip-Hop / Rap / R&B," "Jazz /Blues," "Rock / Pop," and "Electronic / Dance." This set is then padded with an additional 4728 tracks with the same distribution of genres and release dates, being sure that none of the 4728 tracks sample from or are sampled by any other tracks in either the original or derivative datasets (according to whosampled.com). The breakdown of year and genre for all 4728 tracks in the sour can be source dataset can be seen in Table 1

The derivative dataset is composed of the 10 most popular tracks that have directly sampled from each of the 50 most influential tracks, for a total of 466 tracks (since 34 of the tracks sample from two or more of the 50 original works) containing 500 labeled instances of sampling. These samples have been labeled

Table 1: Source dataset year and genre

original genre	count
Rock / Pop	200
Soul / Funk / Disco	3151
Electronic / Dance	100
Hip-Hop / Rap / R&B	1136
Jazz / Blues	191

original year	count
1966-1969	676
1970-1974	2012
1975-1979	277
1980-1984	729
1985-1989	790
1990-1994	294

Table 2: Derivative dataset year and genre, according to Whosampled.com

derivative genre	count
Rock / Pop	23
Jazz / Blues	2
Soundtrack	4
Reggae	2
Other	4
Hip-Hop / Rap / R&B	363
Electronic / Dance	68

derivative year	count
1981-1984	6
1985-1989	77
1990-1994	160
1995-1999	77
2000-2004	27
2005-2009	39
2010-2016	80

as either "drums," "vocals / lyrics," or "hook / riff," the breakdown of instrument labels can be seen in Table 3. Each set of 10 has been chosen only if they have the same label, to improve the likelihood that they have sampled from the same section of the original work, although this is no guarantee, the audio being sampled is likely to be similar. The 466 derivative tracks were released between the years 1981-2016, spanning the genres "Hip-Hop / Rap / R&B," "Electronic / Dance.", "Rock / Pop," "Jazz / Blues," "Soundtrack," and "Reggae." The breakdown of year and genre for the source tracks can be seen in Table 2

All 5244 audio files were taken from the Youtube videos referenced by Whosampled.com. These files were kept in their original format, with varying levels of compression and audio formats, either "opus," "ogg," or "m4a." This variety may not be ideal for a scientific evaluation, however it does represent the format in which these tracks are commonly consumed, and will provide information on how robust the system is to lossy compression.

Table 3: Ground-truth instrument tags for the 500 sample instances

instrument	count
hook / riff	62
vocals / lyrics	231
drums	207

5.2 Procedure

All audio tracks are resampled at a sampling rate of 22050. The system is trained by extracting and storing keypoint descriptors from each of the 4778 audio files in the original dataset. A query is made for each of the 466 derivative tracks. The same descriptors are extracted and their nearest neighbors found. Matches of differing orientations are removed. If the ratio of the distance to the nearest neighbor is greater than θ times the distance to the second nearest neighbor from a different source track, the match is removed. The remaining matches are clustered according to cluster distance δ. Any cluster greater than size σ is a vote for the original track it references as a track sampled by the current query track.

A true positive is determined to be if any votes exist for a correct original track. A false positive is counted as each original track that has a vote and the query track did not actually sample from that track. A false negative is any original track that was actually sampled by the query track, but received no votes.

5.3 Hyperparmeter Optimization

The sample detection system presented here contains a host of hyperparameters at every stage of the process. Many variables exist in the way the spectrograms are created, keypoints are detected, nearest neighbors are found, and thresholding of the matched keypoints is performed. Due to constraints in time and computational power, optimization of the entire system in an exhaustive grid search would not be feasible. The most influential parameters have been tested across a sweep of threshold values. The following parameters were tested. Hop

size and octave bins, β, (time and frequency resolution) when building the CQT spectrogram. The contrast threshold for detecting SIFT keypoints. The ratio threshold for discarding false keypoint matches. And the minimum cluster size for a cluster of keypoints to be considered an instance of sampling.
There is a computational trade-off as increased spectrogram resolution takes longer to compute and increases the time to detect and extract SIFT keypoints. Increasing the SIFT contrast threshold will extract more 128-dimensional keypoint descriptors per track, increasing the size on disk as well as the query per keypoint. The upfront cost of building the database of source descriptors is not a large concern since that only needs to be done once. With the use of ANN, it requires orders of magnitude in the number of descriptors extracted to have a large effect on query times. The major limiting factor for this project is size on disk, since the implementation requires the entire database to fit in RAM in order to query it.

5.4 Evaluation Metrics

Audio fingerprinting systems, including the sample recognition system proposed by Van Balen [25], are often evaluated using mean average precision (MAP) since they return a ranked list of possible matches. In this system, the importance of ratio thresholding means that there is only one returned neighbor for each keypoint. Fingerprinting systems also only need to count the number of matches for the same reference fingerprint in the entire query excerpt and can return a ranked list sorted by this count. It is not this simple for sample detection, since a majority of any query audio file will likely not contain any samples from the reference dataset. Samples may be very short and only used once within a derivative track. The clustering and thresholding of keypoint matches is a vital step of the process since this is not only a retrieval task but a detection task to determine whether a track contains any samples at all.[3] For these reasons, this

[3]The size of a cluster beyond the minimum threshold and the number of clusters referencing the same original track could be used as a confidence measure for how likely the query track sampled from that source track. This was not explored here since I felt it important that a generalized sample detection system be able to detect and give equal importance to very short

will be treated as a multilabel classification problem and will be measured by precision, recall and harmonic mean (F_1-score).

5.4.1 Baseline

With a dataset this large, a random baseline would not be very meaningful. Instead a baseline will be computed using the audio fingerprinting algorithm presented by Wang, original used by Shazam [7]. Many commercial fingerprinting solutions claim to be able to recognize an audio recording even when two recordings are playing at the same time. It is reasonable to assume that an audio fingerprinting system would be able to identify a small subset samples, particularly isolated samples in the foreground of the mix not being obscured by other musical content, and samples that have had little or not processing applied to them. Wang's landmark-based algorithm is particularly sensitive to both pitch shifting and time-stretching and so will miss many samples that may be identified by other state-of-the-art fingerprinting systems. But it should provide a decent baseline to show the comparative improvement of this sample detection system to a standard fingerprinting algorithm.
Audfprint, Ellis' open source implementation of Wang's algorithm is run on the same dataset to determine the baseline [26]. This is the same implementation used in Van Balen's sample recognition system [6], however I am not including the time-scaling modification's he made. A sweep of the threshold of number of fingerprint matches will be made to plot the precision vs. recall and determine the best F_1-score.

6 Results

The overall and interpolated precision and recall curve across all the hyperparameters tested can be seen in Figure 10. The trade off between precision and recall is mainly due to the value of the ratio threshold θ with values between 0.4-

samples that may not ever be repeated within the derivative track.

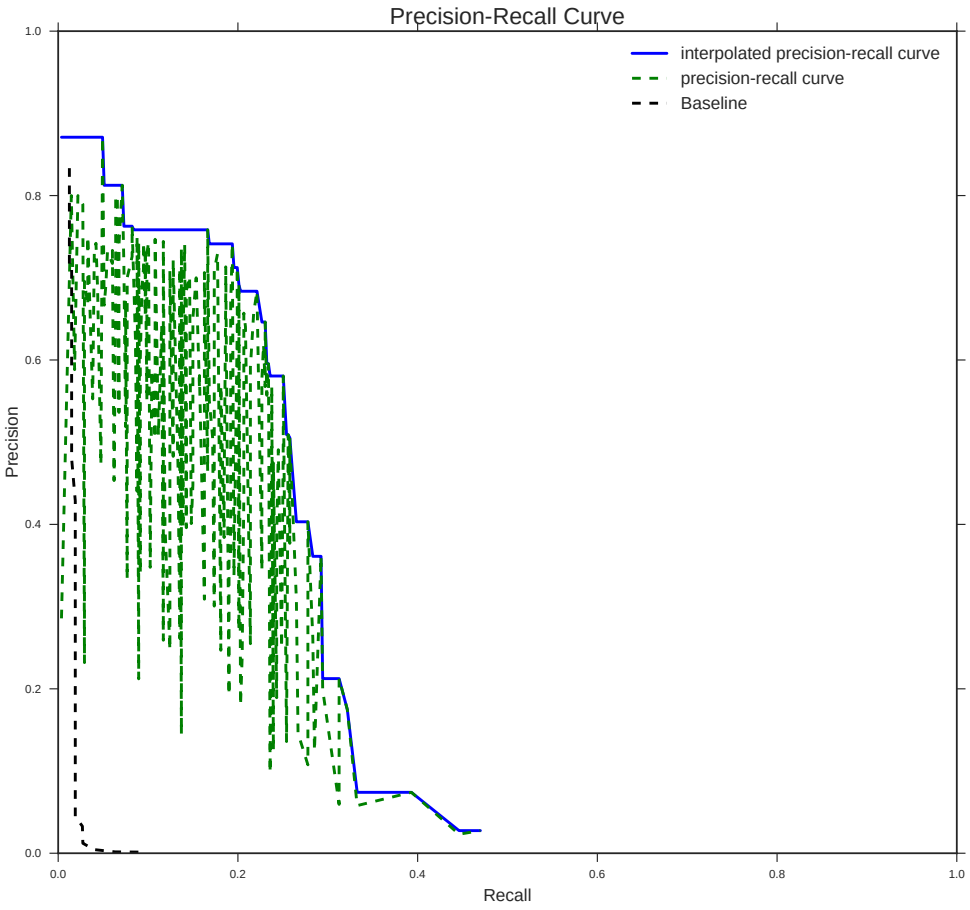

Figure 10: The overall precision-recall curve across all values of every parameter tested.

1.0. The best performance in terms of F_1-score is found with values of $\theta \approx 0.9$. The affect of minimum cluster size can be seen in Figure 11. When the cluster size drops below 3, precision begins to fall off. As θ approaches 1.0 the recall greatly increases compared to the higher minimum cluster sizes, but these values are unimportant since the precision is so low.

The resolution of the computed CQT spectrograms plays an important role where and how many in SIFT keypoints are detected. The SIFT algorithm's contrast threshold was kept relatively consistent such that approximately 5,000-

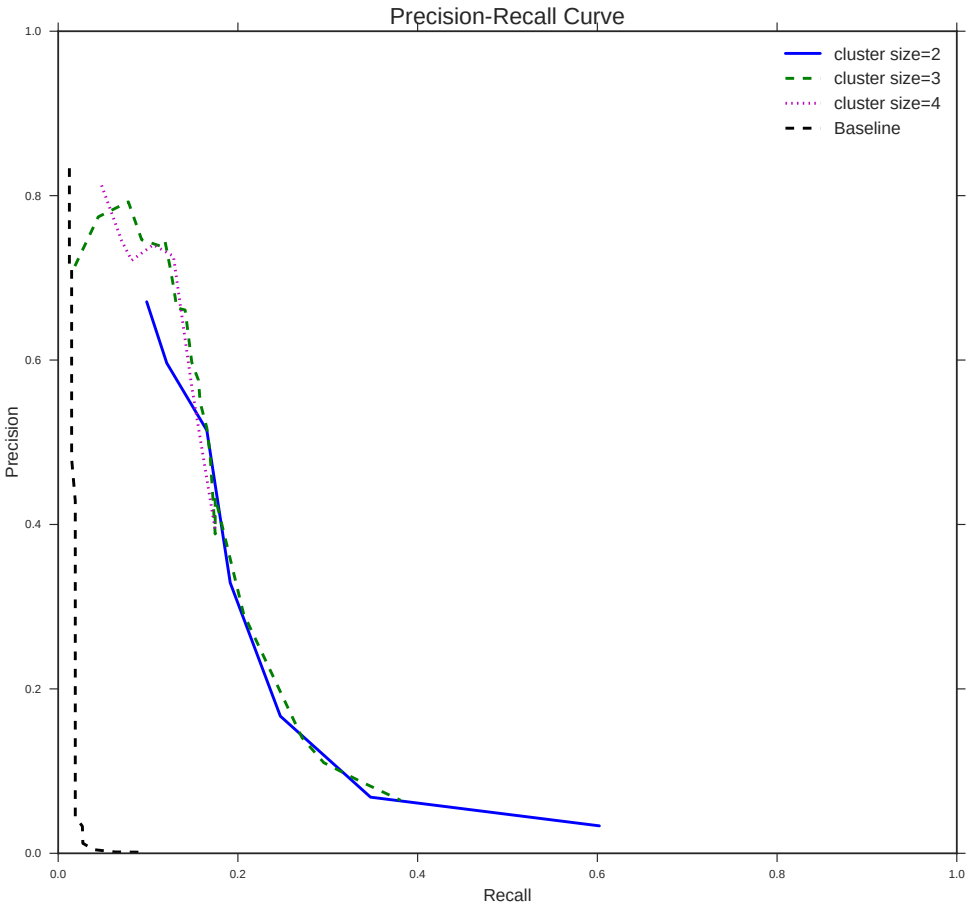

Figure 11: Performance comparison of varying the minimum cluster size to determine an instance of sampling.

Table 4: Performance broken down by hop size (time-resolution)

hop_size	precision	recall	f_1-score
32	0.581	0.250	0.350
64	0.557	0.232	0.328
128	0.436	0.207	0.280
256	0.448	0.150	0.225
512	0.247	0.124	0.165
1024	0.232	0.029	0.052
baseline	0.429	0.019	0.036

20,000 keypoints were detected per track, on average 50 SIFT keypoints per second of audio.

The performance effects of varying the CQT hop size can be seen in Figure 12. Large hop sizes cause a severe decline in both the precision and recall of the system. This is no surprise, as large hop sizes could cause alignment issues for where keypoints are detected. Extracting descriptors from scale-space extrema alleviates the need for perfect alignment between two recordings, but a certain level of resolution is still required. Also, if the image resolution is too low, fewer scale-space extrema will exist. Reducing the SIFT contrast threshold does not help, and fewer keypoints will be extracted per track.

The performance effects of varying the bins per octave of the CQT can be seen in Figure 13 and further expanded upon in Table 5. As long as there is sufficient frequency resolution, the results are comparable. There is a small trending performance improvement as β is increased, however it seems this may peak at $\beta = 72$. Similar to time-resolution, if the frequency-resolution is too low, there will be alignment issues in the vertical axis for where keypoints are detecting, particularly for repitched samples. Again, the low resolution will decrease the total number of keypoints able to be extracted from a track.

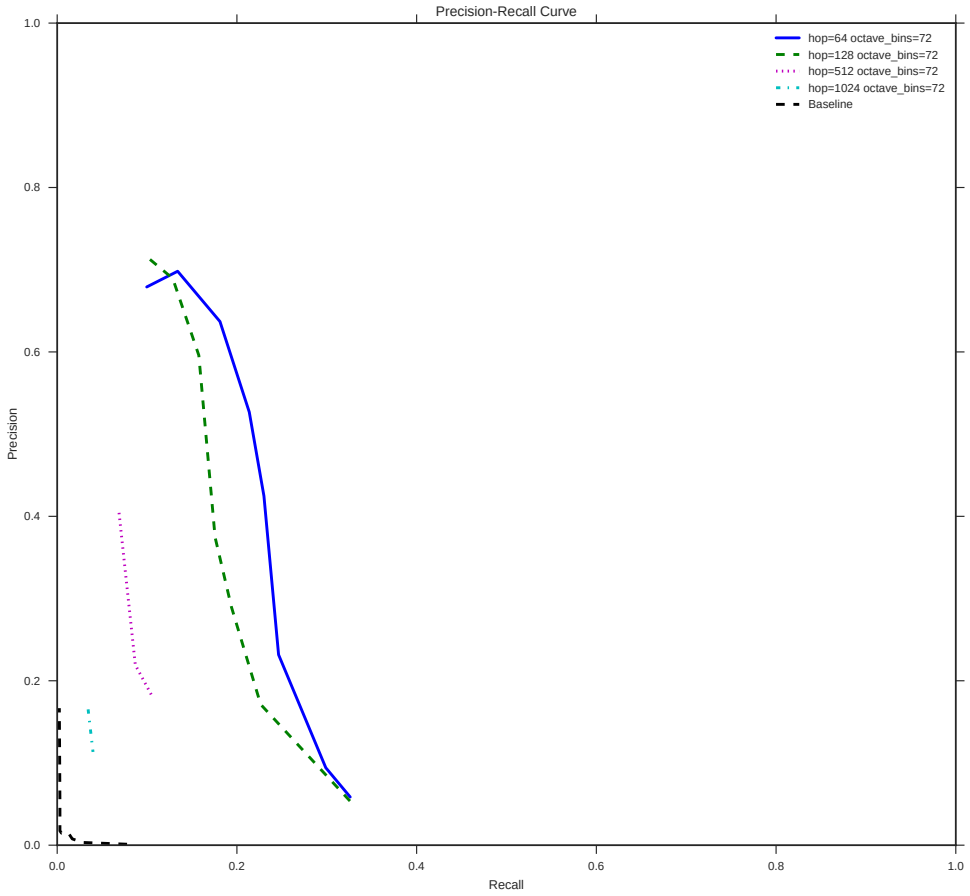

Figure 12: Performance comparison of varying the hop size, to determine the effect of time resolution.

Table 5: Performance broken down by octave-bins, β (frequency-resolution)

octave_bins	precision	recall	f_1-score
96	0.478	0.238	0.317
72	0.581	0.250	0.350
48	0.595	0.234	0.336
36	0.505	0.258	0.341
24	0.577	0.227	0.325
12	0.399	0.119	0.183
baseline	0.429	0.019	0.036

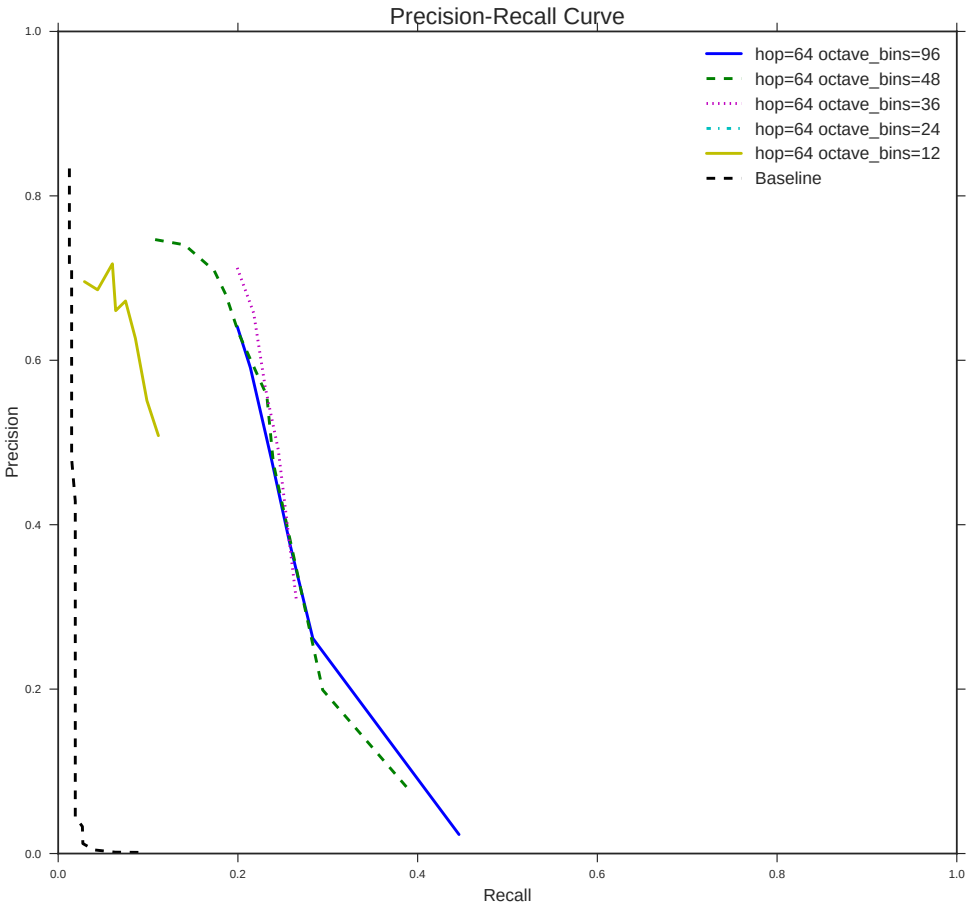

Figure 13: Performance comparison of varying the number of frequency bins per octave, β, to determine the effect of frequency resolution. As long as there is sufficient resolution, increasing β has little effect.

instrument	32	64	128	256
vocals / lyrics	0.245455	0.218182	0.118182	0.118182
hook / riff	0.1875	0.15	0.15	0.15
drums	0.238866	0.271255	0.255061	0.263158

Table 6: Recall by instrument label. Broken down by hop size. Recall of vocal samples greatly improves with increased time resolution, while the other instrument labels are less affected.

original genre	recall	precision	f_score
Electronic / Dance	0.100	1.000	0.182
Soul / Funk / Disco	0.216	0.642	0.323
Hip-Hop / Rap / R&B	0.221	**0.257**	0.238
Rock / Pop	0.200	1.000	0.333
Jazz / Blues	0.222	0.857	0.353

Table 7: Performance of sample detection by genre of the source track

7 Discussion

The best performance achieved is an F_1-score of 0.35, corresponding to a precision and recall rate of 0.581 and 0.250 respectively. This may be arguably low for a retrieval task, but is an order of magnitude above the performance of the baseline fingerprinting system with an F_1-score of 0.036. While it cannot be directly compared, this is on par with Van Balen's system with a MAP of 0.39 evaluated on a much smaller dataset.

In order to learn about the failings of this system the performance was broken down by instrument and genre. In Table 7 the performance can be seen by genre of the original tracks. There is a heavy drop in the precision of samples taken from hip-hop tracks. The recall from electronic tracks is also low, but it is difficult to make any claims based on this due to the under-representation of electronic tracks in the source dataset.

Table 8 shows the performance broken down by the genre of the derivative track. There is comparatively poor performance for electronic/dance and rock/pop genres. It is difficult to say what may be the main cause of this. From qualitative analysis, many of the samples missed in these genres were short and heavily distorted or modified in some way. In hip-hop it is more common that a sample is unaltered and is usually looped or repeated throughout the song. A main aim of this project was to be able to detect these short and modified samples, but it is no surprise that they are more difficult to identify.

While my original hypothosis was that percussion samples would be heavily dependent on time-resolution and vocal or instrumental samples would be identified with better frequency resolution, in Figure 14 it can be seen that the

derivative genre	recall	precision	f_score
Reggae	0.500	0.500	0.500
Electronic / Dance	**0.171**	0.250	0.203
Rock / Pop	**0.087**	0.500	0.148
Soundtrack	0.500	1.000	0.667
Jazz / Blues	0.500	1.000	0.667
Hip-Hop / Rap / R&B	0.222	0.601	0.324
Other	0.250	1.000	0.400

Table 8: Performance of sample detection by genre of the derivative track

recall drum and instrumental samples are largely unaffected and peak in the mid-range, while recall of vocal samples greatly improves with increased time-resolution. This could possibly be due to timbral qualities that exist in vocals and not instrumentals. However, upon inspection a large portion of vocal samples in the dataset are actually very short. Many of these samples are only one word or utterance, and often sped up in the derivative track. On the other hand, drum loops and hook/riff samples tend to be phrases lasting a full bar or longer. Since recall of the other instruments actually begin to decline when the hop size gets too short, it seems there may be no optimal setting for detecting all types of samples. An alternative solution could be to extract SIFT descriptors from multiple spectrograms with different time-frequency resolution to be able to detect both long and short musical samples.

In Figure 15 the effect of frequency resolution can be seen on the recall of different types of samples. The effects are mostly consistent across the different instrument labels. The recall generally improves with increased frequency resolution. Interestingly, odd multiples of 12 appear to have increased performance relative to even multiples. The CQT was computed with the standard $A440 = 440.0Hz$ reference tuning. It is likely that a majority of recordings in the dataset are close to this tuning. This would cause frequency peaks coincide on these 12 bins and be likely locations of scale-space maxima. Even multiples of 12 would not have a bin centered at these locations and possibly cause alignment issues for detecting the location of scale-space maxima.

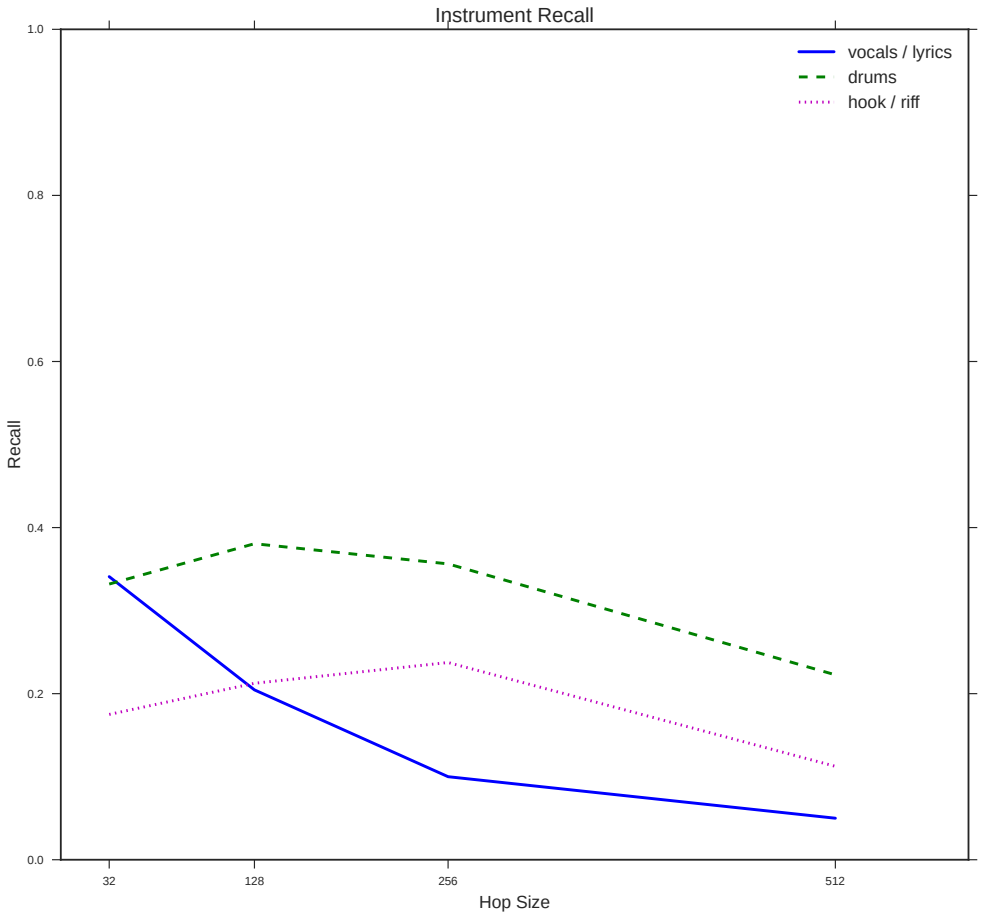

Figure 14: Recall of samples by instrument type. As the hop size decreases (increased time resolution) the recall of vocal samples improves, while the recall of drums and instrumental samples peak with a hop size of 128-256.

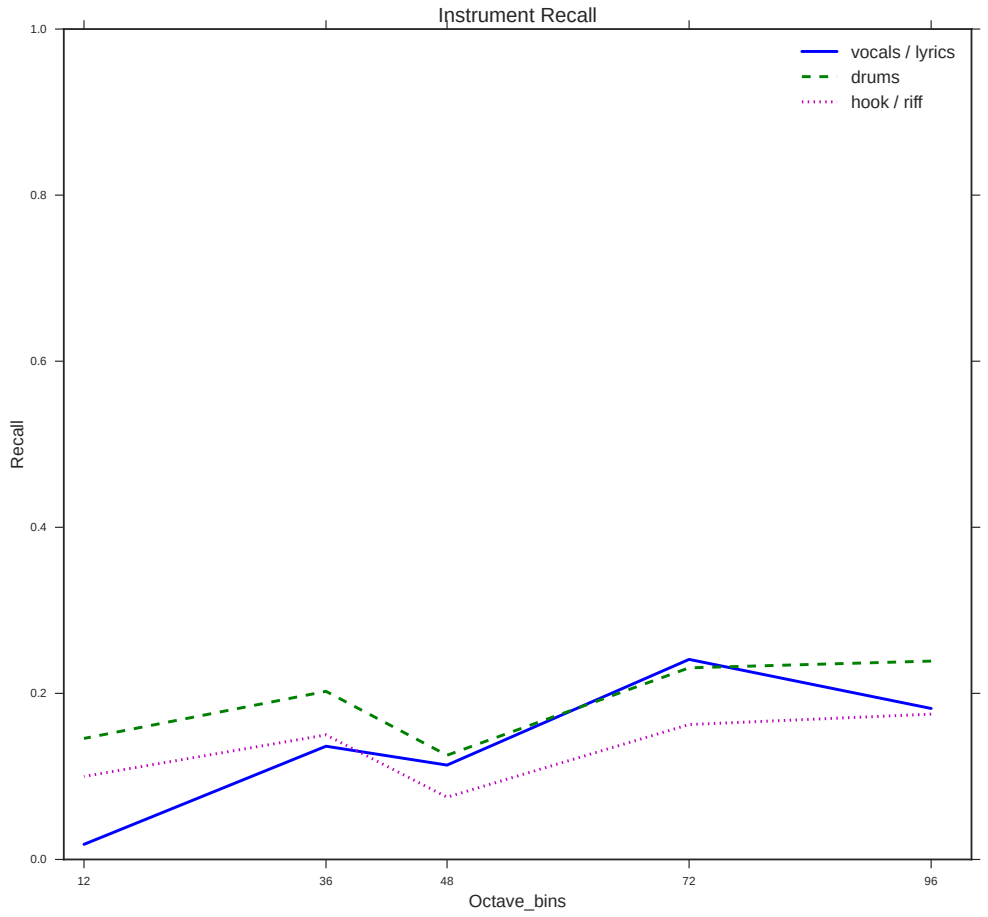

Figure 15: Recall of samples by instrument type. As the frequency resolution changes, the recall rate is fairly consistent across the different instruments.

Table 9 shows the average recall rate for each of the 50 source tracks containing the samples used in the query tracks. For seven of these tracks, the sample was never identified in any of its derivative tracks.

Source Track	instrument	recall
Barry White - I'm Gonna Love You Just a...	drums	0.3
Beside - Change the Beat (Female Version)	vocals / lyrics	0.047619
Billy Squier - The Big Beat	drums	0.181818
Bob James - Nautilus	hook / riff	0.272727
Bob James - Take Me to the Mardi Gras	drums	0.1875
Bobby Byrd - Hot Pants (Bonus Beats)	drums	0.272727
Commodores - The Assembly Line	vocals / lyrics	0
ESG - UFO	hook / riff	0
Eric B. & Rakim - I Know You Got Soul	vocals / lyrics	0.444444
Five Stairsteps - Don't Change Your Love	drums	0.3
Freddie Scott - (You) Got What I Need	drums	0
Funk, Inc. - Kool Is Back	drums	0.1875
Funkadelic - Good Old Music	drums	0.2
George Clinton - Atomic Dog	drums	0.357143
Incredible Bongo Band - Apache	drums	0.545455
James Brown - Funky Drummer	drums	0.25
James Brown - Funky President (People It's Bad)	vocals / lyrics	0.0588235
James Brown - Get Up Offa That Thing	hook / riff	0.0769231
James Brown - Get Up, Get Into It, Get Involved	vocals / lyrics	0
James Brown - Get on the Good Foot	vocals / lyrics	0.0909091
James Brown - Give It Up or Turnit a Loose (Remix)	hook / riff	0.166667
James Brown - Say It Loud, I'm Black and I'm Proud	vocals / lyrics	0.333333
James Brown - The Payback	vocals / lyrics	0.166667
Joe Tex - Papa Was Too	drums	0
Kool & the Gang - N.T.	drums	0.0833333
Lafayette Afro Rock Band - Hihache	drums	0.0909091
Loleatta Holloway - Crash Goes Love (Yell Apella)	vocals / lyrics	0.1

Lyn Collins - Think (About It)	vocals / lyrics	0.25
Malcolm McLaren - Buffalo Gals	vocals / lyrics	0
Melvin Bliss - Synthetic Substitution	drums	0.363636
Mobb Deep - Shook Ones Part II	vocals / lyrics	0.363636
Mountain - Long Red	vocals / lyrics	0.214286
Ohio Players - Funky Worm	hook / riff	0.545455
Public Enemy - Public Enemy No. 1	vocals / lyrics	0
Run-DMC - Here We Go (Live at the Funhouse)	vocals / lyrics	0.384615
Skull Snaps - It's a New Day	drums	0.181818
Sly & the Family Stone - Sing a Simple Song	drums	0.166667
Syl Johnson - Different Strokes	hook / riff	0.25
The Honey Drippers - Impeach the President	drums	0.4
The J.B.'s - The Grunt	hook / riff	0.0909091
The Mohawks - The Champ	vocals / lyrics	0.454545
The Soul Searchers - Ashley's Roachclip	drums	0.461538
The Winstons - Amen, Brother	drums	0.0666667
Wu-Tang Clan - C.R.E.A.M.	vocals / lyrics	0.5
Zapp - More Bounce to the Ounce	drums	0.285714

Table 9: Recall rate for each of the 50 original tracks.

8 Conclusions and Future Work

This evaluation shows the viability of sample detection system based on local keypoint descriptors of an audio spectrogram. While the results are arguably low for an information retrieval task, they are well above the baseline of a simple audio fingerprinting algorithm.

It would be beneficial to perform a more formal evaluation on the performance with different types of sampling (e.g. vocal, instrumental, percussion) as well as with different common signal processing effects applied. Rather than using actual popular music examples, a more controlled evaluation, similar to that of Zhang et al [2], building a dataset of produced recordings using varying types

of processing and occluding the sample in different ways could shed light on where this system is failing. Alternatively, more analysis could be done on the recordings in this dataset to gain more insight into the types of samples that are causing problems. Better ground truth labels could be collected, especially if they could include typological information on the sampled audio which could help determine how to detect those samples that were missed.

This project aimed to explore the viability of SIFT features as fingerprints for sampling identification, but ultimately these features were meant for images and an alternative to SIFT would be beneficial. The analogy between object recognition and sampling detection can only carry so far. SIFT encodes for orientation invariance which is meaningless in an audio spectrogram. These features are also not robust to some signal processing effects such as audio reversal (although this could be accounted for with SIFT by extracted keypoint descriptors from the mirror image of every spectrogram, essentially doubling the size of the database). Several alternatives to SIFT have been developed for object recognition (e.g. SURF, FAST, ORB, and KAZE). Most of them focus on decreased processing time or required storage space rather than improved performance. Still, all these features have the same problem in that they are designed for images, not audio, and thus treat the vertical and horizontal axis in an identical manner. An audio specific alternative would ideally be able to adjust it's time resolution and frequency resolution separately.
The other issue is that the SIFT detector has no way to set the density of keypoints. This means that large areas of an image could return no keypoints. SIFT keypoint matching, even with the use of an ANN algorithm, is fairly consistent and effective at identifying keypoints that are extracted from the same audio source. A main cause for some samples being missed is likely because few or no keypoints are being extracted from that portion of audio. For a fingerprinting system, it is beneficial if keypoints exist in every time interval, or else a sample could be missed resulting in a false negative. A dense SIFT algorithm exists which foregoes the keypoint detection stage and extracts descriptors at every pixel (or at every multiple of n pixels, but this would cause alignment

issues). This would result in a massive amount of information being extracted and severely affect computation time. A more intelligent system would perform keypoint detection while enforcing some guarantee of keypoint density.

This project has proposed one possible system for detecting samples in a query audio file and evaluated this system on a scale that has not been done before in sample recognition studies. The evaluation has brought forth many insights into where future exploration could improve the task of sample detection.

References

[1] Robert Ratcliffe. A proposed typology of sampled material within electronic dance music. *Dancecult: Journal of Electronic Dance Music Culture*, 6(1):97–122, 2014.

[2] Xiu Zhang, Bilei Zhu, Linwei Li, Wei Li, Xiaoqiang Li, Wei Wang, Peizhong Lu, and Wenqiang Zhang. Sift-based local spectrogram image descriptor: a novel feature for robust music identification. *EURASIP Journal on Audio, Speech, and Music Processing*, 2015(1):1–15, 2015.

[3] Nicholas J Bryan and Ge Wang. Musical influence network analysis and rank of sample-based music. In *ISMIR*, pages 329–334, 2011.

[4] WhoSampled. Whosampled: Discover music via samples, cover songs and remixes, 2016. [Online; accessed 2016-06-01].

[5] Jordan L Whitney and Colby N Leider. Automatic sample recognition in hip-hop music based on non-negative matrix factorization. In *Audio Engineering Society Convention 134*. Audio Engineering Society, 2013.

[6] JMH van Balen, Martın Haro, Joan Serra, et al. Automatic identification of samples in hip hop music. In *Proceedings of the 9th International Symposium on Computer Music Modeling and Retrieval (CMMR)*, pages 544–551, 2012.

[7] Avery Wang et al. An industrial strength audio search algorithm. In *ISMIR*, pages 7–13, 2003.

[8] Sébastien Fenet, Yves Grenier, and Gael Richard. An extended audio fingerprint method with capabilities for similar music detection. In *ISMIR*, pages 569–574, 2013.

[9] Joan Serrà, Massimiliano Zanin, and Ralph G Andrzejak. Cover song retrieval by cross recurrence quantification and unsupervised set detection. 2009.

[10] Thierry Bertin-Mahieux and Daniel PW Ellis. Large-scale cover song recognition using hashed chroma landmarks. In *Applications of Signal Processing to Audio and Acoustics (WASPAA), 2011 IEEE Workshop on*, pages 117–120. IEEE, 2011.

[11] Michael Casey and Malcolm Slaney. Fast recognition of remixed music audio. In *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, volume 4, pages IV–1425. IEEE, 2007.

[12] Jaap Haitsma and Ton Kalker. A highly robust audio fingerprinting system. In *ISMIR*, volume 2002, pages 107–115, 2002.

[13] Sébastien Fenet, Gaël Richard, Yves Grenier, et al. A scalable audio fingerprint method with robustness to pitch-shifting. In *ISMIR*, pages 121–126, 2011.

[14] Jacob George and Ashok Jhunjhunwala. Scalable and robust audio fingerprinting method tolerable to time-stretching. In *Digital Signal Processing (DSP), 2015 IEEE International Conference on*, pages 436–440. IEEE, 2015.

[15] Shumeet Baluja and Michele Covell. Waveprint: Efficient wavelet-based audio fingerprinting. *Pattern recognition*, 41(11):3467–3480, 2008.

[16] David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

[17] David G Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004.

[18] Ebrahim Karami, Siva Prasad, and Mohamed Shehata. Image matching using sift, surf, brief and orb: Performance comparison for distorted images.

[19] Spotify Erik Bernhardsson. Annoy. `https://github.com/spotify/annoy`, 2013.

[20] Brian McFee, Colin Raffel, Dawen Liang, Daniel PW Ellis, Matt McVicar, Eric Battenberg, and Oriol Nieto. librosa: Audio and music signal analysis in python. In *Proceedings of the 14th Python in Science Conference*, 2015.

[21] Christian Schörkhuber and Anssi Klapuri. Constant-q transform toolbox for music processing. In *7th Sound and Music Computing Conference, Barcelona, Spain*, pages 3–64, 2010.

[22] Andrea Vedaldi and Brian Fulkerson. Vlfeat: An open and portable library of computer vision algorithms. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 1469–1472. ACM, 2010.

[23] G. Bradski. Open source computer vision library. *Dr. Dobb's Journal of Software Tools*, 2000.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011.

[25] Jan Van Balen. Automatic recognition of samples in musical audio. In *Master's thesis, Universitat Pompeu Fabra*, 2011.

[26] Dan Ellis. Robust landmark-based audio fingerprinting. *web resource, available: http://labrosa.ee. columbia.edu/matlab/fingerprint*, 2009.