Audio Fingerprinting
A Decomposing Study

Master's Thesis

By

Niklas Gälldin and Victor Hultman

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

LUNDS UNIVERSITET

Abstract

Audio fingerprinting is a widely employed technique that involves generating unique fingerprints for given audio signals that later can be used for identification. A well-known example of this is the Shazam application where the concept is to match a short song snippet with a database to find the name of the song and artist. Generally, the audio fingerprints are created by applying a time-frequency transform on the audio signal and extracting the most prominent features in the time-frequency domain. There are different transforms with different properties but the standard choice is the short-time Fourier transform (STFT). This study compares the performance of the STFT with the Hyper Localized Wavelet Transform (HLT) within an audio fingerprinting pipeline, focusing on three key metrics: correctly identifying songs (accuracy), robustness towards noise, and memory. Results indicate that while the STFT and the HLT demonstrate comparable accuracy, the latter exhibits superior noise robustness with a smaller memory usage. The STFT was found to generate approximately 1.23 times more data when creating the fingerprint database compared to the HLT.

Sammanfattning

Ljudfingeravtryck är en välkänd teknik som genererar unika fingeravtryck för ljudsignaler vilka som senare kan användas för identifiering. Ett välkänt exempel på detta är Shazam-applikationen vars koncept är att matcha en kort låtsnutt med en databas för att hitta namnet på låten och artisten. Generellt skapas ljudfingeravtrycken genom att applicera en tids-frekvenstransform på ljudsignalen och extrahera de mest framträdande komponenterna i tids-frekvensdomänen. Det är standard att använda korttids Fouriertransformen (STFT) men det finns också transformer med andra egenskaper. Denna studie jämför prestandan för STFT med Hyper Localized Wavelet Transform (HLT) inom en ljudfingeravtrycksprocess, med fokus på tre viktiga mätvärden: korrekt identifiering av låtar (precision), robusthet mot brus och minnesanvändning. Resultaten visar att medan STFT och HLT uppvisar jämförbar precision, visar den senare överlägsen robusthet mot brus med mindre minnesanvändning. Vidare visade sig STFT generera ungefär 1,23 gånger mer data vid skapandet av fingeravtrycksdatabasen jämfört med HLT.

Preface

In this thesis work, Niklas has mainly been working with Chapters 1, 4 and 5. Victor has mainly been working with Chapters 2, 3 and 6. The simulations, experiments and illustrations were a joint effort.

We would like to thank everyone at LucentWave for giving us the opportunity to collaborate on this Master's Thesis. Thank you to Henrik Jörntell for providing us a workspace at BMC during our thesis work. A special thanks to Kaan Kesgin for guidance, for providing equipment for running simulations, and for introducing us to the Julia programming language. We would also like to give a big thank you to Fredrik Edman for the support and feedback during the writing of this thesis. Finally, we would like to thank our friends and family for supporting us and encouraging us along the way.

Popular Science Summary

Have you ever found yourself in a situation where you hear a captivating song at a pub or a restaurant, but can not recall its name or artist? Searching for it based on fragmented lyrics often proves futile and frustrating. However, audio fingerprinting offers a solution to this common dilemma. By simply recording a snippet of the song with your phone, a unique fingerprint of the song can be extracted and matched with a large database of song fingerprints in no-time.

The first step of audio fingerprinting is to perform a time-frequency decomposition of the audio to find the most characterizing frequencies over time. This Master's thesis explores two different methods for time-frequency decomposition, aiming to enhance the precision and robustness of audio fingerprinting systems for song identification. By comparing the short-time Fourier transform (STFT) and Hyper Localized Wavelet Transform (HLT), this study seeks to evaluate their accuracy in correctly identifying songs.

Time-frequency decomposition methods play a pivotal role in extracting meaningful features from audio signals that are later used to create audio fingerprints. In summary, the decomposition is created by dividing an audio signal into short time-segments; it is then possible to extract the frequencies that occur within each segment from the decomposition. The audio fingerprint is then created by mapping frequencies from each segment in the decomposition in a way that is as unique as possible for each audio signal.

However, more often than not there is some amount of noise or distortions occurring when we want to identify a song. The noise can make it more difficult to get an accurate time-frequency decomposition which is crucial for creating a unique audio fingerprint to match with the database. It is therefore important to use a time-frequency decomposition that is accurate and resistant to noise.

Our experiments reveal that the STFT and the HLT are quite similar in

accurately identifying songs but the HLT is superior when more noise is present. The HLT requires more time to perform the decomposition, but in a practical setting, it can be argued to not be a significant drawback.

In conclusion, this Master's thesis highlights the significance of time-frequency decomposition methods in audio fingerprinting for song recognition. By providing insights into the performance of the STFT and the HLT, this study shows the differences and potential of both methods.

Contents

Chapter 1

Introduction

1.1 Background

Audio fingerprinting is a method that deterministically generates a compact version of a given audio signal [1]. This is most commonly used in audio recognition where a short recording of e.g. a song is captured and its audio fingerprints are matched with others in a beforehand generated fingerprint database. Since the database in many applications can contain many thousands of fingerprints, one typically requires that the lookup should be performed at high speed and that the fingerprints should be unique. To create the fingerprints, a time-frequency decomposition is usually performed on the audio signal where the standard method to use is the short-time Fourier transform (STFT) [2]. However, there exist many other transforms that can be used as well. One such transform is the Hyper Localized Wavelet Transform (HLT) which is a type of wavelet transform [3]. The transform has been shown to achieve great resolution in the decomposition of complex acoustic data and can provide detailed frequency content for short-bursting signals. These properties could make the transform highly interesting in the application of audio fingerprinting and are thus the topic of this Master's Thesis.

A widely used application that utilizes audio fingerprinting is Shazam, www.shazam.com. Although the exact implementation of Shazam is officially undisclosed, there exist several interpretations based on a paper by one of the founders [4]. Shazam is an audio recognition application, where the user records a snippet of a song. A fingerprint of the snippet is created and matched with a large database of song fingerprints. The song name and artist are returned to the user if there is a match between the snippet

and a song in the database.

The user would usually be in a noisy environment, e.g. a pub or a restaurant when recording the snippet. To be able to accurately identify the song it is thus paramount that the Shazam algorithm is robust towards different types of distortions to the audio signal.

1.2 Aim & Objectives

The aim of this project is to evaluate the performance of the HLT within the area of audio fingerprinting. We aim to provide relevant benchmarks for the performance of the HLT by comparing it with the performance of the STFT, which is the standard time-frequency decomposition method.

Since the time-frequency transform is just one part of the algorithm, it can be difficult to evaluate how the HLT and the STFT directly impact the performance. Therefore, to compare the two transforms we will first introduce and analyze the two transforms outside the domain of audio fingerprinting.

The first experiment will be a visual comparison of the time-frequency decomposition generated by the STFT and the HLT to see the difference in time- and frequency resolution. The second experiment will be to see how robust both methods are by calculating the error between a noise-free signal and a noisy signal after both of them have been transformed.

Thirdly, to test if the HLT can outperform the STFT in an audio fingerprint setting, a pipeline based on the Shazam algorithm will be implemented and results from simulations using both the STFT and the HLT will be provided. The evaluation of the performance will be based on benchmarks such as accuracy vs Signal-to-noise ratio (SNR) and the required size of the database. The overall performance of our implementation of the pipeline will be verified by comparing it with the results from the original paper [4]. Shorter snippets than those used in the aforementioned paper will also be tested to see if the HLT is superior to the STFT in these cases.

1.3 Outline

Chapter 2 covers the theory and concepts of time-frequency decompositions. In Chapter 3 the Shazam algorithm is introduced together with our choice of hyper parameters. The experiments and simulations that were carried out are presented in Chapter 4 and the results in Chapter 5. Lastly, the theory and our results are discussed in Chapter 6.

Chapter 2

Time-frequency transforms

In harmonic signal analysis, a 1-dimensional signal in the time domain can be better understood by transforming it into the 2-dimensional time-frequency domain. In audio fingerprinting, this transformation will enable the extraction of the most characterizing features of the signal which will act as a compression due to the removal of insignificant elements. There exist many different time-frequency transforms but the most fundamental one is the short-time Fourier transform (STFT). The STFT can be seen as a special case of a Wavelet transform which is a family of time-frequency transforms with some common properties. The Hyper Localized Wavelet Transform (HLT) is another example of such a transform. Both transforms and their properties will be presented in the following sections.

2.1 Discrete and continuous signals

In practical applications, the signal to be analyzed is a discrete-time signal usually obtained from sampling a continuous-time signal with some sampling frequency F_s. For audio signals, we usually have $F_s = 44.1\,\text{kHz}$ or $F_s = 48\,\text{kHz}$ which is a little bit more than twice the maximum human hearing frequency of 20 kHz [5]. According to the Shannon-Nyquist theorem, these sampling frequencies will create no aliasing, and thus all relevant signals for human perception can be reproduced. For acoustic signals, the highest-pitched instruments usually have a max frequency below 4000 Hz thus even with decimation by a factor of 6, with low-pass filtering, we will have no aliasing [6]. Utilizing downsampling is important to increase the processing speed since the time complexity of time-frequency transforms depends on the length of the signal.

2.2 Discrete short-time Fourier transform

In the most fundamental time-frequency analysis, one usually utilizes the STFT where the Fourier transform is applied on consecutive time windows of the signal. The length of the time windows, W, will impact the frequency and time resolution, and the shape of the window will mitigate some of the possible spectral leakage [7]. Any overlap of the windows will compensate for the loss of information that may happen between two non-overlapping windows. The number of frequency bins produced by the STFT, denoted as $nFFT$, is determined by the value of W. However, $nFFT$ can be larger than W if zero-padding is applied which can be favourable to have $W = 2^k$ where $k \in \mathbb{N}$, so that fast Fourier algorithms will work more efficiently [8]. Other than this the only reason to zero-pad the signal is if one wants a certain value for $nFFT$, since it does not increase the frequency resolution [8]. In the context of resolution, there exists a principle called the uncertainty principle from where it follows that there exists a trade-off between the achievable time or temporal resolution and the frequency resolution [9]. If the signal that is transformed is real-valued, the values at frequency bin $i \in [\frac{nFFT}{2}, nFFT]$ will just be the complex conjugated value at bin $i - \frac{nFFT}{2}$. Thus, only the first $\frac{nFFT}{2}$ bins will carry any valuable information about the frequency content. The time complexity of the STFT is

$$O(T\, nFFT \log(nFFT)), \tag{2.1}$$

where T is the number of windows of length W that fits in the signal.

2.3 Wavelet transform

The HLT is a wavelet transform which means that it uses functions called *wavelets* to perform the transformation to the the time-frequency domain. The wavelets are waveforms that are localized in time and have a finite time duration. Similar to how the signal is decomposed using the STFT, we have for Wavelet Transforms that the signal is represented as a linear combination of scaled and time-shifted versions of a function ψ called the *mother wavelet* [10]. The scaled and time-shifted versions of the mother wavelet are called the *child wavelets* and can be defined in continuous time t as

$$\psi_{a,b}(t) = \frac{1}{\sqrt{a}} \psi \left(\frac{t-b}{a} \right), \tag{2.2}$$

where a is the scaling factor and b is the time shift. The scaling factor will contract the wavelet if a is small and have the reverse effect when a is large,

see Figure 2.1. From the figure, we can see how a change in a is analogous to changing the frequency of the wavelet since the lower wavelet can be viewed to correspond to a lower frequency while the upper wavelet corresponds to a higher frequency. The other parameter b will determine where in time the wavelet is localized.

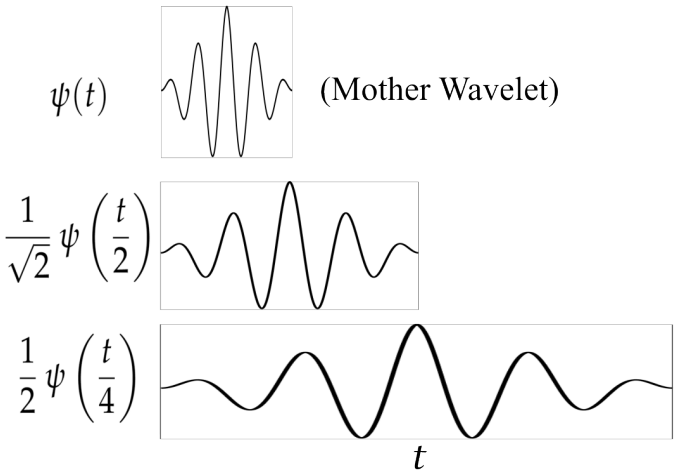


Figure 2.1: The impact of the scaling factor a on the wavelet ψ using $a = 1$ (Mother wavelet), $a = 2$ and $a = 4$.

When calculating the wavelet transform we convolve the child wavelets with the signal and receive a time-frequency decomposition of the signal. However, since directly performing the convolution operation would require $O(N^2)$ number of operations we, in practice, transform the wavelets and the signal using the Fast Fourier Transform (FFT). After this, the transformed wavelet and signal are multiplied element-wise and the result is transformed back to the time domain using the Inverse Fast Fourier Transform (IFFT). The element-wise multiplication is possible due to the convolution theorem which states that

$$g * h = \mathcal{F}^{-1}\left(G \cdot H\right), \tag{2.3}$$

where

$$G = \mathcal{F}(g), H = \mathcal{F}(h) \tag{2.4}$$

and $\mathcal{F}(.)$ denotes the Fourier transform operator. Both the FFT and IFFT require $O(N\log(N))$ operations which means that we reduce the complexity by using this method instead of directly computing the convolutions.

2.4 Hyper Localized Wavelet Transform

The wavelets of the HLT are defined as a modified version of

$$\psi_f(t) = -\left(2\log\left(\frac{2f}{n}\right) + \log(t+\epsilon)^2\right) e^{2\pi i f t}\left(e^{-t^{2\log(n)}\left(\frac{2f}{n}\right)^{4\log(n)}}\right), \quad (2.5)$$

where t is the time in seconds, f [Hz] is the frequency of interest, and the constants ϵ and n are positive and non-zero [3]. The value of f can be interpreted as the scaling factor a as previously mentioned, while the localization factor b has not been defined but would correspond to where in the analyzed signal the wavelet should be non-zero. An example of what the wavelet looks like for $f = 10\,\text{Hz}$ can be seen in Figure 2.2 below. The modified version of the wavelet in 2.5 that is used in this project can not be disclosed due to company confidentiality.

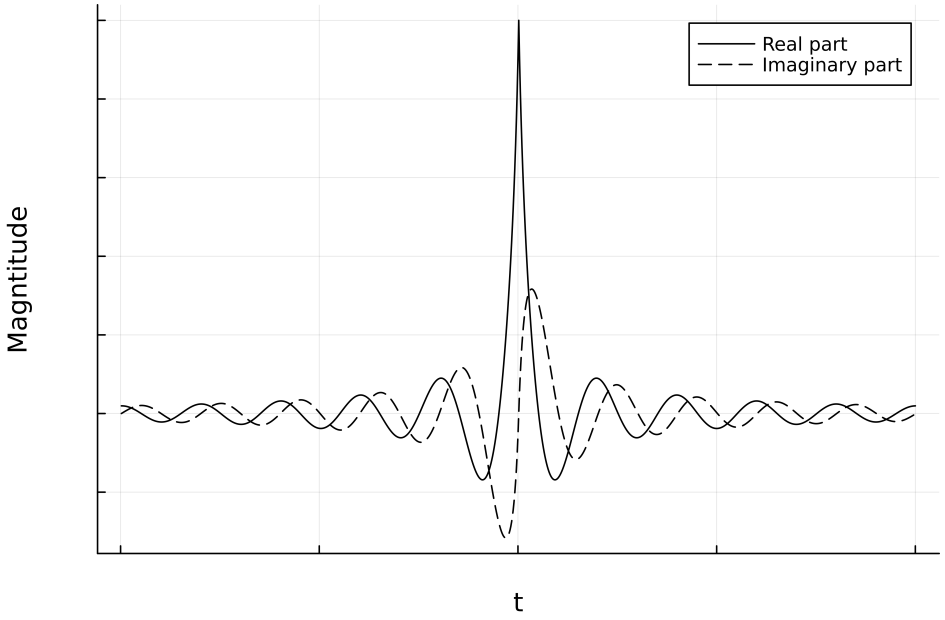


Figure 2.2: The Hyper Localized Wavelet for $f = 10\,\text{Hz}$.

When transforming a signal using the HLT we can assume that the length of the analyzed signal is N_s and we put the length of the wavelets to be N. We then decompose the signal by dividing it into $T = \lceil \frac{N_s}{N} \rceil$ consecutive windows where we can zero-pad the signal so that the new signal

length $\hat{N}_s \mod N = 0$. This will give T windows of data with N samples each. We then proceed as previously described where both the wavelet in (2.5) for different values of f and the window of data is Fourier transformed. The two resulting signals are then multiplied element-wise and after that inverse transformed in order to obtain the time-frequency decomposition. If we denote M as the number of frequencies analyzed, i.e. the number of wavelets, we have that $M = \frac{N}{2}$ is the optimal value since if it is lower then we will get a decreased frequency resolution, and if it is larger then we gain nothing since we can only represent $\frac{N}{2}$ number of frequencies in a signal of length N without aliasing. The time complexity of the HLT is thus

$$O(T\,M\,N\log(N)) = O(T\,N^2\log(N)), \tag{2.6}$$

since we perform the FFT and IFFT operations M number of times for all T windows of data.

Chapter 3

Shazam Algorithm

To evaluate the performance of the HLT within the area of audio fingerprinting we have implemented a version of the original Shazam algorithm described by Wang [4]. Although the details of the algorithm have not been made public due to commercial reasons, there still exist many different publicly available implementations based on the original paper [11], [12] and [13]. All implementations follow the same general structure that is described in Figure 3.1. When creating our implementation we were inspired by the implementation by Strauss [11]. In the following sections, we will go through each part of the algorithm and our implementation.

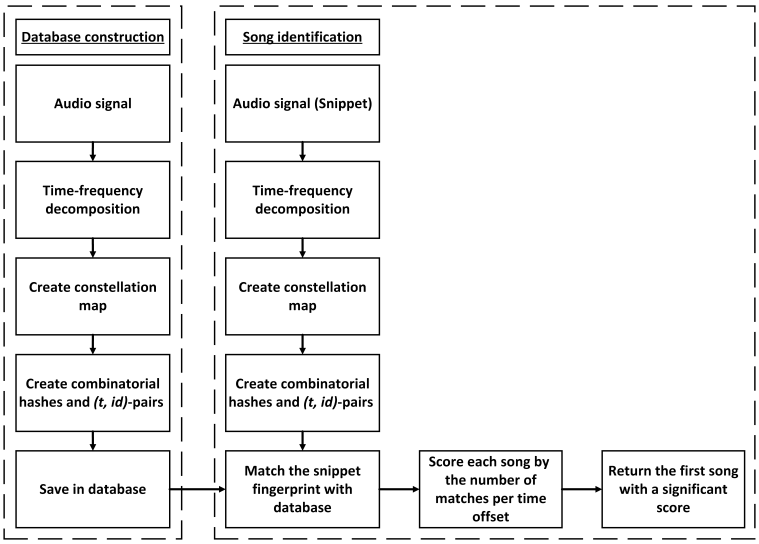


Figure 3.1: Overview of the Shazam pipeline.

3.1 Audio fingerprints

An audio fingerprint is a unique and compact representation of an audio signal. In the context of music, the different frequencies of the musical notes that are played are what makes the song unique. However, not only which notes are played contribute to the uniqueness of the song, but also how they relate to each other in time. This observation is the basis that the fingerprints rely on as we will describe in more detail.

In order to produce a compact representation of the audio signal a time-frequency decomposition is performed which gives information about the frequency content of the signal at different time instances. The most characteristic frequencies will correspond to a peak in the spectral density which can be retrieved from the decomposition. A certain number of spectral peaks are selected at each discrete time instance and can be seen as the most important features in that time window.

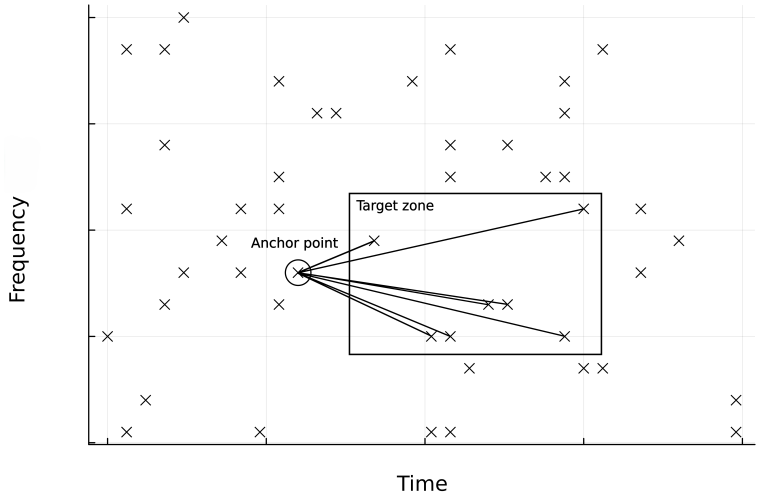


Figure 3.2: The pairs that are made for a certain constellation point (anchor point) and how the selection is limited by a time-difference bound and an absolute frequency-difference bound (target zone).

All selected peaks for a given audio signal create a so-called *constellation map* as it is a set of points in the time-frequency plane [4]. However, the audio fingerprint is not the created constellation map but instead a set of pairs of constellation points. A pair between two points is created if one constellation point (anchor point) has another constellation point within its target

zone. A constellation point belongs in the target zone of an anchor point if two criteria are fulfilled: the time difference is within a certain bound and the absolute value of the difference in frequencies is within another bound. The effects of the criteria are illustrated in Figure 3.2.

The pair of points are used to create a combinatorial hash where the two frequencies and their time difference are used to create the hash-value. For example, if we have 1024 frequency bins then we need 10 bits each to represent the two frequencies and if we let the time difference be represented by a 12 bit integer we get a 32 bit integer as a hash, see Figure 3.3. By creating these hashes we enable fast access when the fingerprint of the snippet is to be matched with the fingerprint of all songs in the dataset. In the resulting hash table, the time belonging to the anchor point is stored together with the id of the song which we will refer to as a (t, id)-pair. The set of hashes and their corresponding values for a certain song is the resulting audio fingerprint of that song.

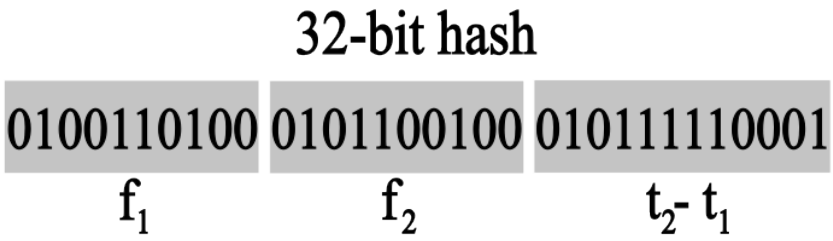


Figure 3.3: Illustration of how the combinatorial hashes in the Shazam pipeline are calculated. In this example a hash is created from two constellation points (t_1, f_1) and (t_2, f_2) where the first point is the anchor point.

3.2 Database construction

When the set of hashes and corresponding (t, id)-pairs have been created it is stored in a hash table that will work as a database. The hash table will then have the hash keys described in the previous section and the values being a list of (t, id)-pairs where a certain song-id may appear multiple times but with a different time value. Since there is no noise in the audio signals used to create the database (they are the original tracks) there is little to no need to tune the hyper parameters to take into account robustness. For example, the number of peaks could be high in order to have a detailed representation of the spectral density.

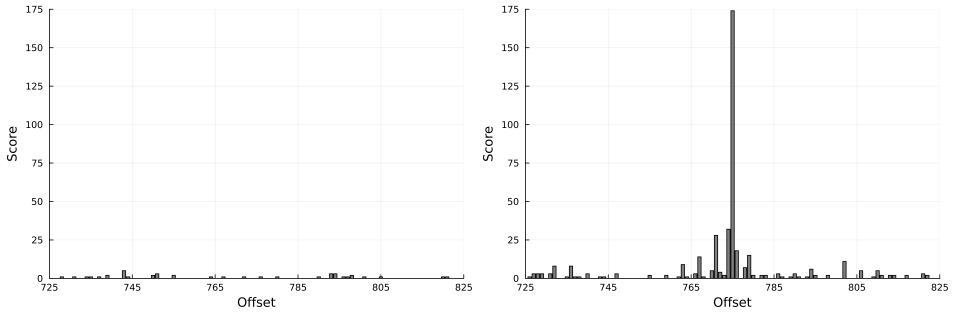
3.3 Song identification

The goal of the Shazam algorithm is to identify a song based on a short and possibly distorted snippet of the song. In order to match the snippet with the database the audio fingerprint of the snippet is created. This gives a set of combinatorial hashes mapping to (t, id)-pairs where the song-id is arbitrary since the song is at this point unknown. After creating the hashes they are used to retrieve the (t, id)-pairs in the database. A match is created between each (t, id)-pair for the hash in the database and the (t, id)-pair of the snippet. For each match, we store the following information in a list

$$\text{matches}[id] = (t_{\text{snippet}}, t_{\text{song}}), \tag{3.1}$$

where t_{snippet} is the time value of the (t, id)-pair of the snippet, t_{song} and id comes from the (t, id)-pair from the hash in the database. This means that each matches$[id]$ will have a list of values of the type that is assigned in (3.1). After the list has been completed another list of scores by offset is created for each song. This list will be created by going through each value in (3.1) and incrementing the score by 1 for the value at bin $offset = t_{\text{song}} - t_{\text{snippet}}$. After this, the final score for the song when matched to the snippet will be given by the max-value in the offset-list. Each song that has matched with the snippet will have a threshold score that determines whether the score that is produced by the matching step is significant. The level of significance, α, can be chosen arbitrarily but to ensure that the number of false-positives is low it should be selected so that $\alpha \leq 0.1\,\%$. The threshold scores are determined by creating a probability distribution, f, of the highest score from a matching with an incorrect song, which we will refer to as the highest incorrect score. Given α, the threshold can be determined by solving for x in $F_X(x) = P[X \leq x] = 1 - \alpha$ where F_X is the cumulative distribution and X is a stochastic variable that describes the observations of the highest incorrect scores.

The reason for using the offset in time as a factor in the evaluation of the score is that the constellation points of the snippet will relative to each other occur at the same time as the constellation points of the correct song. For the incorrect songs, all the matches will have very different offsets as the likelihood that several combinatorial hashes not only match with the snippet but also have the same offset is rather small [4]. In Figure 3.4, the scores for different offsets of a song have been plotted in a diagram where a 5 s snippet was taken from the middle of a song and matched with each song in the dataset. As can be seen, the incorrect song has several matches with the snippet but no clear maximum value for a certain bin while the



(a) Incorrect Song (b) Correct Song

Figure 3.4: The scores for different offsets in time when matching a snippet with (a) an incorrect song and (b) the correct song.

correct song has a clear maximum value. The maximum height for a certain bin would correspond to the score for the snippet with that particular song. In this example, it is clear that the score for the snippet and the correct song will be higher than the incorrect one. To correctly identify the song, this score needs to be higher than the threshold score for the correct song.

3.4 Peak Extraction

In this thesis, the number of peaks that are selected at each time instance is a hyper parameter that can be tuned and its value will impact the robustness of the algorithm. To increase the robustness for any number of peaks the mean value of some of the largest peaks for each frequency bin can be computed and only the peaks that are larger than this mean value multiplied by a coefficient, c, are kept. The coefficient c is another hyper parameter that needs to be tuned for the algorithm to perform well. Additionally, another improvement of the detection of local maxima in the spectrum for each time instance is to divide the frequency bins into frequency bands and only keep the highest peak in each respective band. This method will give peaks that are evenly spread out over the entire spectrum and ensure that not only lower frequencies that tend to have higher peaks will be selected [13]. A linear division into bands would for example correspond to the bands seen in Figure 3.5.

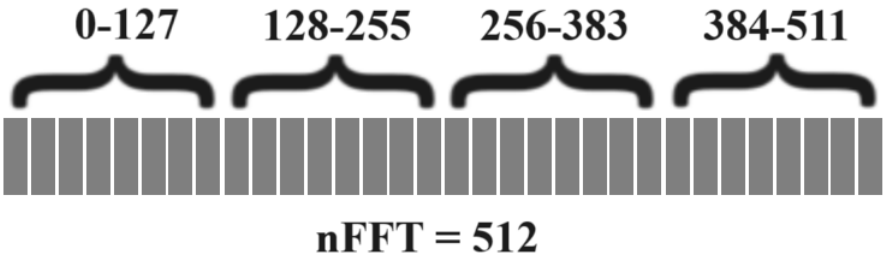


Figure 3.5: Linear division into 4 bands using $nFFT = 512$ number of frequency bins. The numbers correspond to the bins that belong to a certain band.

Since the octaves for musical notes follow an exponential relation the bands can also be logarithmic [14]. Using the previous example this would correspond to the bands seen in Figure 3.6 below.

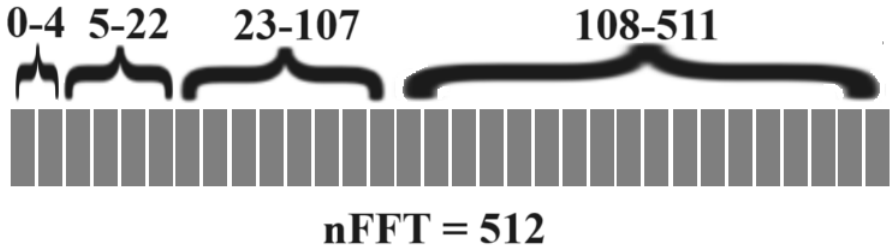


Figure 3.6: Logarithmic division into 4 bands using $nFFT = 512$ number of frequency bins. The numbers correspond to the bins that belong to a certain band.

The described steps of the peak extraction are summarized in pseudo-code in Algorithm 1.

Algorithm 1 Peak extraction

procedure PEAK EXTRACTION(S) ▷ S is the audio signal
  $T \leftarrow$ Number of time windows
  $nFFT \leftarrow$ Number of frequency bins
  $N \leftarrow$ Number of peaks
  $M \leftarrow \emptyset$ ▷ Mean value $1 \times nFFT$-vector
  $\boldsymbol{S} \leftarrow \boldsymbol{F}(S)$ ▷ $\boldsymbol{F}$ is the transform, $\boldsymbol{S}$ is a $T \times nFFT$-matrix
  for $j \leftarrow 0 : nFFT - 1$ **do**
    find set $\mathbb{I}_j$ of time instances for the 100 largest local maxima for $\boldsymbol{S}[:, j]$
    $M[j] \leftarrow c \cdot \frac{1}{100} \sum_{i \in \mathbb{I}_j} \boldsymbol{S}[i, j]$
  end for
  for $i \leftarrow 0 : T - 1$ **do**
    find all local maxima (peaks) for $\boldsymbol{S}[i, :]$
    select largest peak $\boldsymbol{S}[i, j]$ in each frequency band $1 \rightarrow N$
    select peaks $\boldsymbol{S}[i, j]$ that fulfill $\boldsymbol{S}[i, j] \geq M[j]$
  end for
  return $\boldsymbol{S}[i, j]$
end procedure

3.5 Significance

In a real-life situation when trying to identify a song by using the Shazam algorithm, there is a possibility that the song is not part of the fingerprint database. To handle this the threshold scores for every song in the dataset were generated. As explained in section 3.3, the threshold specifies the minimum score needed to be able to predict that particular song. In this thesis, the threshold was calculated by observing the highest score of all incorrectly matching songs when no noise was added and then computing the fraction of the highest incorrect score divided by the correct score. We will refer to this fraction as the *relative threshold*. The value of the relative threshold for all songs can be seen as observations from some unknown distribution. Initial testing indicated that a suitable distribution for the observations was the Gamma distribution and the parameters were estimated by using maximum likelihood, see example in Figure 3.7. A false positive rate could then be selected to determine the joint relative threshold which was multiplied with the correct score for each song to obtain each threshold. When trying to identify a snippet during simulation the snippet was

only matched with the correct song and if the score was higher than the threshold the song was considered to be correctly identified. The false positive rate was equal to $\alpha = 0.1\,\%$.

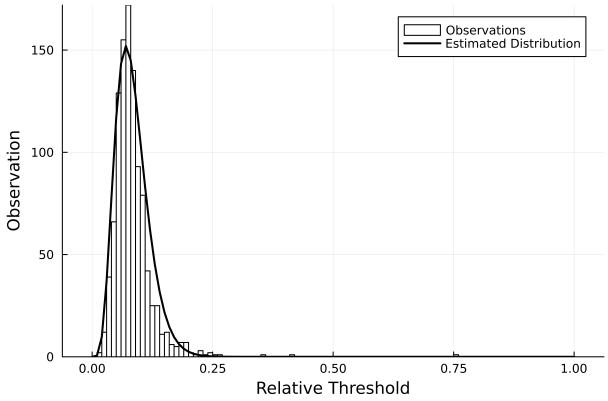


Figure 3.7: Estimated Gamma distribution based on the histogram of observed relative thresholds for each song in the dataset. The Gamma distribution is scaled to match the height of the data.

3.6 Settings and tuning

As has been described in previous sections there exists several hyper parameters that introduce many degrees of freedom to the tuning of the algorithm. Although all parameters in some way will impact the performance, only parameters that directly impact the measurements in each experiment were focused on. This section will go through the investigated and fixed hyper parameter values for the Shazam algorithm implemented in this thesis.

3.6.1 Number of peaks

The number of peaks N that are selected from the spectral density at each discrete time instance is a tunable hyper parameter. This number will impact how detailed the fingerprint will be and thus how much the data is compressed. Optimally, from a uniqueness perspective, N should be equal to the actual number of frequencies that are present in the audio signal in a certain time window, but this information is not known and even if it was known it would vary between different songs. To manage this we intro-

duced the hyper parameter c. By having $c > 0$, the relatively low peaks will get filtered out which means that the number of selected peaks from a time window will be smaller than or equal to N. If $c = 0$, the maximum possible number of peaks/constellation points, given N, would be obtained. This maximum would then be given by $N \cdot T$, where T is the number of time windows.

When tuning c the trade-off is that with more data we need more memory to store the information and we want a compression of the audio signal by producing the audio fingerprint, thus c should be low. On the other hand, c should be high so that insignificant spectral peaks are filtered out. Suitable values that were found were $c = 0.6$ when creating the database and $c = 0.9$ when matching the snippets to the database. The number of peaks was selected to be $N = 6$ after some initial testing where the trade-off between accuracy and memory was considered.

3.6.2 Target zone

When creating the combinatorial hashes only points that lay within a target zone are paired with the anchor point. The larger the target zone is the more hashes will be created for each anchor point which means that the compression will decrease as we get a more detailed fingerprint. One special case is to have no bounds and thus create combinatorial hashes with all points for each anchor point. This would however drastically increase the number of generated hashes which is not desirable with regard to memory. Other than this observation the exact values of the bounds are not well defined. We decided to use the bounds $t_l \leq \Delta t \leq t_u$ and $0 < |\Delta f| \leq \frac{nFFT}{6}$ where $nFFT$ is the number of frequency bins used in the transform, Δt is the time-difference measured in time-windows and Δf is the frequency-difference respectively between an anchor point and another constellation point. The upper and lower time bounds, t_l and t_u, can be varied depending on how many hashes that is wanted on average since a larger target zone will lead to more point-pairs. Suitable values that were found for $nFFT = 2048$ were $t_l = 2$ and $t_u = 4$ which means that for each anchor point, we will create a pair with another point that occurs between 256 ms and 768 ms from the time of the anchor point since we have 50% overlap. The upper bound t_u was varied to observe how the size of the fingerprint database impacts the overall performance, the analyzed values were $t_u = 2, 3, ..., 10$. The lower bound $t_l = 2$ was held constant and selected to be equal to 2 since this is the lowest value it can be without having any overlap with the time window of the anchor point.

3.6.3 Number of frequency bins and window length

The number of frequency bins, $nFFT$, that are used when performing the time-frequency decomposition is another hyper parameter. If it is larger than the window length W it will imply that each window is zero-padded. As has been previously discussed in section 2.2, the songs are real-valued which means that we will obtain $\frac{nFFT}{2}$ unique frequency bins using these settings, and the window length in seconds would then be equal to $\frac{W}{F_s}$. To not have too many hash collisions when we do not zero-pad each window and to not exceed the 10 bits of frequency information that is used when creating the combinatorial hashes, we will get a lower and upper bound respectively for W. The upper bound will be $W = nFFT = 2048 = 2^{11}$ while the lower bound is not well defined but could probably be around $W = 64 = 2^6$. When comparing with the reference implementation by Wang [4] we used $W = nFFT = 2048$ but when we tried shorter lengths of the snippets i.e. smaller than 5 s we tried $W = nFFT = 2^k$ for $k = 6, 7, ..., 11$. When we did zero-pad we had $nFFT = 2048$ while $W = 2^k$ for $k = 6, 7, ..., 11$. The reason for trying shorter window lengths was to achieve a more detailed representation of the audio signal in the time domain in order to possibly achieve a higher accuracy for the shorter snippet lengths. However, in the conducted simulations this did not give a better performance and the zero-padding did not improve the results significantly. The reason why this might be the case is that when $W = nFFT < 2048$ we get too few frequency bins to combine thus increasing the risk of hash collisions which implies less unique fingerprints. Since the zero-padding does not increase the resolution the same problem will appear when $W < nFFT = 2048$ because even though we have more frequency bins, adjacent spectral peaks will not be resolved thus leading to a degradation in performance. For these reasons, we let $W = nFFT = 2048$.

3.6.4 Frequency bands

By dividing the frequency bins into different bands as explained in section 3.1 a better performance could perhaps be achieved. After some initial testing, it was observed that logarithmic bands gave the best performance with regard to overall accuracy for different SNR:s. Because of this, it was decided that all subsequent tests would use logarithmic bands.

3.6.5 Summary of settings

The parameter values that were used for the comparison with the reference implementation are summarized in Table 3.1 below. These parameter values were also used when analyzing the impact of the database size, except that $t_u = 2, 3, \ldots, 10$.

Table 3.1: The parameters for the comparison with the reference implementation.

Parameter	N	W	$nFFT$	t_l	t_u	c_{database}	c_{snippet}
Value	6	2048	2048	2	4	0.6	0.9

Chapter 4

Experiments & Data preprocessing

Three different comparative experiments were conducted for the HLT where the performance of the STFT was seen as the benchmark. The experiments were a time-frequency decomposition of different sinewave signals, evaluation of the robustness by using audio signals with and without Additive White Gaussian Noise (AWGN) and accuracy performance in the implemented Shazam pipeline. The idea with the first two experiments was to compare the two transforms in a setting where the transform and its hyperparameters make the difference. In the third experiment, the performance of the HLT and the STFT was compared in the same Shazam pipeline implementation.

4.1 Time-frequency decomposition

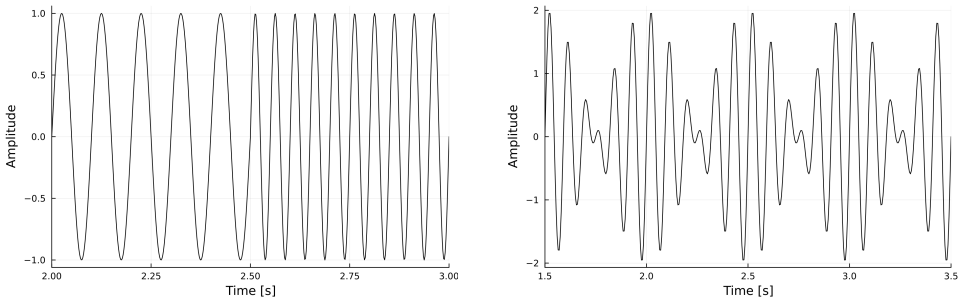
A comparison of the time-frequency decomposition between the STFT and the HLT was done for two different signals in the time domain with different frequency content. For the HLT, wavelets of the same size as the input signal were used, resulting in a time-frequency decomposition of each input sample. For the STFT rectangular windows were used with full overlap, $nFFT = 256$, and four different window lengths $W = \frac{nFFT}{8}, \frac{nFFT}{4}, \frac{nFFT}{2}, nFFT$ to show how the window length impacts the resolution. The first signal that was tested was

$$y_1 = \begin{cases} \sin 2\pi t F_1, & t \in [0, 2.5] \\ \sin 2\pi t F_2, & t \in [2.5, 5], \end{cases} \tag{4.1}$$

with a sampling frequency of $F_s = 512\,\text{Hz}$ and where $F_1 = 10\,\text{Hz}$, $F_2 = 20\,\text{Hz}$, and t is the time in seconds. A one-second snippet of signal y_1 is shown in Figure 4.1a. The second signal tested was

$$y_2 = \sin 2\pi t F_1 + \sin 2\pi t F_2, \quad t \in [0, 5], \tag{4.2}$$

with a sampling frequency of $F_s = 200\,\text{Hz}$ and where $F_1 = 10\,\text{Hz}$, $F_2 = 12\,\text{Hz}$, and t is time in seconds. A two-second snippet of signal y_2 is shown in Figure 4.1b. These two signals was chosen to visualize the resolution of the two transforms in time and frequency respectively.



(a) One second snippet of y_1 in Eq. (4.1).

(b) Two second snippet of y_2 in Eq. (4.2).

Figure 4.1: The two tested signals.

4.2 Noise robustness

As a measurement of the robustness, the Mean Squared Error (MSE) between a transformed noisy signal and a transformed noise-free signal was calculated. Using the Frobenius norm, the definition of the MSE for two matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ is given by

$$\text{MSE} = \frac{\|\boldsymbol{A} - \boldsymbol{B}\|_F^2}{N} = \frac{\text{trace}\left((\boldsymbol{A} - \boldsymbol{B})^* (\boldsymbol{A} - \boldsymbol{B})\right)}{N} \tag{4.3}$$

where * denotes Hermitian transpose and N is the number of elements in $\boldsymbol{A}$ and $\boldsymbol{B}$ [15]. Note that the transformed signals are matrices hence the use of a matrix norm. The noise used was AWGN with different amplitudes corresponding to SNR:s in the range of $-15\,\text{dB}$ to $15\,\text{dB}$ and the signals were the same songs used in the Shazam pipeline.

To explain any possible difference in the MSE when using the HLT and the STFT, a heatmap of the difference between a transformed signal with and without noise was provided. The signals analyzed were two randomly selected songs and the noise was AWGN with an amplitude corresponding to an SNR of 0 dB. The full procedure is summarized in Figure 4.2.

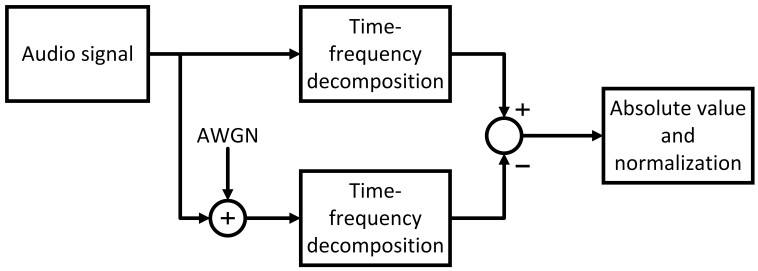


Figure 4.2: The calculation of the difference between a transformed signal with and without noise.

4.3 Reference implementation

To make sure that our implementation gives relevant results we compare the results from our implementation when using the STFT with the results obtained by Wang [4]. Wang used a dataset consisting of 10000 songs and measured the accuracy by matching snippets taken from the middle of 250 selected songs. The accuracy was measured for different levels of SNR and different lengths of the song snippet. In this project, a dataset of 1041 songs was used and each song was matched to the database using different snippet lengths and levels of AWGN. The snippets were taken in the middle of the song, just as Wang did. Each snippet was matched with the database and the accuracy for each level of SNR and snippet length was calculated. The accuracy can be seen as an estimation of the probability of identifying a random snippet with a certain length and noise level. The settings used for the implementation are presented in Table 3.1.

A comparison with Wang's implementation with 10000 songs and our implementation with 100, 550, and 1041 songs in the dataset was made to evaluate how the number of songs in the dataset impacts the accuracy estimation. All estimations were done with snippets that were 5 s long and taken from the middle of each song. Although not explicitly stated in the paper by Wang we will use the same window length $W = 2048$ and number of frequency bins $nFFT = 2048$ as the paper did. This can be deduced

from that the paper uses 10 bits of frequency information when creating the combinatorial hashes, see Section 3.1. Since the data is real valued, the choice of $nFFT = 2048$ gives 1024 unique frequency bins, see Section 2.2, which corresponds to 10 bits of frequency information. The other settings for the algorithm are seen in Table 3.1.

4.4 Comparison of the HLT and the STFT

After the benchmark performance was obtained, the accuracy when using the HLT for the same settings was collected. The average number of (t, id)-pairs in the fingerprint database for the HLT and the STFT respectively was calculated as a measurement of the size of the database. This was done for different values of t_u to analyze how the size of the fingerprint database impacts the accuracy. The other settings can be seen in Table 3.1. Snippets of length $1, 2, \ldots, 5$ s were also tested with the settings in Table 3.1 to possibly observe any difference in performance between the STFT and the HLT for short snippets.

4.5 Data preprocessing

The 1041 songs that were used in the dataset were retrieved from the royalty-free music website Epidemic Sound, www.epidemicsound.com.
Each song was picked at random but it was ensured that songs with varying beats per minute (BPM) were included to have sufficient generalization for the Shazam implementation.
Some songs were Monophonic (mono) and some were Stereophonic (stereo) meaning that a song could have one or two audio channels respectively. Since the Shazam algorithm is based on mono audio signals the stereo songs were transferred to mono audio

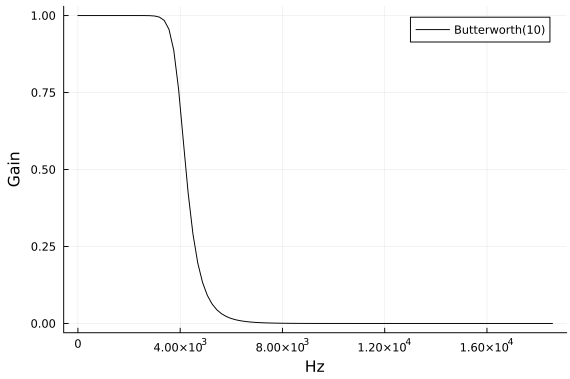


Figure 4.3: The applied low-pass filter which was used as an anti-aliasing filter for the songs with sampling frequency F_s =48 kHz.

by taking the mean of the two channels. To speed up computations and use data that is similar to what was used by Wang, the audio signals were downsampled to have a sampling rate of 8 kHz. The anti-aliasing filter was a 10-pole Butterworth filter with a cut-off frequency $\frac{F_s}{2 \cdot 6}$, see Figure 4.3. The factor $\frac{1}{6}$ comes from the fact that we decimate by 6 to get a sampling rate of 8 kHz since $F_s = 48$ kHz.

Chapter 5

Results

5.1 Time-frequency decomposition

The time-frequency decompositions of y_1 from Eq. (4.1) are shown in Figure 5.1 and 5.2 below.

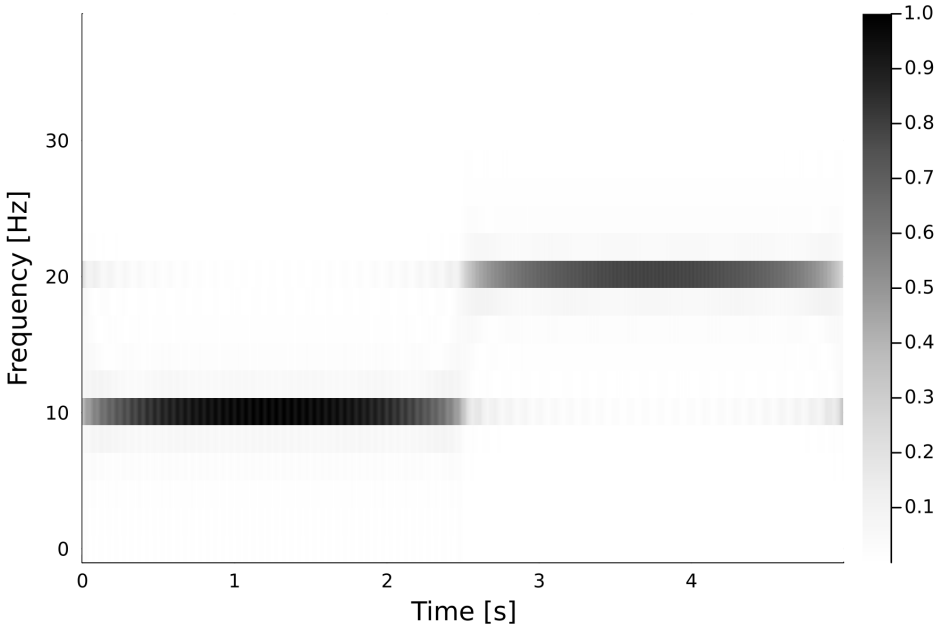
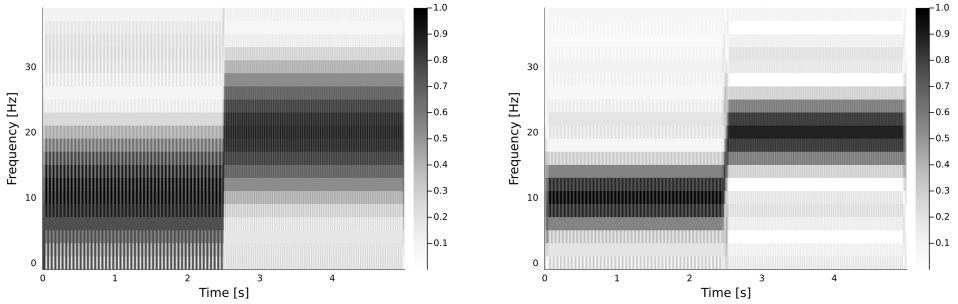


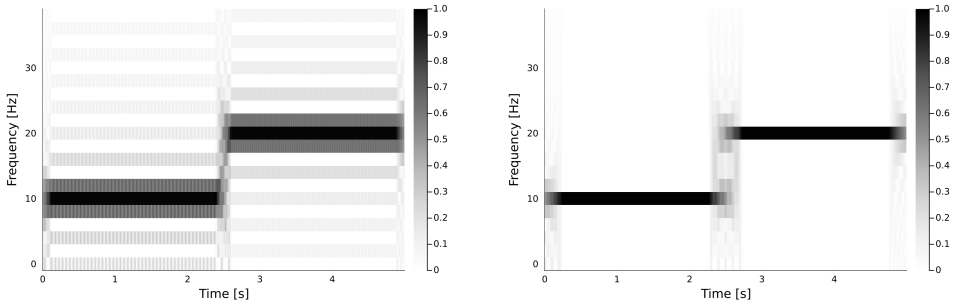
Figure 5.1: Scalogram of y_1 using the HLT.

Figure 5.2 shows the trade-off between time- and frequency resolution using the STFT. As the size of the time window increases for the STFT it becomes more difficult to see when in time y_1 moves from 10 Hz to 20 Hz, i.e. the time resolution decreases but the frequency resolution increases. This is visually represented in the spectrogram seen in Figure 5.2a, 5.2b, 5.2c and 5.2d where the time window is 32, 64, 128 and 256 samples respectively. If we compare this to Figure 5.1 we see that the HLT manages to both locate when in time the frequency changes in the signal i.e. at 2.5 s and which frequencies that are present. The reason for this is that the HLT gives a frequency decomposition for each sample in y_1 which, due to the multi-resolution, leads to improved resolution in both time and frequency.



(a) $W = 32$

(b) $W = 64$



(c) $W = 128$

(d) $W = 256$

Figure 5.2: Spectogram of y_1 using the STFT with different window lengths, W.

In Figure 5.3 and Figure 5.4 the scalogram and spectrogram of y_2 from Eq. (4.2) can be seen respectively. In Figure 5.3 we can see that the HLT manages to separate the two frequencies and also locate when in time the signal has the highest amplitude.

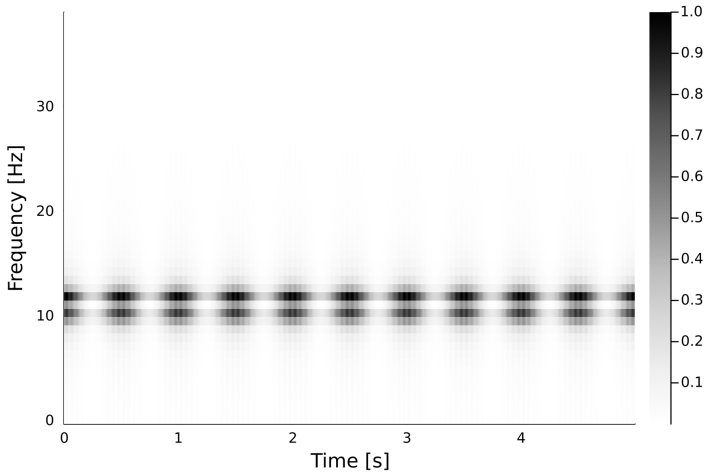
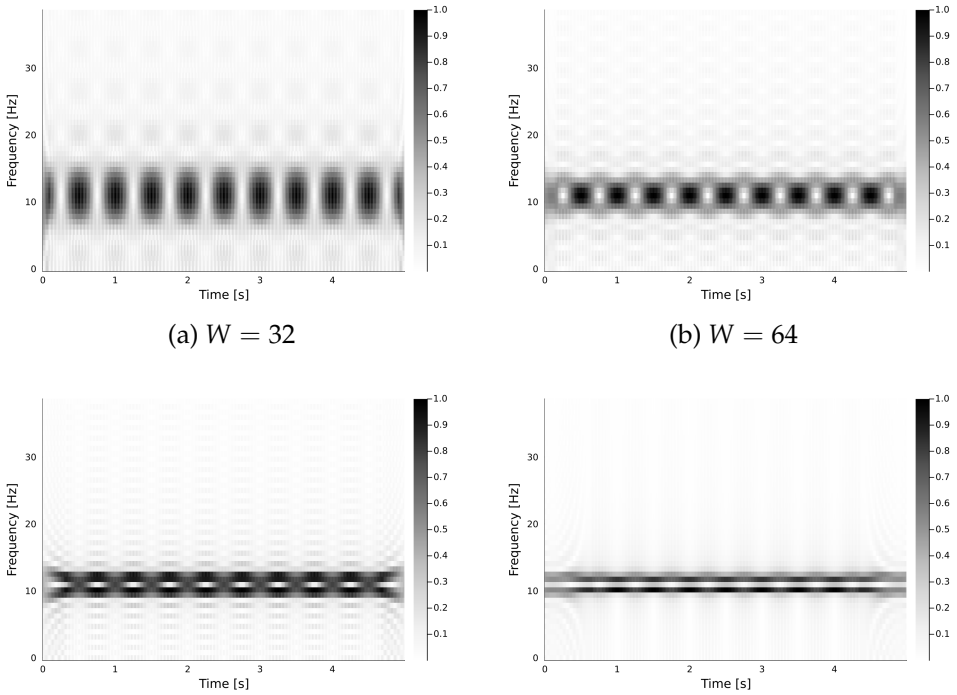


Figure 5.3: Scalogram of y_2 using the HLT.



(a) $W = 32$

(b) $W = 64$

(c) $W = 128$

(d) $W = 256$

Figure 5.4: Spectrogram of y_2 using the STFT with different window lengths W.

When analyzing the time-frequency decomposition of y_2 using the STFT it becomes evident that, just as according to theory, the frequency resolution is worse for smaller time windows. For window lengths 32 and 64, the STFT has difficulties separating the two frequencies 10 Hz and 12 Hz, as can be seen in Figure 5.4a and 5.4b respectively. For window lengths 128 and 256, the STFT manages to separate the frequencies as can be seen in Figure 5.4c and 5.4d respectively. After analyzing the time-frequency decomposition of both y_1 and y_2 it is evident that the STFT suffers from the uncertainty principle, while the HLT does not.

5.2 Robustness towards noise

The logarithm of the average MSE of the 1041 decomposed songs, using the HLT and the STFT, can be seen in Figure 5.5. The figure shows that the HLT has a lower MSE compared to the STFT for all levels of SNR that were tested.

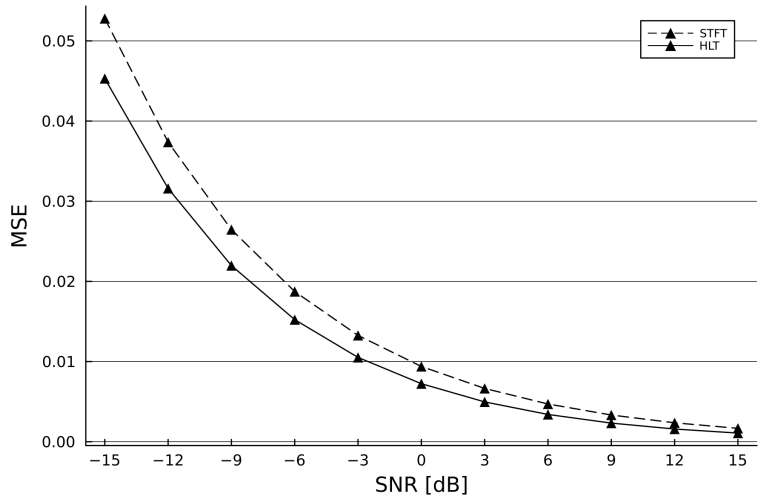
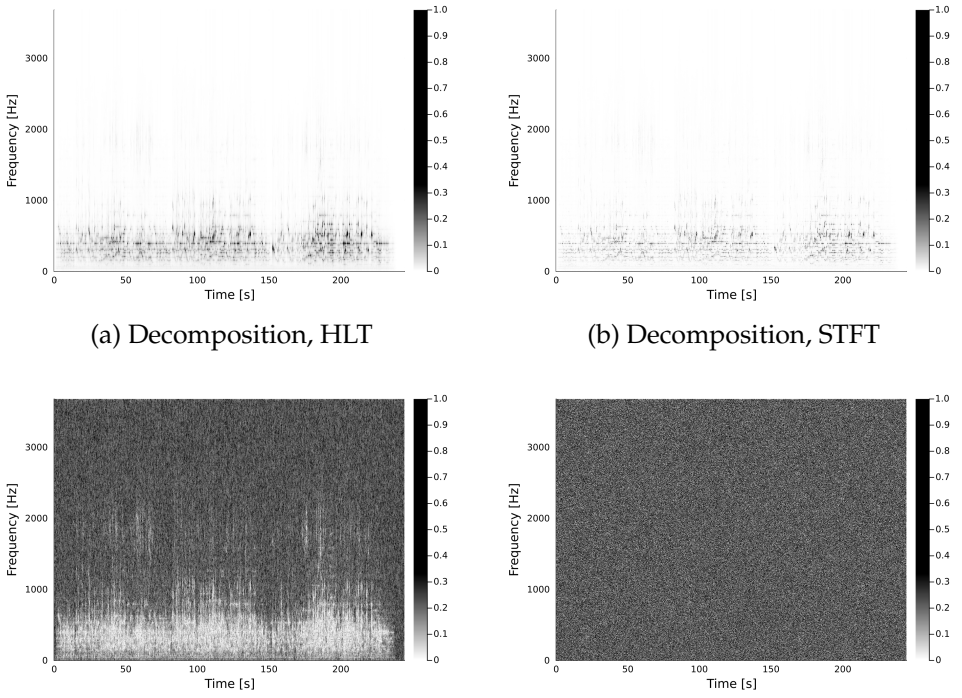


Figure 5.5: The average MSE for 1041 noisy and noise free songs using the HLT and the STFT respectively.

To understand why a difference in MSE was observed, a time-frequency decomposition of two randomly selected songs referred to as "Song 1" and "Song 2" with and without AWGN was also calculated. For Song 1 we see that by using the HLT, the smallest difference, see Figure 5.6c, is obtained at the time-frequency points with the highest amplitude seen in Figure 5.6a.

This corresponds to where we have actual non-zero frequency content in the song which is also verified by the negative Pearson correlation of -0.56. In contrast, when using the STFT, we can not see any clear structure in the difference, see Figure 5.6d, and the correlation is also estimated to be 0.



(a) Decomposition, HLT

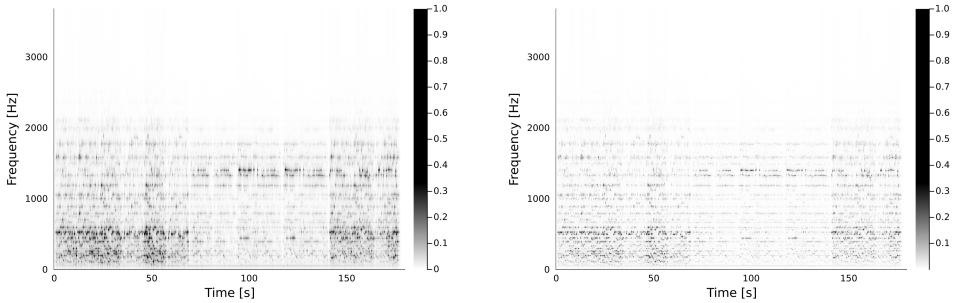
(b) Decomposition, STFT

(c) Difference, HLT

(d) Difference, STFT

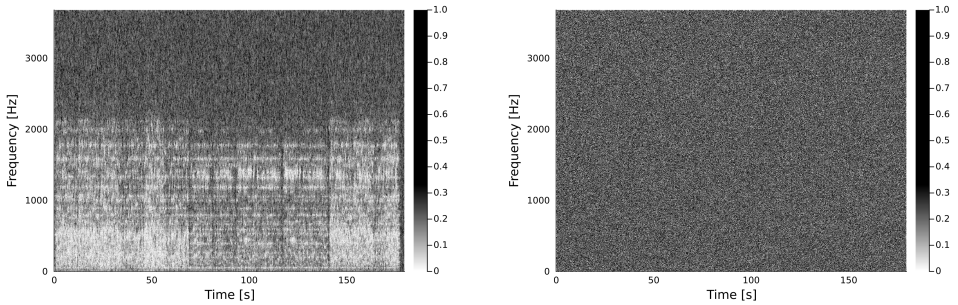
Figure 5.6: (a) and (b) show the time-frequency decomposition of Song 1 using the HLT and the STFT respectively, normalized to values between 0 and 1. (c) and (d) show the difference between the time-frequency decomposition of Song 1 with and without noise. The difference is normalized with the largest maximum value obtained from both the HLT and the STFT decomposition.

The same observation of where the smallest time-frequency difference is obtained can be made for Song 2, see Figure 5.7. When using the HLT, we see the smallest difference in Figure 5.7c where the amplitude is the highest in Figure 5.7a. This observation is verified with the Pearson correlation coefficient which is estimated to -0.6. For the STFT, Figure 5.7d shows that the difference is random and evenly distributed across the spectrum.



(a) Decomposition, HLT

(b) Decomposition, STFT



(c) Difference, HLT

(d) Difference, STFT

Figure 5.7: (a) and (b) show the time-frequency decomposition of Song 2 using the HLT and the STFT respectively, normalized to values between 0 and 1. (c) and (d) show the difference between the time-frequency decomposition of Song 2 with and without noise. The difference is normalized with the largest maximum value obtained from both the HLT and the STFT decomposition.

These results explain why we have a difference in the MSE when we compare both transforms and why the HLT might be considered to be more robust towards noise. Since the transforms can not distinguish between what is noise and what is the pure signal, the results also imply that the HLT emphasizes stronger frequency components over uniform frequency content, see Figures 5.6c and 5.7c.

5.3 Reference implementation

The purpose of comparing our implementation with a reference is to show that the results obtained with our implementation are relevant when we later compare the STFT and the HLT within the same framework. As can be seen in Figure 5.8, our implementation using the STFT has a higher accuracy for all snippet lengths compared to the reference.

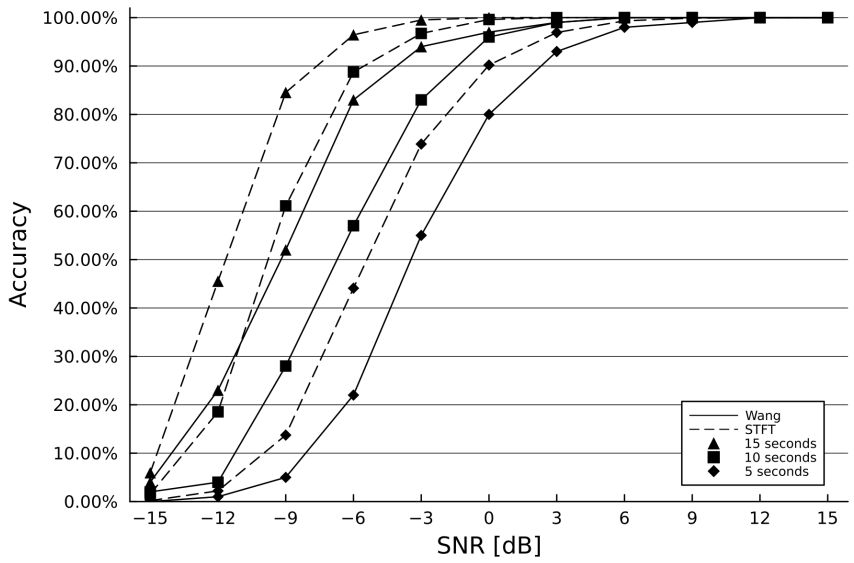


Figure 5.8: Accuracy vs SNR for the reference and our implementation using the STFT and settings according to Table 3.1.

While these results indicate that our implementation works well we must also consider the differences in estimating the accuracy between this implementation and the reference. One major difference is the size of the dataset being used. In our implementation, we used a dataset of 1041 songs while Wang used a dataset of 10000 songs which is almost 10 times bigger. As illustrated in Figure 5.9, the curves for accuracy vs SNR get shifted to the right when the size of the dataset increases. The relative difference in dataset size seems to have the most impact on accuracy vs SNR rather than the difference in the number of songs. Going from 100 to 550 songs, which implies an increase with a factor of 5.5, makes for a larger shift of the curves compared to going from 550 to 1041 songs, a relative size increase of around 1.89.

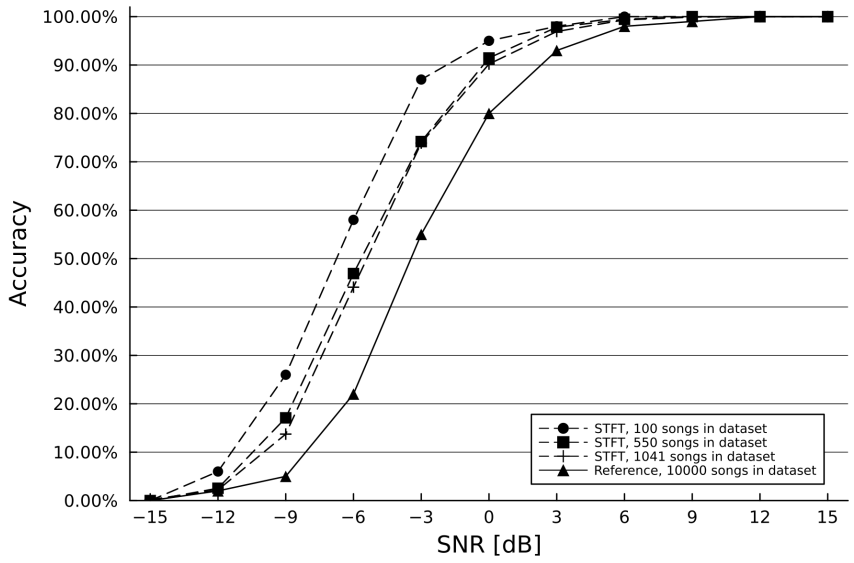


Figure 5.9: Accuracy vs SNR for the reference and our implementation using the STFT with different numbers of songs in the dataset. We used the settings according to Table 3.1. The accuracy was estimated using 5 s snippets taken from the middle of each song.

5.4 Comparison of the HLT and the STFT

This section contains the results obtained when comparing the HLT and the STFT using our Shazam implementation. Figure 5.10 depicts the accuracy vs SNR for the Shazam implementation for the HLT and the STFT with the settings in Table 3.1. Overall both transforms gave quite similar accuracy vs SNR. In Figure 5.10 we can see that the shape of the curves for the HLT and the STFT are similar and that this holds for different lengths of the snippets. However, the HLT has a higher accuracy overall and this becomes particularly evident for lower SNR levels.

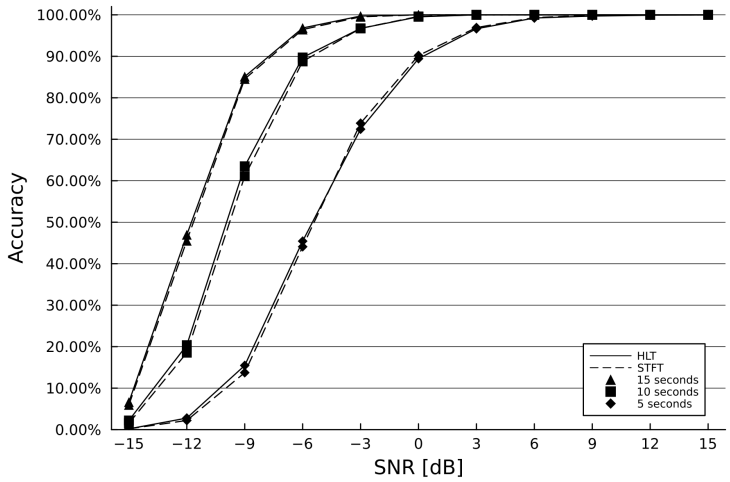


Figure 5.10: Accuracy vs SNR for the Shazam implementation using the STFT and the HLT.

From Figure 5.11 we can see that an increase in t_u generally corresponds to an increased accuracy. The curves shown in the aforementioned figure also indicate a strong correlation since the Pearson correlation coefficient is above 0.9 for almost all curves, see Table 5.1.

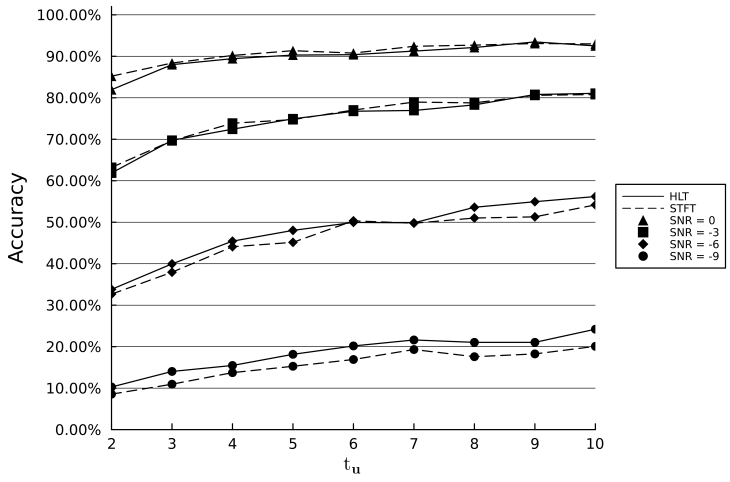


Figure 5.11: Accuracy vs t_u for the Shazam implementation using the STFT and the HLT.

Table 5.1: The Pearson correlation coefficient, r, between accuracy and t_u for different noise levels.

	SNR	**0**	**-3**	**-6**	**-9**
r	**HLT**	0.866	0.931	0.954	0.948
	STFT	0.901	0.926	0.934	0.934

Another observation is that by comparing the average number of (t, id)-pairs for the STFT and the HLT respectively, the STFT generates on average more pairs, see Table 5.2. This relation remains the same for all values of t_u which is reasonable since the values are an average of the results from all songs. While the same settings lead to similar accuracy with both decomposition methods, the STFT generates 1.23 times more (t, id)-pairs, which would suggest that the HLT is more efficient considering memory usage.

Table 5.2: Average number of (t, id)-pairs for different values of t_u.

	(t, id)**-pairs**		
t_u	**STFT**	**HLT**	**STFT rel. to HLT**
2	13326	10874	1.23
3	26927	21842	1.23
4	40489	32840	1.23
5	54071	43869	1.23
6	67671	54916	1.23
7	81275	65970	1.23
8	94892	77033	1.23
9	108515	88103	1.23
10	122142	99178	1.23

Figure 5.12 depicts the accuracy vs SNR for the STFT and the HLT when $t_u = 5$ and $t_u = 6$ respectively, which is an example of when the number of (t, id)-pairs is similar for both transforms. In this example, the STFT generated 1.5% less (t, id)-pairs than the HLT. The accuracy vs SNR is quite similar but the HLT has higher accuracy for lower levels of SNR.

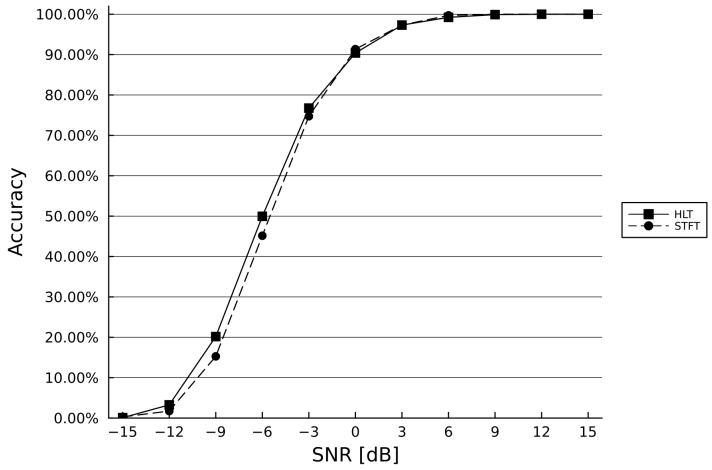


Figure 5.12: Accuracy vs SNR for the Shazam implementation using the STFT and the HLT. The STFT with $t_u = 5$ and the HLT with $t_u = 6$.

When reducing the snippet length there is no major change in the accuracy vs SNR for the HLT relative to the STFT, see Figure 5.13. As expected the accuracy is decreased as the snippet length gets shorter which we can see is also the case for an arbitrary SNR. This holds for both the STFT and the HLT but we can see that the latter has better accuracy for all snippet lengths for the lower SNR levels of -6 dB and -9 dB, see Figure 5.13.

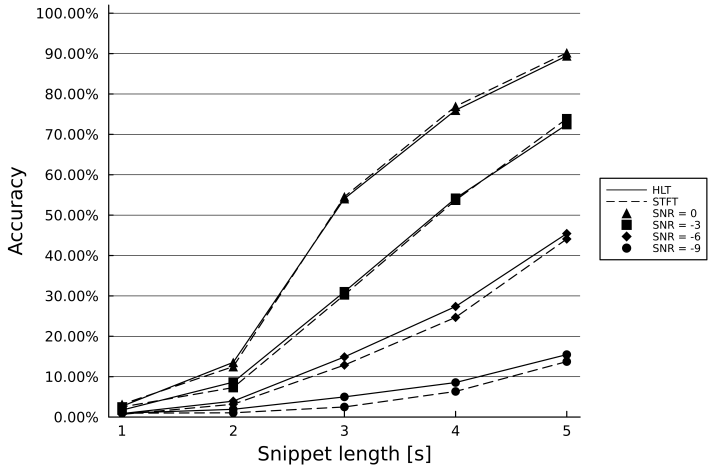


Figure 5.13: Accuracy vs snippet length for the Shazam implementation using the STFT and the HLT.

Chapter 6

Discussion

6.1 Shazam implementation

The comparison between the results of the Shazam implementation in this thesis and the reference showed that the former gave better results overall. However, from Figure 5.9 we saw that the number of songs in the dataset did impact the accuracy. If we assume that the dataset of 1041 songs used in this thesis is somewhat representative of all songs then we should not expect a much worse result going from 1041 to a dataset of 10000 songs. This is because we estimate a joint relative threshold corresponding to a false positive rate of $\alpha = 0.1\%$ and then compute a threshold score for each song in the dataset. As the number of songs increases we get a more reliable estimate of the thresholds which can explain why the difference in accuracy for 550 and 1041 songs in the dataset is very small compared to 100 songs, see Figure 5.9.

It is also important to consider what songs are included in the dataset. A dataset of very similar songs, e.g. one genre, could get a lower accuracy compared with a dataset with a lot of different genres since in the first case it is more difficult to create unique fingerprints compared to the second case. We used 1041 songs picked at random, while our reference does not disclose which songs are included in the dataset, just that it contains 10000 popular songs at the time. Another important factor is the amount of songs being matched with the dataset to create the accuracy estimation. When producing our results we matched all songs in the dataset while the reference only used a subset of 250 songs from the dataset of 10000 songs. Given that Wang's article is a conference paper, the selection of songs is undisclosed, potentially affecting the efficacy of the results.

6.2 Database size

The average number of (t, id)-pairs can be seen as a measurement of how much data is extracted from an audio signal. This means that the more (t, id)-pairs that are generated for an audio signal, the larger the fingerprint is, and thus the likelihood that we get a higher number of correct matches will increase. On the other hand, a larger database does not necessarily mean that the likelihood of correctly identifying a song will increase since we also likely will have a set of less unique hashes for each fingerprint. The loss of uniqueness for the set of hashes of each song will increase the incorrect matches, which can lead to wrongly predicting a song. This effect probably becomes more tangible for larger datasets since there is a finite number of combinatorial hashes that can be created and, if the subset of hashes for each song is large then the number of incorrect matches will increase. From the results in 5.11 we see no decrease in accuracy for larger values of t_u, instead, the accuracy is strongly correlated with t_u. However, it is likely that the accuracy curve would flatten if even larger values of t_u would have been tested.

6.3 Reducing snippet length

As was seen in Figure 5.13 the accuracy decreased when the snippet length decreased which is likely an effect of the fingerprint becoming smaller. The fingerprint of a short snippet has a greater probability of being a subset of other songs than the correct one which is equivalent to a higher probability of misidentification. A solution to this would be to reduce the window length W which would lead to the extraction of more spectral peaks that are used to create more hashes, thus increasing the size of the fingerprint. However, this would also require the same window length to be used when creating the fingerprint database which would decrease the compression rate. If W is selected to be very small there is also a possibility that the frequency content of two neighbouring time windows is very similar or identical which would not lead to a more unique fingerprint thus contradicting the reason to reduce W. The reduction of W was tested but did not yield any improved results, thus this was not included in the report.

6.4 Conclusions

Overall the HLT performed slightly better than the STFT in the conducted experiments; with better visual time-frequency decomposition and slightly higher accuracy in the Shazam pipeline. Based on the visual time-frequency decomposition, it can be concluded that the HLT has a better time-frequency resolution compared to the STFT as it does not suffer from the uncertainty principle. When implementing the Shazam pipeline it was noticed that the HLT generated fewer (t, id)-pairs and was more robust towards noise than the STFT.

In the experiments, only AWGN was used but when using a song recognition application in a real setting the noise might have a different structure, e.g. people talking in the background or traffic noise. It could thus be interesting to further test the performance with other types of noise and distortions.

While the HLT performed better in terms of resolution and accuracy it is also important to consider some potential drawbacks. The HLT has a significantly higher time complexity, which has to be considered in the choice of transform. If the computational time is the limiting factor, then the STFT is preferable over the HLT. However, if the complexity of the problem is high then the HLT is a valid option. In the case of the Shazam pipeline, the time it takes to create the database is not as consequential as the time it takes to match the snippet to the database. Thus the higher time complexity of the HLT can be argued to not be as unfavorable since the snippets are not very long and the other parts of the Shazam algorithm will take a relatively long time.

To get a time-frequency decomposition and not only a frequency decomposition of the entire audio signal, the songs were divided into T overlapping windows of length W as is standard for the STFT. However, since the HLT is a continuous transform this method discretizes the HLT. The reason for the discretization was to be able to extract the most significant spectral peaks using the same method in order to produce comparable results by using both transforms. However, for further work, it would be interesting to define the wavelets over the entire duration of the song and utilize the multi-resolution properties. With this implementation, one could extract peaks by considering both time and frequency instead of extracting the most significant peaks within a time window, as was done in this study.

It is worth mentioning that the Shazam pipeline implemented in this project is representative of what was used 20 years ago. Today, one would likely use a machine learning-based approach to solve the problem. How-

ever, the purpose of the pipeline is to compare the two transforms rather than implementing a state-of-the-art song identification pipeline.

To further study the capabilities of the HLT, it can be worth exploring other domains within audio fingerprinting that present greater complexity. For instance, voice recognition tasks typically involve shorter snippet lengths which allows for higher sampling frequencies and thus more samples for a short snippet.

Bibliography

[1] C. Philip and S. G. (2020) Audio fingerprinting: Understanding the concept, process, application. Accessed: 2024-01-30. [Online]. Available: https://www.pathpartnertech.com/audio-fingerprinting-understanding-the-concept-process-application

[2] P. Cano, E. Batle, T. Kalker, and J. Haitsma, "A review of algorithms for audio fingerprinting," in *2002 IEEE Workshop on Multimedia Signal Processing.*, 2002, pp. 169–173.

[3] K. Kesgin and H. Jörntell, "Singular superlet transform achieves markedly improved time-frequency super-resolution for separating complex neural signals," *bioRxiv*, 2023. [Online]. Available: https://www.biorxiv.org/content/early/2023/02/28/2023.02.27.530211

[4] A. Wang, "An industrial strength audio search algorithm." 01 2003.

[5] S. Haskel and D. Sygoda, *Biology: A Contemporary Approach*. New York: Amsco School Publications, Inc., 1996.

[6] S. S. of Music. (2023) What are high pitch instruments? a complete guide. Accessed on: 2024-02-15. [Online]. Available: https://sloanschoolofmusic.com/what-are-high-pitch-instruments/

[7] M. Müller and F. Zalkow. (2024) Stft - window functions. Accessed on: 2024-03-11. [Online]. Available: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C2/C2_STFT-Window.html

[8] N. Instruments. (2023) Zero padding in fft analysis. Accessed on: 2024-02-16. [Online]. Available: https://www.ni.com/docs/en-US/bundle/labwindows-cvi/page/advancedanalysisconcepts/lvac_zero_padding.html

[9] M. Sandsten, "Time-frequency analysis of time-varying signals and non-stationary processes," 2020, available at https://www.maths.lu.se/fileadmin/maths/personal_staff/mariasandsten/TFkompver4.pdf, accessed pages 10-12.

[10] J. O. S. III, *Spectral Audio Signal Processing*. Center for Computer Research in Music and Acoustics (CCRMA), Stanford University, 2011. [Online]. Available: http://ccrma.stanford.edu/~jos/sasp/

[11] M. Strauss. (2021, January 29) How shazam works - an explanation in python. [Online]. Available: https://michaelstrauss.dev/shazam-in-python

[12] V. Khatri, L. Dillingham, and Z. Chen, "Song recognition using audio fingerprinting," *Dept. of Electrical and Computer Engineering, University of Rochester*, 2019. [Online]. Available: https://hajim.rochester.edu/ece/sites/zduan/teaching/ece472/projects/2019/AudioFingerprinting.pdf

[13] Christophe. (2015) How does shazam work? Updated: August 6, 2015; Posted: May 23, 2015. [Online]. Available: https://www.academia.edu/33464339/How_does_Shazam_work_Coding_Geek

[14] L. Mottola. (2020) Frequency table. Accessed on: 2024-02-13. [Online]. Available: https://www.liutaiomottola.com/formulae/freqtab.htm

[15] Z. Wang and A. C. Bovik, "Mean squared error: Love it or leave it? a new look at signal fidelity measures," *IEEE SIGNAL PROCESSING MAGAZINE, vol. 26, no. 1, pp. 98-117*, Jan. 2009.