Pontifícia Universidade Católica
do Rio de Janeiro

Matheus Adler Soares Pinto

A Method for Real-Time Generation of Videoke from Video Streaming

Dissertação de Mestrado

Dissertation presented to the Programa de Pós–graduação em Informática, do Departamento de Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática.

Advisor: Prof. Sérgio Colcher

Rio de Janeiro
September 2023

Pontifícia Universidade Católica
do Rio de Janeiro

Matheus Adler Soares Pinto

A Method for Real-Time Generation of Videoke from Video Streaming

Dissertation presented to the Programa de Pós–graduação em Informática of PUC-Rio in partial fulfillment of the requirements for the degree of Mestre em Informática. Approved by the Examination Committee:

Prof. Sérgio Colcher
Advisor
Departamento de Informática – PUC-Rio

Prof. Julio Cesar Duarte
IME

Prof. Antonio José Grandson Busson
BTG Pactual

Rio de Janeiro, September 28th, 2023

Matheus Adler Soares Pinto

Graduated in 2021 from the Universidade Estadual do Maranhão (UEMA) in Computer Engineering.

Bibliographic data

Pinto, Matheus Adler Soares

A Method for Real-Time Generation of Videoke from Video Streaming / Matheus Adler Soares Pinto; advisor: Sérgio Colcher. – 2023.

56 f: il. color. ; 30 cm

Dissertação (mestrado) - Pontifícia Universidade Católica do Rio de Janeiro, Departamento de Informática, 2023.

Inclui bibliografia

1. Informática – Dissertação. 2. Sistemas Multimídia. 3. Processamento em Tempo Real. 4. Arquitetura de Software. 5. Inteligência Artificial. I. Colcher, Sérgio. II. Pontifícia Universidade Católica do Rio de Janeiro. Departamento de Informática. III. Título.

CDD: 004

To my parents, my grandparents and my life partner for their unconditional support and encouragement.

Acknowledgments

Firstly, I would like to express my gratitude to God, the creator of all things and the source of blessings in my life and in the life of my family.

I am immensely grateful to my advisor, Prof. Sérgio Colcher, for believing in me and for his fundamental support in the development of this work. I would also like to express my deep gratitude to Busson, for the exchange of experiences, advice, and opportunities. Without your guidance and help, this dissertation would not have been possible.

I thank the members of the examining board who kindly accepted the invitation to collaborate with their expertise in evaluating this study. I also thank the professors, staff, and colleagues at PUC who shared their time and knowledge with me.

The Telemídia family deserves my sincere gratitude for their hospitality and for exchanging experiences over the years in my master's program.

I would like to thank my girlfriend and life partner, Giulia, for her encouragement, support, patience, and understanding.

I deeply thank my parents, Jony Arrais and Mônica Soares, who have always been my greatest motivation, doing everything in their power to help me get to this point. To my brother, Gabriel Victor, for all his concern seeing me spend hours in front of the computer. To my brother, Jony Júnior, for his concern and for the exchange of academic experiences he provided me with at this final moment.

Finally, I am extremely grateful to my grandparents, Hermano José and Maria de Fátima, who have always provided me with fundamental support, whether emotional or financial.

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Finance Code 001

Abstract

Pinto, Matheus Adler Soares; Colcher, Sérgio (Advisor). **A Method for Real-Time Generation of Videoke from Video Streaming**. Rio de Janeiro, 2023. 56p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Traditional karaoke systems typically use pre-edited videos, which limits the creation of videoke experiences. In this dissertation, we propose a new method for generating videoke in real-time from video streaming sources, called the videoke Generator. This method combines video and audio processing techniques to automatically generate videoke and is designed to perform processing in real-time or near real-time. The main objectives of this study are to formulate a methodology to process videos in continuous flow and to generate videoke in real-time while maintaining essential features such as the suppression of the main voices of the music and the automatic generation of subtitles highlighting words. The results obtained represent a significant contribution to the field of real-time multimedia generation. The method was implemented in a client/server architecture for testing. These contributions represent a step forward in the field of entertainment and multimedia systems as they introduce a new methodology for the creation of videoke experiences. To our knowledge, this is the first work that addresses the development of a real-time videoke generator that performs automatic synchronization and highlighting at the word level, based on a literature review.

Keywords

Multimedia Systems; Real-Time Processing; Software Architecture; Artificial Intelligence.

Resumo

Pinto, Matheus Adler Soares; Colcher, Sérgio. **Um Método para Geração em Tempo Real de Videokê a partir de Streaming de Vídeo**. Rio de Janeiro, 2023. 56p. Dissertação de Mestrado – Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro.

Sistemas tradicionais de karaokê geralmente utilizam vídeos pré-editados, o que limita a criação de experiências de videokê. Nesta dissertação, propomos um novo método para a geração de videokê em tempo real a partir de fontes de streaming de vídeo, chamado Gerador de Videokê. Este método combina técnicas de processamento de vídeo e áudio para gerar automaticamente videokê e é projetado para realizar o processamento em tempo real ou próximo a isso. Os principais objetivos deste estudo são formular uma metodologia para processar vídeos em fluxo contínuo e gerar videokê em tempo real, mantendo características essenciais como a supressão das vozes principais da música e a geração automática de legendas destacando palavras. Os resultados obtidos representam uma contribuição significativa para o campo da geração de multimídia em tempo real. O método foi implementado em uma arquitetura cliente/servidor para testes. Essas contribuições representam um avanço no campo dos sistemas de entretenimento e multimídia, pois introduzem uma nova metodologia para a criação de experiências de videokê. Até onde sabemos, este é o primeiro trabalho que aborda o desenvolvimento de um gerador de videokê em tempo real que realiza sincronização automática e destaque a nível de palavras, com base em uma revisão da literatura.

Palavras-chave

Sistemas Multimídia; Processamento em Tempo Real; Arquitetura de Software; Inteligência Artificial.

Table of contents

List of figures

List of tables

List of Abreviations

ASR – Automatic Speech Recognition
CNNs – Convolutional Neural Networks
CPU – Central Processing Unit
Demucs – Deep Extractor of Music Sources
DNNs – Deep Neural Networks
FPS – Frames per second
HLS – HTTP Live Streaming
HTML – HyperText Markup Language
HTTP – Hypertext Transfer Protocol
ML – Machine Learning
MM – International Multimedia Conference
NLP – Natural Language Processing
NMF – Non-Negative Matrix Factorization
PCM – Pulse Code Modulation
PTMs – Pre-trained Models
RNNs – Recurrent Neural Networks
SRT – SubRip

1
Introduction

Karaoke, which originated in Japan in 1971, is a form of musical interaction in which participants perform musical compositions without the presence of the original vocals (HOSOKAWA; MITSUI, 2005). Its consolidation as a globally widespread form of entertainment continues to this day and is one of the most important leisure activities both in Japan and worldwide. In 2011, an experiment proved its popularity, ranking it as the 7th most popular practice in Japan with 38.4 million participants, directly competing with other entertainment activities such as watching movies, playing video games, and listening to music (TACHIBANA et al., 2016).

A variation of this concept, which we will refer to as videoke, includes additional elements such as video and synchronized subtitles alongside the corresponding audio, eliminating the original vocal track. The experience offered by videoke is even more enriching as it allows participants to not only express the music vocally but also interact with the content visually, which has significantly boosted its growing popularity.

However, it is essential to point out that the creation of this content, whether in videoke or karaoke format, is still primarily the responsibility of specialists such as sound engineers. Generating these tracks in high quality for musical compositions requires mastery of specialized media manipulation software, which is a challenge that often exceeds the capabilities of the public (MEHENDALE et al., 2021). These limitations lead to significant constraints, such as the availability and variety of easily accessible karaoke and videoke tracks.

1.1
Motivation

This dissertation arose from the motivation to overcome the limitations inherent in traditional videoke systems, as described in the previous section. In an era characterized by the presence of artificial intelligence in various sectors of society, including the entertainment industry, we have observed a growing interest in the study and development of techniques related to automatic karaoke generation. This is evidenced by the research conducted by Patel (PATEL et al., 2022). This study analyzes recent trends and advances in this area, highlighting notable examples such as Spleeter, Hybrid Demucs, D3Net, Open-Unmix, Sams-Net, and others.

The formulation of a methodology that significantly contributes to the advancement and exploration of this emerging topic forms the basis for the conceptualization of this dissertation. In this context, we present an approach called *Videoke Generator*. This method enables the real-time generation of videoke from video streaming sources. By filling this research gap, the dissertation aims to contribute to the current landscape of studies related to artificial intelligence used to create experiences in the field of musical entertainment.

1.2 Objective

The main focus of this research is to introduce an approach for real-time generation of videoke from video streaming sources. To achieve this goal, our proposed method involves the implementation of a methodology consisting of an interrelated network of modules, each of which performs a specific function in the generation of videoke in real-time.

In addition to the main objective outlined, this study aims to make additional contributions. In this context, three specific objectives should be highlighted: the implementation of an automatic subtitle synchronization system with word-level highlighting to improve the user experience; the introduction of a granularization mechanism in the videoke generation process that enables real-time or near-real-time application; and the validation of the applicability of the proposed methodology through the development of a functional prototype that will be evaluated.

Therefore, this work aims to advance studies on the topic of videoke generation by introducing the *Videoke Generator*. This contributes to the development of the field by addressing specific aspects such as the automatic synchronization of subtitles and the granularization of the process. The prototype resulting from the method provides a tangible tool for practical evaluation that will serve as a guide for future work, thus contributing to the body of knowledge in the field of automated real-time videoke generation.

1.3 Justification

The relevance of the research proposed in this dissertation is grounded in two different areas: theoretical and practical.

From a theoretical point of view, the study of real-time videoke generation from video streaming sources represents an advance in the fields of media processing and artificial intelligence. This research fills a gap in the literature and is the first to specifically address this problem. It contributes to the under-

standing of the integration of algorithms and techniques aimed at extracting, manipulating, and synchronizing different modalities of media, such as audio, video, and subtitles. In addition, the proposal covers the area of automatic generation of multimedia content in real time, an area with potential applications in various fields such as entertainment, education, and communication.

On the practical side, the implementation of the proposed *Videoke Generator* will have a tangible impact. The ability to generate videoke in real-time from streaming video sources has the potential to positively impact the way people interact with musical and audiovisual content. This development can provide opportunities for enhanced experiences and also serve as a starting point for future research in the entertainment industry. Furthermore, automating this process has the potential to reduce the reliance on audio and media experts to create videokes, democratizing the technology and making it accessible to a wider audience. This approach could drive the production of musical entertainment content and enable more people to create and share videoke in a simplified way.

Therefore, the research in question not only helps to advance theoretical knowledge in the field of media processing and artificial intelligence but also contributes to the generation of videoke content, creating opportunities for future work and improvements in this field of study.

1.4 Methodology

This Section describes the methodology used to conduct the proposed research, covering all phases from problem definition to the evaluation of results. The methodology was developed to ensure the generalizability and reproducibility of this study.

1.4.1 Literature Review

In the first phase, the literature on the generation of real-time videoke from video streaming sources was reviewed. The aim of this review was to capture the current state of knowledge and identify previous contributions, approaches used, gaps, and unresolved issues. Various sources of information were consulted, including academic databases, scientific journals, professional conferences, and specialized literature. The review was conducted systematically, using appropriate keywords and selection criteria.

The analysis of the sources enabled the development of a knowledge base on the topic and provided context for the research. This extensive literature

review provided insights that guided the methodology and approach of the study.

1.4.2 Method Definition

The *Videoke Generator* method represents the proposed solution in this research. Based on the knowledge gained from the literature review, a framework was designed to be followed in all stages of the automatic real-time videoke generation process. This methodology includes audio and video processing and manipulation, automatic subtitle synchronization, videoke generation, and subsequent evaluation of the results.

This phase is important to create a solid methodological framework that will guide all subsequent steps of the study. The methodology is designed to ensure a systematic approach at all phases of the process, aiming for efficiency and reliability of results.

1.4.3 Experimentation and Validation

In this phase, practical experiments were carried out according to the proposed methodology. Different video media sources were selected to simulate the transmission for experiments with the developed system. These media were categorized into groups, each representing a particular style of music. The categorization was based on the hypothesis that different music styles can influence the process of videoke generation in different ways, taking into account characteristics such as rhythm, melody, dynamics, and instrumentation.

To test this hypothesis, an experimental approach was chosen in which each set of video media representing a musical style was used to generate videoke with the system. In this way, it was possible to verify whether the system exhibited significant differences in performance based on the music style of the video media sources used in the creation of the videoke.

1.4.4 Results Evaluation

The results achieved, such as synchronization, subtitle quality, and harmonization, were then analyzed and compared between the groups. The analysis also included a subjective evaluation of satisfaction with the generated videoke in different musical styles, based on predefined criteria. In addition, the processing time and the effectiveness of the state-of-the-art models used as tools in the system were evaluated.

1.5 Text Structure

This document follows a structure to present the essential elements of the research. Chapter 2 introduces basic concepts for a complete understanding of the proposed method. Chapter 3 reviews and discusses research related to the problem at hand. Chapter 4 explains the proposal in detail. Chapter 5 presents the experiments and results. Finally, Chapter 6 concludes the research and outlines possible directions for future work.

2
Theoretical Reference

This chapter is dedicated to introducing concepts that underpin the understanding of this study. By establishing a solid foundation of theoretical knowledge, we aim to provide readers with the necessary tools to fully contextualize and understand the real-time videoke generation method proposed in this study. In this context, we will cover important concepts related to karaoke, audio and video processing, music source separation, speech recognition, subtitle synchronization, embedding elements in videos, video streaming, and the HTTP Live Streaming (HLS) protocol. Each of these concepts is essential to the development and understanding of the method and helps to provide a solid foundation that supports all the work presented in this dissertation. Therefore, this chapter is fundamental for the reader to understand and evaluate the contribution of this study.

2.1
Karaoke: A Form of Musical Entertainment

Karaoke is a popular form of musical entertainment that gained prominence in the early 2000s (ZHOU; TAROCCO, 2013). In this Section, we will examine karaoke as a significant cultural manifestation and discuss its origin, evolution, social and cultural influence, and role in today's society.

Karaoke has its roots in Japan, where it was invented in the 1970s by Daisuke Inoue, a Japanese musician. The word karaoke is a combination of the Japanese words *kara* (meaning empty) and *oke* (meaning orchestra). Inoue developed a system that allowed people to sing popular songs accompanied only by instrumental recordings, so that everyone could feel like the artist of the song. This concept quickly spread in Japan and later all over the world. Over time, karaoke evolved to include a variety of songs in different languages and musical genres. Karaoke machines were equipped with video screens, animated lyrics, and scoring systems to evaluate the singers' performances. Karaoke bars and special venues where people could meet to sing (LUM, 2012).

Raymond Williams (WILLIAMS, 2020) says that a culture should be understood as a specific way of life that expresses meanings and values not only through art and learning, but also through institutions and everyday behaviors. Based on this definition, cultural analysis involves the study of the implicit and explicit meanings and values that permeate a particular way of life and culture. The study of a particular cultural system therefore involves

examining the symbiotic interaction between the human, material, symbolic, and institutional elements of a society that express that particular way of life. If we apply this approach to the practice of karaoke singing, we can recognize it as a cultural manifestation in itself. Karaoke represents a process of interaction and human practices that shapes certain values, meanings, and social realities as an integral part of a specific culture—a unique way of life. It plays an important role in promoting social interaction and communication between people, and in many cultures, karaoke is a common social activity.

Karaoke also has a significant impact on people's psychological well-being. Singing is a form of emotional and artistic expression that releases endorphins and lowers cortisol levels. This reduces stress and contributes to a better quality of life, according to a study by scientists from the Department of Experimental Psychology at the University of Oxford (DUNBAR et al., 2012). Participating in a karaoke session can provide a sense of achievement and personal satisfaction, even for those who are not professional singers. In addition, karaoke offers the opportunity to overcome shyness and social anxiety, which are very common in today's society (BUSS et al., 2022).

Currently, karaoke is a popular form of entertainment worldwide. With the advancement of technology, online karaoke apps and programs have become more prevalent, allowing people to sing and share their performances with others. This shows how karaoke has adapted to technological change and remains relevant in today's society. This provides scope for research into artificial intelligence systems for karaoke generation.

2.2 Audio and Video Processing

In this Section, we will look at audio and video processing, highlighting key definitions and algorithms related to music source separation, speech recognition, subtitle synchronization, and embedding elements in videos. These topics are crucial for understanding the real-time videoke generation method proposed in this study.

2.2.1 Music Source Separation

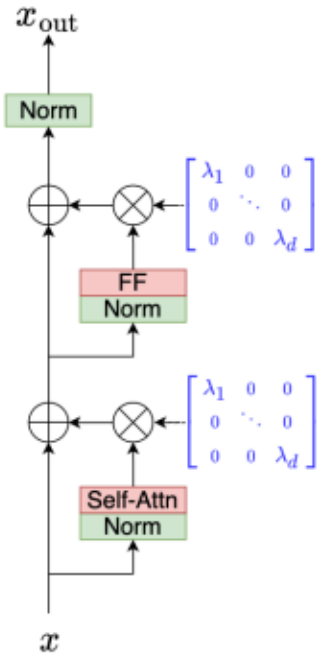
Music source separation is a technique in audio processing that plays a crucial role in real-time videoke generation. It refers to the ability to decompose a music recording into its individual components, such as vocals, instruments, and other sound sources. This allows the creation of karaoke tracks, for example, where the artists' original voices are isolated from the

musical accompaniment (CANO et al., 2018). A well-known model in this area is Demucs (Deep Extractor of Music Sources), which is used as a tool in this study.

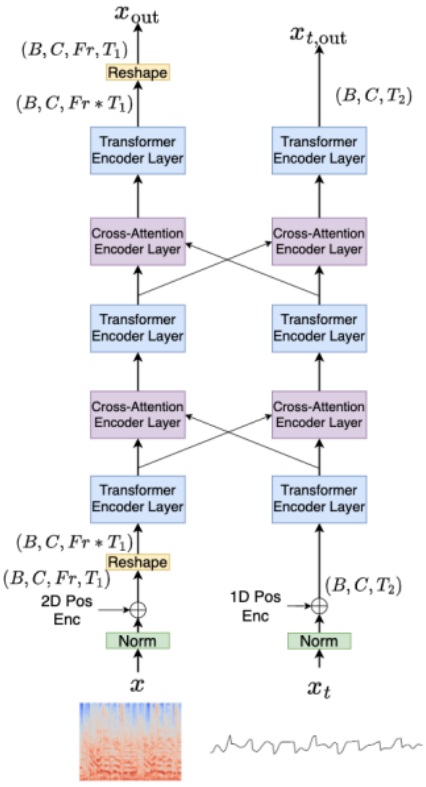
Demucs (ROUARD; MASSA; DÉFOSSEZ, 2023) is a state-of-the-art solution in the field of music source separation and is known for its ability to isolate different musical elements in a recording. This includes, for example, the ability to separate drums, bass, and vocals from the rest of the musical accompaniment. Its effectiveness is based on a convolutional U-net architecture inspired by the Wave U-net and optimized for the separation of music sources.

In this work, we have chosen to use the latest version of Demucs, known as Demucs v4. It introduces the Hybrid Transformer Demucs, a hybrid version that combines spectrogram processing techniques with waveform separation using transformer technology. This hybrid approach further enhances the model's ability to separate complex music sources. The Hybrid Transformer Demucs builds on the Hybrid Demucs with internal layers that have been replaced by a cross-domain transformer encoder.

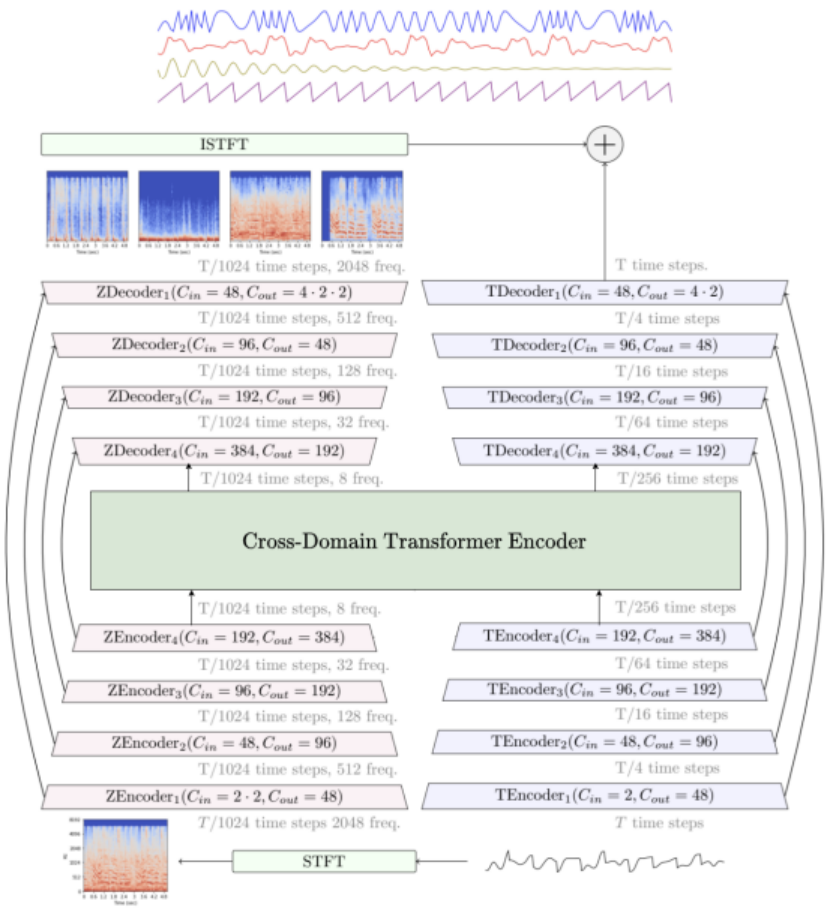
The Transformer Encoder of the Hybrid Transformer Demucs is equipped with self-attention mechanisms that allow the model to analyze and understand the relationships between different parts of the music more deeply. This results in high performance and makes the model state-of-the-art in music source separation.



(a) Transformer Encoder Layer



(b) Cross-domain Transformer Encoder of depth 5



(c) Hybrid Transformer Demucs

Figure 2.1: Details of the Hybrid Transformer Demucs architecture (ROUARD; MASSA; DÉFOSSEZ, 2023).

The architecture of the Hybrid Transformer Demucs consists of three main elements, which are shown in Figure 2.1. In subfigure 2.1a, we highlight the presence of the transformer encoder layer, characterized by self-attention mechanisms and layer scaling, which are essential for information analysis and processing. In Subfigure 2.1b we have the Cross-domain Transformer Encoder, designed to handle both spectral and temporal signals simultaneously. This module contains layers from both the transformer encoder and the cross-attention encoder, providing a comprehensive approach for the representation of heterogeneous data. Finally, in the subfigure 2.1c, the figure shows an overview of the entire architecture in which the dual U-Net encoder/decoder structure is integrated. This structure is designed to optimize feature extraction and reconstruction and ensure effective performance when separating sound sources in a complex musical environment. Each component plays a role in the Hybrid Transformer Demucs' overall ability to perform advanced audio processing tasks.

The Hybrid Transformer Demucs retains the four outermost layers of the encoder and decoder of the Hybrid Demucs, adding a cross-domain transformer encoder in between them. This structure is critical to the model's high performance in separating complex music sources and enables in-depth analysis of the relationships between the parts of the music.

Demucs' ability to effectively separate music sources plays an important role in creating videoke tracks. In this research, we will explore how Demucs can be integrated into the real-time videoke generation process at the stage of removing the original vocals and incorporating into the videoke only the musical accompaniments.

2.2.2 Speech Recognition

Speech recognition is another well-known technique in audio processing, playing a fundamental role in the transcription of verbal information contained in audio recordings. In this Section, we will explore some concepts related to speech recognition, focusing primarily on the task of audio transcription.

This task involves converting audio signals containing speech into corresponding text, and requires approaches capable of handling the variety of voices, accents, and linguistic contexts present in recordings. A notable model in this domain is Whisper, a state-of-the-art solution for audio transcription.

Whisper is a sequence-to-sequence transformer model that stands out as a state-of-the-art solution for audio transcription. This model is trained on different speech processing tasks, such as multilingual speech recognition,

speech translation, spoken language identification, and speech activity detection. These tasks are jointly represented as a sequence of tokens to be predicted by the decoder, allowing a single model to replace multiple stages of a traditional speech processing pipeline.

Whisper's multitask training uses a set of special tokens that act as task specifiers or classification targets. This multifunctional approach contributes to the generalization and robustness of the model, enabling it to handle a variety of voices, accents, and linguistic contexts present in audio recordings. Figure 2.2 provides an overview of the architecture proposed in the Whisper paper (RADFORD et al., 2023).

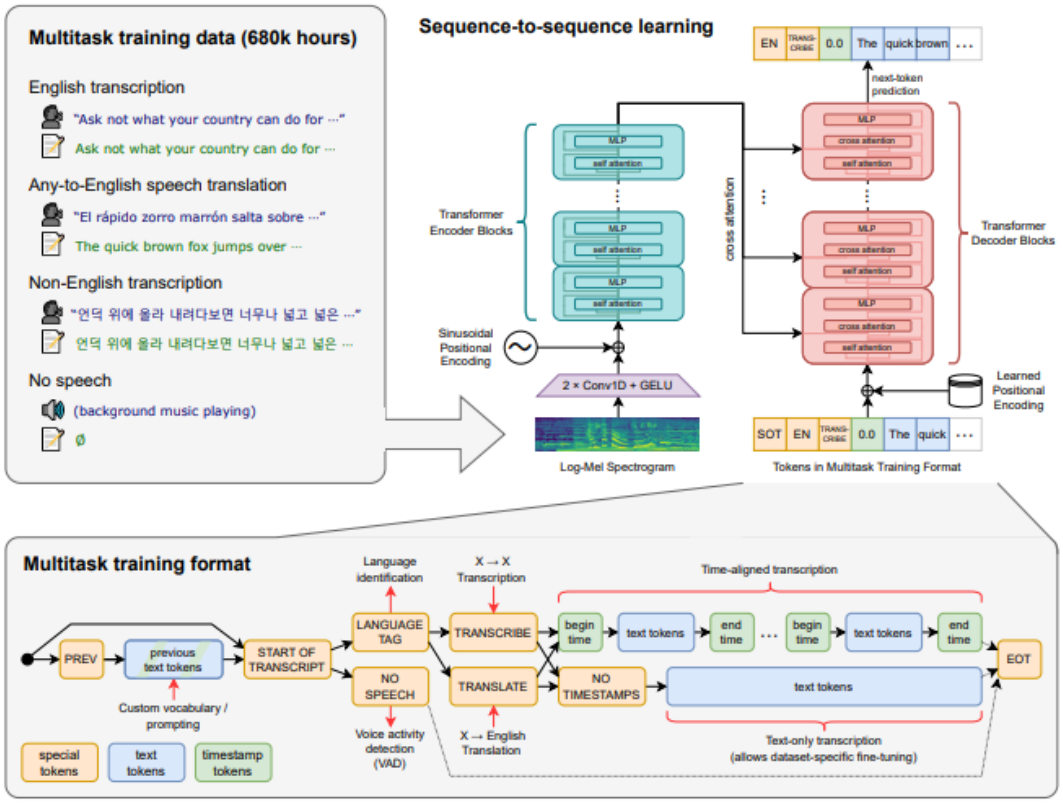


Figure 2.2: Overview of the Whisper architecture (RADFORD et al., 2023).

Whisper has been trained on large datasets of transcribed audio, allowing the model to learn complex patterns and linguistic nuances. Its generalization ability is enhanced through techniques such as transfer learning, where the model is initially trained on large datasets and then fine-tuned for specific tasks. Whisper's results stand out by demonstrating effectiveness in the speech transcription task across different scenarios and various reference datasets.

For this research, within the scope of speech recognition, we utilized Faster-Whisper (KLEIN, 2023), a reimplementation of the OpenAI Whisper model. It utilizes the fast-inference mechanism CTranslate2 for transformer models. This implementation achieves speeds up to four times faster than the

OpenAI Whisper model while maintaining the same accuracy and requiring fewer memory resources. Faster-Whisper stands out as an efficient alternative that provides equally satisfactory results, which is crucial for our real-time application.

The accuracy of speech transcription directly contributes to the quality of the generated subtitles and, consequently, to the user experience. In this research context, we will examine how models like Faster-Whisper can be integrated into the videoke generation process, enhancing the accuracy and usefulness of real-time generated transcriptions.

2.2.3 Subtitle Synchronization

Subtitle synchronization is an important step in the videoke generation process concerning the user experience, ensuring that subtitles are aligned with the corresponding segments of the audio recording (BRITTO, 2018).

Subtitle synchronization refers to the precise determination of when each subtitle should be displayed during video playback. This task is essential to ensuring that subtitles adequately accompany the audio content, allowing for coherent visualization of the song lyrics.

2.2.3.1 SubRip Subtitle

A widely used file format for storing subtitle synchronization information is the SubRip Subtitle (SRT) format. This format adopts a simple text structure, where each subtitle is associated with a sequential number, followed by temporal information indicating the start and end of the subtitle display. SRT uses the convention of hours, minutes, seconds, and milliseconds to accurately represent temporal instants.
Example structure of an SRT file:

```
1
00:05:00,400 --> 00:05:15,300
This is an example of a subtitle.
```

The typical structure of an SRT file consists of three main parts: the sequential subtitle number, the time interval during which the subtitle should be displayed, and the textual content of the subtitle itself. This organization facilitates reading and interpretation by media playback systems, ensuring the synchronized presentation of subtitles throughout the corresponding video.

The accuracy of subtitle synchronization is responsible for the overall quality of the videoke experience, allowing users to follow song lyrics as they are played. In this

research context, based on the results obtained by the audio transcription model, we will seek to automate the process of subtitle generation and synchronization. This process will be integrated into the real-time videoke generation process, ensuring synchronization between subtitles and the corresponding audio playback.

2.2.4 Incorporating Elements into Videos

The incorporation of multimedia elements refers to the insertion of additional data into an existing video composition. In our research, this data will be the subtitles. This process allows the inclusion of generated subtitles into music video clips. For this stage, we will use a widely used tool in studies related to this topic, FFmpeg.

2.2.4.1 FFmpeg

FFmpeg is an open-source software tool that has become prominent for audio and video manipulation, including the efficient incorporation of additional elements such as subtitles. Through the command line, FFmpeg offers a wide range of functionalities, allowing the addition of subtitles, overlays, or any other graphic element to videos with different formats and resolutions (NEWMARCH; NEWMARCH, 2017).

To incorporate subtitles using FFmpeg, you can use the subtitles filter. This filter allows overlaying subtitles onto a specific video track, adjusting their position, size, and appearance as needed. An SRT file is the input file sent on the command line in FFmpeg.
Command line example to embed an SRT file into a video:

```
$ ffmpeg -i video.mp4 -vf subtitles=subtitle.srt output_video.mp4
```

The effectiveness of FFmpeg in incorporating elements into videos lies in its ability to process various input formats and offer advanced configuration options. This flexibility makes FFmpeg the ideal tool for us to use in creating videokes with multimedia elements generated during the process.

In the scope of this research, we will explore how FFmpeg can be integrated into the real-time videoke generation process, enabling dynamic subtitle incorporation and manipulation of video and audio in the generated videoke medias.

2.3 Video Streaming

Video streaming is a technique for efficiently delivering multimedia content, enabling the continuous playback of audio and video in real-time without the need to wait for the complete file download. In this Section, we will explore the essential definitions related to video streaming and examine in detail the use of the HTTP Live

Streaming (HLS) protocol to establish a client/server architecture, thus simulating a streaming flow.

2.3.1 HTTP Live Streaming (HLS) Protocol

The HLS protocol stands out as a widely adopted solution for implementing video streaming. HLS divides the video into small segments, which are subsequently transmitted to the client through HTTP requests. This approach facilitates dynamic adaptation to the available bandwidth, adjusting the transmission quality as necessary to avoid playback interruptions.

One of the main benefits of this protocol is related to its compatibility features. Unlike other streaming formats, HLS is compatible with a wide variety of devices and firewalls.

The client-server architecture employed by HLS involves the sequential transmission of video segments, with the client continuously requesting new segments as playback progresses. The server, in turn, maintains a list of available segments and manages delivery according to client requests. This dynamic interaction between client and server creates the illusion of a continuous video stream, even as data is transmitted incrementally. Figure 2.3 illustrates the three components of an HTTP Live Stream: the server component, the distribution component, and the client software.

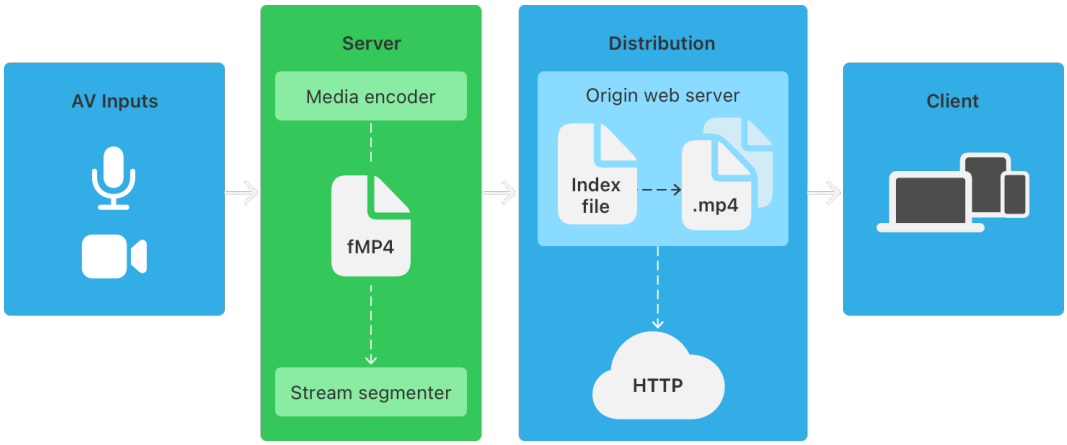


Figure 2.3: Components of an HTTP Live Stream

In a standard scenario, a hardware encoder receives audio and video signals and performs encoding into formats such as HEVC video and AC-3 audio, resulting in the production of a fragmented MPEG-4 file or an MPEG-2 transport stream. Subsequently, a stream segmenter, operating through software, divides this stream into multiple short media files, which are then stored on a web server. Simultaneously, the segmenter creates and updates an index file containing a list of the media files. The URL of this index file is published on the web server. The client software, upon reading the index, requests the media files in the presented sequence and plays them

continuously, without interruptions or noticeable gaps between segments (PANTOS; MAY, 2017).

In the context of this research, we will analyze the integration of the HLS protocol into the real-time videoke generation process. We will understand how adopting this video streaming technique enables the delivery of multimedia content, aiming to ensure a real-time videoke experience.

3
Related Works

This Section aims to analyze works related to the research proposed in this study, resulting from a systematic literature review. The literature review aimed to build a knowledge base on the current state of studies on this topic, aiming to identify opportunities for contribution to the proposed method. To achieve this purpose, the literature review was conducted specifically, using keywords and consulting various academic and technical sources.

Next, we will detail the process that guided this review, addressing the keywords used, the repositories consulted, the considered time window, and the inclusion and exclusion criteria adopted for the selection of works.

To identify studies related to the creation of an approach for real-time videoke generation from video streaming sources, the following keywords were used: real-time videoke, automatic karaoke generation, and real-time video streaming processing.

The investigation of related works covered various sources of information, including academic databases such as the ACM Digital Library and Google Scholar, as well as preprint repositories such as arXiv. Additionally, specialized portals for multimedia conferences, such as the International Multimedia Conference (MM), were explored.

The literature review encompassed works published from early 2013 to the most recent date available, which is September 2023. This temporal scope was established to ensure the inclusion of the most relevant and up-to-date works in the analysis.

To ensure the quality and relevance of the selections, inclusion, and exclusion criteria were applied. Regarding the inclusion criteria, works were considered that directly addressed the challenges associated with automatic karaoke and videoke generation or explored approaches related to real-time video streaming processing. Additionally, these works needed to be available in Portuguese and/or English.

In contrast, the exclusion criteria involved studies that did not maintain a direct connection with the theme of karaoke or videoke generation. As well as outdated works or those employing obsolete methods, not significantly contributing to the understanding of the problem. Additionally, works that were not freely accessible in full were excluded.

Based on these criteria, the works (TACHIBANA et al., 2016), (KWAN; SUN; YUDITSKAYA,), (CHEN; CHEN; JANG, 2021), and (FERNANDO et al., 2023) were selected to compose the related works Section of this dissertation, allowing for a comprehensive and comparative analysis of the state of the art regarding the solution to the issue at hand.

3.1 A real-time audio-to-audio karaoke generation system for monaural recordings based on singing voice suppression and key conversion techniques

In (TACHIBANA et al., 2016), published in 2016, a real-time karaoke audio-to-audio generation system based on techniques of sung voice suppression and pitch conversion is proposed. The general idea of the study focuses primarily on audio manipulation, presenting an automated system capable of suppressing the singer's voice in audio recordings of songs and performing real-time pitch conversion.

The authors developed an interactive music player called Euterpe. The system allows users to lower the vocal part and change the pitch of accompanying sounds. The system is basically a combination of already-known techniques, and therefore does not claim novelty in each technical component. However, they claim that this article has the following contributions: they demonstrated that a sung voice enhancer is as promising as a sung voice suppressor, and they developed a system that works in real-time by choosing lightweight technical components.

The problem with this approach is the singularity concerning the elements found in a karaoke system, as the study only addresses audio manipulation. The authors suggest as future work the construction of more intelligent karaoke systems, with the addition of displaying song lyrics.

3.2 Karaoke of Dreams: A multi-modal neural-network generated music experience

The work presented in (KWAN; SUN; YUDITSKAYA,), published in 2020, proposes a multidimensional approach to the karaoke experience. The Karaoke of Dreams (KoD) is a deep learning karaoke environment that generates music and videos based on user input of song titles.

The generation of karaoke songs consists of three components: lyrics, music, and the corresponding music video. The generation pipeline is modular, where both the music and the video are interconnected with the generated lyrics. In each generation module, a finely tuned open-source neural network model was employed.

In this process, KoD operates in the auditory dimension in harmony, melody, lyrics, and style. For the musical dimension, it reproduces 5-track pop songs generated by a Generative Adversarial Network (GAN), rendered through a Pure Data patch. In the visual dimension, it operates on images associated with words using attention GANs. In the linguistic dimension, it is generated by the fine-tuned GPT-2 in pop song lyrics. For the spatial dimension, it is a quadruple-rotating tesseract, delineating the three-dimensional space. The human element is the participant's voice and performance in the generated concert experience. Figure 3.1 shows the prototype created of the multidimensional space.

Figure 3.1: Participants singing in the KoD (KWAN; SUN; YUDITSKAYA,).

The major limitation mentioned in the work is the inability to generate on-demand content. For the authors, a pipeline could be constructed so that the song title, lyrics, music, and images can be generated in real-time immediately after the participant's input, in order to make the experience more engaging.

3.3 Singer separation for karaoke content generation

In (CHEN; CHEN; JANG, 2021) published in 2021, the authors address a specific challenge within the context of karaoke content generation, which is the separation of musical sources. As explained in Section 2, musical source separation can be used to separate the main singer's voice from the rest of the music.

Each song includes many components, such as drums, piano, vocals, harmonies, etc. They divide these components into accompaniment and vocals. Therefore, a singer separation system (SSSYS) includes two steps, as shown in Figure 3.2, where step 1 separates the music into accompaniment and vocals, and step 2 divides the mixed vocals into two vocal tracks. In the vocal separation step, an enhanced version of the Wave-U-Net model was used to accurately differentiate between vocals and accompaniment. Then, the resulting vocal files were used as training materials for the second step. In the main vocal separation step, the vocals are mixed to form various audio files and undergo a model self-selection. The models used in this step were the DPRNN and DPTNet, which convert mixed vocal data into two vocal tracks.

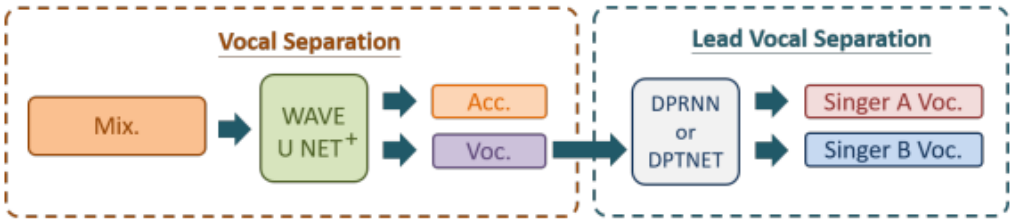


Figure 3.2: System flowchart of singer separation system (CHEN; CHEN; JANG, 2021).

This work focuses exclusively on a delineated stage within our karaoke generation method, specifically on the separation of musical sources. However, the approach of isolating the vocals and generating distinct audio tracks for one or more singers in a musical composition emerges as a potential enhancement to significantly improve the proposal of this dissertation.

Exploring the ability to separate and individualize voices in a musical track represents a promising extension of this work. This extension can enrich the user experience, allowing for greater customization in karaoke playbacks. By considering singer separation as an additional focal point, we can broaden the horizons of this study, delving into the complexity and practical benefits of this unique approach in the context of real-time karaoke generation.

3.4 Hybrid Y-Net Architecture for Singing Voice Separation

Finally, in (FERNANDO et al., 2023) published this year, a new deep learning-based approach, the Y-Net architecture, is proposed. It is designed to separate hybrid musical sources, and it is an end-to-end solution capable of extracting features from both spectrogram and waveform domains. The architecture combines the advantages of raw audio and spectrogram representations to estimate a time-frequency mask for separating the singing voice.

The Y-Net architecture consists of two main components: a learnable filter and a Y-net. The learnable filter offers the advantage of learning filters capable of extracting crucial features that are not present in the STFT spectrogram, while incorporating phase information. The Y-net extracts features from both domains and reconstructs the spectrogram of the singing voice.

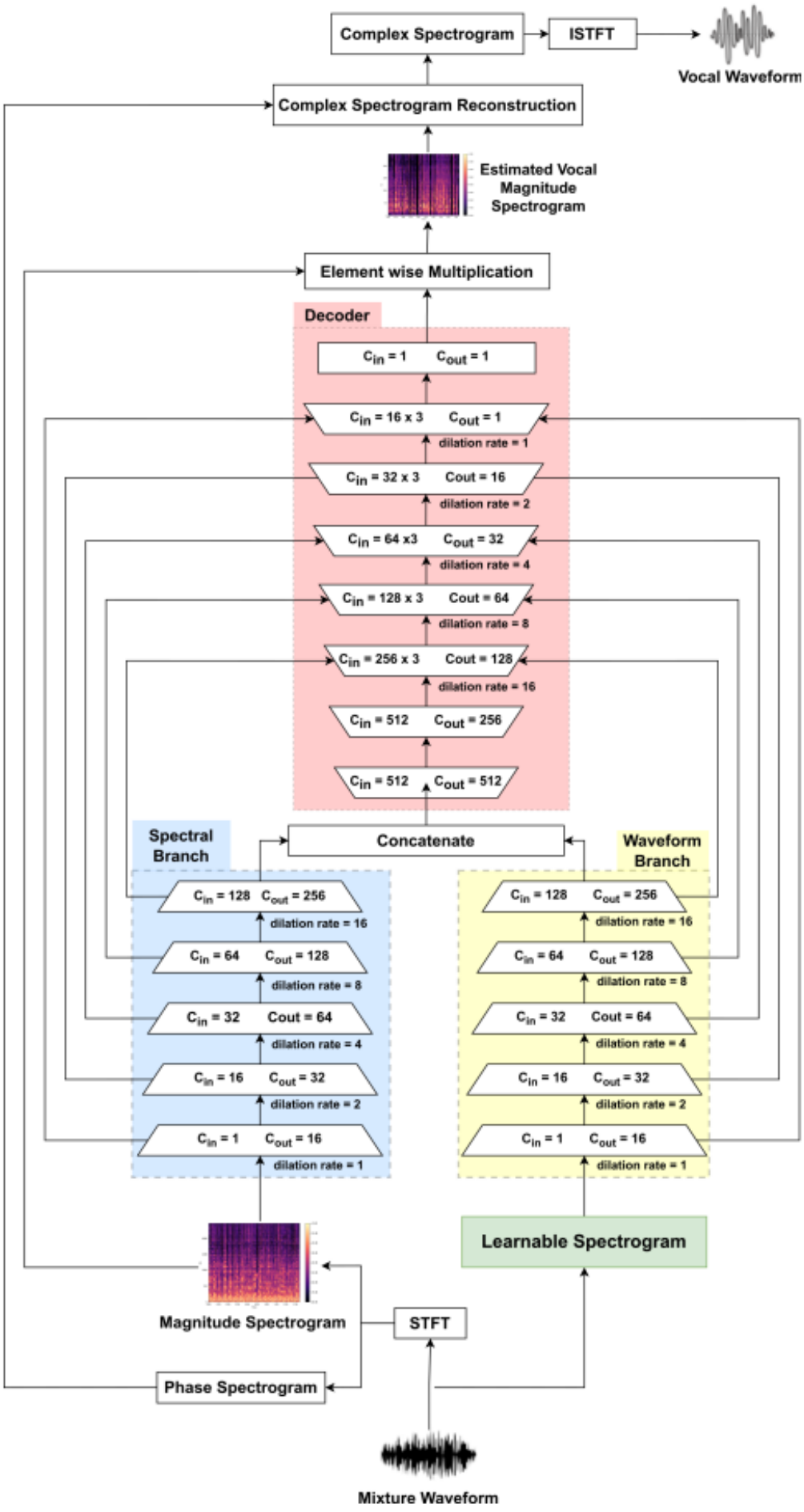


Figure 3.3: Y-Net Architecture (FERNANDO et al., 2023).

The network proposed in this related work consists of two encoding branches, namely the waveform branch and the spectral branch, which deal with raw audio and the spectrogram, respectively. The front part of the raw audio encoding branch includes a learnable spectrogram module, which converts raw audio to a spectrogram format. Then, two encoders extract features from both inputs, which are then merged into the core of the network and sent to the decoder. The decoder consists of a stack of upsampling layers that decode the features to the original size of the spectrogram, as shown in Figure 3.3.

This work, like (CHEN; CHEN; JANG, 2021), also focuses on improving a delineated stage within our karaoke generation method, which is the separation of musical sources. Thus, the limitation of generating karaoke content solely by manipulating audio also fits into this study, opening up possibilities to enhance the generated content by adding other elements, such as video and subtitles.

3.5 Discussion of related work

It is noteworthy that there is considerable emphasis placed on the task of vocal separation when examining related works in karaoke generation, as this stage plays a crucial role in the process. In our research, we are dedicated to applying established techniques of vocal separation and speech recognition, aiming to achieve high-quality results in these specific stages of videoke generation.

Among the related works, our proposal stands out as the first to explore a holistic approach. We are encompassing the video experience, audio devoid of original vocals, and synchronized subtitles, with word-level highlights, in the generation of real-time videoke from video streaming sources. Compared to the works presented in this Section, our study stands out for following this approach.

Table 3.1 provides a comparison of the elements addressed in karaoke generation among the works listed in this Section. The presence or absence of each element is indicated by “yes” or “no”, offering a clear view of the distinct contributions and gaps in existing research on karaoke generation. This comparative analysis reinforces the relevance of our work in the current landscape, highlighting its contributions in the field of real-time videoke generation.

Reference	Video	Audio	Subtitles	Word Level Sync	Real Time
(TACHIBANA et al., 2016)	No	Yes	No	No	Yes
(KWAN; SUN; YUDITSKAYA,)	Yes	Yes	Yes	No	No
(CHEN; CHEN; JANG, 2021)	No	Yes	No	No	No
(FERNANDO et al., 2023)	No	Yes	No	No	No
Ours	Yes	Yes	Yes	Yes	Yes

Table 3.1: Elements covered in each work.

4
Proposal

In this chapter, we will present in detail the method we have developed for real-time videoke generation. Our method, which we refer to as the *Videoke Generator*, is a solution composed of a series of interconnected modules. Each module performs a function in achieving the desired result, which is the automatic generation and playback of videoke media in real time.

The goal in generating videoke media is to provide users with a complete experience, including the original music video, the audio without vocals, and subtitles synchronized with word-level highlights to be sung at the correct moment of the melody. It is worth noting that our proposal was designed to perform this videoke generation process in real time or near real time, optimizing it especially for continuous video streaming.

To better understand the operation of the *Videoke Generator*, we will explore, in the subsequent Sections, the detailed analysis of each component of the proposed method, as illustrated in Figure 4.1. We will highlight the specific functionalities of each module, their relevant contributions to the overall system, and how the cohesive integration of these components results in a dynamic videoke experience.

This description of the method aims not only to clarify the technical operation of the *Videoke Generator*, but also to highlight the potential practical applications and benefits for end users.

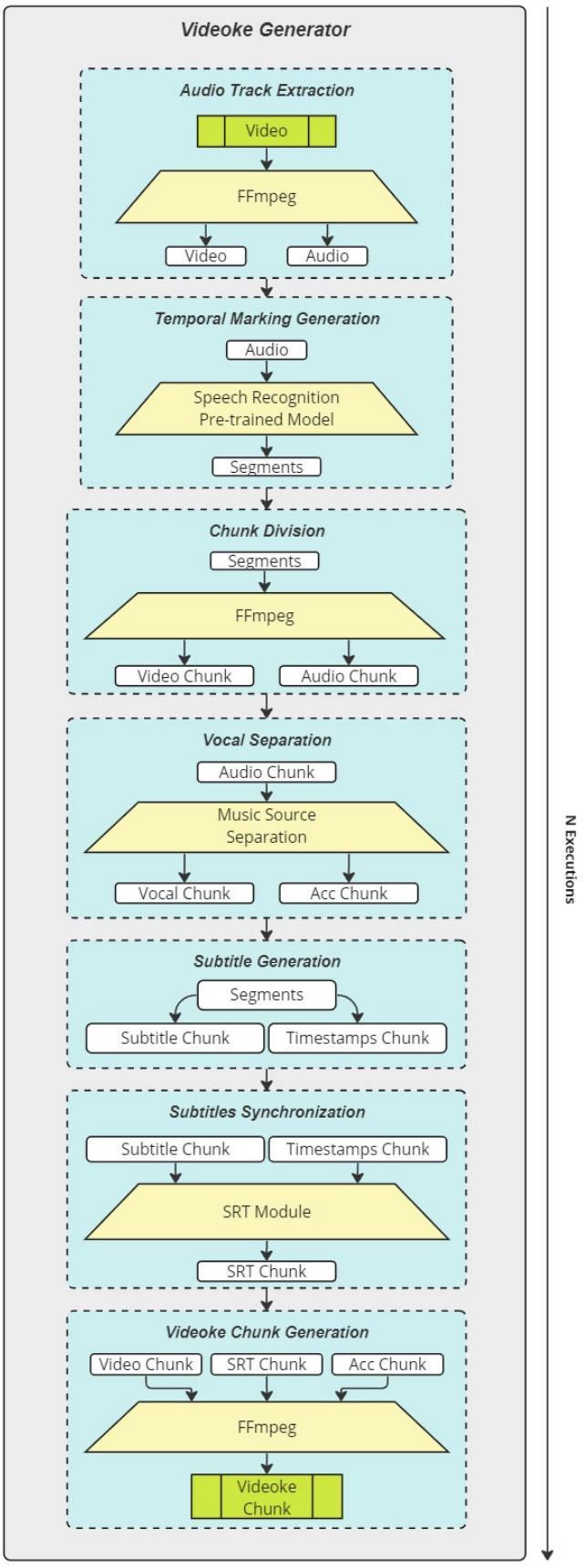


Figure 4.1: Method for Real-Time Automatic Videoke Generation

4.1 Audio Track Extration

The process begins with the Audio Track Extraction module, as illustrated in Figure 4.2. This module receives the video stream and performs a basic audio extraction operation, generating temporary files containing the unaltered audio from the video. This extraction aims to create two types of independent files: one containing the video and another containing only the audio, allowing them to be processed separately in the subsequent modules.

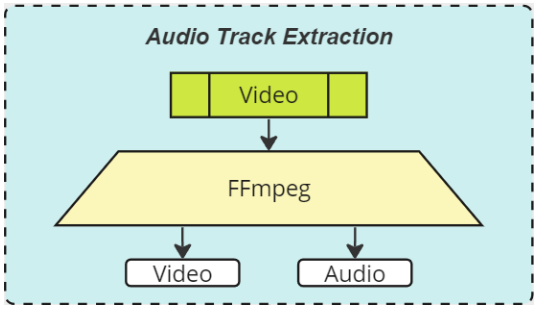


Figure 4.2: Audio Track Extraction Module

As the temporary audio files are successfully generated, the method can process them independently of the corresponding video. This is particularly important because, throughout the process, various operations and modifications will be applied to the audio to fit the videoke proposal. Therefore, the Audio Track Extraction module plays an initial role in the method, serving as the starting point of the processing flow.

4.2 Temporal Marking Generation

In the subsequent stage of the videoke generation process, the Temporal Marking Generation module, as illustrated in Figure 4.3, plays a significant role in creating a central element called *Segments*. These *Segments* are generated by applying a pre-trained model to the audio transcription task.

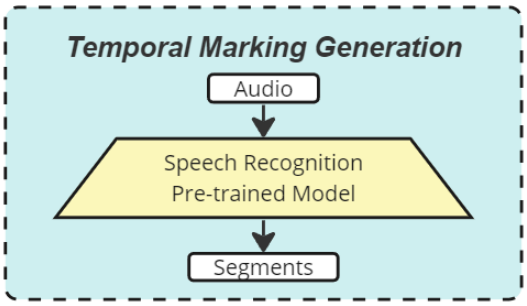


Figure 4.3: Temporal Marking Generation Module

The *Segments* are a data representation that incorporates essential information such as identifiers, transcriptions, and the start and end times of each segment. These temporal details are crucial for guiding the division of the video and audio into smaller, more manageable parts. These smaller parts are referred to as video and audio chunks, the purpose of which will be detailed in the following Sections.

The resulting data serves as a temporal map, allowing the system to locate specific moments when parts of the music are sung, ensuring that the division of the video into smaller Sections preserves all relevant information between segments. In the upcoming Sections, we will explore in detail how the *Segments* guide the division of the video and audio into smaller parts, as well as the transcription and synchronization of song lyrics. It is important to note that although the *Segments* contain transcription information, the file with the transcribed text has not yet been generated at this point in the process.

4.3 Chunk Division

The creation of chunks represents the granularization of the flow to enable the processing and playback of videoke in real time, or close to it. The importance of this step stems from a series of interrelated factors that have a direct and significant impact on the overall effectiveness of the system. For example, the processing time of each chunk is crucial to ensuring uninterrupted video playback. The structure of this module of the process is represented in Figure 4.4.

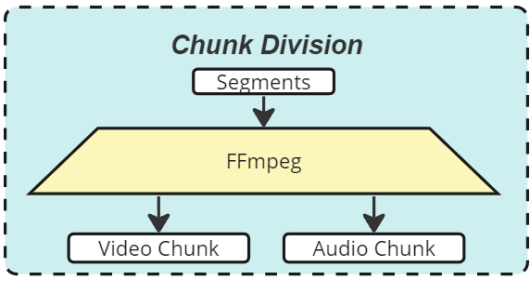


Figure 4.4: Chunk Division Module

Initially, the chunks are generated from the previously extracted video and audio files, following the temporal markings defined in the *Segments*. This procedure creates synchronized segments that correspond to specific parts of the song, ensuring the absence of abrupt cuts that could impair the user experience. The concern to avoid unexpected cuts arises due to the possibility of losing sung information between the song segments, which would compromise the synchronization of lyrics with the videoke images. Each chunk corresponds to a section of the song in which the lyrics are displayed according to the rhythm and timing of the music, providing a natural experience for users. This eliminates possible perceptible discontinuities that could disrupt the continuity of the videoke.

Additionally, the subdivision into chunks allows for efficient video processing, as the larger the video being processed, the longer the runtime of the models. By creating smaller segments, the system can handle each chunk independently, enabling immediate playback after processing is completed, without waiting for the other chunks. This approach aims to provide users with access to processed video segments while others are still being processed, significantly improving the efficiency of the videoke generation flow. This aspect is especially important in real-time processing scenarios, where latency can be considered the most critical point of the process.

In summary, the creation of chunks plays a role in ensuring coherence in the generated segments and minimizing runtime during processing to maintain seamless playback.

4.4 Vocal Separation

In the vocal separation stage (Figure 4.5), the audio chunks undergo special processing using a pretrained model for source separation. This stage aims to remove the original vocals from the music and create a temporary audio track containing only the instruments. This process is necessary to fulfill the core concept of videoke, which is to allow users to sing along with their own interpretations.

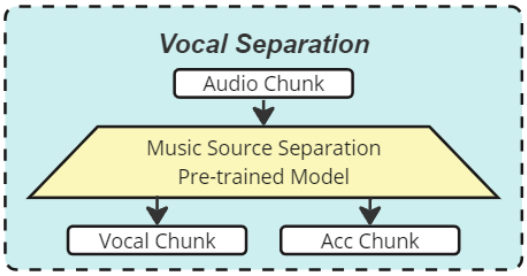


Figure 4.5: Vocal Separation Module

Vocal separation is a notoriously complex task, and this is where the pretrained model comes into play as an auxiliary tool in our method. This model is trained on a vast amount of data, enabling it to distinguish between vocal elements, representing the singer's voice, and background elements, representing the instrumental music. This distinction is absolutely essential, as it allows our system to isolate and separate the original vocals from the music, resulting in an instrumental track that will serve as the basis for user performances.

Therefore, the vocal separation stage, driven by a pretrained model, plays a fundamental role in our videoke creation process, enabling the extraction of original vocals from the background music in each chunk.

4.5 Subtitle Generation and Synchronization

At this stage, we resume the use of the object generated by the pretrained transcription model, previously created based on the original audio. The process of generating transcribed texts occurs at this point in the flow. The *Segments* object contains transcription information; however, to generate the transcription files, it is necessary to access them by iterating through the object and relating its identifiers to the chunk being processed. By accessing this information, we obtain detailed transcriptions for each chunk, capturing each word of that chunk along with its respective timestamps. This procedure is carried out to cover all audio segments corresponding to each chunk.

The quality of word-level transcriptions is of paramount importance for the user experience, as it allows lyrics to be displayed synchronized with the melody. This information is combined with a specialized script we developed internally, called the SubRip (SRT) Module. This module automates the process of generating SRT files. Its script accesses the information provided by the *Segments* and automatically writes the corresponding SRT file for each chunk. This procedure aims to generate subtitle synchronization files that will be incorporated into the video, specifically designed for this purpose.

The resulting SRT files incorporate synchronized subtitles and word-level highlights for each chunk. The highlights emphasize the words as they are sung in the music, creating a visual effect that significantly enhances the perception of song lyrics.

This approach is illustrated in Figure 4.6 and not only ensures that song lyrics are synchronized with the melody but also enriches the user experience with word-level highlights, making it more intuitive to follow along with a song.

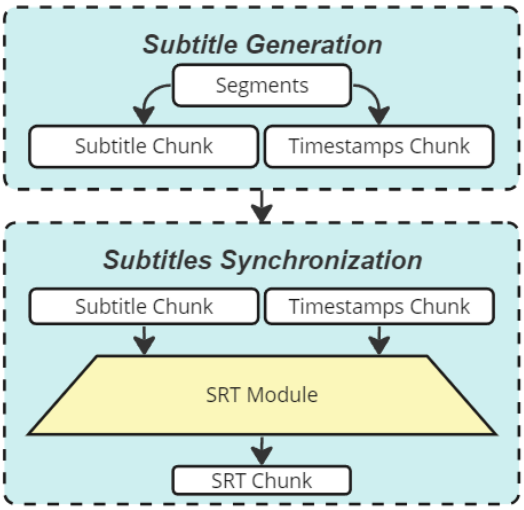


Figure 4.6: Subtitle Generation and Synchronization Module

4.6 Videoke Chunk Generation

Finally, Figure 4.7 illustrates the phase where the video chunk is merged with the subtitles and the corresponding audio, resulting in a videoke chunk. This stage marks the tangible outcome of the entire videoke generation process, transforming a series of separate elements into a single video media. This videoke chunk is temporarily stored as a file on the servers, awaiting delivery to the user.

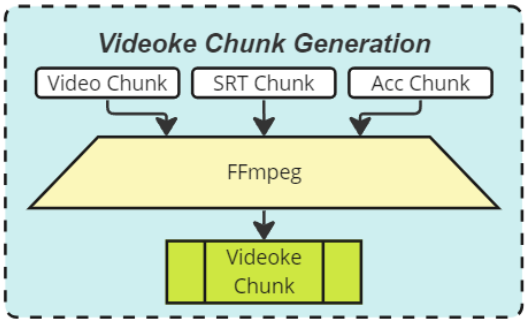


Figure 4.7: Videoke Chunk Generation Module

4.7 Real-time processing and buffering

To conclude the real-time videoke generation process, it is crucial to understand the integration of all the previously mentioned modules and steps. The entire method has been designed to operate within a client-server architecture, where processing occurs on the server side and playback on the client side. To enable this structure, an effective means of communication between these two parts must be established.

The central point for controlling and managing the chunks is the use of a video transmission protocol. This protocol plays the role of coordinating and synchronizing each component of the videoke, allowing the chunks generated by the *Videoke Generator* flow to be transmitted continuously and in real-time, or close to it.

The mechanism for managing the video transmission protocol is a playlist file, also known as a manifest file (m3u8). This file is responsible for coordinating the playback of the chunks, serving as the basis for continuous communication between the server and client. It contains essential metadata, such as the total duration of the videoke chunk, the playback order of the chunks, and other details relevant to ensuring proper synchronization between audio, video, and subtitles. It acts as a guide for the media player, allowing it to request and play the chunks in the correct sequence, ensuring smooth playback.

To set up a server with a video transmission protocol, several technical details must be considered to ensure efficient performance. Firstly, the choice of server software is crucial. Popular options include Wowza Streaming Engine, Adobe Media

Server, and NGINX. Each of these software options offers specific features and requires proper configurations to meet the project's needs.

After installing the server software, it's important to properly configure the video transmission properties. This includes setting the available bandwidth for transmission, video resolution, frame rate (FPS), and encoding format, such as H.264 or H.265. These settings should be adjusted according to the project specifications and the capabilities of the available network infrastructure.

To configure the client side, a strategy that can be used is to use an HTML player. Typically, these players are JavaScript libraries that allow playback of video streams based on some video transmission protocol directly in the browser, without the need for additional plugins.

In addition to the basic configuration, it's also significant to consider the player's compatibility with different browsers and devices. Although HTML players are widely supported in most modern browsers, there may still be performance and behavior differences among different playback environments. Therefore, it's advisable to perform tests on a variety of browsers and devices to ensure a consistent playback experience for users.

Real-time processing requires careful management of the buffer, which temporarily stores video data before it's played back on the client. In the specific case of HTML players, the client-side buffer plays the role of ensuring smooth and continuous video playback. While the server is processing the initial block of chunks and waiting for their completion to add them to the manifest file, the HTML player's buffer is displayed to the user on the screen.

During this period, the HTML player's buffer remains constantly loading, as the video files have not yet been added to the manifest file and thus are not ready for playback. The duration of this loading can vary depending on the number of chunks in the initial block and the server's processing capacity.

An initial latency ranging from 10 to 30 seconds is considered reasonable according to the documentation of some video transmission protocols, such as HLS (PANTOS; MAY, 2017) and MPEG-DASH (SODAGAR, 2011). For most broadcasters, this is not an issue, and a live stream can handle such a delay without causing user dissatisfaction.

It's important to note that processing capacity is only considered on the server side. On the client side, the HTML player only plays the video. Therefore, the system can be used anywhere with the ability to open an HTML player, such as a computer or a mobile device.

Once the server completes the processing of the initial block of chunks and adds them to the manifest file, playback can be initiated on the HTML player. At this point, the player's buffer begins to be filled with the video data transmitted by the server, allowing playback to occur smoothly and without interruptions.

While the processed block is being played back on the client side, the server continues to process the remaining chunks and adds them to the manifest file

whenever each one finishes its videoke generation process. This allows the video to be processed while it's being played back, ensuring that the user does not even notice the processing, thus making it happen in real-time, or close to it.

It's important to highlight that, to ensure an optimal viewing experience, it's essential to optimize the buffer. This includes adjusting the chunk block size to balance latency and playback quality, as well as designing a server with reasonable processing power to ensure stable and efficient transmission.

5
Experiments and Results

In this section, we will provide a detailed description of the experimental settings, the third-party software used, the implementation and execution of the method, as well as the analysis of the results obtained. Our goal is to ensure the reproducibility of the work, enabling other researchers and developers to use and evaluate the prototype developed in various contexts.

Initially, we will describe the experimental settings, including details about the hardware and software used during the experiments. Next, we will present the third-party software employed, highlighting its functionalities and contributions to the project's development. Subsequently, we will address the implementation of the proposed method, providing information about the system development process and the steps involved in real-time videoke generation.

Following the implementation description, we will detail the execution of the experiments, including the procedures adopted for data collection and system performance evaluation. We will analyze the results obtained in terms of video quality, audio and subtitle synchronization, transmission latency, and other parameters relevant for evaluating the method.

Finally, we will discuss the conclusions drawn from the conducted experiments, highlighting the strengths and limitations of the developed prototype, as well as possible directions for future research and enhancements. Through this Section, we aim to offer a detailed insight into the experimental process, providing insights for the academic and professional community interested in the development of real-time videoke generation systems.

5.1
Configuration

Before delving into the details of the experiments, it is essential to present the settings used in the development and experimentation of the method. This includes information about the hardware used in the experimentation, including details about the processor, RAM, storage capacity, and other components relevant to the system's performance.

5.1.1
Hardware

The experiments were conducted on a remote server with the following specifications:

- **Processor:** Intel(R) Core(TM) i7-5960X CPU @ 3.00GHz
- **RAM:** 62GB

– **Graphics Processing Unit:** NVIDIA GeForce GTX 1080

5.1.2 Third Party Software

To implement and execute our prototype, we utilized a variety of third-party software, including:

– **Python 3.8:** The primary programming language for code development.
– **FFmpeg:** A command-line tool for video and audio processing, used for manipulating and converting media formats.
– **SubRip (SRT):** A widely used subtitle file format for creating and synchronizing subtitles.
– **Demucs:** Pre-trained model for the Task of Music Source Separation (ROUARD; MASSA; DÉFOSSEZ, 2023)
– **Faster-Whisper:** Pre-trained model for Audio Transcription Task (KLEIN, 2023)
– **HTTP Live Streaming (HLS):** Widely used media streaming protocol for transmitting audio and video content over the internet

5.2 Experimentation

In this Section, we will detail the experimental procedures carried out within the scope of this project. The main objective of these experiments is to implement a prototype using the proposed method for real-time videoke generation outlined in this research. To achieve this goal, a series of tests were conducted, encompassing a variety of music genres and execution conditions. This approach allowed us to assess the generalization capability of our prototype in different scenarios.

The experiments were structured according to the method described in Section 4, following a sequence of steps. Initially, we prepared the experimental environment, configuring the server and client according to the previously defined specifications. Next, we selected a sample of music representative of various genres to compose our test base.

5.2.1 Audio Track Extraction

In this Section, we will describe the experiments conducted in the Audio Track Extraction stage, which constitutes the first step of the proposed method for real-time videoke generation. The main objective of this stage is to extract the audio track from a video stream, creating a temporary file on the server containing the unaltered audio, which will be subsequently processed in the following stages.

The central component of this module is the class called Audio Track Extractor. This class is responsible for performing the extraction of the audio track from the input video stream. To do this, the module uses the FFmpeg tool, which operates via command line to generate two temporary files on the server, one with the video and the other with the audio.

In the process of executing the FFmpeg command, various necessary arguments are specified. For example, the audio codec to be used is determined, opting for PCM (Pulse Code Modulation) with 16-bit little-endian. Additionally, a sampling rate of 44100 Hz is established, a widely recognized and employed standard in the audio industry. These parameters are crucial to ensuring the fidelity and quality of the extracted audio track.

5.2.2 Temporal Marking Generation

In this phase of the process, we use a specific pre-trained model for audio transcription. We opted for the Faster-Whisper model, a reimplementation of the original Whisper model from OpenAI using CTranslate2, an inference acceleration engine for Transformer models. We chose this adaptation due to its effectiveness, being up to four times faster than the original model while maintaining the same accuracy and requiring fewer memory resources, as documented by Guillaume Klein in (KLEIN, 2023).

In practice, we developed a simple class called Transcriber, which plays the role of utilizing the inference capability of the pre-trained model. This class is responsible for processing the audio track of the video, extracted in the previous stage, and generating the object called Segments. These segments are structured representations containing relevant information for videoke generation. This approach allows for the mapping of important parts in the audio, providing the necessary details to synchronize subtitles with the video during playback.

5.2.3 Chunk Division

In the next stage of our process, we implemented the ChunkGenerator class, which is responsible for creating video and audio chunks based on the information contained in the Segments, as described in Section 4.3. To perform this task, we used the FFmpeg tool as the operational base.

The ChunkGenerator class receives three parameters: the start time of the chunk, the end time of the chunk, and a unique identifier for the chunk. Based on this information, the code generates temporary files representing the video and audio chunks, using the provided identifier.

These temporary files are created with the same video and audio codecs as the input file, eliminating the need for reformatting or additional processing. This

approach aims to optimize latency, as the video is simply copied directly, reducing processing overhead.

An interesting feature was incorporated to enhance the real-time videoke generation process. It was observed that most songs start with an instrumental introduction devoid of vocals. Thus, it was realized that this non-sung initial segment could be treated as a videoke chunk without the need to go through processing stages. Based on the information provided by the Segments, we identified the moment when the first sung part of the song begins. Therefore, we considered the entire video segment until the start of the first sung part as the first videoke chunk, thus minimizing processing time.

The files stored on the server to be sent to the client are of the Transport Stream (.ts) type, also generated with the FFmpeg tool. We specify a bit stream filter for the video, converting the H.264 video codec to a format called Annex B, suitable for video streaming (SUBHASH, 2023). This file plays a fundamental role in managing the streaming flow and is incorporated into the manifest file.

In cases where the song does not have an instrumental introduction, the first chunk follows the same process as the others, resulting in a slight delay in the start of the transmission. However, this delay is only the time needed to process the first chunk, and details about the processing time for each chunk will be discussed later in the results' subsection.

5.2.4 Vocal Separation

Within the fundamental concept of a videoke, the stage of removing vocals from songs is present. This phase is implemented through the VocalSeparator class, whose objective is to separate the vocal parts from the audio, which can contain both music and vocals. To perform this task, we utilize the Demucs model, a pre-trained model for the task of music source separation, developed by the Facebook research group (ROUARD; MASSA; DÉFOSSEZ, 2023).

Demucs represents a cutting-edge technology model for source separation in music, capable of isolating components such as drums, bass, and vocals from the rest of the music. However, for our prototype, we chose to focus on separating only two sources: vocals and instruments. The Demucs model is built on a convolutional U-Net architecture, inspired by the Wave-U-Net, and has been adapted to become a hybrid model, combining spectrogram separation techniques and waveform methods with the use of transformers.

Vocal separation is internally performed by the VocalSeparator class, utilizing the inference capability of the pre-trained model. This results in the generation of new temporary audio files: one containing only the vocals, called the vocal chunk, and another with the instrumental accompaniment of the music, called the accompaniment chunk. These two chunks constitute the basis for the subsequent stage of the videoke generation process.

5.2.5 Subtitle Generation and Synchronization

One of the contributions of this work is the automatic synchronization of subtitles at the word level. This synchronization follows the highlights that coincide with the part being sung in the video. This result was achieved through the module we developed, called the SubRip Module.

The implementation of this module involves the creation of the SRTFileGenerator class, whose objective is to generate temporary subtitle files in the SRT format. The files are generated with subtitles that have dynamic highlights based on the word information contained in a segment. These subtitles enrich the video experience as they assist users in singing along.

Here is where we access the information from the *Segments* object through their identifiers, and the written transcription generation occurs. This process iterates through each of the identified segments, and each generated video and audio chunk corresponds to a generated segment.

The creation of the temporary SRT file occurs in a loop that iterates through the words in the segment. The script calculates the start and end times of the subtitles and the moments when each word should be highlighted, based on the temporal information of the words in the segment. Then, this information is added to the file in SRT format.

The highlighting of the words is implemented by applying styling to the words. We use yellow when they are being sung, while the others remain white. This highlighting is achieved by formatting each word as an HTML tag and incorporating it into the SRT file. Figure 5.1 represents an example of this styling in the subtitles.

Figure 5.1: Example of styles in videoke

At the end of this process, we obtain an SRT chunk, which contains synchronized subtitles configured to highlight each word being sung in the video chunk. This element is essential for the user experience, making the interpretation of song lyrics more dynamic.

5.2.6 Videoke Chunk Generation

In the final stage of processing, we integrate all the elements generated so far. At this point, we have a backing track chunk, an SRT chunk, and the video chunk. Now, we need to merge these components to form a videoke chunk. To achieve this, we implemented the VideokeGenerator class, whose main objective is to create videoke chunks from the various mentioned sources. The process of generating videoke involves the merging of video, audio, and subtitles. Thus, in this phase, we again resort to the FFmpeg tool to manipulate these files.

It is important to note that, as these files will be used in a streaming flow, each of them has a container with information, including timestamps for playback within the playback stream. Therefore, it is necessary to inform these timestamps to the videoke chunk to ensure continuous playback.

To accomplish this, when using the VideokeGenerator class, an offset is always provided, representing the temporal marker of each videoke chunk. This offset is calculated based on the accumulation of the duration of each chunk, referring to the starting point of each videoke chunk in relation to the original video.

With all this information properly organized, the VideokeGenerator class uses the FFmpeg tool to combine the video chunk, the backing track chunk, and the subtitles, thus creating a videoke chunk. After merging these elements, we have a temporary file in the Transport Stream (.ts) format, suitable to be added to the manifest file and thus be requested by the client to continue the playback flow in the player.

5.2.7 Architecture

The architecture proposed in this project is based on a client-server model, designed to enable real-time or near-real-time processing of videoke generation. This structure consists of two main components: the client, responsible for sending requests, and the server, tasked with processing these requests and returning the resulting videoke chunks.

Figure 5.2 visually illustrates this architecture. On the client side, users interact exclusively with the HTML player, which makes requests to the server to play the video. On the other hand, the server is composed of several modules, each responsible for a specific stage of the videoke generation process, as detailed in Section 4. For simplification, we abstract the *Videoke Generator* module.

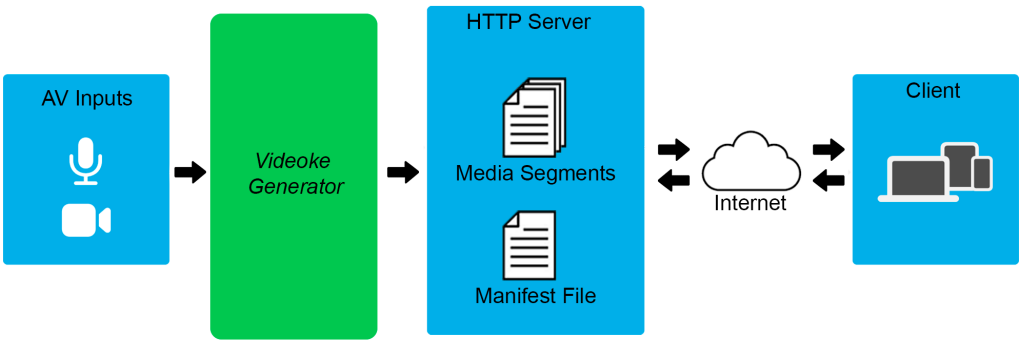


Figure 5.2: Client-Server Architecture for Streaming

The communication flow between client and server is continuous. To achieve this, the server was configured with an NGINX web service and uses the HTTP Live Streaming (HLS) video transmission protocol.

NGINX provides an event-driven and asynchronous architecture, allowing servers to handle HTTP requests efficiently. This protocol, in turn, is responsible for transferring information between devices connected to a network, enabling communication between client and server. The NGINX server is configured to receive the videoke chunks and store and manage the manifest file, which contains information about the available segments, their URLs, and their durations. The NGINX server is represented by the HTTP Server block in Figure 5.2.

Regarding HLS, this protocol originates from the server where the streaming is being created. As it is based on HTTP, any common web server can originate the streaming. Video data is reformatted so that it can be recognized and interpreted by any device.

The flow of the architecture is intuitive: the video is initially divided into segments, encoded, and sent to client devices over the internet when streaming is requested. The client device, such as a smartphone or laptop, receives the stream and plays the video, using the manifest file as a reference to assemble the video in the correct order for playback.

5.3 Results

During the experimentation phase, we directed our attention to several key points in evaluating the results, including transcription accuracy, vocal separation quality, and synchronization between audio and subtitles. Each of these aspects was analyzed to assess the prototype's performance and usability.

In this Section, we will explore the results obtained from the prototype development. The analysis also considers the processing times of the chunks, aiming to validate the applicability of the solution for real-time or near-real-time contexts.

5.3.1 Qualitative evaluation of videokes

To assess the quality of the generated videoke, we conducted 15 experiments across the “Rock”, “Forró”, “MPB”, “Rap” and “Pagode” categories. These experiments were generated by the prototype, enabling us to perform evaluations. The primary objective of this analysis was to provide a qualitative assessment of transcription accuracy, vocal separation effectiveness, and synchronization between audio and subtitles. Additionally, we aimed to understand the impact that different musical genres have on the system's performance.

The results of the experiments highlighted several relevant aspects. It was observed that, regardless of the musical category, the experiments yielded satisfactory transcriptions, quality audio, and a high rate of accuracy in audio-subtitle synchronization. These results demonstrate that the music category is not a direct factor affecting the system.

However, it was identified that other factors may impact the system's performance, which are not directly associated with musical style but rather with the song's rhythm and the level of interaction between the singer and the audience. Specifically, the fast pace of some songs, such as Eminem's “Rap God” and Haikaiss's “RapLord” posed significant challenges. The very fast and complex passages in these songs represented a difficulty even for human perception, which consequently impacted the transcription accuracy and the overall quality of the generated videoke.

In addition to challenges related to the fast pace of certain songs, other factors were also identified as significant influencers of the videoke generation system's performance. One of these factors is the level of interaction between the singer and the audience during the performance of the songs. During the experiments, it was observed that the presence of interactions, such as applause, shouts, or audience conversations, introduces additional noise in the audio. These noises from the interaction between the singer and the audience can cause distortions in the audio, resulting in difficulties for the transcription model to correctly recognize and interpret the song's words. As a result, transcription failures occur, and consequently, synchronization between audio and subtitles is compromised, compromising the overall quality of the generated videoke. This phenomenon was especially noted in the “Pagode” and “Forró” genres, where singers interact heavily with the audience, generating noise that is not related to the song's lyrics.

These observations can guide future improvements and refinements in the architecture and algorithms used, aiming to ensure a more consistent and satisfactory experience for users in a variety of musical and interaction scenarios.

5.3.2 Chunk processing time

As explained in 4.7, it is important to highlight that the chunk processing time is directly related to the processing capacity of the server on which the system is hosted. The server used for our experiments did not have a high-performance GPU, unlike the benchmarks used in the music source separation and audio transcription models. Therefore, the processing time values presented reflect the conditions of the available hardware during the experiments. It is essential to understand that these values are relative and subject to variations, as they can be improved or hindered depending on the processing capacity of the available hardware.

To establish parameters, the music source separation (Demucs) and audio transcription (Faster-Whisper) models were evaluated on an NVIDIA Tesla V100S GPU, recognized for its effectiveness in artificial intelligence tasks. Considering the transcription model's processing time as a reference point, since it provides the audio used in the tests, we based it on the time required to process an audio of 13 minutes and 19 seconds (available at <https://www.youtube.com/watch?v=0u7tTptBo9I>). On the NVIDIA Tesla V100S GPU, the average processing time for this audio was approximately 59 seconds, while on our server, the same audio required an average of 320 seconds. This difference highlights the direct impact of hardware on model processing times. An advantage is that, being a client-server architecture, the client side does not need to deal with these computational demands, as all processing is performed on the server. Thus, the client only requests chunks that have already been processed and plays them on the player, significantly simplifying the client side.

We conducted a series of tests with different numbers of chunks in the initial block to be processed, aiming to find a number that, suitable for our hardware, established an initial buffer between 10 and 20 seconds. After several attempts, we determined that the ideal value for the initial buffer size is three chunks. This buffer allows the system to maintain a margin between the chunks being played and those being processed, avoiding possible bottlenecks during playback.

Additionally, we analyzed the implementation of sung introductions detection, mentioned in subsection 5.2.3. We observed that songs with unsung introductions had a smaller initial buffer compared to those without these introductions, and they had a larger safety margin, which further contributes to avoiding interruptions during playback. It is important to highlight that the HTML player has the ability to create a loading buffer for the user in case of playback interruptions. This buffer allows the user to wait while the next chunk is made available in the manifest file to be played. Once the chunk is available, playback resumes normally.

5.3.3 Evaluation of the models used

5.3.3.1 Faster-Whisper

Speech recognition systems are typically evaluated based on the Word Error Rate (WER) metric. However, WER, which is based on string edit distance, penalizes all discrepancies between the model output and the reference transcription, including benign differences in transcription style. As a result, systems that produce transcriptions considered correct by humans may still exhibit a high WER due to minor formatting variations. While this is a problem for all transcribers, it is particularly pronounced for zero-shot models like Faster-Whisper, which lack access to examples of transcription formats from specific datasets.

This is not a new finding; developing evaluation metrics that better correlate with human assessment is an active area of research. While there are some promising methods, none have been widely adopted yet for speech recognition. The Faster-Whisper model chooses to address this issue through extensive text normalization before calculating WER, in order to minimize the penalty for non-semantic differences. The text normalizer was developed through iterative manual inspection to identify common patterns where naive WER penalized Whisper models for insignificant differences. All details of this process can be found in the paper published by OpenAI (RADFORD et al., 2023).

WER measures the percentage of incorrect word transcriptions across the entire set. The lower the WER, the more accurate the system is. In this regard, Faster-Whisper has demonstrated good performance in multilingual transcription tasks. Tests were conducted on datasets specific to this task, including the Multilingual LibriSpeech (MLS) (PRATAP et al., 2020), Common Voice 9 (ARDILA et al., 2019), and FLEURS (CONNEAU et al., 2023). Faster-Whisper achieved a WER of 6.8 on the MLS dataset, 6.3 on Common Voice 9, and 4.3 on FLEURS, which are results that outperform the evaluation of other models for this task, justifying its choice to be integrated into our system.

5.3.3.2 Demucs

For the task of music source separation, the evaluation is based on the Signal-to-Distortion Ratio (SDR) metric, as defined by SiSEC18 (STÖTER; LIUTKUS; ITO, 2018). The tests proposed in the paper (ROUARD; MASSA; DÉFOSSEZ, 2023) were conducted using the MUSDB dataset (RAFII et al., 2017), along with additional training data.

These extra data were compiled into an internal dataset consisting of 3500 songs, with tracks from 200 artists spanning various musical genres. Each track was assigned to one of four sources based on the name provided by the music producer (e.g., “vocals2”, “sound effects”, “bass” etc.). However, this labeling is susceptible to noise as these names are subjective and sometimes ambiguous. For

150 of these tracks, manual verification was performed to ensure the accuracy of automated labeling, discarding ambiguous tracks. With this, a first Hybrid Demucs model was trained on the MUSDB and these 150 tracks.

The evaluation conducted in the paper published by the Facebook research team compared the proposed model with various state-of-the-art baselines and also provided baselines trained without any extra training data for reference. Comparing with the baselines, it was observed that the enhanced sequence modeling capabilities of transformers increased the SDR, reaching a value of 9.20 dB of SDR, which outperformed the other evaluated models in the comparison. Based on these results, the Demucs model was chosen to assist in the task of music source separation in our system.

6
Conclusion and future work

Throughout this dissertation, we have presented a method for generating videoke in real-time or near-real-time and built a prototype to validate the proposed methodology. The research began with the motivation to overcome the limitations inherent in traditional videoke systems, as detailed in Section 1. From this point, we examined the challenges and opportunities associated with audio transcription, music source separation, and word-level subtitle synchronization within the context of automatic videoke generation.

The main goal of the work was to propose a method for automatic videoke generation that could be implemented for real-time processing streams. This goal was achieved by demonstrating the feasibility of the proposed client/server architecture and highlighting the transcription generation, audio quality, and subtitle synchronization demonstrated in the results.

Additionally, the complementary contributions, which involved subtitle synchronization with word-level highlights and the incorporation of a granularization mechanism to allow chunk playback as they were processed, were also achieved. We can also highlight as a contribution the effective use of cutting-edge audio transcription and music source separation models in the method's processing flow, as well as the approach of a client/server architecture to enable the involved challenges. These points contribute to the advancement of studies and development of systems focused on the multimedia and entertainment field.

As possible directions for future work, we can mention some possibilities for research evolution. The field of music source separation and audio transcription is constantly evolving, so it would be interesting to investigate machine learning techniques to improve the quality of vocal separation and audio transcription. Another identified aspect is the opportunity to implement singer detection, as described in (CHEN; CHEN; JANG, 2021), to allow duets and reduce the possibility of errors in the generated transcriptions. Thinking about users, it would be pertinent to create a scoring system for users and other ways to enrich the videoke experience. In summary, this work is another step in a journey of continuous exploration and innovation in videoke generation. The possibilities are vast, and we hope that this study inspires future researchers and developers to continue enriching and expanding the capabilities of this project.

7
Bibliography

ARDILA, R. et al. Common voice: A massively-multilingual speech corpus. **arXiv preprint arXiv:1912.06670**, 2019.

BRITTO, J. O. **Análise do Impacto da Sincronia de Legendas na Qualidade de Experiência do Usuário**. Tese (Doutorado) — Master's dissertation. UFES, ES, Brazil. http://repositorio. ufes. br/handle . . . , 2018.

BUSS, Z. da S. et al. Projeto "12&30": um relato de experiência integrando karaokê e saúde mental na universidade. **Extensio: Revista Eletrônica de Extensão**, v. 19, n. 44, p. 72–80, 2022.

CANO, E. et al. Musical source separation: An introduction. **IEEE Signal Processing Magazine**, IEEE, v. 36, n. 1, p. 31–40, 2018.

CHEN, H.-Y.; CHEN, X.; JANG, J.-S. R. Singer separation for karaoke content generation. **arXiv preprint arXiv:2110.06707**, 2021.

CONNEAU, A. et al. Fleurs: Few-shot learning evaluation of universal representations of speech. In: IEEE. **2022 IEEE Spoken Language Technology Workshop (SLT)**. [S.l.], 2023. p. 798–805.

DUNBAR, R. I. et al. Performance of music elevates pain threshold and positive affect: Implications for the evolutionary function of music. **Evolutionary psychology**, SAGE Publications Sage CA: Los Angeles, CA, v. 10, n. 4, p. 147470491201000403, 2012.

FERNANDO, R. et al. Hybrid y-net architecture for singing voice separation. **arXiv preprint arXiv:2303.02599**, 2023.

HOSOKAWA, S.; MITSUI, T. **Karaoke around the world: Global technology, local singing**. [S.l.]: Routledge, 2005.

KLEIN, G. **Faster Whisper transcription with CTranslate2**. 2023. Disponível em: <https://github.com/guillaumekln/faster-whisper>.

KWAN, D.; SUN, S.; YUDITSKAYA, S. Karaoke of dreams: A multi-modal neural-network generated music experience.

LUM, C. M. **In search of a voice: Karaoke and the construction of identity in Chinese America**. [S.l.]: Routledge, 2012.

MEHENDALE, N. et al. Vocal separation using karaoke u-net. **Available at SSRN 3983514**, 2021.

NEWMARCH, J.; NEWMARCH, J. Ffmpeg/libav. **Linux sound programming**, Springer, p. 227–234, 2017.

PANTOS, R.; MAY, W. **HTTP live streaming**. [S.l.], 2017.

PATEL, P. et al. Karaoke generation from songs: recent trends and opportunities. In: IEEE. **2022 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference (APSIPA ASC)**. [S.l.], 2022. p. 1238–1246.

PRATAP, V. et al. Mls: A large-scale multilingual dataset for speech research. **arXiv preprint arXiv:2012.03411**, 2020.

RADFORD, A. et al. Robust speech recognition via large-scale weak supervision. In: PMLR. **International Conference on Machine Learning**. [S.l.], 2023. p. 28492–28518.

RAFII, Z. et al. Musdb18-a corpus for music separation. 2017.

ROUARD, S.; MASSA, F.; DÉFOSSEZ, A. Hybrid transformers for music source separation. In: IEEE. **ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)**. [S.l.], 2023. p. 1–5.

SODAGAR, I. The mpeg-dash standard for multimedia streaming over the internet. **IEEE multimedia**, IEEE, v. 18, n. 4, p. 62–67, 2011.

STÖTER, F.-R.; LIUTKUS, A.; ITO, N. The 2018 signal separation evaluation campaign. In: SPRINGER. **Latent Variable Analysis and Signal Separation: 14th International Conference, LVA/ICA 2018, Guildford, UK, July 2–5, 2018, Proceedings 14**. [S.l.], 2018. p. 293–305.

SUBHASH, V. Using ffmpeg filters. In: **Quick Start Guide to FFmpeg: Learn to Use the Open Source Multimedia-Processing Tool like a Pro**. [S.l.]: Springer, 2023. p. 83–117.

TACHIBANA, H. et al. A real-time audio-to-audio karaoke generation system for monaural recordings based on singing voice suppression and key conversion techniques. **Journal of Information Processing**, Information Processing Society of Japan, v. 24, n. 3, p. 470–482, 2016.

WILLIAMS, R. **Culture and materialism**. [S.l.]: Verso Books, 2020.

ZHOU, X.; TAROCCO, F. **Karaoke: The global phenomenon**. [S.l.]: Reaktion Books, 2013.