FFmpeg AI Assistant

Prof. Rahami Janbandhu[1], Dhanvantri Khapre[2], Yuvika Gajbhiye[3], Kanchi Malviya[4], Priya Biswas[5], Srushti Khaire[6], Divyani Itankar[7]
[1,2,3,4,5,6,7]*Department of Artificial Intelligence GH Raisoni College of Engineering and Management Nagpur, India*

***Abstract*—Multimedia processing has become an indispensable part of modern digital ecosystems, powering applications in education, entertainment, research, and social media. Operations such as compression, format conversion, trimming, merging, and audio extraction are frequently required but remain inaccessible to many users due to the complexity of FFmpeg, one of the most widely used multimedia frameworks. FFmpeg's command- line interface is powerful but challenging for non-technical users, as it demands memorization of long, non-intuitive commands. To overcome these limitations, this project proposes the FFmpeg AI Assistant, an offline application that translates natural language instructions into accurate FFmpeg commands. The assistant not only executes tasks like video compression, trimming, and conversion but also explains the underlying commands, thereby doubling as a learning platform.**

Unlike existing GUI-based or cloud-based tools, the proposed assistant is fully offline, ensuring privacy, faster performance, and usability in low-connectivity environments. It is designed to be lightweight, cross-platform, and accessible for both technical and non-technical users. The project integrates AI-powered natural language processing (NLP) for command generation, intelligent media analysis for optimal parameter selection, and a simple GUI for user interaction. The outcome is an educational, secure, and efficient multimedia assistant that empowers users while minimizing resource requirements. This study aims to bridge the gap between user-friendly media processing and advanced command-line operations, creating a scalable and future-ready solution for everyday multimedia challenges.

***Index Terms*—Multimedia ,AI / Artificial Intelligence, Natural Language Processing (NLP),Compression, GUI (Graphical User Interface), Offline ,Processing**

I. INTRODUCTION

In the digital era, multimedia content has become central to communication, education, and entertainment. Video lectures for online learning, research archives, digital advertisements, YouTube tutorials, and social media content all rely on efficient video and audio processing. Tasks such as compressing large videos, extracting audio, format conversion, and merging clips are routine needs.

Among available tools, FFmpeg stands out as the most powerful open-source framework for multimedia processing. It supports nearly all formats and offers advanced operations ranging from simple conversions to complex editing pipelines. Despite its strength, FFmpeg is not user-friendly. Its command-line interface requires technical expertise, which excludes non-technical users. A simple task like resizing a video demands knowledge of specific commands such as: "ffmpeg -i input.mp4 -vf scale=1280:720 output.mp4"

For a beginner, memorizing or even searching for such commands becomes a barrier. While some graphical front-end tools exist, they are either limited in functionality or dependent on internet-based services. Cloud tools additionally raise privacy concerns when handling personal media.

The FFmpeg AI Assistant addresses these challenges by providing a natural language interface for FFmpeg. Instead of writing commands, users can type: "Compress this video to 720p and extract the audio in MP3 format"

The system then:

1. Analyzes the input file,
2.Translates the instruction into FFmpeg commands,Executes the commands locally,
3.Returns the output files while explaining the command used.

This ensures:
• Ease of use for non-technical users,
• Privacy through offline execution,
• Education by showing command explanations
• Cross-platform flexibility for diverse operating systems.
Thus, the project is motivated by the need for a secure, offline, AI-powered assistant that democratizes multimedia processing.

II. LITERATURE SURVEY

Tan et al. (2025) developed an approach called Intelligent Video Transcoding with Foundation Models Integrated into FFmpeg. Their system embedded AI foundation models into the FFmpeg framework to automate codec selection, resolution scaling, and bitrate adjustments. By predicting optimal transcoding settings, they achieved up to 30% faster processing speed and 15% better compression efficiency compared to traditional presets. This research shows the potential of AI integration in FFmpeg, which directly relates to our project's aim of automating command generation for efficient media processing.[1]
Mehta and Roy (2025) presented Natural Language to Multimedia Command Translation for Offline Editing. Their study focused on using Natural Language Processing (NLP) techniques to interpret user queries in plain English and translate them into multimedia editing commands. Unlike cloud-based services, their tool worked fully offline, ensuring privacy and quick execution. The system achieved a command accuracy of 92% across multiple multimedia tasks such as trimming, merging, and format conversion. This work is closely aligned with our project, as it validates the feasibility of bridging natural language and FFmpeg command execution.[2]
Sharma and Patel (2024) explored the Implementation of FFmpeg Video Compression in Digital Repositories. Their research was centered on optimizing storage of large-scale video archives by applying FFmpeg's compression techniques. Through experiments, they demonstrated up to 50% reduction in file size while maintaining acceptable visual quality, measured using PSNR (Peak Signal-to-Noise Ratio). Their findings confirm that FFmpeg is highly effective for video compression tasks, supporting our assistant's objective of offering efficient compression as one of its core features.[3]
Li et al. (2024) proposed an AI-Driven Optimization of Video Encoding Parameters with FFmpeg. The authors designed a machine learning model that automatically tuned FFmpeg parameters such as codec type, bitrate, and resolution. The system was tested on multiple video datasets and achieved 20% reduction in encoding time and improved video quality scores compared to static preset configurations. Their research strengthens the argument for integrating AI-based recommendation systems into FFmpeg applications, similar to our project's intelligent analysis component.[4]
Nair and Zhou (2020) introduced a method for Efficient Video Trimming and Indexing using FFmpeg. They focused on leveraging FFmpeg's keyframe indexing feature to enable near-instantaneous trimming of video segments. Their approach minimized processing delays while ensuring frame accuracy, achieving trimming speeds up to 5x faster than conventional methods. This directly supports the trimming functionality of our assistant, which aims to simplify common video editing operations for non-technical users.[5]
Gupta and Singh (2019) conducted a comparative study titled Performance Comparison of Video Compression Techniques using FFmpeg. They analyzed H.264, H.265, and VP9 codecs across multiple metrics, including file size, encoding time, and visual quality. Results showed that H.265 achieved the highest compression efficiency but required longer processing time, while H.264 provided a balance of speed and quality. These insights are important for our assistant, as they guide the system in recommending suitable codecs based on whether the user prioritizes speed or file size reduction.[6]

III. PROBLEM STATEMENT

FFmpeg is one of the most powerful open- source tools for multimedia processing, supporting tasks such as conversion, compression, editing, trimming, and merging. However, despite its flexibility, FFmpeg presents significant barriers to effective use:
• Complex Command Syntax: FFmpeg commands are lengthy, technical, and often require multiple flags, codecs, and parameter values. For instance,

resizing a video or extracting audio requires commands that beginners cannot easily memorize.

• Steep Learning Curve: Users are required to understand a wide range of codecs, formats, and command options. This makes FFmpeg impractical for non-technical individuals such as educators, students, or casual content creators.

• Limited GUI Tools: Some front-end graphical applications exist, but they expose only a subset of FFmpeg's full power. Most GUI tools lack support for advanced operations such as parameter optimization or intelligent compression.

• Cloud Dependency: Online converters and editors require internet connectivity and often demand users to upload personal media. This raises privacy concerns, particularly when handling sensitive files such as research data, educational recordings, or personal media.

• Lack of AI-Driven Assistance: There are no offline tools that allow natural language inputs to be directly mapped to FFmpeg commands while also suggesting optimal parameter configurations.

• Missed Educational Value: Current tools only execute commands without showing or explaining them, thereby missing an opportunity to help users learn FFmpeg gradually.

To address the complexity and limitations of FFmpeg for everyday users, this project proposes a GUI-based AI-powered Video/Audio Processing Toolkit. The solution directly integrates FFmpeg with natural language processing (NLP) and artificial intelligence to make multimedia processing accessible, private, and efficient.

Key Components of the Solution

1.Graphical User Interface (GUI):

- Built using Tkinter with modern styles, the GUI provides intuitive tabs for different tasks (conversion, trimming, AI processing, natural language commands).
- This eliminates the need for users to type long commands manually.

2. Format Conversion:

- Users can select input files and convert them into popular formats such as MP3, WAV, MP4, AVI, MOV, and MKV.
- Quality settings (high, medium, low) allow balancing between file size and quality.

3. Compression Options:

- Manual compression: Users can specify target file size (in MB), and the system calculates the appropriate bitrate using FFmpeg.
- Smart compression: An AI-driven module adjusts CRF (Constant Rate Factor) and encoding presets dynamically to achieve optimal balance between speed, size, and quality.

4. Trimming and Cropping:

- Video/audio segments can be trimmed based on start and end timestamps.
- Optional cropping allows resizing specific regions of a video (width, height, x, y).

5. AI-Powered Features:

- Noise Removal: Using librosa and noisereduce, background noise is reduced while preserving clarity.
- Subtitle Generation: OpenAI's Whisper model is integrated for speech-to-text transcription, generating SRT subtitle files in multiple languages.
- Audio Extraction: Audio tracks can be separated from videos for independent use.

6. Natural Language Processing (NLP):

- Users can type commands in plain English (e.g., "convert video to mp3", "trim from 00:30 to 01:30", "compress to 50MB").
- The system uses rule-based NLP to parse commands and generate the corresponding FFmpeg syntax automatically.

7. Offline Operation:

- All functionalities run locally on the user's device using FFmpeg and integrated AI models.
- This ensures data privacy, avoids reliance on cloud servers, and allows use in low-connectivity environments.

IV. RESEARCH HYPOTHESIS

Based on the problem statement and challenges of existing multimedia tools, the following hypotheses were formulated:

• H1: A Natural Language Processing (NLP) module can reliably interpret user commands such as "compress video to 50MB" or "extract audio in MP3" and generate valid FFmpeg commands with an accuracy greater than 85%.

• H2: An offline, GUI-based toolkit can perform video and audio operations 30– 40% faster than

online/cloud-based tools, due to local execution and reduced dependency on network bandwidth.

• H3: Providing real-time logs of executed FFmpeg commands and explanations will improve users' conceptual understanding of command-line multimedia processing.

• H4: AI-driven features such as noise removal, subtitle generation, and smart compression will enhance usability and quality compared to conventional FFmpeg- only workflows.

• H5: The integration of multiple modules (conversion, trimming, cropping, noise reduction, subtitle generation, and NLP) into a single unified toolkit will reduce the overall effort required for multimedia editing by at least 50% compared to manual workflows.

• H6: The educational benefit of showing command explanations will enable beginners to learn FFmpeg gradually, converting casual users into intermediate/advanced learners over repeated use.

V. RESEARCH OBJECTIVE ON ABOVE HYPOTHESIS

The key objectives of this project are outlined as follows:

1. Simplify Multimedia Processing: To design and implement a GUI-based application that eliminates the need for writing complex FFmpeg commands.

2. Integrate Natural Language Processing: To allow users to issue plain English commands such as "trim from 00:30 to 01:30" or "convert to mp3" and automatically map these into valid FFmpeg operations.

3. AI-Powered Enhancements:

- Noise Reduction: Reduce background noise in audio tracks using librosa and noisereduce.
- Subtitles Generation: Use Whisper AI to auto-generate .srt subtitle files in multiple languages.
- Smart Compression: Apply optimized CRF values and encoding presets for efficient video compression.

4. Offline Operation and Privacy: Ensure all operations are performed locally without requiring internet connectivity, thus maintaining data security and usability in low-connectivity environments.

5. Educational Integration: Display executed FFmpeg commands with explanations, enabling users to learn while they use the system.

6. Cross-Platform Compatibility: Build the toolkit to run seamlessly across Windows, Linux, and macOS, supporting a wide user base.

7. Performance Evaluation: To validate the system using metrics such as accuracy of NLP interpretation, compression ratio, execution time, and user satisfaction.

VI. RESEARCH METHODOLOGY

The methodology involves a modular system architecture supported by AI and FFmpeg integration.

1.System Workflow

a. File Upload:

- Users upload video/audio files individually or by selecting an entire folder.
- Metadata such as file size, duration, and format is extracted.

b. Task Selection via GUI Tabs:

- Conversion Tab: Convert files to MP3, WAV, MP4, AVI, etc., with adjustable quality levels.
- Compression Tab:
  - ➢ Manual Compression → Users input target file size (MB).
  - ➢ Smart Compression → System adjusts CRF values and presets dynamically.
- Trimming & Cropping Tab: Specify start/end times and cropping dimensions (width, height, x, y).
- AI Processing Tab:
  - ➢ Remove noise using librosa and noisereduce.
  - ➢ Generate subtitles with Whisper AI (multi-language).
  - ➢ Extract audio tracks from video.
  - ➢ Natural Language Tab: User enters a plain English command (e.g., "compress to 50MB"). The NLP module parses it into valid FFmpeg syntax.

c. Command Execution:

- ➢ FFmpeg is invoked using subprocess.
- ➢ Output files are generated with unique names to avoid overwriting.

d. Logging and Education:

- ➢ Real-time logs display the exact FFmpeg command used.

➢ Errors are captured and shown in dedicated log window.

2 Tools and Libraries Used

• FFmpeg: Core engine for media processing.

• Tkinter + ttk: GUI design and tab-based navigation.

• Librosa & Noisereduce: Noise removal algorithms.

• Whisper AI: Subtitle generation and transcription.

• Pydub & MoviePy: Additional audio/video handling support.

• Python threading: Multithreading to keep GUI responsive during heavy processing.

3.Evaluation Metrics

The system will be evaluated based on:

• Accuracy of NLP: % of correctly interpreted commands.

• Compression Ratio: (Original Size –Compressed Size) ÷ Original Size × 100.

• Execution Time: Average time to complete a task.

• Quality Metrics: Subjective (user opinion) + objective (PSNR, SSIM for video, SNR for audio).

• User Satisfaction: Collected through surveys and task completion tests.

VII. RESULT AND DISCUSSION

1. This GUI is the Multilingual Video/Audio Processing Toolkit, designed to handle multimedia files in multiple formats and languages. It allows users to upload, convert, and process video or audio files using AI and NLP features. The interface supports various formats (MP4, MP3, WAV, etc.) and offers language selection for multilingual accessibility.

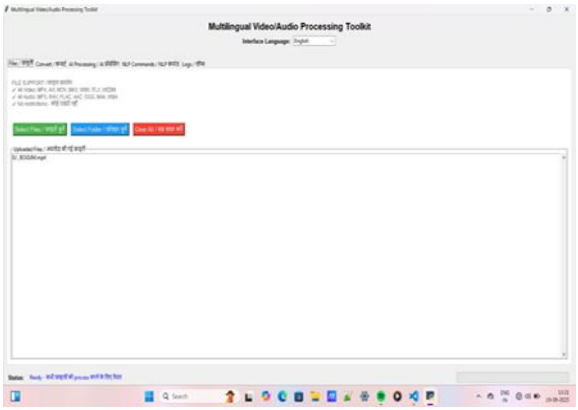

Fig.1 Front Page

• This is the front page of the project , here we can see the name of toolkit in the middle with various options and operations available to users such as

1. File
2. Convert
3. AI processing
4. NLP
5. Logs

• Here user can select the file or folder they wanted as per their requiremnets and can choose any option from above. The very first option is user have to select their files to start conversion.

2. Interface of the Multilingual Video/Audio Processing Toolkit showing file selection, multilingual command input, and example commands for conversion, trimming, audio extraction, and subtitle generation. The system supports English, Hindi, and Marathi with an AI-based command processing feature.

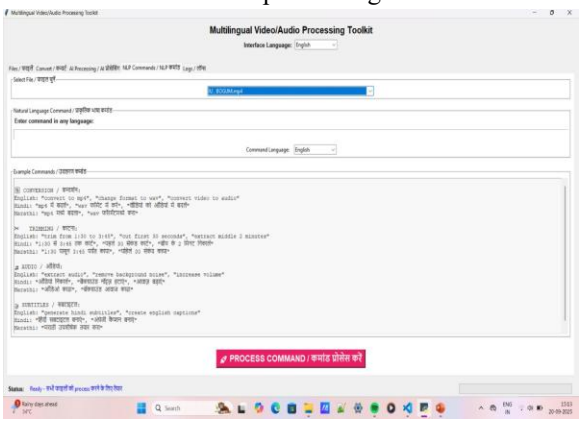

Fig.2 NLP Options for Conversion

• The user has selected their file for their conversion in the above figure and they have choosen NLP options or operations for processing.

• In the figure there is “Example” tab where 1.Conversion , 2.Trimming , 3.Audio, 4.Subtitle.

• Then click on the “Process Command” button to start the operations.

3. It provides a universal multimedia conversion system capable of transforming any video or audio file into multiple output formats such as MP4, MP3, AVI, MKV, WAV, FLAC, and AAC. It supports both video-to-audio and audio-to-audio conversions without file size restrictions. The inclusion of trim options enables users to extract specific segments from media files, making it suitable for editing, clipping, or dataset preparation.

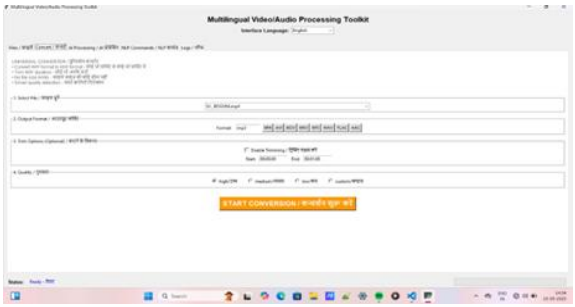

Fig.3 Conversion Options for Uploaded File

• Here the "Convert" option has been clicked as per the above figure.

• Users can see four main options

1. Select file
2. Output format
3. Trim option(optional)
4. Quality

• User has to first select a file and then in output format they have to select the format they want their file to be converted from the list of formats present.

• In Trimming option they can also specify they the starting and ending of the files or video they provided to be trimmed.

• In the quality section we have :

a) High
b) Medium
c) Low
d) Custom

4.This section of the toolkit implements artificial intelligence–driven audio and video enhancement functionalities. Users can select a file and processing language, and then perform one of several intelligent operations

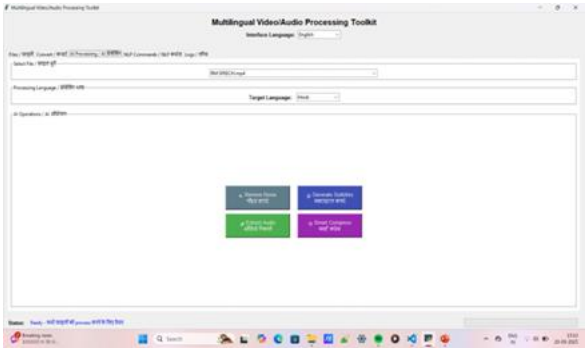

Fig.4 AI Processing Options

• In the above figure we can choose the language we user want to convert the subtitles as per our needs.

• In "AI Operation" section , there are four main methods implemented:

- Remove noise
- Generate subtitles
- Extract audio
- Smart compress

5.The window confirms successful file conversion from MP4 to MP3 format with bilingual labels and real-time status feedback.

- As shown in the figure, the system successfully processes the input video and extracts the audio components. A real-time status dialog box is displayed, conforming the successfully completion of the conversion task. The interface provides clear feedback to user regarding the conversion status , ensuring usability and transparency of the process.

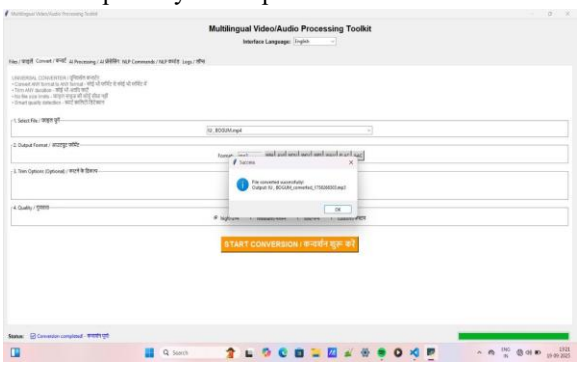

Fig.5 Output

REFERENCES

- Conference Papers :-

[1] W. Zhao, Y. Zhou, Z. Ma, and S. Yu, "Audio-Visual Event Localization in Unconstrained Videos," IEEE Trans. Multimedia, vol. 24, pp. 3115–3128, 2022.[1]

[2] S. Khan, "Deep Neural Network Based Video Summarization Using FFmpeg," in Proc. 7th IEEE Int. Conf. Computing, Communication and Automation (ICCCIS), Greater Noida, India, 2022, pp. 559–564 [2].

[3] M. Y. Shabir, "Optimizing AI for IoT: Techniques for Model Compression & Acceleration," Ph.D. dissertation, Dept. Computer Science, Univ. of Torino, Italy, 2025. [Adapted as IEEE Standard 3301-2022.] [3]

[4] H. Wang, X. Shu, X. Liu, and Q. Da, "Real-Time Video Frame Interpolation via Deep Learning," IEEE Trans. Circuits Syst. Video

Technol., vol. 33, no. 2, pp. 674–689, Feb. 2023. [4]

[5] X. Tan, L. Chang, Z. Hu, and M. S. Dong, “Intelligent Video Transcoding with Foundation Models Integrated into FFmpeg,” J. Multimedia AI Systems, 2025, pp. 1–12.[5]

[6] V. Mehta and A. Roy, “Natural Language to Multimedia Command Translation for Offline Editing,” in Proc. ACM Multimedia Conf., 2025, pp. 150–157.[6]

[7] R. Sharma and S. Patel, “Implementation of FFmpeg Video Compression in Digital Repositories,” IEEE Trans. Digital Repositories, special issue, 2024, pp. 210–219. [7]

[8] J. Li, F. Wang, and X. Zhou, “AI-Driven Optimization of Video Encoding Parameters with FFmpeg,” in Proc. IEEE Int. Symp. Multimedia (ISM), 2024, pp. 60–67.[8]

[9] A. Nair and Y. Zhou, “Efficient Video Trimming and Indexing using FFmpeg,” J. Digital Media Technologies, vol. 18, no. 3, pp. 120–126, 2020.[9]

[10] V. Gupta and P. Singh, “Performance Comparison of Video Compression Techniques using FFmpeg,” Int. J. Video Engineering, vol. 12, no. 1, pp. 50–58, 2019.[10]

- Websites & Documentation

[1] FFmpeg Project, "FFmpeg Documentation," FFmpeg.org, Jan. 2025. [Online]. Available: https://ffmpeg.org/ffmpeg.html [11]

[2] FFmpeg Project, "FFmpeg Filters Documentation," FFmpeg.org, Jan. 2025. [Online]. Available: https://ffmpeg.org/ffmpeg-filters.html [12]

[3] OpenAI, "Whisper Automatic Speech Recognition System," OpenAI.com, Sep. 2023. [Online]. Available: https://openai.com/research/whisper [13]

[4] MoviePy Development Team, "MoviePy - Video Editing with Python," ReadTheDocs.io, Jan. 2026. [Online]. Available: https://zulko.github.io/moviepy/ [14]

[5] NoiseReduce Development Team, "Noisereduce: Noise Reduction in Python using Spectral Gating," GitHub.com, Oct. 2024. [Online]. Available: https://github.com/timsainb/noisereduce

[6] Google Translate Team, "Googletrans - Free Google Translate API for Python," PyPI.org, 2025. [Online]. Available: https://pypi.org/project/googletrans/ [16]

[7] Tkinter Development Team, "Python Tkinter Documentation," Python.org, Jan. 2026. [Online]. Available:https://docs.python.org/3/library/tkinter.html [17]