***Hands-Free Video Player: Enhancing Accessibility with Voice-Controlled Navigation***

**V.Karthika1., Mr.A.Siva Ganesh2.,**

1Department of MCA,

2Assistant Professor, Department of MCA,

1,2 Mepco Schlenk Engineering College, Sivakasi– 626124

1vkarthika2892\_mca25@mepcoeng.ac.in

2 sivaganesha@mepcoeng.ac.in

**Abstract**

This paper presents a technology-centric approach to improve Over-The-Top (OTT) services for Smart TVs through the use of advanced speech recognition, video analysis and natural language processing tools. The system integrates TransNetV2 for AI-based scene boundary detection, Porcupine for hotword detection, and state of the art Automatic Speech Recognition (ASR) engines with Vosk, Whisper and DeepSpeech for real-time speech to text conversion. Natural Language Processing (NLP) incurs the use of BERT and spaCy to determine user intent and time-based commands from spoken directives. Videos are ingested and processed using FFmpeg and OpenCV for frame manipulation and visualization, and create Intelligent content classification and scene understanding with YOLO and ResNet. The platform is established and engineered with Flutter (to include a shared codebase cross-platform to all Smart devices), as well as a Python Flask backend for module cohesion and functionality. Once developed, the system demonstrates the ability to perform real-time, hands-free media control and allow an intuitively heavier and softer way of engagement for the consumer experience for modern OTT applications.

**Keywords:** *Voice Control, Speech Recognition, Natural Language Processing (NLP), Automatic Speech Recognition (ASR), Hands-Free Video Playback, Voice*

*Commands, Media Accessibility, AI-Powered Navigation, Video Processing, Smart Interaction, Real-Time Playback Control*

**1. Introduction**

Voice-activated systems serve as alternatives to the traditional control of video playback through manual means, with the goal of enhancing accessibility and convenience. This study provides a voice-activated video playback system that utilizes speech recognition and natural language processing (NLP) technology to support seamless interaction. Users demand seamless control over video playback using simple spoken words such as "play," "pause," "rewind 1 min," "skip 10 sec," etc. Similarly, while cooking, the user can command the system to "rewind 30 sec" to return to a step in the video instructions without touching the device. This system helps with media accessibility, assistive technology, hands-free entertainment, and compliance temporarily. By employing the use of Automatic Speech Recognition (ASR) technology and AI-enabled NLP models, it enables the successful command and execution in real time and with accuracy. This study reflects on the design, implementation, and associated uses of this system to demonstrate its effectiveness in today's modern machine-interactive smart environments.

* 1. **Evolution of Voice-Controlled Media Playback**

The idea of voice-controlled interfaces came into being with the improvement of speech recognition and artificial intelligence (AI) technology. First, voice instructions were limited to basic keyword identification, which was often resulting in errors and misunderstandings. But with Automatic Speech Recognition (ASR) and Natural Language Processing (NLP) joined together, recent systems can currently process complex voice inputs, detect natural speech behavior, and ascertain human intent very accurately. This has brought a massive change in media interaction, which is now more responsive and user-centric. Technology giants such as Google, Amazon, and Apple have led this industry with voice assistants such as Google Assistant, Alexa, and Siri, demonstrating the feasibility of hands-free interaction in day-to-day applications.

* 1. **Importance of Hands-Free Interaction in OTT Platforms**

Platforms like Netflix, Amazon Prime Video, and Disney have huge libraries of content, making it more difficult to negotiate, and for users to find a specific scene, jump to a different part of the content, or control playback options without compromising their viewing experience. Hands-free navigation improves accessibility, convenience, and efficiency for: People with Disabilities: Individuals with limited movement capabilities can utilize voice-controlled assets to navigate OTT applications. Multitaskers: Users can send voice commands while cooking, working out, or working without holding a remote. Smart Home Control: Users can interact with OTT platforms that are attached to voice-controlled platforms, to manage everything seamlessly and evenly throughout their home in a more exciting way.

**2. Literature Survey**

[1] This paper proposes a CNN-based Streamed Speech Command Recognition (SSCR) system using a custom crowdsourced dataset. The model is optimized for low memory footprint, enabling on-premise deployment. A Voice Controlled Media Player (VCMP) demonstrates real-world applicability, with testing revealing key insights on performance. The work highlights edge-compatible speech recognition for IoT applications.

[2] This research combines ultrasonic hand gesture recognition and voice commands for device control. Trilateration and Leap Motion sensors enable 3D spatial tracking, while voice recognition supplements interaction. The system aims for intuitive, touch-free media control, particularly in smart environments.

[3] Google’s voice search development focuses on ubiquity and performance, addressing challenges in speech recognition accuracy and user interface design. The study notes rapid adoption of voice search, emphasizing repeat user engagement as a success metric.

[4] A C#-built media player uses voice recognition libraries to assist users with disabilities. The system replaces keyboard/mouse inputs with wireless voice commands, enhancing accessibility. The work underscores voice technology’s role in inclusive design.

[5] This paper examines voice technology adoption among people with intellectual disabilities, highlighting barriers like typing dependency. It advocates for voice interfaces to improve digital accessibility and independence.

[6] "Pepper Media Player" is a Java-based, cross-platform application supporting MP3/MP4 playback via voice commands. The wake word "HELLO" initiates interaction, aiding visually impaired users. The system eliminates reliance on physical input devices.

[7] Avuçlu et al. present a C#-implemented media player designed specifically for users with disabilities, enabling complete control through voice commands. The system utilizes existing voice recognition libraries to replace traditional input methods, allowing operation via wireless headsets. This work primarily focuses on accessibility applications, demonstrating how voice technology can empower users with physical limitations. The implementation shows practical deployment of basic speech recognition for media control but lacks advanced AI features.

 [8] A React and Alan Studio-based web app delivers news via voice interactions, integrating News/Weather APIs. The system reduces manual effort, offering hands-free access to categorized news. Targets time-constrained users preferring auditory consumption.

[9] Vosk provides offline speech recognition with low-latency performance, supporting multiple languages and platforms. Its modular architecture enables efficient deployment on edge devices and embedded systems. The toolkit uses deep neural networks for accurate transcription without cloud dependencies. Vosk's API flexibility makes it suitable for real-time voice-controlled applications.

[10] Porcupine offers highly accurate wake-word detection with minimal computational overhead, supporting custom wake-word creation. Its lightweight design (sub-100KB memory footprint) enables on-device processing for privacy-sensitive applications. The engine supports multiple platforms including Raspberry Pi and mobile devices. Porcupine's low false-alarm rate makes it ideal for always-listening voice interfaces.

[11] FFmpeg provides comprehensive cross-platform solutions for video/audio processing, including decoding, encoding, and transcoding. Its modular architecture supports hundreds of codecs and formats through libavcodec. The toolkit enables frame-level manipulation and streaming capabilities for media applications. FFmpeg's CLI utilities and libraries are widely used in commercial and open-source projects.

[12] OpenCV delivers optimized implementations of 2500+ computer vision algorithms for real-time image/video analysis. The library supports deep learning inference through integration with TensorFlow and PyTorch. Its cross-platform nature (C++, Python, Java APIs) facilitates deployment from embedded to cloud systems. OpenCV's GPU acceleration enables high-performance video processing pipelines.

[13] Flutter enables natively compiled multi-platform applications from a single codebase using Dart programming language. Its widget-based architecture provides pixel-perfect rendering across iOS, Android, and web platforms. The framework's hot reload feature accelerates development cycles significantly. Flutter's rich plugin ecosystem supports integration with native platform features.

[14] Flask offers minimalist Python web development with extensible microservices architecture. Its Jinja2 templating engine and Werkzeug WSGI toolkit provide core web functionality without bloat. The framework's modular design allows seamless integration with AI/ML backends. Flask's RESTful request dispatching is ideal for building lightweight API servers.

[15] TensorFlow provides end-to-end machine learning workflows from model training to deployment across diverse hardware. Its high-level Keras API simplifies neural network development while supporting custom low-level operations. The framework's TensorFlow Lite variant enables efficient on-device inference. TensorFlow's ecosystem includes tools for visualization (TensorBoard) and production serving.

[16] The video\_player plugin delivers performant inline video playback integrated with Flutter's widget tree. It supports platform-native players on iOS (AVPlayer) and Android (ExoPlayer) with unified Dart API. The plugin enables custom video controls and subtitle rendering. Its texture-based implementation ensures smooth frame synchronization with Flutter's rendering pipeline.

[17] Porcupine's SDK provides C++/Python bindings for embedding wake-word detection in custom applications. The SDK includes pre-trained models for common wake words and tools for custom model training. Its real-time processing capabilities achieve <100ms latency on commodity hardware. The SDK's thread-safe design supports integration with complex multimedia systems.

[18] Souček and Lokoč present TransNet V2, an improved deep learning architecture for efficient shot boundary detection in videos. The model achieves real-time performance while maintaining high accuracy on standard benchmarks. It introduces novel temporal modeling techniques for better transition classification. The work provides practical insights for video indexing and content analysis applications.

[19] Soni investigates accuracy improvements in the Vosk speech recognition system through domain-specific language model adaptation. The study demonstrates significant WER reduction when using tailored linguistic models for specialized vocabularies. The paper provides methodologies for optimizing open-source ASR systems. Results show particular benefits for technical and medical terminology recognition.

[20] Al-Hajri et al. develop a novel visualization system for personal video browsing history using temporal thumbnails. The interface enables non-linear navigation through watched content with spatial memory cues. User studies demonstrate improved seek-time performance compared to traditional timelines. The work pioneers user-centric video interaction paradigms for media players.

**2.1. Introduction to Voice-Controlled Video Playback**

Voice-activated technologies are changing how we engage with media, providing convenient and hands-free interaction. Conventional video watching involves manual controls, which can be challenging when impaired or when multitasking. Sophisticated AI (ASR solutions like Whisper and DeepSpeech), and NLP (like BERT and GPT) enable real-time, serverless verbatim transcription with exceptional accuracy. Users might even verbally issue basic natural commands like "skip forword 10 seconds" or "rewind 1 minute" to give the technology basic control over video. Meanwhile, the use of OpenCV, FFmpeg, and other video segmentation software, improves scene detection and intelligent navigation—this is combined with powerful edge computing on mobile devices, which ensures offline low-latency counting. This paper investigates combining voice or speech recognition and natural language processing and visual processing for an effective, voice-controlled media experience. By incorporating this technology, smart media applications will be improved in terms of automation, usability, and accessibility.

**2.2. Existing Systems and Their Limitations**

Voice-controlled media systems like Google Assistant, Alexa, and Siri are pretty handy for basic tasks, letting you say things like "pause" or "play." However, they really struggle when it comes to understanding more complex commands or the context behind what you're asking. For instance, if you want to "skip the intro" or "rewind the last conversation," these systems often can't keep up. They also don't allow for personalized or dynamic commands, which can be a bit frustrating. When it comes to navigating through content, they usually stick to fixed timestamps and don't recognize scene changes or the meaning behind the content. Plus, they miss out on using AI for video content analysis, which means you can't easily skip over ads or specific action scenes. Most existing media players, built with languages like C# or Java, or using basic ASR tools like Vosk and Porcupine, are limited to simple recognition and manual controls. They also depend heavily on a constant internet connection for cloud processing, which raises privacy issues and limits offline use. On top of that, these systems are often designed for specific platforms or just for TV interfaces, leaving mobile devices and cross-platform integration in the dust. Because of all this, current OTT media solutions really don't deliver the personalized, intelligent, and hands-free experiences that users are looking for in today's smart ecosystems.

**2.3. Role of ASR and NLP in Voice-Controlled Systems**

The proposed voice-based OTT system relies on Automatic Speech Recognition (ASR) and Natural Language Processing (NLP) to facilitate interaction between users and the computing device. Specifically, the system uses Vosk, an efficient offline ASR engine that provides near real-time transcription of spoken user commands into text. Vosk is optimized for multiple languages and works offline on the user's device, which is especially important for privacy-sensitive technologies on devices, such as a smart television. For wake word detection (i.e., "Hey Google"), Porcupine is used to continuously listen for the wake phrase so that the system only becomes active when the user chooses to engage it. Once ASR has transcribed the spoken command into text, NLP techniques will be used to parse and understand the intent behind the command. The intent can include keywords that reference a temporal command (i.e., "skip forward 30 seconds!") or commands that reference content (i.e., "go to the fighting scene"). Each interpreted intent will then be mapped to an action through a pre-established command-action mapping system. The combination of ASR and NLP are an integral part of the voice-based interaction loop which gives users accurate, responsive, and intuitive control over content playback and navigation.

**2.4. Video Content Analysis and Scene Detection**

The suggested OTT platform contains sophisticated video content analysis, which will allow for smart scene skipping, as well as other navigation beyond simple time controls. The heart of this scene skipping is TransNetV2, a deep learning scene boundary detection model that focuses on semantic scene transitions across a sequence of video frames. TransNetV2 processes consecutive frames and generates a timestamp at which the semantic transition occurred in the video process. This timestamp can be called on to allow the user to skip the entire scene or jump to sections of scenes that are action-sequence-heavy, dialogue-heavy, etc., based on voice command. All video data has been pre-processed with OpenCV to extract each frame efficiently during playback. FFmpeg is being used to decode and segment the media stream. OpenCV and FFmpeg work seamlessly in an asynchronous manner to isolate key moments of the scene in real time while not interrupting playback. Additionally, lightweight content classification heuristics are being applied to classify scenes by type (e.g., fight, romantic, dramatic, etc.) to provide context-aware navigation. As designed, the approach intelligently allows skipping, which ultimately minimizes search effort and enhances user experience through consumption that matches intent and preference.

**2.5. How it’s differed from the other video player**

Typically, the feature of a voice command or manual interaction defines a video player, but there has been actual intelligent and context-aware interaction with the media with our system. It combines various sophisticated techniques: an advanced speech recognition system based on Vosk, Whisper, and DeepSpeech; hotword detection with Porcupine; and natural language processing supported by BERT and spaCy. The features enable comprehension of scenarios such as "skip the intro" and "replay the last dialogue." Unlike most of the other players, which can use simple seek commands, our player can dynamically identify TransNet V2 scene boundaries while the AI-based YOLO and ResNet models classify casual videos. Most of the existing players restrict themselves to be deployed on Smart TVs or limited on the platforms of mobiles. The solution developed for this is a fully cross-platform Flutter-Python Flask application that supports everything-including TV, mobile, and web-tethering across platforms for a single experience. In debate, it has included hands-free and real-time interactivity without relying heavily on cloud services. Thus, it offers superior privacy, offline operation, and personalization above any other player of videos.

**2.6. Challenges in Real-Time Video Navigation**

One of the biggest challenges in implementing real-time, offline voice-controlled video navigation is achieving low-latency response while maintaining high accuracy. Edge computing and on-device ASR models are emerging as potential solutions, but most existing research focuses on cloud-based architectures, limiting usability in offline environments.

**3.Proposed System**

**3.1 System Architecture**

The architectural framework of the proposed OTT platform is intended to integrate technologies that support the warm user experience across smart TVs. The higher-level components consist of a Real-Time Voice Control Module, AI-Based Video Segmentation, as well as an Interactions Layer. The Voice Control Module uses Porcupine for its hotword function, and Vosk for the speech recognizing function (e.g., commands such as "skip forward 30 seconds" or "go to the fight scene"). After this, the command is forwarded to the Intelligent Navigation System that uses TransNetV2 for scene boundary recognition, to understand where to go within the media based on the spoken command. This process circumvents any subjective navigation or feedback during the experience. The system incorporates separate frame extraction algorithms (OpenCV, FFmpeg, etc.) to extract each frame, and emphasizes real-time video interaction and analysis. All of these technological components are implemented using Flutter, providing the ability to deploy on smart TVs running Android TV, Fire TV, Tizen, webOS and Apple TV. The framework is established so that each component can operate independently and communicate smoothly with the hotword processing & voice recognition module, video segmentation, and playback system.

**3.2 Workflow**

The system operates by the Voice Control Module starting by listening for hotwords via Porcupine, causing the system to only enter an active state when the user speaks the phrase "Hey Google". Once in this state, the system listens for a command, which the Vosk system will then recognize and return as a string. The recognized command is then processed by the Natural Language Processing (NLP) engine to extract the user's intent, such as to skip forward at a specific time, or to navigate to a specific scene. Once the intent has been processed, the Intelligent Scene Navigation calls on the TransNetV2 algorithm to detect the scene boundaries, and the OpenCV and FFmpeg modules will process the video stream to adjust the playback to reflect the user’s intent. For example, when the user says a command such as "skip forward 30 seconds", the video will be analyzed for the correct amount of time to skip, then will resume play. The output will be displayed on the smart TV, creating the experience that interactions with the system and navigation through the content occurs in a natural, quick manner. Flutter acts as the user interface, controlling UI updates and communication with all of the modules to allow for a smooth experience across all devices. Fig.3.2: System workflow diagram shows the System workflow

 

**Fig. 3.2. System workflow Diagram**

**4. Implementation Methodology**

The proposed OTT platform will require voice control, AI-based scene detection, and real-time video processing capabilities, integrated into a single system built for smart TVs. The development process is organized around modularity, with independent components dedicated to voice recognition, video segmentation, video playback controls, and UI rendering, that communicate to one another using an established interface. The backend consists of the machine learning models and multimedia processing tools, while the frontend will be built with Flutter in order to maintain a consistent deployment on multiple platforms. Overall, this modular and layered approach to the software architecture provides for an expandable and maintainable system in the future that can be modified to include, for example, gesture recognition, personalized scene tagging, etc.

**4.1 Voice Control and Command Interpretation**

The speech interaction part starts with Porcupine, a low-power on-device hotword detection engine, which is always listening for the hotword "Hey Google." After the hotword is detected, the Vosk speech recognition engine will transcribe the user’s voice command in real time. The transcriptions are passed to a lightweight Natural Language Processing (NLP) layer to determine intent (for example, to skip forward 10 seconds or to access a semantic content, such as to go to the fighting scene). Once the commands are interpreted, these commands are mapped to corresponding functionality in the video player which may include navigating to a scene or skipping time. The interaction allows the user to communicate naturally and hands-free to the application, with low-latency and accuracy, even when offline.

**4.2 Scene Detection and Video Processing**

The process segmenting scenes in video is performed by TransNet V2, a deep learning model that has been trained to accurately detect the boundaries between scenes. OpenCV was used to process the video stream and extract a video frame, and FFmpeg was used to decode the video and conduct media transitions. TransNet V2 processes the frame transitions and detects scene transitions which are meaningfully distinguishable in context to the video being watched - These transitions of meaning are grouped with timestamps as scene boundaries. These boundaries allow the system to use all the frames to create a scene map of the video. This scene map will be referred to in response to user navigation requests. When a user requests to jump to a particular scene type (ex: "fight scene"), we use a pre-labelled index(s) or utilize AI classification and can seek to the relevant segment from the transcription produced previously. The entire operation has been performed as effectively as possible producing results which can work on resource constrained smart TV devices.

**5. Results**

The combined voice-controlled OTT system enhances content navigation on standard smart TVs. The time from speech to action (on screen) is responsively fast, creating a seamless experience. Additionally, users can control playback using basic voice commands that remove the need for remotes. The voice recognition engine was able to accurately understand common playback directions, such as “skip forword,” “rewind 10 seconds,” and “go to the action scene.” The scene detection module is able to accurately divide different segments in video content medial, such as intro, ads, and action scenes, which can enable smart content skipping and content-aware navigation. During user testing, verticals indicated they could navigate video content easily and effectively, and consistently could navigate faster than with remotes. The system behaves very naturally, regularly supports context-driven playback, and supports hands-free and smart video general experience overall. Fig.: 5.1., 5.2. ,5.3. are the screenshots of the output.



 **Fig. 5.1. Hotword Detection**

  

 **Fig. 5.2. Listing Commands ‘Play’ and ‘stop’**

 

**Fig. 5.3. Listing Commands ‘skip forward’ and ‘rewind’**

**6. Conclusion**

The project offers a new and useful way to improve user experience within OTT platforms using innovative artificial intelligence and voice recognition capabilities. The platform aims to rethink the viewing experience and improve interactions with the content by avoiding drawbacks of traditional linear playback systems, especially in smart TV environments where touch interaction is unachievable or remote-based content browsing is inefficient and boring. Utilizing voice-controlled functionalities through Vosk and Porcupine allows for a flat and intuitive model of user interaction without the process of hands-on interaction. A user can say commands these will allow them to control playback intuitively create playback commands such as "Go to the climax" or "Rewind 30 seconds", rendering the user experience dramatically more accessible, especially with additional support for visually impaired users, or personal preference for a quicker alternative to navigate through inherently complex or time-consuming content. TransNetV2 is used for scene segmentation and brings intuitive content understanding and intelligence to the vast understanding of video. It broadly and intuitively segments videos into discrete units of meaning, objects and/or events occurring within action, dialogue and advertisement. The identified segments through the transitions are visualized and integrated into the video player with labelled segments extracted using FFmpeg and OpenCV. This functionality enables a user to say something like "rewatch that encase there was something I missed" and jump to the next slice of meaningful human experience by avoiding or skipping fragmented moments of the video replay and enhance personalization markers. Flask architecture is the backend server that enables components of the project to communicate smoothly and allows for a seamless experience for users. The frontend framework using Flutter, will be cross-platform to TV, as well as mobile devices. Thorough testing shows the system features low latency and high usability, confirming its real-time performance and efficiency. The end-to-end pipeline from hotword hearing to playback execution is reliable, scalable, and adaptable for future improvements. This establishes a strong base for the next generation of OTT applications that are smarter, more equitable, and better able to respond to user intent. In summary, this research represents a significant advancement in the overlap of AI, speech processing and human-computer interaction. Future research will build on this architecture to consider semantic scene understanding, supporting multi-languages and GPU based acceleration, thereby further bridging the gap between AI capability and user context in modern day entertainment ecosystems.

**7. References**

[1] A. K. Singh et al., "Voice Controlled Media Player: A Use Case to Demonstrate an On-Premise Speech Command Recognition System," Adv. Signal Process. Intell. Recognit. Syst., Springer, 2020.

[2] Raj, "Control Video With Gesture and Voice Recognition," *IJCSP*, 2024.

[3] M. Bacchiani et al., "Your Word is My Command: Google Search by Voice," Adv. Speech Recognit., Springer, 2010.

[4] E. Avuçlu et al., "An Application to Control Media Player with Voice Commands," J. Polytech., vol. 23, no. 4, pp. 1311–1315, 2020.

[5] R. A. Dhukate et al., "A Study on Use of Voice Control in Day to Day Life," Dnyanasadhana Coll., 2021.

[6] J. A. R. and J. A. B., "Media Player Using Voice Recognition," *IJETIE*, vol. 5, no. 6, 2019.

[7] E. Avuçlu, A. Özçifçi, and A. Elen, "An Application to Control Media Player with Voice Commands," J. Polytech., vol. 23, no. 4, pp. 1311-1315, Jan. 2020.

[8] D. A. Pathrabe et al., "Conversational Voice Controlled News Application," *NIELIT Aurangabad*, 2022.

[9] "Vosk Speech Recognition Toolkit," Alphacephei, 2020. [Online]. Available: <https://alphacephei.com/vosk/>

[10] "Porcupine: Wake Word Engine," Picovoice, 2025. [Online]. Available: <https://picovoice.ai/products/porcupine/>

[11] "FFmpeg," FFmpeg Developers, 2025. [Online]. Available: <https://ffmpeg.org/>

[12] "OpenCV: OpenSource Computer Vision Library," OpenCV Team, 2025. [Online]. Available: <https://opencv.org/>

[13] "Flutter: UI Toolkit," Flutter Team, 2025. [Online]. Available: <https://flutter.dev/>

[14] "Flask: Web Development," Python Software Foundation, 2025. [Online]. Available: <https://flask.palletsprojects.com/>

[15] "TensorFlow," TensorFlow Authors, 2025. [Online]. Available: <https://www.tensorflow.org/>

[16] "video\_player: Flutter Plugin," Flutter Team, 2025. [Online]. Available: <https://pub.dev/packages/video_player>

[17] "Porcupine SDK," Python Software Foundation, 2025. [Online]. Available: <https://github.com/Picovoice/porcupine>

[18] T. Souček and J. Lokoč, "TransNet V2: An Effective Deep Network Architecture for Fast Shot Transition Detection," arXiv:2008.04838, 2020.

[19] A. A. Soni, "Improving Speech Recognition Accuracy Using Custom Language Models with the Vosk Toolkit," arXiv:2503.21025, 2025.

[20] A. Al-Hajri et al., "Visualization of Personal History for Video Navigation," in Proc. SIGCHI Conf. Hum. Factors Comput. Syst., 2014, pp. 1187-1196