Real-Time Gesture-Based Sound Control System

by

Mahya Khazaei
B.Eng., Iran University of Science and Technology, 2018

A Thesis Submitted in Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

in the Department of Computer Science

We acknowledge and respect the Lək^{w}əŋən (Songhees and Esquimalt) Peoples on whose territory the university stands, and the Lək^{w}əŋən and W̱SÁNEĆ Peoples whose historical relationships with the land continue to this day.

Real-Time Gesture-Based Sound Control System

by

Mahya Khazaei
B.Eng., Iran University of Science and Technology, 2018

Supervisory Committee

Dr. George Tzanetakis, Supervisor
(Department of Computer Science)

Dr. Alex Thomo, Departmental Member
(Department of Computer Science)

ABSTRACT

This thesis presents a real-time, human-in-the-loop music control and manipulation system that dynamically adapts audio outputs based on the analysis of human movement captured via live-stream video. This project creates a responsive link between visual and auditory stimuli, fostering an interactive experience where dancers not only respond to music but dynamically influence it through their movements. The system enhances live performances, interactive installations, and personal entertainment, creating an immersive experience where users' movements directly shape the music in real time. This project demonstrates how machine learning and signal processing techniques can create responsive audio-visual systems that evolve with each movement, bridging human interaction and machine response in a seamless loop.

The system leverages computer vision techniques and machine learning tools to track and interpret the motion of individuals dancing or moving, enabling them to participate actively in shaping audio adjustments, such as tempo, pitch, effects, and playback sequence in real time. Constantly improving through ongoing training, the system allows users to generalize models for user-independent use by providing varied samples; around 50–80 samples are typically sufficient to label a simple gesture. Through an integrated pipeline of gesture training, cue mapping, and audio manipulation, this human-centered system continuously adapts to user input. Gestures are trained as signals from human to model, mapped to sound control commands, and then used to naturally manipulate audio elements.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to thank:

Dr. George Tzanetakis for granting me the freedom to explore my research independently and for encouraging me to pursue my passions, making this journey both fulfilling and inspiring,

my beloved Ali whose love, patience, and understanding made this accomplishment possible,

My friends and family, for supporting me in the low moments.

To achieve great things, two things are needed: a plan and not quite enough time.
Leonard Bernstein

DEDICATION

I dedicate this thesis to those who have encouraged me to seek knowledge, to question the world, and to believe in the power of perseverance. To all the dreamers and believers who inspire others to push beyond limits, this is for you.

Chapter 1

Introduction

1.1 Why Connect Gesture and Sound?

Sound control through motion and gesture detection is an innovative field that bridges human-computer interaction, music, and sensory technologies. Interactive sound control systems aim to interpret physical movements and gestures, translating them into meaningful modifications of sound or music. The performer or user's body becomes the primary interface, allowing for dynamic, intuitive control over musical elements like volume, pitch, playback speed, or even individual instrument tracks within a composition. Gesture-based systems are not only reshaping live performances but also contributing to interactive multimedia experiences, where sound can respond to human motion in real-time. This creates a more immersive and embodied musical engagement, enriching the relationship between the performer and the sound being created[1].

One of the critical challenges in designing gesture-based sound control systems is accurately mapping human movements to sound parameters. Gesture recognition technologies, such as motion sensors, cameras, and physiological sensors, are often used to track body movements. However, the data gathered from these devices is complex and multi-dimensional, requiring sophisticated methods to interpret and apply to sound manipulation. Mapping motion to sound is a nuanced process, as gestures may vary in speed, intensity, and form depending on the individual or the desired musical expression. Despite these complexities, gesture-based systems enable expressive music and sound control that can be completely different than traditional musical interfaces, such as keyboards or touchscreens. Such systems can also be

combined with traditional music interfaces.

Traditional interfaces like keyboards or touchscreens typically have discrete inputs (e.g., pressing a key or touching a screen), which limits the amount of expressive control that can be exerted in real-time. In contrast, gesture-based systems can capture continuous, fluid movements from the body, allowing for more dynamic and nuanced control over sound. For instance, by tracking the speed, intensity, or direction of a performer's movement, gesture-based systems can modify sound in real-time, enabling more subtle and continuous variations that are harder to achieve with buttons or sliders. This creates a deeper level of interaction where sound can respond to a wide array of body motions, providing more expressive possibilities, especially in live performance or interactive sound design. The idea is supported by research that explores how gesture and embodiment become core concepts in multimedia and musical interaction. The performer's body is deeply engaged in sound creation through motion and physiological sensing, which traditional interfaces don't capture as effectively [2, 3, 4].

Moreover, these systems rely on advances in technology and interdisciplinary research, particularly in how humans perceive sound and gesture. Research into sonic affordance, for example, explores how sound invites action and suggests that certain sounds naturally evoke specific physical responses from users [5]. This knowledge can help inform the design of more intuitive gestural interfaces, where users can naturally interact with the system in a way that feels organic and expressive. Such systems hold promise not only in music performance but also in therapeutic, educational, and multimedia contexts, where sound and motion interact to create a deeper sensory experience.

These systems can offer unique interactive experiences in other fields as well, such as:

- **Therapeutic contexts:** Gesture-based systems can be used in music therapy to engage patients in active physical movement while controlling sound. This can enhance motor skills, provide cognitive stimulation, and offer emotional expression, creating a therapeutic environment that is responsive to both movement and sound.

- **Educational settings:** These systems can be used to teach both music and dance in an interactive and engaging way. Students can experiment with sound and movement, using their bodies to explore musical and dance concepts, mak-

ing learning more immersive and intuitive. This approach is especially beneficial for young learners or individuals with diverse learning needs, as it transforms abstract concepts into physical actions, allowing them to understand rhythm, timing, and expression through direct interaction.

- **Multimedia contexts:** In areas like interactive art installations, virtual reality (VR), or gaming, combining motion detection and sound control can create deeply immersive experiences. For example, in a VR environment, users' gestures can control the auditory landscape, making the experience more engaging by synchronizing sound with the user's movements.

The notion of creating a deeper sensory experience is rooted in the concept of embodied engagement, where the body's interaction with sound creates a multisensory experience, as discussed in research on human-computer interaction and embodied musical practices [6, 2, 3]. The integration of sound, gesture, and motion provides an experience that engages both the auditory and kinesthetic senses, which is why these systems have potential in areas that require sensory interaction and engagement.

In summary, gesture-based sound control systems leverage human body movements to manipulate musical sound, enhancing both creativity and interactivity. Despite the challenges in designing effective mapping strategies between gestures and sound parameters, these systems offer a more natural and immersive way for users to engage with music. They represent the convergence of technology, art, and cognitive science, opening up new possibilities for creative expression and interaction.

1.2 Methods for Capturing Body Motions Data

Gesture-based sound control systems rely on several methods to capture and interpret human movement, translating these motions into musical or sound-related outputs. The primary methods of motion capture in these systems include optical motion sensing, inertial sensors, and muscle tension sensing. Each of these methods captures different aspects of human movement and offers unique advantages and challenges in translating motion into sound.

- **Optical sensing**, particularly through video cameras, is one of the most widely used methods for capturing body motion in various fields, including gesture-based sound control. Video-based systems can analyze movements captured by

single or multiple cameras, using sophisticated techniques such as depth-sensing or stereoscopic vision to track and record body motion. Optical motion capture (MoCap), which often uses infrared sensors and reflective markers, is still considered one of the most reliable methods for tracking complex movements in 3D space, despite being several decades old. MoCap systems provide precise data on the movement and orientation of markers attached to the body, making it possible to capture fine details like finger motions and facial expressions. Advanced MoCap systems also offer real-time streaming capabilities, making them suitable for live performance applications. However, while highly accurate, optical systems can be costly and require complex setups, limiting their accessibility in certain contexts [7, 8].

- **Inertial sensors**, commonly known as **Inertial Measurement Units (IMUs)**, offer a more portable and cost-effective alternative to optical systems. IMUs integrate accelerometers, gyroscopes, and often magnetometers to track movement in multiple dimensions. These sensors, which are increasingly used in mobile and wearable devices, can detect acceleration, rotational velocity, and orientation in real-time, making them well-suited for tracking bodily motions during musical performance. IMUs are particularly advantageous for their portability and affordability, allowing performers to move freely without the need for external cameras or markers. However, a key limitation of IMUs is their inability to provide accurate absolute position data due to sensor drift, making them less precise for tracking spatial relationships between performers or objects [7]. While they are excellent for detecting relative movements and postures, IMUs are less effective than optical systems for capturing complex motion in a 3D environment.

- **Muscle tension sensing** the third method, relies on **Electromyography (EMG)** to capture the electrical signals produced by muscle contractions. EMG sensors, widely used in biomedical fields, can detect subtle muscle activity and translate these signals into motion data for controlling sound. In musical contexts, EMG sensors offer a unique way of capturing not just movement, but also the effort and intention behind a performer's physical actions. This makes EMG particularly useful for capturing expressive, nuanced gestures that other sensing methods might miss. However, the EMG signals are highly sensitive and can be difficult to process due to noise from the environment or variabil-

ity in individual muscle usage. Machine learning techniques are often used to help interpret these noisy signals, making EMG an attractive but technically challenging option for gesture-based sound control [9, 10, 11].

Each of these motion-sensing technologies—optical systems, inertial sensors, and muscle tension sensing—brings different strengths and limitations to the design of gesture-based sound control systems. Optical systems provide high precision, inertial sensors offer portability and ease of use, and muscle tension sensing captures the subtleties of muscle effort and expression. Together, these methods contribute to creating more immersive and responsive musical experiences, pushing the boundaries of how performers and audiences engage with sound.

1.2.1 Analyzing Data and Feature Extraction from Body Motions Data

Once body movements are captured by motion sensing systems, the next step is to analyze the raw input data and extract relevant features that can be used for gesture-sound interaction. This process involves transforming raw sensor data into meaningful descriptors, which help systems understand and interpret the user's movements in ways that are musically expressive. Feature extraction is crucial because it directly influences how machine learning (ML) algorithms will interpret the body movements, and consequently, how the system will respond with sound output.

High-Level Motion Descriptors are commonly employed to analyze and describe body movement more effectively. These descriptors go beyond raw data by summarizing motion in ways that are easier to interpret, such as speed, acceleration, or smoothness of movement. In expressive movement analysis and music performance, these features provide insight into how gestures can influence sound. Tools like Eyesweb [12] and the Musical Gesture Toolbox [13] have been designed specifically to support real-time analysis of human movement and to extract such features. Initially, these tools were focused on analyzing video data, but they have been extended to work with various types of motion capture data, including optical MoCap and inertial sensors [14].

Some of the most widely used motion features include:

- **Fluidity:** This feature, inspired by research on human motion dynamics, represents how smooth or continuous a movement is. It is derived by calculating

the inverse of "jerk," which is the rate of change of acceleration over time. A higher fluidity index indicates smoother movements, and it has been used in emotion recognition tasks from full-body movement data [15, 16].

- **Quantity of Motion (QoM):** This feature measures the total amount of motion in a given gesture, calculated as the sum of the speeds of a set of points, often scaled by their mass. In musical performance, QoM helps quantify how much energy a performer is putting into their gestures, which can then be translated into sound dynamics [17, 18].

- **Contraction Index:** The contraction index captures whether the performer's body is contracting inward or expanding outward, based on the distances between points on the body and the centroid of the motion. This feature can indicate changes in posture or gesture expressiveness, often related to emotional expression [16, 17].

- **Bounding Shapes:** Bounding shapes, such as bounding boxes or convex hulls, provide a 3D spatial representation of the performer's body. The volume or dimensions of these shapes can indicate how much space the body is occupying, which can be useful for analyzing gestures in dance or performance art. This data helps track how the performer's posture changes over time, offering further insights into the relationship between gesture and sound [18, 19].

For systems that utilize **inertial measurement units (IMUs)** or **electromyography (EMG)** data, additional features are extracted:

- **Periodic Quantity of Motion (PQoM):** This feature measures how body movements resonate with the rhythm of the music, capturing the periodic nature of gestures in relation to musical timing, such as quarter notes or eighth notes. It is useful for aligning body motion with the musical beat, enhancing the interaction between movement and sound [20, 21].

- **EMG Signal Features:** For systems that track muscle tension, EMG sensors provide a wealth of information about muscular activity. Key features include **Signal Amplitude**, which measures the strength of the muscle contraction, and **Mean Absolute Value (MAV)**, which is a popular feature in EMG analysis for gesture recognition [22, 23]. Other commonly used features include **Root Mean Square (RMS)** for gesture classification tasks [24], and the **Zero**

Crossing Rate (ZCR), which tracks how often the signal changes direction and is useful for analyzing periodic muscle activity [25, 26].

By extracting these features from the raw motion data, gesture-based sound control systems can translate physical actions into meaningful musical expressions. High-level descriptors allow for more intuitive mapping of gestures to sound, and when combined with ML algorithms, they can support the development of systems that learn from user behavior and adapt their responses in real time.

1.3 Real-Time Sound Alteration

In gesture-based sound control systems, altering sound in real-time is essential for creating dynamic and interactive performances. This involves manipulating audio parameters in response to gestures as they are captured and processed, allowing the sound to change fluidly with each movement. The system must map incoming gesture data to relevant sound parameters to ensure immediate and expressive auditory feedback.

Several key sound parameters that can be manipulated in real-time include:

- **Pitch:** Adjusting the pitch of a sound based on specific gestures can modify the musical notes or tones. For instance, raising or lowering a hand can correspond to an increase or decrease in pitch, adding expressiveness to the performance.

- **Volume:** Gestures can control the loudness of sound in real-time. Large, forceful gestures might increase the volume, while smaller, more subtle movements decrease it, allowing performers to dynamically adjust sound intensity. Volume control can also apply to different instruments or *stems* within a music track, where movements can emphasize specific components of the music.

- **Playback Speed:** Gestures can also influence the speed of sound playback. Slow, deliberate movements might slow down the sound, creating a stretched effect, while rapid gestures could speed it up, energizing the performance.

- **Filtering:** Filters such as low-pass or high-pass can be applied in real-time, with gestures modulating parameters like cutoff frequency and resonance. This alters the tonal quality of the sound, giving performers control over the brightness or warmth of the audio.

- **Spatialization:** More advanced systems allow gestures to control the spatial properties of sound, such as panning it across different speakers or positioning it within a 3D sound space. This creates an immersive audio experience, with the sound appearing to move through the environment in response to the performer's gestures.

To achieve real-time manipulation, systems continuously process gesture data and map it to these parameters. For instance, **granular synthesis** offers detailed control over sound playback by manipulating playback start time, pitch shift, and playback speed. As the performer moves, the synthesis engine responds by adjusting these parameters in real-time, creating seamless transitions between different sound states [27].

Corpus-based concatenative synthesis (CBCS) is another powerful technique for real-time sound manipulation. CBCS works by breaking down an audio file into small units or "grains," each analyzed for specific auditory features, such as loudness, frequency, and energy. In real-time performance, gestures can navigate the feature space of these sound units, allowing the performer to control which parts of the sound are played based on their movements. This method enables highly expressive control over the timbre and texture of sound by selecting audio units that best match the desired auditory features [28]. CBCS offers a more advanced and flexible way to manipulate pre-recorded audio, providing rich, multidimensional control over sound output during live performance.

Through systems like granular synthesis and CBCS, gesture-based sound control systems can offer performers a wide range of real-time sound alterations. Whether generating new sounds or manipulating existing audio tracks, these systems allow users to interact with sound on a deeper, more intuitive level, enhancing both the creativity and expressiveness of live performances.

1.4 Mapping Motion and Gesture Data to Sound Control

Mapping motion-related data to the sound-controlling part of a gesture-based system is a fundamental process that transforms physical gestures into meaningful auditory outcomes. This mapping enables real-time interaction between the performer and the sound, allowing movements to be seamlessly translated into musical expressions.

The effectiveness of these systems largely depends on how well the motion data is analyzed and connected to specific sound parameters, making the design of these mappings crucial for achieving expressive and responsive sound control.

In a gesture-based sound system, the mapping is often facilitated by **Machine Learning (ML)** techniques, which allow for flexible and adaptive responses to the user's movements. During the **training phase**, motion sensors capture data from the performer's gestures, which is then associated with corresponding sounds or sound parameters. The system "learns" these associations by analyzing the relationships between the input gestures and the resulting sound output. Once the model is trained, it enters the **testing phase**, where it can interpret new gestures in real-time, responding based on the patterns it learned during training. This approach enables the system to generalize and apply learned mappings even in unpredictable environments or with different users, ensuring a consistent and flexible interaction [29].

Effective gesture-sound mappings involve several strategies. For example, in **supervised learning** , specific gestures are paired with the desired sound output during training. This can allow for both **classification** and **regression** tasks, where classification assigns specific gestures to sound events (e.g., a circular gesture might trigger a snare drum), while regression allows continuous control over sound parameters, such as adjusting volume or pitch in real-time based on the smoothness or intensity of a movement [30]. These mappings can be further enhanced using probabilistic classifiers, which provide a range of possibilities rather than discrete choices, making the interaction more fluid and adaptable to nuanced movements.

Reinforcement learning (RL) offers another layer of flexibility, where the system continuously refines the mapping between gestures and sounds based on user feedback. For instance, a performer might use a gestural interface to control sound synthesis, and the system can adjust the mapping based on how the performer responds to different sound outputs. Over time, the system learns to optimize its mappings, offering a more personalized and adaptive experience [31].

Incorporating temporal modeling into the mapping process is also essential, as gestures evolve over time. Techniques like **Dynamic Time Warping (DTW)** or **Hidden Markov Models (HMM)** are used to align incoming motion data with pre-recorded gesture templates, allowing the system to track not just the gesture's shape but also its temporal progression. This enables real-time adjustments in sound based on how a gesture unfolds, further enhancing the expressivity and responsiveness of the system [32, 33].

By employing these ML techniques and mapping strategies, gesture-based sound systems can effectively convert motion data into intricate and responsive sound control. This mapping process not only enables real-time sound manipulation but also ensures that the system remains flexible and expressive across different contexts and users.

1.5 Contributions

This research presents a significant contribution to the field by developing an innovative system for gesture acquisition, enabling users to perform gestures in sync with musical beats. The system offers configurable sample counts, allowing users enough preparation time for positioning and coordination relative to the camera, enhancing gesture accuracy and consistency. A key aspect of this work involved data collection from diverse users, creating a valuable dataset that strengthens the reliability and applicability of the system.

To demonstrate the effectiveness of the developed system, few experiments were conducted, showcasing its capability in real-time gesture recognition and audio manipulation. An optimized configuration within Max was established to capture live data and manage audio, ensuring that raw video data remains contained within the Max environment while essential landmarks are relayed to Python, which facilitates advanced machine learning applications.

The system is designed for accessibility, allowing seamless interaction without the need for programming knowledge, making it usable by both technical and non-technical audiences in interactive audio applications. In its design, careful consideration was given to optimizing performance for live use, configuring system resources such as music alteration processes to ensure artifact-free, natural output. This approach enhances the system's practicality in real-time environments, delivering a fluid, high-quality experience that maintains the integrity of the audio while supporting responsive gesture-based interaction.

1.6 Outline

Chapter 2 Related Works - This chapter reviews existing research in gesture recognition and sound synthesis, focusing on relevant advancements in machine learning and interactive systems. Key topics include gesture recognition techniques, real-time sound control frameworks, and the integration of emotion mapping in sound-based interaction systems. This review establishes a foundation by highlighting how previous works have influenced the system's development and identifying research gaps.

Chapter 3 System Description - This chapter details the architecture of the proposed system, covering the components such as Max MSP, MediaPipe, and Open Sound Control (OSC) for communication. It explains the technical aspects of integrating gesture tracking and sound manipulation, providing a blueprint of the data flow and processing stages. Emphasis is placed on the modularity of the design, allowing for real-time responsiveness and flexible interaction between user gestures and sound output.

Chapter 4 Experiment and Evaluation - This chapter describes the experimental setup and evaluation metrics used to assess the system's performance. It includes scenario design for testing real-time responsiveness and accuracy in gesture recognition, along with specific metrics such as latency and classification accuracy. The chapter also presents a comparative analysis of the system's effectiveness against other gesture recognition tools.

Chapter 5 Future Works - This chapter delves into potential advancements and research directions to expand the system's capabilities. Suggestions include enhancing motion detection algorithms to handle more complex and subtle gestures, increasing system adaptability for personalized gesture recognition, and exploring additional applications such as in live performance, interactive installations, or rehabilitation therapies. It also proposes integrating advanced machine learning models to improve real-time responsiveness and adaptability.

Chapter 6 Conclusion - This final chapter summarizes the thesis findings, emphasizing the system's contributions to real-time, gesture-based sound control. It reflects on the challenges encountered, the system's strengths, and limitations,

and provides closing thoughts on the future trajectory of research in interactive sound systems.

Chapter 2

Related works

In recent years, the intersection of gesture recognition, machine learning, and sound synthesis has attracted significant attention from researchers, driven by advancements in human-computer interaction (HCI), wearable technologies, and computer vision. This chapter reviews the most relevant contributions in the field of gesture-controlled sound, highlighting how these efforts have influenced the development of the prototype and techniques presented in this thesis.

Several key areas of research provide the foundation for this work: gesture and body movement recognition systems, real-time sound control systems, Gesture-based interactive sound and emotion mapping and machine learning techniques for adaptive and real-time interaction/Real-time machine learning for gesture-sound mapping. A number of studies have employed various approaches, including deep learning for gesture classification and signal processing for sound synthesis, to enhance the user experience in creative and interactive sound environments.

2.1 Gesture and Body Movement Recognition Systems

Deep Learning for Gesture Recognition

Gesture recognition has undergone a significant evolution with the adoption of deep learning techniques, driven by several groundbreaking developments. One of the key milestones is the integration of Convolutional Neural Networks (CNNs) for pose estimation tasks, such as in Convolutional Pose Machines (CPMs), which sequentially

predict increasingly accurate body part locations by learning implicit spatial relationships between parts. This approach overcomes the limitations of traditional models by refining predictions iteratively and addressing challenges like occlusions and varying poses [34].

Another evolutionary finding is the use of graph-based models, like the Spatial-Temporal Graph Convolutional Networks (ST-GCN), which model human actions by treating skeleton sequences as graphs. These models can capture both spatial and temporal dynamics of human joints, leading to improved generalization and action recognition. ST-GCN eliminates the need for hand-crafted features by automatically learning joint connectivity patterns over time[35]. Furthermore, frameworks like OpenPose have enabled real-time multi-person 2D pose estimation using Part Affinity Fields (PAFs), providing a nonparametric method to associate body parts, resulting in high-accuracy, scalable solutions [36]. These advancements collectively push the boundaries of gesture recognition, making it more precise, efficient, and adaptable to real-world applications. In gesture recognition, the MediaPipe framework is notable for its ease of use in prototyping pipelines for tasks such as 2D and 3D hand and body pose detection, leveraging models like Google's Body Landmarks and hand tracking. It allows real-time, high-accuracy detection, making it a popular choice for applications in human-computer interaction [37].

Ensemble-Driven Pose Estimation

Deep learning has significantly advanced gesture recognition, particularly through the use of carefully designed pipelines like 3D hand pose estimation. In the paper 3D Hand Pose Estimation Based on Five-Layer Ensemble CNN, the authors present a novel approach where the task of hand pose estimation is broken down into smaller sub-tasks, each focusing on individual fingers. This hierarchical method, known as the Five-Layer Ensemble CNN (5LENet), addresses the challenges of self-geometric ambiguities and occlusions common in hand pose estimation using a single RGB images. The system utilizes five separate layers to estimate the 3D poses of individual fingers and then fuses them to produce an accurate full 3D hand pose. By dividing the task into layers, the system captures finer details of each finger before fusing them to create a complete 3D hand pose, resulting in more accurate and detailed estimations compared to holistic approaches [38].

The use of a tailored ensemble pipeline ensures that finer details of hand move-

ment are captured, and combining the individual finger estimates leads to more precise overall results. Additionally, by utilizing the natural topology of the hand, the system enhances the structural representation, further improving accuracy. The model takes advantage of the hand's natural structure, particularly by connecting the middle finger to the palm, which improves the representation of hand movements. This breakdown into smaller, manageable tasks, supported by methods like RNNs that can process sequences step by step, provides more accurate results compared to techniques that analyze the hand as a whole. By focusing on parts of the hand in a structured sequence, this method offers greater precision, especially in applications like virtual reality (VR), augmented reality (AR), and human-computer interaction (HCI) [39]. Jobanputra et al. (2019) provide a broad overview of human activity recognition (HAR) systems, which primarily focus on differentiating between physical activities such as walking and running. While these systems excel at analyzing sequences of movements with strong temporal dependencies, the problem addressed in this thesis is distinct. Instead of recognizing extended activities, this work focuses on classifying discrete gestures for real-time audio manipulation. This key difference highlights that HAR systems operate with a broader temporal scope, whereas the gesture-based system described here emphasizes precise, momentary classifications tailored to interactive applications.[40] The research "Gesture-Based Affective Computing on Motion Capture Data" explores the use of full-body skeletal motion capture to interpret gestures in the context of affective computing. This work is significant in the domain of gesture and body movement recognition systems as it demonstrates how detailed motion data, captured through video-based sensors, can be analyzed to understand human gestures and emotions. The system utilizes motion capture data to extract key features from body movements, enabling the recognition of affective states through gestures. This approach contributes to the development of more sophisticated gesture recognition systems that integrate emotional context, improving human-computer interaction, particularly in applications like virtual reality, gaming, and social robotics.[41]

Real-time

Real-time gesture recognition systems face key challenges like latency, accuracy, and environmental variability. Instant feedback is crucial in applications like virtual reality, gaming, and interactive systems, where even slight delays can impact user ex-

perience. Achieving high accuracy is difficult due to the complexity of gestures, movement variations, and changing conditions like lighting or background noise. In a gesture-based sound control system, recognition is the first step in the pipeline, and its latency and accuracy directly affect the rest of the system, making it essential to address these challenges effectively.

One approach to addressing the challenges of real-time gesture recognition is the development of systems that employ computer vision algorithms for natural human-computer interaction. For example, researchers have implemented a system using techniques like Camshift and Haar-like classifiers to track hand gestures in real time, enabling interaction with virtual environments. This work aims to replace traditional input devices, such as a mouse, with hand gestures for manipulating 3D objects, which is especially useful in applications like gaming or virtual browsing. While the system shows promise, it faces challenges in noisy environments and varying illumination conditions, which impact recognition accuracy.[42]

Another notable effort involves using machine learning methods for gesture recognition based on skeletal data captured by sensors like the Kinect. In this work, researchers employed Support Vector Machines (SVM) to classify gestures associated with emotional states based on body movements, such as recognizing gestures that indicate emotions like "friendly" or "in distress." By optimizing the system for real-time performance on embedded hardware, this research showcases the potential of integrating gesture recognition into smaller, more efficient devices, pushing the boundaries of real-time performance [43].

2.2 Real-time Sound Control Systems

When discussing the contributions to real-time sound synthesis, particularly systems that respond dynamically to environmental changes, the challenges faced by gaming sound synthesis are a prominent reference point. Real-time systems in this domain must address issues such as maintaining low latency, optimizing resource utilization, and ensuring seamless interaction between the virtual environment and the generated sound. The research in this field has explored various methods to overcome these challenges, ranging from leveraging hardware capabilities like GPUs to employing efficient algorithms and computational models. Notably, these studies often focus on creating adaptive and responsive soundscapes, whether for virtual environments, live performances, or interactive applications like gaming. Each work reviewed here

contributes distinct techniques and solutions to improve real-time sound synthesis, pushing the boundaries of interactive audio experiences.

The first study under review highlights a system that synthesizes realistic sounds in real-time for virtual environments by utilizing GPU-based recursive and non-recursive filters. The challenge of implementing recursive filters on GPUs, traditionally suited for tasks without recursive dependencies, was a key obstacle. To address this, the authors developed a novel method to efficiently propagate recursion across parallel GPU threads, ensuring that the system could handle the dynamic audio requirements of a virtual world without significant computational overhead. This breakthrough allowed for lightweight processing while maintaining the high performance necessary for interactive applications such as gaming. By synthesizing sounds, such as the collision of objects made from different materials, this system responded in real-time to changes in the virtual environment, ensuring a seamless and immersive experience for users [44].

A different approach is taken by a study that modeled the sound of a historical acoustic wind machine using digital synthesis techniques. The researchers employed the Sound Designer's Toolkit (SDT) in Max/MSP to create a computationally efficient system that could replicate the mechanical sounds of friction between wooden slats and cloth. The real-time aspect of the system was achieved by coupling a rotary encoder with an Arduino, which controlled the sound synthesis engine. To manage computational load, the system paused sound generation when slats were not in contact with the virtual cloth, effectively minimizing unnecessary processing. This optimization was crucial for maintaining real-time performance, allowing the system to generate complex physical modeling with minimal strain on the CPU. The use of polyphony management further streamlined the processing, making this solution suitable for live performances where real-time sound manipulation is essential.[45]

Another significant contribution to the field comes from a system that synthesizes perceptually relevant sounds based on interactions between morphing solids, such as impacts, friction, and rolling events. The authors designed a double source-filter model to simulate the sounds of these interactions efficiently. By focusing on key parameters like impact speed and friction, they created a system that responded dynamically to changes in the environment while keeping computational demands low. Real-time performance was ensured by carefully managing the number of filters used, allowing the system to handle complex audio tasks without overwhelming the processor. The use of the Open Sound Control (OSC) protocol facilitated seamless

communication between the synthesizer and the game engine, ensuring that the sound evolved in sync with the virtual environment, making it ideal for applications that require real-time interaction, such as video games [46].

A further study explored the real-time synthesis of footstep sounds for virtual reality applications. This research focused on generating dynamic audio feedback based on the user's physical interaction with virtual surfaces, such as gravel, wood, or sand. The system captured real footstep sounds using microphones and estimated the corresponding ground reaction forces (GRF), which were then used to control a sound synthesis engine based on physical models. By doing so, the system produced realistic audio feedback that enhanced user immersion in the virtual environment. The use of C++ algorithms embedded within the Max/MSP platform enabled real-time processing, with compatibility for Pure Data. Unlike previous approaches that relied on sensor-embedded footwear, this system allowed users to interact with the environment while wearing their own shoes, making it more flexible and accessible. The integration of real-time sound synthesis with user input offered a significant improvement in the realism of virtual experiences, particularly in gaming and VR environments [47].

Lastly, a study on real-time vocal harmonization systems for live performances addressed one of the core challenges in real-time sound synthesis: maintaining low latency while preserving sound quality. The authors implemented a frame-based processing system that allowed the system to detect pitches and chords in near real-time, ensuring that vocal harmonization could be applied without delay during live performances. To achieve this, they utilized the YIN pitch detection algorithm and PSOLA for pitch shifting, both optimized for real-time processing. Template matching for chord detection was used to reduce computational overhead, avoiding the need for more complex statistical models. This approach ensured that the system remained lightweight and responsive, making it suitable for live performance scenarios where any delay in sound synthesis could disrupt the flow of the performance [48].

In conclusion, each of these works provides valuable insights and solutions to the challenges of real-time sound synthesis. From leveraging hardware like GPUs to optimize processing, to employing lightweight algorithms for live performance contexts, these studies push the boundaries of what is possible in interactive audio. Their contributions not only improve the technical capabilities of real-time sound systems but also enhance the immersive experience for users, whether in gaming, virtual reality, or live performances.

2.3 Gesture-based Interactive Sound Mapping systems

Wekinator [49] is a machine learning software created by Rebecca Fiebrink that enables users to use real-time, interactive machine learning to create new musical instruments, sound controllers, and other creative tools. Designed with artists and musicians in mind, Wekinator simplifies the process of training machine learning models, allowing non-experts to experiment with gesture, sound, and image recognition. Wekinator supports various input devices like cameras, microphones, and sensors, and it can translate these inputs into data for machine learning algorithms that generate outputs, such as sounds or visual effects. By using supervised learning techniques, users can "teach" Wekinator through demonstrations, making it accessible for interactive applications without requiring programming skills.

Wekinator's popularity in the creative technology field stems from its compatibility with software like Max/MSP, Pure Data, and Processing, which are widely used in interactive art, music, and multimedia projects. It supports multiple machine learning algorithms, such as k-nearest neighbors, neural networks, and support vector machines, allowing users to select models that best suit their needs. By focusing on real-time interactivity and facilitating rapid prototyping, Wekinator has become a valuable tool for artists, musicians, and designers interested in machine learning-based interaction, enabling projects where traditional coding might be a barrier.

In Continuous interaction with a smart speaker via low-dimensional embeddings of dynamic hand pose (CISS)[50] the authors introduce a novel method for controlling smart speakers through mid-air hand gestures. The system utilizes MediaPipe Hands to extract 21 hand landmarks from video frames, which are then embedded into a two-dimensional pose space using an autoencoder. This low-dimensional embedding facilitates intuitive interaction by mapping hand poses to corresponding music track profiles. A PointNet-based model is employed to classify gestures, enabling users to control music playback and explore music spaces seamlessly. By jointly optimizing the autoencoder with the classifier, the system enhances gesture discrimination, allowing users to select different musical moods through variations in hand pose. This combination ensures efficient and responsive control of smart speaker functions through dynamic hand gestures.

In the paper Mugeetion[51], the authors present a novel musical interface that captures users' facial gestures to influence musical features in real-time. The sys-

tem utilizes FaceOSC software to track facial gestures and employs the Facial Action Coding System (FACS) to interpret these gestures as emotional states. By mapping specific facial action units (AUs) to corresponding musical parameters, Mugeetion enables users to control sound generation through their facial expressions, creating an emotion-driven musical experience. This approach leverages computer vision techniques for real-time facial expression detection, allowing for dynamic modulation of sound based on the performer's emotional state.

In their 2020 work[31], Towards assisted interactive machine learning(AIML), Federico Ghelli Visi and Atau Tanaka introduce a system that employs reinforcement learning to develop adaptive gesture-sound mappings. This approach allows musicians to interactively train the system, enabling it to learn and refine the associations between specific gestures and corresponding sounds over time. By integrating reinforcement learning, the system adapts to the performer's unique gestures, facilitating a personalized and responsive musical experience. This method contrasts with traditional static mappings by offering dynamic, real-time adaptability, thereby enhancing expressive potential in musical performances.

Table 2.1: Comparison of Systems Based on Accuracy, Gesture Types Supported, and Python Compatibility

System	**Accuracy**	**Gesture Types Supported**	**Python Compatibility**
Proposed	Hypothetically 85-90% based on setup	Full-body and hand gestures	Yes
CISS	Not specified	Mid-air hand gestures	Yes
Mugeetion	Not specified	Facial movements, emotion-driven gestures	No
Wekinator	Dependent on user configuration and training iterations	Multiple gesture types based on user-defined inputs	No

Continued on next page

System	Accuracy	Gesture Types Supported	Python Compatibility
AIML	Dependent on RL training and adaptive learning performance	Hand and body gestures with reinforcement-based adjustments	Yes

Table 2.2: Comparison of Proposed System and Other Gesture-Based Sound Control Systems

System	Criterion	Description
Proposed	Purpose	Real-time gesture-to-sound control
	Gesture Acquisition Method	Body landmarks via camera
	Data Processing Pipeline	OSC and MediaPipe with Python for ML and audio triggering
	Real-Time Interaction	Yes
	Hardware Requirements	Camera and computer
	Adaptability	Customizable for various gestures
	Strengths	Low-cost setup, real-time adaptability
	Limitations	Noise impact untested, possible lighting sensitivity
	Unique Features	Customizable, adaptable for personalized sounds
CISS	Purpose	Continuous hand gesture control for smart speakers
	Gesture Acquisition Method	Camera for hand pose detection, PointNet-based model

Continued on next page

System	Criterion	Description
	Data Processing Pipeline	Embedding hand pose data for continuous interaction
	Real-Time Interaction	Yes
	Hardware Requirements	Camera and speaker for smart speaker setup
	Adaptability	Fixed hand pose gestures
	Strengths	Continuous interaction, intuitive for smart devices
	Limitations	Dependent on user hand position, gesture library limited
	Unique Features	Low-dimensional embedding for interaction continuity
Mugeetion	Purpose	Emotion-based music control
	Gesture Acquisition Method	Facial gestures captured via camera
	Data Processing Pipeline	Facial gesture recognition pipeline
	Real-Time Interaction	Yes
	Hardware Requirements	Camera setup for facial recognition
	Adaptability	Limited gesture variety, emotion-driven
	Strengths	Unique emotion-based sound control
	Limitations	Lighting dependence, may not support complex expressions
	Unique Features	Emotion-driven control for an immersive experience
Wekinator	Purpose	Real-time interactive ML platform for gesture-to-sound mapping
	Gesture Acquisition Method	Input via sensors, controllers, or other external sources

Continued on next page

System	Criterion	Description
	Data Processing Pipeline	Interactive learning pipeline for mapping gestures to audio outputs
	Real-Time Interaction	Yes
	Hardware Requirements	PC, any sensor or input device
	Adaptability	High adaptability with customizable gesture-sound mappings
	Strengths	Flexibility in training and adjusting gesture-sound mappings
	Limitations	Initial setup requires user input, learning may take time
	Unique Features	User-defined mapping with interactive machine learning
AIML	Purpose	Interactive gesture-sound mapping with reinforcement learning
	Gesture Acquisition Method	Sensors for gesture detection with adaptive reinforcement learning
	Data Processing Pipeline	Reinforcement learning for dynamic gesture-sound mapping
	Real-Time Interaction	Yes
	Hardware Requirements	Reinforcement learning setup with gesture-sensing hardware
	Adaptability	Adaptive through reinforcement, with potential for personalized mappings
	Strengths	Dynamic, adaptive mappings via reinforcement learning
	Limitations	Requires reinforcement learning adjustments and potentially longer setup
	Unique Features	Reinforcement learning approach enables continuous adaptation to gestures

The comparative analysis presented in tables 2.1 and 2.2 highlights the unique features and limitations of various gesture-based sound control systems. Notably, the proposed system stands out for its real-time adaptability, low-cost setup, and support for a wide range of customizable gestures, differentiating it from other systems that may have limited gesture types, adaptability, or complex hardware requirements. This comparison helps to underscore the proposed system's strengths and areas for improvement, informing future development directions. In the next chapter, we delve into a detailed description of our proposed system

Chapter 3

System Description

The proposed system processes live video streams to alter sound based on body movements and postures, providing an interactive, real-time experience. Body detection and landmark extraction are handled by Google's MediaPipe framework, which identifies key body, hand, and facial landmarks necessary for interpreting gestures. MediaPipe runs within the Max/MSP programming environment through an emulation setup, leveraging Max MSP's real-time processing strengths in multimedia applications. Since the most effective machine learning tools and models are available in the Python environment, we used the OSC (Open Sound Control) protocol to bridge Max MSP and Python. OSC ensures fast, structured communication between these environments, allowing gesture data extracted in Max MSP to be processed by machine learning models in Python for adaptive sound synthesis. This setup enables a dynamic response between visual input and audio output, enhancing the system's real-time interactivity and interfacing the large state of the art ML ecosystem of the Python language.

3.1 System architecture

The system architecture is designed for real-time gesture-based sound control and integrates components across Max MSP and Python environments, with Open Sound Control (OSC) facilitating communication between them. Within the Max environment, a live camera feed provides raw video input, which is processed by a MediaPipe module to detect body gesture landmarks. These landmarks, representing key positions of the user's body, are then passed through Max's patchers and formatted

as JSON data before being sent to a Python console via OSC. This modular design allows for efficient handling and transfer of gesture data from Max MSP to Python, where more intensive processing takes place.

In the Python environment, the incoming gesture data undergoes a series of transformations: preprocessing, cleaning, and classification using machine learning algorithms. The Python module not only performs real-time gesture recognition but also incorporates a memory system, which enables the storing of gesture data for future use. This feature allows the system to adapt and recall previously used gesture data, enhancing user interaction by allowing for consistency and customization over time. The recognized gestures are mapped to specific audio manipulations based on preconfigured associations set by the user. Once a gesture is identified, Python sends a corresponding message back to Max MSP via OSC, activating a designated audio control patcher in real-time.

This architecture, as illustrated in the figure 3.1, clearly delineates the workflow and responsibilities of each component. The Max environment handles video capture, landmark detection, and initial data handling, while Python manages machine learning processing and gesture memory. The OSC link serves as a bidirectional channel, enabling continuous communication between the environments to achieve dynamic, responsive sound manipulation tied to user-defined gestures. This design ensures flexibility and allows for scalable expansion of gesture-sound mappings, making it suitable for interactive sound applications in performance or experimental contexts.

3.2 External Components

3.2.1 MAX

Max/MSP is a visual programming language and development environment primarily designed for music, audio, and multimedia applications. Created by Cycling '74, it enables users to build interactive and real-time systems by visually connecting objects that represent various functions, from sound synthesis to video manipulation. Max MSP is highly modular and extensible, making it adaptable to many creative and technical projects. Its graphical interface allows for straightforward building of interactive systems, which is ideal for applications that require real-time processing, such as interactive music generation or responsive sound control.

Max/MSP offers faster response times for real-time multimedia processing com-

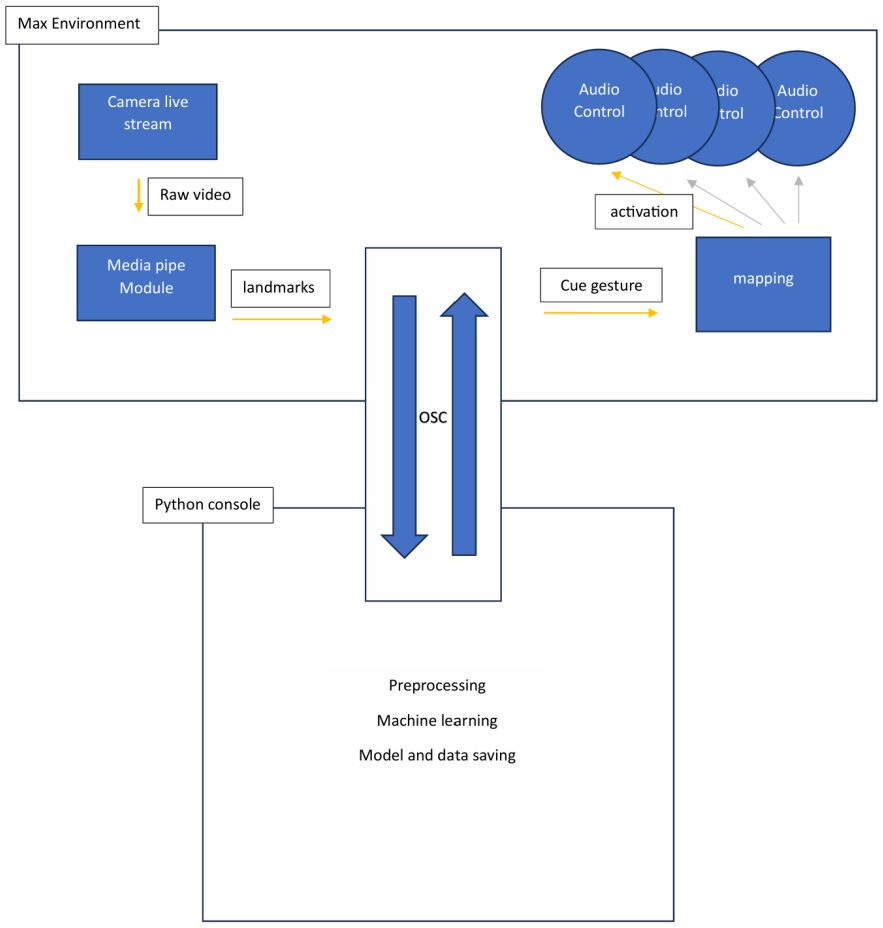

Figure 3.1: system data flow

pared to alternatives like Pygame or OpenCV, particularly when running on standard laptop processors. This performance advantage stems from Max MSP's design as a low-latency environment optimized for real-time applications. In contrast, Pygame, though effective for simpler 2D game graphics, was not built for the demands of high-speed multimedia or audio processing. Pygame relies heavily on Python's interpreted nature, which can introduce delays in scenarios requiring real-time responsiveness and interactivity, such as gesture-to-sound control applications.

Another consideration is OpenCV, commonly used for real-time image processing. While OpenCV is capable of handling live video streams, it generally requires more system resources and can introduce latency, especially on non-specialized hardware. OpenCV also lacks the direct, native support for real-time audio and multimedia processing found in Max MSP, making it less efficient for projects that combine multiple

media types. Max MSP's architecture, designed specifically for low-latency, multimedia processing, allows it to handle complex interactions in real-time, making it the preferred choice for applications involving responsive sound manipulation and multimedia interactions on general laptop hardware.

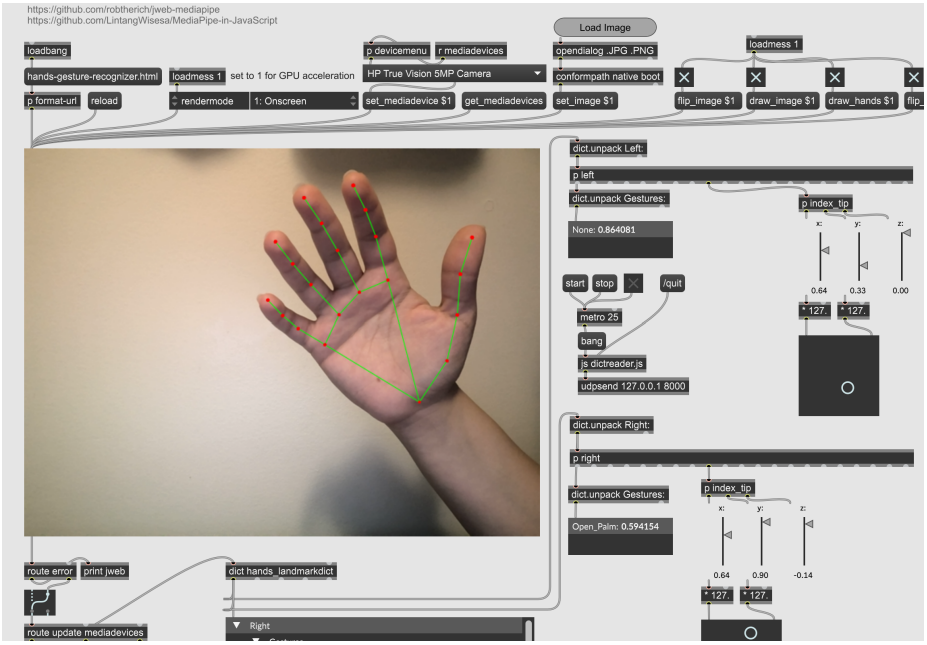

Figure 3.2: Mediapipe hand gesture detection module in Max environment connected to a patch that sends data to the Python console.

3.2.2 MediaPipe

The MediaPipe framework, developed by Google, is a cross-platform framework for building customizable, real-time machine learning pipelines for video, audio, and sensor data. It is widely used for applications like object detection, hand and body tracking, and face detection. MediaPipe provides several pre-built solutions, each designed for specific use cases. For instance, the MediaPipe Hands model detects 21 landmarks per hand, enabling precise hand tracking for gestures, while MediaPipe Face Mesh captures 468 facial landmarks, allowing for highly detailed facial expression analysis. The MediaPipe Pose model offers a 33-point body landmark system, capturing a full-body skeleton for activity and motion tracking.it also has a built in hand gesture detection.

MediaPipe's cross-platform capabilities are one of its defining strengths, allowing it to run on a wide range of devices, from laptops to mobile devices and embedded

systems. Its modular design and support for multiple programming languages, including Python, C++, and JavaScript, make it adaptable across different operating environments. MediaPipe's framework can be embedded within mobile applications on Android and iOS, enabling real-time gesture or face recognition on consumer-grade devices. Additionally, its compatibility with popular machine learning libraries, such as TensorFlow, means that users can extend MediaPipe's functionalities or integrate it within larger machine learning workflows.

To bring MediaPipe's body landmark detection into the Max MSP environment, developers have utilized the jweb-mediapipe [https://github.com/robtherich/jweb-mediapipe] integration, a project that adapts MediaPipe for Max MSP via JavaScript. The jweb-mediapipe library, is based on an initial project by Lintang Wisesa, MediaPipe-in-JavaScript[https://github.com/LintangWisesa/MediaPipe-in-JavaScript]. This original library rewrote MediaPipe in Javascript, making it accessible to JavaScript environments such as browsers or applications that use Node.js. By enabling MediaPipe in a JavaScript form, jweb-mediapipe can be embedded into Max MSP as a JavaScript object, allowing the system to leverage MediaPipe's body and facial landmark detection directly within the Max environment. This integration allows Max MSP to process body landmark data in real time, enabling gesture-driven interactions with sound, video, and other media in a seamless, low-latency format ideal for interactive multimedia applications.

As discussed in Chapter **??**, our system utilizes both body pose landmarks and hand pose landmarks for gesture recognition. The MediaPipe framework provides three-dimensional coordinates for each detected point, capturing position data in terms of x, y, and z values. For hand gestures, we utilized all available landmarks to ensure comprehensive hand motion tracking, as each point contributes to an accurate representation of the gesture.

However, for body pose recognition, we selected only a subset of the landmarks, specifically focusing on those directly relevant to gesture detection. Certain points, such as face landmarks, were excluded from our analysis, as they do not play a significant role in identifying body movements or gestures. Among the remaining body landmarks, the nose position was employed as a central reference for head orientation, providing a stable anchor for body position without unnecessary complexity.

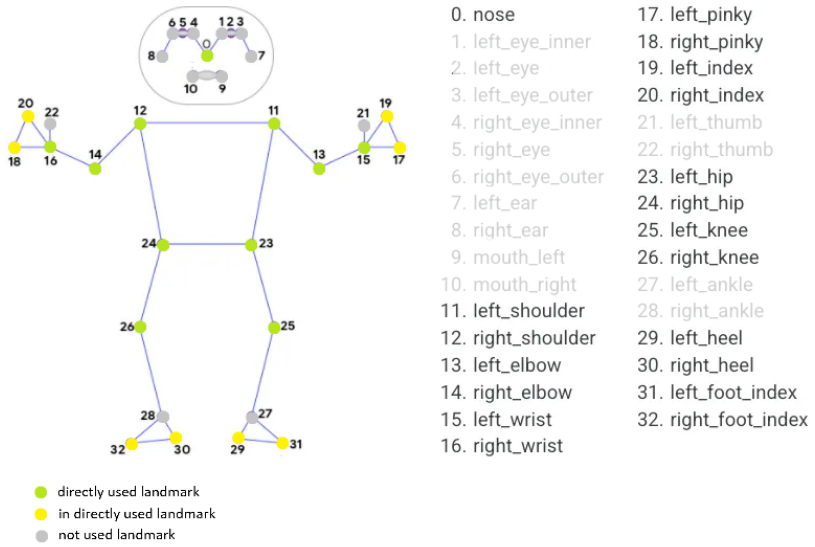

Figure 3.3: Mediapipe body pose landmarks and feature selection

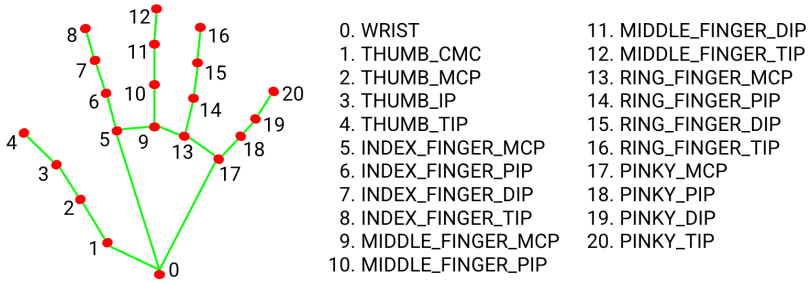

Figure 3.4: Media pipe hand pose landmarks. All features have been used in training

3.2.3 Open Sound Control

Open Sound Control (OSC) is a communication protocol designed to facilitate real-time, low-latency data exchange between multimedia applications, allowing them to interact and synchronize seamlessly. Originally developed in the 1990s for music applications, OSC has since gained popularity across a range of fields, including interactive installations, gaming, and virtual reality, due to its flexibility and efficiency. OSC transmits data as messages containing both address patterns and arguments, allowing for structured, dynamic communication between devices and applications over a network. This structure enables developers to create complex, interactive systems where sound, video, and other media can be precisely controlled and synchronized across various applications, regardless of platform. Its lightweight design ensures that even time-sensitive interactions, like those in gesture-controlled sound environments,

can maintain responsiveness and accuracy, making OSC a valuable tool in multimedia and real-time systems.

OSC (Open Sound Control) is an ideal choice for sending data from Max MSP to Python due to its speed, flexibility, and capacity to handle real-time, inter-application communication. In a project where gesture data from MediaPipe in Max MSP must be transmitted to Python for additional machine learning processing, OSC's efficient, low-latency communication protocol enables smooth, responsive data flow.

The flexibility of this system is enhanced by its compatibility with major machine learning frameworks in Python, such as PyTorch and TensorFlow. Using OSC (Open Sound Control) for data exchange between Max MSP and Python allows seamless integration of these frameworks, facilitating real-time processing of gesture data. This compatibility means that, in addition to custom implementations, the system can leverage a wide array of pre-trained models and machine learning tools provided by these popular libraries. PyTorch and TensorFlow support makes it straightforward to experiment with different algorithms and architectures, offering adaptability for various research or performance settings. This design choice aligns with the system's goal of being both modular and extendable, allowing researchers to modify or upgrade the machine learning component as needed without altering the overall structure.

3.2.4 Machine Learning model

In our initial experiments, we explored several classifiers, including Random Forest, Gradient Boosting Classifier, and a Multilayer Perceptron (MLP), to determine the best model for classifying body gestures based on body landmarks. While Random Forest and Gradient Boosting offered reasonable performance, the MLP stood out by delivering superior accuracy and consistency in recognizing complex gesture patterns, making it the most effective choice among our baseline models.

An MLP is particularly well-suited for body gesture classification due to its ability to handle complex, non-linear relationships in high-dimensional data. Body gestures are represented by the spatial configuration of multiple joints and points, and the MLP's fully connected layers excel at capturing dependencies among these positions. This capability allows the MLP to differentiate between subtle variations in gestures, identifying patterns within body landmarks with precision.

Moreover, MLPs are straightforward and adaptable in architecture, easily tailored to different dataset sizes without excessive computational overhead. Unlike convo-

lutional networks that require grid-like data, MLPs work well with flattened data structures, such as coordinates of body landmarks, without requiring extensive preprocessing. Additionally, they are computationally lightweight compared to more complex models, making them ideal for real-time gesture recognition systems, where quick and accurate classification is crucial. This balance of efficiency and performance makes MLPs well-suited for body gesture classification applications.

Chapter 4

Experiment and Evaluation

In this chapter, we present our experimental approach to demonstrate and validate the concept of gesture-based sound control, with an emphasis on its potential applications in traditional performance settings. Our goal is to explore how performers can use distinct gestures to interact with and manipulate sound in real time, offering a new level of creative expression and control. To thoroughly investigate this, we designed two main test scenarios that simulate performance conditions, providing insight into the technical requirements necessary for accurate and responsive gesture recognition.

The experimental process is divided into distinct phases, each serving a specific role in preparing, training, and evaluating the system's response to performer gestures. The initial training phase introduces the performer to the system through a metronome-guided sequence, during which they execute their primary gesture at designated intervals multiple times. This helps the system learn to differentiate the primary gesture from other incidental movements. Following training, the system undergoes an evaluation phase, where its accuracy in gesture recognition is assessed. To ensure reliability, users can refine the model by adding additional gesture instances or corner cases, enhancing the model's robustness for live use.

Finally, once the system has been trained and validated, users map their gestures to specific sound actions within Max MSP, enabling real-time control during performance. Our experimental setup allows performers to adjust playback timing, modify sound features like pitch and gain, and apply effects based on detected gestures. We also evaluate the system's feedback time and classification accuracy in comparison with established tools, aiming to achieve over 90% similarity in classification results. Through this methodical approach, we validate the concept of gesture-based sound control, offering a (novel) interaction manipulation for sound in performance contexts.

4.1 Experimental Overview

The primary goal of this experiment is to validate the feasibility and effectiveness of the proposed real-time gesture-controlled sound synthesis system. Specifically, the experiment aims to demonstrate that gestures captured via body tracking (e.g., MediaPipe) can successfully and responsively modulate audio output within a practical, real-time framework. Key objectives include assessing the system's responsiveness (e.g., low latency between gesture recognition and sound synthesis), accuracy in gesture detection, and overall reliability in maintaining continuous sound control without interruptions.

Additionally, the experiment seeks to explore the ease of integration among different components (such as Max MSP, MediaPipe, and Python via OSC) and evaluate the system's usability in terms of intuitive control and adaptability to different user gestures. These objectives collectively aim to establish that the concept is practical for real-time applications, providing a foundation for further development and potential use in dynamic environments like live performance or interactive installations.

4.1.1 Scenario Design

The scenarios developed for the system are grounded in real challenges that performers often encounter during live shows. In Broadway musicals, for instance, live music plays a crucial role, with a conductor synchronizing the orchestra with actors and actresses for sudden, unpredictable musical cues. In smaller or local performances of these productions, however, it can be challenging to find skilled local musicians and conductors who can precisely coordinate with the dance sequences. As a result, many productions rely on pre-recorded full recorded track or instrumental version of it instead of live music.

In these cases, because the performers are delivering lines or moving without a live conductor or instruments and vocals to give some cues about the timing, it can be challenging to complete dialogues or movements with precise timing. As a result, actions intended to coincide with specific sound cues—like a dramatic beat or bang—often occur early or late, disrupting the intended impact. The Dance performance scenario addresses this issue by allowing performers to send a signal to the music system, prompting it to continue from a designated point in the track. This approach helps synchronize dialogue and movements with critical musical moments, creating a more cohesive and impactful performance experience. the "dance perfor-

mance scenario" has been designed based on this concept.

Another performance scenario explores creative, expressive control over various sound elements, enabling users to manipulate audio in real time. In this setup, performers can adjust volume, change pitch, and modify the gain of different sound streams, such as drums, instruments, or vocals, using designated gestures. This approach opens up a vast range of possible mappings and connections, allowing users to convey emotion and artistry through the control they have over a playback track. the "real time sound control scenario" has been designed based on this concept.

This scenario resembles the functionality of platforms like Wekinator, often considered a tool for crafting new digital instruments. However, the focus of this study is on enabling performers to make real-time modifications to an existing track, maximizing the capabilities of Max MSP's real-time sound editing environment. By offering flexible sound control, this scenario provides performers with the tools to personalize their interaction with the music, enriching the performance experience and enhancing creative expression.

4.2 Training phase

benefits: fun, dont need user to press any key before the recording phase.

4.2.1 System Learning Process: user view

The training phase for the machine learning model is designed to guide the user through providing effective samples in an intuitive way. Here's how it works:

1. Setting Up the Tempo: The session begins with setting a tempo, and the user can choose between hearing plain beat or the main audio track.

2. Sample Count Selection: The user will specify the number of samples they want to provide in each session—initially, 4 or 8 samples are recommended for ease.

3. Visual Countdown and Readiness to Cue: To prepare the user, a visual countdown will display "3, 2, 1," and along plain beats signaling them to get ready for the initial cue gesture (a gesture that acts as the primary signal for the system to initiate a response).

4. Guided Beat System for Sample Collection: To begin, a green light will turn on, accompanied by a distinct beat sound. by these two signs user knows system is recording a sample of cue gesture. After this initial signal, a repeating sequence of three yellow beats followed by one green beat will play and show. Each green beat serves as the cue for the user to perform the "cue gesture," marking the sample point for the system. This cycle continues until the required number of "cue gestures" has been collected.

   During the yellow beats, the user can either remain neutral or perform other movements typical of their performance. This flexibility allows the model to recognize distinct cues while distinguishing other movements that may occur during real-time performances

5. Confirmation: Once the required number of samples has been collected, the user is prompted with a confirmation step. This allows them to review and ensure that each sample aligns with the intended gestures, providing the option to discard any inaccurate samples before they are fed into the model. This added flexibility ensures only precise data is used for training, enhancing the model's accuracy and reliability.

4.2.2 Machine learning training

During each green light moment, a snapshot of all body position landmarks detected by the MediaPipe module in the Max MSP environment is captured based on a live stream from the user's laptop camera. This snapshot includes the full set of landmarks, representing the user's posture at that exact moment. These data points are then sent as JSON objects to the Python environment via the OSC (Open Sound Control) protocol. Each snapshot taken during a green light moment is labeled as the "main class," indicating the primary gesture intended for the system.

Yellow beats follow a similar data capture process, with each snapshot labeled as an "other" class, representing movements outside the primary gesture. This labeled dataset is then used to train machine learning models capable of classifying gestures based on these captured landmarks. By distinguishing between the main class and other movements, the system learns to accurately identify and respond to the designated gesture in real-time applications. The data collected undergoes preprocessing to standardize and scale each input, ensuring consistency and enhancing the model's ability to recognize gestures effectively. Each body landmark is represented by three

coordinates (x, y, z) from MediaPipe's landmark recognition. However, to simplify the process, only the x and y coordinates are used in this experiment. This approach maintains essential spatial information while reducing complexity.

For gesture mapping, three foundational models were evaluated: Random Forest (RTF), Multi-Layer Perceptron (MLP), and Decision Tree. Each model's performance was assessed based on the number of available data samples to determine which model provided the most accurate and reliable results. After testing multiple gestures and analyzing the outcomes, the MLP model consistently performed best in the majority of experiments, leading to its selection as the primary model for our system. The MLP model's superior accuracy and adaptability make it particularly suited for capturing and interpreting user gestures in real time, providing a reliable solution for our gesture-mapping needs. [a table needed here]

4.3 Training Evaluation

After completing the training phase, the model processes the collected data to learn the intended gesture, providing evaluation metrics to indicate how well it recognizes the movement. If the accuracy falls below an acceptable threshold, the training phase must be repeated to add up more samples and improve the model's understanding and performance.

To ensure optimal quality, the user has the option to test the trained model in a live environment, observing its responsiveness to their gestures in real time. This step allows the user to gauge the system's reaction and confirm that it accurately interprets the gesture. Similar to platforms like Wekinator, this system also allows users to add specific corner cases where the model may struggle, ensuring it learns to handle a wide range of variations. This iterative approach enhances the model's precision, ensuring it reliably interprets the desired gesture across various performance conditions.

After training a gesture, the user can assign a custom name to it, making it easier to identify and manage within the system. Additionally, the system generates a model file and a standard scaler file, which store the trained model and the scaling parameters used during preprocessing. These files can be saved for future use, allowing the user to reload the trained gesture model and apply it in other sessions without retraining, ensuring consistency and ease of deployment across different applications.

in this part user can test different saved gestures to make sure system is able to distinguish different cue gestures.

4.4 Mapping Phase

The gestures saved during the final stage of the training phase are now fully recognizable by the system. At this point, the user can select from various scenarios, each tailored to different performance contexts. Based on the chosen scenario, the user is prompted to provide specific information that links each gesture to the desired audio control functions.

4.5 Action Phase

The system is now ready to recognize the cue gesture and trigger the corresponding action. To accomplish this, it requires an observation window—a defined time period during which it actively monitors and sends snapshots of body landmarks to estimate the probability of detecting the cue gesture. Once the system confidently identifies the desired gesture from the performer, it will execute the assigned action mapped to that gesture.

This observation window can remain open throughout the entire performance, providing continuous responsiveness. However, in scenarios like dance performances, where extraneous movements are common, shortening the observation window can help reduce the chance of misinterpretation, making the system more accurate in detecting only the intended cue gestures.

4.5.1 Scenarios

Dance Performance Scenarios

In the first scenario, the user defines an observation window by setting a start and end time, allowing the system to focus on detecting the cue gesture within a specific timeframe, which enhances accuracy. Additionally, the user identifies a precise cue moment in the soundtrack—such as a bang sound at time t—that is intended to align with the performer's gesture.

The user also configures a latency tolerance window to account for potential timing variations. During the performance, the system begins playback, and once it enters the observation window, it monitors for the cue gesture. If the cue gesture is detected before the observation window ends (indicating an early performance), the system will trigger the bang sound immediately at the specified cue moment (time t) in the

soundtrack. If the system does not detect the cue gesture within the observation window, it extends the listening period by the latency tolerance window to allow for a slightly delayed performance. This latency tolerance window serves as an extension of the observation window, giving the system additional time to detect the gesture. If no cue gesture is detected by the end of this window, the system proceeds with the playback from the bang sound.

These steps are carefully designed to integrate seamlessly with traditional dance performances that use playback music, enhancing performance opportunities without disrupting established timing or choreography.

Real-time sound control

In this scenario, the system enables flexible, intuitive control over sound elements by allowing each gesture to convey both a directional command (e.g., increase or decrease) and a unit value. For example, a gesture like pointing upward can signal an increase in volume by one unit, while pointing downward decreases it by one unit. To add more dynamic sound effects, the user could create a different gesture, such as a whole hand up, to increase the volume by five units, producing a more pronounced change in the audio. This approach empowers users to make both subtle and sudden adjustments to sound levels based on the expressiveness of their gestures.

To ensure reliable gesture recognition, the system scans for gestures at each beat, capturing only a brief series of landmark snapshots. System will response under milliseconds to indicate if any of saved gestures or not. This targeted observation period makes the gesture recognition process more responsive, reducing the likelihood of accidental commands while maintaining precise control timing.

Before activating this functionality, the audio track is divided into separate stems (such as drums, vocals, or other instruments), and the user can assign specific gestures to adjust the gain of each stem independently. This allows for custom gain arrangements for individual stems, enabling performers to create complex audio interactions and nuanced soundscapes. The combination of directional commands, adjustable units, and independent stem control provides an adaptable and expressive tool for live sound manipulation, giving performers creative freedom to shape the auditory experience in real time.

4.6 System Report and Evaluation

The system has been successfully established and tested with two different users, each performing multiple gesture cues. To evaluate performance more rigorously, we measured the system's feedback time, ensuring that it met real-time responsiveness standards. Additionally, we conducted a comparative analysis between our gesture recognition system and MediaPipe's hand gesture recognizer, allowing us to assess relative accuracy and reliability. This comparison provided insights into the system's effectiveness, validating its suitability for dynamic, performance-based applications.

4.6.1 participants analytics

In this study, two users participated in a training phase involving four distinct gesture cues: raising the right hand, raising the left hand, raising the right leg, and raising the left leg. The sample count for each gesture class is detailed in Table 4.1. Given that each user followed their preferred tempo and varied in the number of samples before each confirmation, the total training duration was 6 minutes for User A and 5 minutes for User B. The average time per sample, including both active interaction with the system and rest intervals, was approximately 2.6 seconds.

Gesture	**Participant A (Samples)**	**Participant B (Samples)**
Right Hand Up	58	50
Right Leg Up	36	50
Left Hand Up	36	60
Left Leg Up	24	50

Table 4.1: Number of Samples for Each Gesture by Participants A and B

Training was continued until the system provided accurate, real-time feedback for each user's gestures, responding correctly to distinct, recognizable gestures. The model's precision, recall, F1 score, and accuracy are reported in Table 4.2 and 4.3. Since the number of samples varies between classes and participants, using macro and weighted averages provides a more balanced and meaningful basis for comparison.

As depicted in figure 4.1 and 4.2 , after approximately 60 samples, the model accuracy stabilized, with no significant improvement noted with additional samples. However, this threshold may vary depending on the user and the specific gesture; in our experiment, around 50 samples were sufficient for the system to reliably classify the gestures.

Class	Precision	Recall	F1-Score	Support
Right Hand Up	1.00	1.00	1.00	19
Left Hand Up	1.00	0.89	0.94	9
Right Leg Up	0.80	1.00	0.89	12
Left Leg Up	1.00	0.71	0.83	7
Accuracy		0.94		47
Macro Avg	0.95	0.90	0.92	47
Weighted Avg	0.95	0.94	0.94	47

Table 4.2: Classification Report for participant A based on validation dataset

Class	Precision	Recall	F1-Score	Support
Right Hand Up	1.00	1.00	1.00	16
Left Hand Up	1.00	0.84	0.91	19
Right Leg Up	0.82	1.00	0.90	14
Left Leg Up	1.00	1.00	1.00	14
Accuracy		0.95		63
Macro Avg	0.96	0.96	0.95	63
Weighted Avg	0.96	0.95	0.95	63

Table 4.3: Classification Report for participant B based on validation dataset

Subsequently, we evaluated the cross-user performance of models by testing a model trained on User A's data with User B's input. The lower performance of the User B-trained model on User A's data suggests that model performance can be highly user-specific, benefiting from user-specific data to accelerate convergence and enhance real-time accuracy. Alternatively, a more generalized and stabilized model(like the model trained over participant A), trained with a diverse dataset, could achieve broader applicability if the selected features (landmarks) are not body-specific. Comparative performance data for these models can be found in Table 4.4.

4.6.2 Real-time performance

To evaluate the real-time responsiveness of our system, we measure the total time taken for each complete cycle from capturing body landmarks to processing them, recognizing the cue, and applying corresponding changes in the audio output. This sequence includes capturing body movements through the camera, transmitting these landmarks to the processing unit, recognizing and interpreting the cue, and ultimately triggering the audio response. Our tests show that this entire process consistently occurs within 0.2 seconds. From a human perspective and based on the temporal

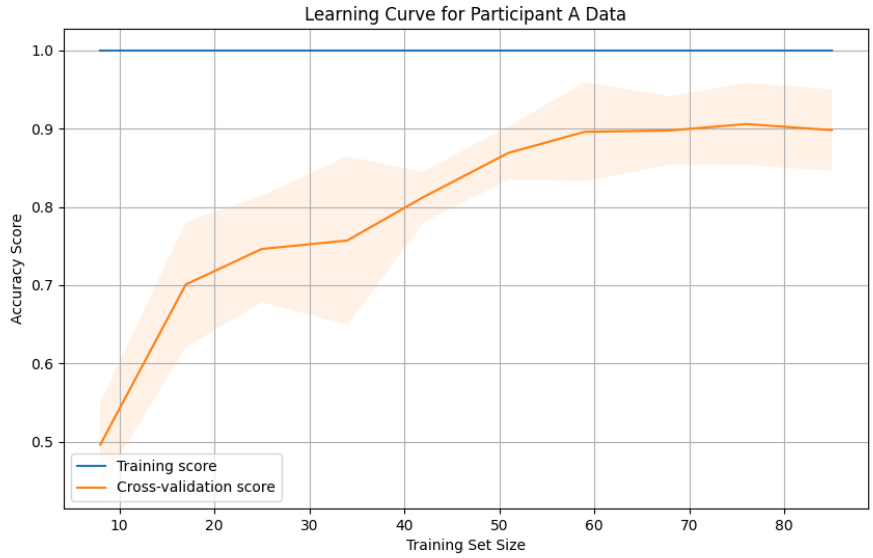

Figure 4.1: user A learning curve

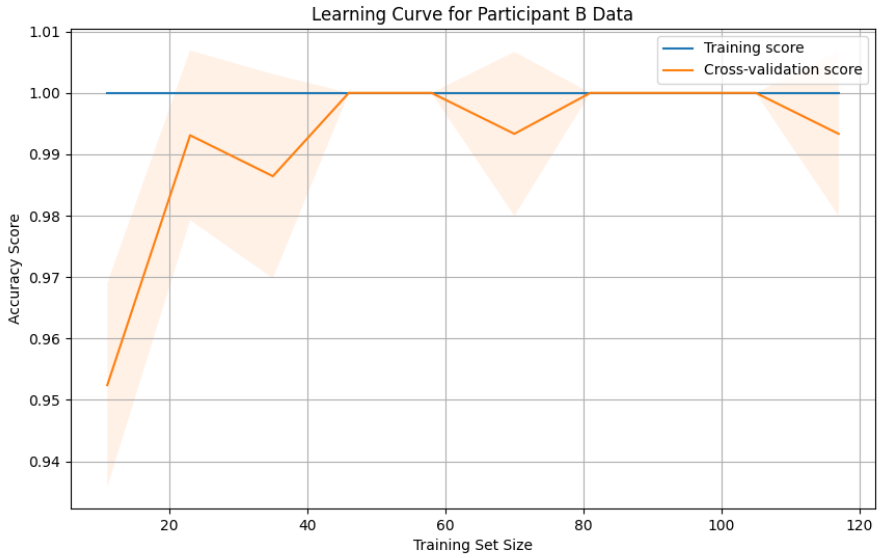

Figure 4.2: user B learning curve

sensitivity requirements of our application, this latency meets the criteria for real-time responsiveness. Achieving this sub-200ms response time ensures that the user experiences immediate feedback to their gestures, creating a seamless and intuitive interaction that aligns with the demands of real-time audio manipulation.

Test description	Class	Precision	Recall	F1-Score	Support
A on B	Right Hand Up	0.71	0.88	0.78	58
	Left Hand Up	0.64	0.81	0.72	36
	Right Leg Up	0.88	0.58	0.70	36
	Left Leg Up	0.31	0.17	0.22	24
	Accuracy	0.68			154
	Macro Avg	0.63	0.61	0.60	154
	Weighted Avg	0.67	0.68	0.66	154
B on A	Right Hand Up	1.00	0.98	0.99	50
	Left Hand Up	0.98	1.00	0.99	50
	Right Leg Up	1.00	0.97	0.98	60
	Left Leg Up	0.96	1.00	0.98	50
	Accuracy	0.99			210
	Macro Avg	0.99	0.99	0.99	210
	Weighted Avg	0.99	0.99	0.99	210

Table 4.4: Cross-User Performance of Models: User A Data Tested with Model Trained on User B Data (A on B) and Vice Versa (B on A)

4.6.3 comparative classification evaluation

To evaluate the performance of our gesture classification system in comparison to an established system, we conducted an experiment using MediaPipe's hand gesture recognition model as a benchmark. For this experiment, we replicated the setup of our main experiment with one key difference: this time, we focused exclusively on hand landmarks instead of full-body landmarks. Our aim was to determine how closely our model could approximate the accuracy of MediaPipe's well-regarded system.

We categorized the hand gestures into three main classes for this comparison:

- Open Palm: A gesture where the hand is open with the palm visible.
- Closed Fist: A gesture where the hand is closed into a fist.
- Others: A category that encompasses at least five different common hand

gestures that did not fit into the "Open Palm" or "Closed Fist" categories. In this setup, we used MediaPipe's hand gesture labeling as the ground truth for accuracy measurement. The purpose was to assess our model's performance in approximating MediaPipe's results and to understand the effectiveness of our training system when applied to hand gesture classification. Our model achieved an impressive 90% similarity in performance to the MediaPipe model, with high precision and recall across most categories.

Class	F1-Score	Support
Open Palm	0.96	64
Close Fist	0.89	71
Others	0.89	107
Accuracy	0.91	
Macro Avg	0.91	242
Weighted Avg	0.91	242

Table 4.5: Comparison of Our Model with MediaPipe Hand Gesture Recognition System

4.7 Summary of Findings

In this chapter, we examined the feasibility and effectiveness of a gesture-based sound control system in performance contexts. By simulating realistic scenarios, we assessed both technical requirements and user interaction methods for integrating gesture recognition with sound manipulation. Our approach involved two primary test cases: a dance performance simulation and real-time sound control scenario, each illustrating distinct performance needs and operational workflows.

Key findings include:

1. Gesture Recognition Accuracy: Through systematic training and real-time testing, we achieved over 90% similarity to other gesture detection systems, demonstrating the system's reliability for live, interactive environments.

2. Responsiveness and Latency: The system showed minimal latency, meeting the real-time responsiveness essential for synchronizing gestures with sound cues, which is vital for dynamic and improvisational settings.

3. User-Friendly Training Process: The training phase, which utilized a metronome-guided sequence, effectively separated the primary gesture from incidental movements, enhancing both user experience and model precision.

4. Flexibility in Sound Control: By mapping gestures to various sound modifications, such as pitch, volume, and gain control, the system enabled expressive, intuitive sound manipulation, offering performers creative autonomy over audio in real time.

Our comparison with established tools like MediaPipe's hand gesture recognizer further validated the system's practicality and adaptability in performing arts and

interactive installations. The study's findings lay a foundation for future developments in gesture-based sound control, enabling performers to explore novel ways of connecting movement and sound, thus expanding the expressive potential in live performances.

Chapter 5

Conclusions

This thesis has introduced a real-time gesture-based sound control system that offers performers an intuitive and customizable interface for manipulating sound through body movements. This work represents a significant step in interactive music technology, empowering artists to shape sound in ways that suit their unique style and needs. By bridging human movement with machine learning, the system delivers a highly adaptable and expressive tool, making it accessible to users with or without technical backgrounds. This conclusion will review the key contributions of the research, reflect on its limitations, discuss potential directions for future work, and offer closing thoughts on the system's broader implications.

Summary of Key Contributions

1. **Empowering Performers with Customizable Training:** One of the novel aspects of this system is its adaptability to the performer's needs. The user can configure specific parameters, such as the listening window, to refine the system's accuracy and responsiveness. This degree of customization allows performers to personalize their interaction with sound, making it possible to adapt the system to various performance contexts. By giving performers control over model training and real-time adjustments, this system goes beyond typical gesture recognition tools, enabling users to tailor the system to their specific movements and style, which significantly enhances their creative expression.

2. **Versatile Data Generalization for Broader Usability:** This research demonstrates that with a sufficiently comprehensive dataset, the system can generalize well across different users, making it adaptable and versatile for var-

ious applications. The ability to capture diverse gestures and create a reliable, reusable model offers a foundation for broader usability without requiring constant recalibration. This adaptability means that the system could be employed in settings like live performances, interactive installations, or even therapeutic environments. The findings underscore that with a rich dataset, gesture-based sound control systems can cater to a wide audience, expanding their use beyond specific individual performances.

3. **Machine Learning for Creative Control in Non-Technical Applications:**By integrating machine learning, this system allows performers to control sound through expressive, natural movements, bringing advanced sound manipulation within reach for non-technical users. Traditionally, adjusting sound parameters through code has posed a barrier for many artists. This system eliminates that obstacle, enabling intricate sound manipulations through intuitive body gestures without requiring programming skills.

   Moreover, this approach enables seamless use of the Python environment, allowing the system to leverage powerful machine learning frameworks such as PyTorch and TensorFlow. These libraries, recognized as state-of-the-art tools in machine learning, allow for the implementation of sophisticated models, enhancing the accuracy and flexibility of gesture recognition. This research demonstrates that by integrating Python-based tools, machine learning can effectively support creative tasks in real time, transforming complex coding functions into interactive, user-friendly experiences that are accessible, adaptable, and creatively enriching.

Reflections on Limitations

Despite its success, this work encountered some limitations, particularly in handling diverse environmental conditions and fine-grained gesture distinctions. Factors like lighting variations and background interference can affect the system's gesture recognition accuracy, especially in uncontrolled settings. Another limitation is the need for users to undergo initial model training, which may require dedicated time for optimal performance. Although the system allows for flexible configuration, individual differences in body movements may still affect the generalizability of the model. Addressing these limitations would enhance the system's robustness, allowing it to function reliably in more varied and complex environments.

Future Work and Directions

Building on these findings, several promising directions for future research emerge:

1. **Enhanced Environmental Robustness:** To broaden its usability, future versions of this system could incorporate algorithms that adapt to changing lighting and environmental factors, making the system more resilient in varied settings. Exploring additional sensors or data sources, such as depth cameras or inertial sensors, could improve gesture recognition under challenging conditions.

2. **Adaptive and Reinforcement Learning for Personalized Interaction:** Combining adaptive learning, online learning, and reinforcement learning (RL) offers significant potential to enhance the system's gesture-to-sound mapping capabilities. RL could be utilized to dynamically refine mappings by learning from user interactions and feedback during performances, ensuring optimal responsiveness and personalization. For example, RL can adjust mappings to suit a performer's style, preferences, or specific context.

   Online learning, as a subset of adaptive learning, would enable the system to continuously update its gesture models in real-time, accommodating variations such as new users, evolving gestures, or changing environmental conditions. This would eliminate the need for extensive retraining while maintaining high accuracy and responsiveness. By combining RL's long-term optimization with the immediate adaptability of online learning, the system could evolve seamlessly during live use, providing performers with an increasingly intuitive and personalized experience.

3. **Leveraging Python's Machine Learning Ecosystem** By interfacing with Python, the system can utilize a vast array of machine learning libraries such as TensorFlow, PyTorch, and Scikit-learn. These tools offer state-of-the-art algorithms and pre-built components for RL, online learning, and adaptive systems, facilitating rapid experimentation and implementation. Python's ecosystem ensures that advanced learning models can be seamlessly integrated, enabling dynamic, real-time learning and improving both the performance and adaptability of gesture-based sound control systems.

4. **Advanced Motion Detection for Complex Gestures:** In the current experimental setup, gesture detection relies solely on a single frame to identify

each gesture. While this approach captures a static snapshot, it may fall short of effectively capturing full-body expressions or movements that unfold over time. Complex gestures and body expressions are often best understood as a sequence of motions, where the timing and order of actions are essential for conveying the intended meaning.

Expanding the system to recognize movements as a series of gestures allows for more nuanced interpretation and the detection of more intricate patterns in user actions. By defining movement as a sequence of gestures, the system can begin to understand and classify complex actions, such as dance sequences or choreographed motions, which are composed of smaller, ordered gestures. This extension enables the system not only to detect isolated gestures but also to interpret multi-step actions in a continuous flow, adding depth to the gesture recognition capabilities.

To implement this, methods such as Hidden Markov Models (HMM) and Viterbi Decoding can be employed. HMMs provide a probabilistic framework for modeling sequences, where each state represents a specific gesture in the movement sequence. By training the model on gesture sequences, the system learns the likely transitions between gestures, effectively recognizing movement patterns over time. Viterbi Decoding complements this by finding the most probable sequence of gestures from the observed data, allowing the system to accurately predict the intended movement even in the presence of slight variations. Together, these methods enhance the system's ability to recognize structured movements, providing a powerful tool for real-time movement recognition in performance and interactive applications.

5. **Integration with Advanced Gesture Recognition Technologies:** By incorporating more sophisticated gesture recognition techniques—such as 3D pose estimation or deep learning-based classifiers—the system could better capture complex or subtle gestures. This enhancement would allow performers to experiment with intricate gestures, adding a new layer of expressiveness to the sound control interface.

6. **Expansion into Therapeutic and Educational Applications:** Given its interactive and user-friendly design, this system has potential applications in therapy and education, where gesture-based control can provide engaging, hands-

on experiences. Future research could explore these applications, adapting the system for use in environments like music therapy, physical rehabilitation, or interactive learning.

Closing Remarks

In summary, this work has presented an innovative, real-time gesture-based sound control system that empowers performers to shape sound dynamically and interactively. By providing a customizable, accessible interface, the system brings advanced sound manipulation within reach for artists and performers without programming knowledge. This approach opens new possibilities for creative expression, allowing users to interact intuitively with music and sound in real time.

The achievements of this research highlight the potential for future developments in gesture-controlled systems, showing that complex, adaptive sound manipulation can be achieved with minimal setup. By seamlessly integrating human movement with machine learning, this system lays the foundation for a future where performers can interact naturally and expressively with technology, creating a richer and more immersive experience in sound-based interactions. Through this research, we have demonstrated that the intersection of gesture recognition, machine learning, and real-time sound control offers a promising pathway to more inclusive, engaging, and innovative multimedia experiences.

Bibliography

[1] Federico Ghelli Visi and Atau Tanaka. Interactive machine learning of musical gesture, 2020.

[2] Atau Tanaka and Marco Donnarumma. The body as musical instrument. *The Oxford handbook of music and the body*, pages 79–96, 2019.

[3] Federico Visi, Esther Coorevits, Rodrigo Schramm, and Eduardo Miranda. Musical instruments, body movement, space, and motion data: music as an emergent multimodal choreography. 2017.

[4] Stefan Östersjö. Go to hell: towards a gesture-based compositional practice. *Contemporary Music Review*, 35(4-5):475–499, 2016.

[5] Alessandro Altavilla, Baptiste Caramiaux, and Atau Tanaka. Towards gestural sonic affordances. 2013.

[6] Rolf Inge Godøy and Marc Leman. *Musical gestures: Sound, movement, and meaning*. Routledge, 2010.

[7] Antonio Camurri, Shuji Hashimoto, Matteo Ricchetti, Andrea Ricci, Kenji Suzuki, Riccardo Trocca, and Gualtiero Volpe. Eyesweb: Toward gesture and affect recognition in interactive dance and music systems. *Computer Music Journal*, 24(1):57–69, 2000.

[8] Jofish Kaye and ACM Special Interest Group on Computer-Human Interaction. *Proceedings of the 2016 CHI conference extended abstracts on human factors in computing systems*. ACM, 2016.

[9] Atau Tanaka. Intention, effort, and restraint: The emg in musical performance. *Leonardo*, 48(3):298–299, 2015.

[10] Atau Tanaka, Balandino Di Donato, Michael Zbyszynski, and Geert Roks. Designing gestures for continuous sonic interaction. 2019.

[11] Michael Zbyszyński, Balandino Di Donato, Federico Ghelli Visi, and Atau Tanaka. Gesture-timbre space: Multidimensional feature mapping using machine learning and concatenative synthesis. In *International Symposium on Computer Music Multidisciplinary Research*, pages 600–622. Springer, 2019.

[12] Antonio Camurri, Barbara Mazzarino, and Gualtiero Volpe. Analysis of expressive gesture: The eyesweb expressive gesture processing library. In *Gesture-Based Communication in Human-Computer Interaction: 5th International Gesture Workshop, GW 2003, Genova, Italy, April 15-17, 2003, Selected Revised Papers 5*, pages 460–467. Springer, 2004.

[13] Alexander Refsum Jensenius, Rolf Inge Godøy, and Marcelo M Wanderley. Developing tools for studying musical gestures within the max/msp/jitter environment. 2005.

[14] Alexander Refsum Jensenius. The musical gestures toolbox for matlab. 2018.

[15] Tamar Flash and Neville Hogan. The coordination of arm movements: an experimentally confirmed mathematical model. *Journal of neuroscience*, 5(7):1688–1703, 1985.

[16] Stefano Piana, Alessandra Staglianò, Francesca Odone, and Antonio Camurri. Adaptive body gesture representation for automatic emotion recognition. *ACM Transactions on Interactive Intelligent Systems (TiiS)*, 6(1):1–31, 2016.

[17] Diego Fenza, Luca Mion, Sergio Canazza, and Antonio Roda. Physical movement and musical gestures: a multilevel mapping strategy. *Proceedings of Sound and Music Computing*, 5:24–26, 2005.

[18] Donald Glowinski, Nele Dael, Antonio Camurri, Gualtiero Volpe, Marcello Mortillaro, and Klaus Scherer. Toward a minimal representation of affective gestures. *IEEE Transactions on Affective Computing*, 2(2):106–118, 2011.

[19] Kozaburo Hachimura, Katsumi Takashina, and Mitsu Yoshimura. Analysis and evaluation of dancing movement based on lma. In *ROMAN 2005. IEEE International Workshop on Robot and Human Interactive Communication, 2005.*, pages 294–299. IEEE, 2005.

[20] Federico Visi, Rodrigo Schramm, and Eduardo Miranda. Gesture in performance with traditional musical instruments and electronics: Use of embodied music cognition and multimodal motion capture to design gestural mapping strategies. In *Proceedings of the 2014 International Workshop on Movement and Computing*, pages 100–105, 2014.

[21] Meinard Müller. Information retrieval for music and motion. *Information Retrieval for Music and Motion*, 2007.

[22] Angkoon Phinyomark, Pornchai Phukpattaranont, and Chusak Limsakul. Feature reduction and selection for emg signal classification. *Expert systems with applications*, 39(8):7420–7431, 2012.

[23] Zainal Arief, Indra Adji Sulistijono, and Roby Awal Ardiansyah. Comparison of five time series emg features extractions using myo armband. In *2015 international electronics symposium (IES)*, pages 11–14. IEEE, 2015.

[24] Kang Soo Kim, Heung Ho Choi, Chang Soo Moon, and Chi Woong Mun. Comparison of k-nearest neighbor, quadratic discriminant and linear discriminant analysis in classification of electromyogram signals based on the wrist-motion directions. *Current applied physics*, 11(3):740–745, 2011.

[25] Rajesh G Bachu, S Kopparthi, B Adapa, and Buket D Barkana. Voiced/unvoiced decision for speech signals based on zero-crossing rate and energy. In *Advanced techniques in computing sciences and software engineering*, pages 279–282. Springer, 2010.

[26] Baptiste Caramiaux, Marco Donnarumma, and Atau Tanaka. Understanding gesture expressivity through muscle sensing. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 21(6):1–26, 2015.

[27] Cycling '74 (2019) max software. `https://cycling74.com/products/max`. Accessed: 2024-01-01.

[28] Diemo Schwarz. Corpus-based concatenative synthesis. *IEEE signal processing magazine*, 24(2):92–104, 2007.

[29] Mehryar Mohri. Foundations of machine learning, 2018.

[30] Trevor Hastie, Robert Tibshirani, Jerome H Friedman, and Jerome H Friedman. *The elements of statistical learning: data mining, inference, and prediction*, volume 2. Springer, 2009.

[31] Federico Ghelli Visi and Atau Tanaka. Towards assisted interactive machine learning: exploring gesture-sound mappings using reinforcement learning. In *ICLI 2020—the fifth international conference on live interfaces*, pages 9–11, 2020.

[32] Pavel Senin. Dynamic time warping algorithm review. *Information and Computer Science Department University of Hawaii at Manoa Honolulu, USA*, 855(1-23):40, 2008.

[33] Frédéric Bevilacqua, Bruno Zamborlin, Anthony Sypniewski, Norbert Schnell, Fabrice Guédy, and Nicolas Rasamimanana. Continuous realtime gesture following and recognition. In *International gesture workshop*, pages 73–84. Springer, 2009.

[34] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 4724–4732, 2016.

[35] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[36] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7291–7299, 2017.

[37] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019.

[38] Lili Fan, Hong Rao, and Wenji Yang. 3d hand pose estimation based on five-layer ensemble cnn. *Sensors*, 21(2):649, 2021.

[39] Matthew Chen and Melvin Low. Recurrent human pose estimation.

[40] Charmi Jobanputra, Jatna Bavishi, and Nishant Doshi. Human activity recognition: A survey. *Procedia Computer Science*, 155:698–703, 2019.

[41] Asha Kapur, Ajay Kapur, Naznin Virji-Babul, George Tzanetakis, and Peter F Driessen. Gesture-based affective computing on motion capture data. In *Affective Computing and Intelligent Interaction: First International Conference, ACII 2005, Beijing, China, October 22-24, 2005. Proceedings 1*, pages 1–7. Springer, 2005.

[42] Siddharth S Rautaray. Real time hand gesture recognition system for dynamic applications. *International Journal of ubicomp (IJU)*, 3(1), 2012.

[43] Yann Maret, Daniel Oberson, and Marina Gavrilova. Real-time embedded system for gesture recognition. In *2018 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 30–34. IEEE, 2018.

[44] Fernando Trebien and Manuel M Oliveira. Realistic real-time sound re-synthesis and processing for interactive virtual worlds. *The Visual Computer*, 25:469–477, 2009.

[45] Fiona Keenan and Sandra Pauletto. A mechanical mapping model for real-time control of a complex physical modelling synthesis engine with a simple gesture. In *International Conference on Digital Audio Effects (DAFx17)*, pages 25–31, 2017.

[46] Laurent Pruvost, Bertrand Scherrer, Mitsuko Aramaki, Sølvi Ystad, and Richard Kronland-Martinet. Perception-based interactive sound synthesis of morphing solids' interactions. In *SIGGRAPH Asia 2015 Technical Briefs*, pages 1–4. 2015.

[47] Rolf Nordahl, Stefania Serafin, and Luca Turchet. Sound synthesis and evaluation of interactive footsteps for virtual reality applications. In *2010 IEEE Virtual Reality Conference (VR)*, pages 147–153. IEEE, 2010.

[48] Adrian von dem Knesebeck, Sebastian Kraft, and Udo Zölzer. Realtime system for backing vocal harmonization. In *Proc. of the 14th Int. Conference on Digital Audio Effects*, 2011.

[49] Rebecca Fiebrink and Perry R Cook. The wekinator: a system for real-time, interactive machine learning in music. In *Proceedings of The Eleventh International*

Society for Music Information Retrieval Conference (ISMIR 2010)(Utrecht), volume 3, pages 2–1. Citeseer, 2010.

[50] Songpei Xu, Chaitanya Kaul, Xuri Ge, and Roderick Murray-Smith. Continuous interaction with a smart speaker via low-dimensional embeddings of dynamic hand pose, 2023.

[51] Eunjeong Stella Koh and Shahrokh Yadegari. Mugeetion: Musical interface using facial gesture and emotion, 2018.

Appendix A

Additional Information

Sonic Affordance: In human-computer interaction and sound design, sonic affordance refers to the potential for action or interaction that a sound implicitly communicates to a user. It denotes how sound characteristics (such as timbre, pitch, and rhythm) suggest particular ways of engagement or evoke specific responses. This concept originates from affordance theory, which explores how elements in the environment convey cues for action. In the context of gesture-based sound control, sonic affordance helps guide users by making sound cues intuitively understandable, facilitating seamless and natural interaction within the system.

Embodied Engagement: In the context of human-computer interaction and gesture-based sound control, embodied engagement refers to the immersive, physical involvement of users as they interact with a system through bodily movements. This concept highlights the role of the body in cognition, suggesting that understanding and interaction arise not only from cognitive processing but also through physical actions and sensory feedback. Embodied engagement allows users to experience a direct and intuitive connection with sound, as their gestures translate into real-time auditory responses, reinforcing a sense of presence and control within the interactive environment.

Wind Machine Instrument: A wind machine instrument is a device used in sound design and music to simulate the sound of blowing wind. It typically consists of a large cylindrical drum with slats or paddles, which are rotated by a handle or motor. When turned, the air moving through the slats creates a turbulent sound reminiscent of wind, and variations in speed produce changes in intensity and tone. This instrument is commonly used in theatrical productions, film scoring, and soundscapes to evoke natural ambiance and atmospheric effects.