WaveTune: Harnessing Radar Sensors to Compose Music Beats with Body Gestures

Juneja, Suchdeep Singh ; Vaidya, Girish ; Zuniga, Marco

DOI
10.1109/DCOSS-IoT61029.2024.00029

Publication date
2024

Document Version
Final published version

Published in
Proceedings of the 2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT)

2024 20th International Conference on Distributed Computing in Smart Systems and the Internet of Things (DCOSS-IoT) | 979-8-3503-6944-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/DCOSS-IoT61029.2024.00029

WaveTune: Harnessing Radar Sensors to Compose Music Beats with Body Gestures

Suchdeep Singh Juneja*, Girish Vaidya*†, Marco Zuniga*
* *Delft University of Technology, Delft, Netherlands*
† *Amsterdam Institute for Advanced Metropolitan Solutions, Amsterdam, Netherlands*
Email: s.s.juneja@student.tudelft.nl, g.vaidya@tudeflt.nl, m.a.zunigazamalloa@tudeflt.nl

***Abstract*—In the field of music technology, the ability to connect body movements with music generation has led to highly engaging applications. Many existing solutions use cameras or wearable devices to capture physical movements and translate them into commands for a music generation tool. While the camera-based systems are privacy-invasive and demand good lighting conditions, the wearable devices adversely impact the fluidity of movements. We propose WaveTune, a privacy-friendly and device-free interface for creating music beats through body gestures. Wavetune has the following components: (i) a millimeter-wave radar as the frontend sensor that captures body movements, (ii) a gesture recognition algorithm that performs gesture delimitation and identification in real-time, and (iii) a music generation component that maintains a seamless and pleasant experience. WaveTune also provides an option to continuously map random dynamic movements into musical parameters to encourage further interaction. We recruited 24 users to train and test WaveTune's ability to map body gestures to musical commands. To the best of our knowledge, WaveTune is the first mmWave system that allows a *layered* composition of music *beats*.**

I. Introduction

There has been a long-standing interest to interact with music through physical movements. 'The Hands' project proposed music generation through body motion way back in 1984 [1]. Controlling and generating music through motion adds a visual element to the sound, making the performance more expressive and impactful. Existing approaches for translating motion into music include: (a) Camera-based systems and (b) Wearable devices. For instance, the LEAP Motion Controller uses a camera to infer physical movements [2]. This approach offers a natural and intuitive way to interact with music. On the other hand, Mi.Mu Gloves [3] and SOMI-1 [4] use wearable sensors to capture motion. In both of these approaches, the movements are translated into commands that alter specific musical parameters.

Limitations of current approaches. Camera-based solutions, however, demand optimal lighting conditions and unobstructed line-of-sight. Furthermore, they could be perceived as intrusive since they capture full body images, potentially compromising the privacy of people in the scene [5]. On the other hand, wearable devices preserve privacy at the expense of using body-mounted devices. The use of wearable devices could impact the spontaneity and aesthetics of movements. Furthermore, using interfaces such as Mi.Mu Gloves requires an extensive calibration process and familiarity with the system to master fine movements and understand their impact on the generated music [6].

Technology	Privacy	Ease of use
Camera	×	✓
Wearable-devices	✓	×
mmWave radar	✓	✓

TABLE I: Comparison of sensors for music generation.

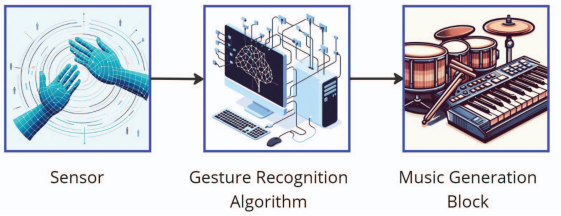


Fig. 1: WaveTune's components. [Images generated using Image Creator from Microsoft Bing.]

A novel approach to simultaneously address the challenges of privacy and fluidity has been explored in *O Soli Mio* [7]. The platform uses a millimeter wave (mmWave) radar to capture physical movements. The radar represents objects within its range as *point clouds*. Unlike cameras, these point clouds do not provide fine-grained information about people, but together with machine learning methods, the point clouds can be used to identify gestures that modify music tones.

Soli Mio inspires our work since it provides privacy, but it has two shortcomings. *First*, there is not much flexibility for music composition. Soli Mio *uses conventional machine learning methods that can recognize only one gesture*. Having a single command limits interaction because users can only modify one parameter, such as the tempo. Music, however, has multiple layers (melody, drums, tempo), and exploring these layers requires a richer set of gestures. *Second*, Soli Mio is not open source. It uses a radar system exclusively integrated with Google products such as Pixel Phones and Nest [8]. To unleash the potential of a tool that composes music with body gestures, it is desirable to have an open-source project that builds upon proven state-of-the-art methods for gesture recognition and music composition.

As summarized in Table I, mmWave sensors provide privacy and ease of use compared to camera-based and wearable-based approaches, but the SoA (Soli Mio) allows minimal levels of music interaction.

Contribution. We propose an alternative to create richer musical beats. Our system, called *WaveTune*, generates beats through body gestures in a layer-by-layer fashion. For example, a user can add a layer of percussion to an existing melody or

2325-2944/24/$31.00 ©2024 IEEE
DOI 10.1109/DCOSS-IoT61029.2024.00029

add a bass layer to an underlying rhythm. WaveTune enables this layered composition by recognizing body movements and translating them into instructions for a music generation tool.

We envision WaveTune as a platform that encourages people to create music beats in a playful manner. For example, nowadays, many artists first create beats with music tools and then add lyrics. With WaveTune, schools could have more engaging music lessons, where children use body movements to create a beat, learning in that way the basics of modern music composition without invading the children's privacy with cameras or wearables.

Overall, WaveTune makes the following contributions.

1) A beat generation tool that does not require wearable devices or cameras [Sections II and III]. WaveTune protects user privacy while retaining movement fluidity. WaveTune's components, shown in Figure 1, consist of (i) a radar sensor to capture cloud points, (ii) a gesture recognition algorithm that maps cloud points into commands, and (iii) a music controller that maps commands to different musical layers.

2) An optimized algorithm to recognize gestures for beat composition [Section IV]. We analyze the state-of-the-art to identify the best gesture recognition algorithm and modify it to suit the needs of music composition. Our algorithm filters irrelevant gestures, identifies the beginning and end of the desired gestures, and decodes them in real time.

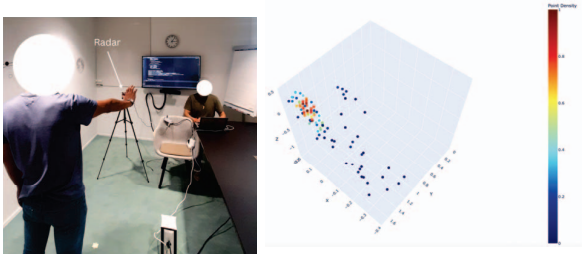
3) A controller that creates beats on a layer-per-layer basis [Section V]. The controller maps gestures to musical commands and its design focuses on *continuity* and *ease of use*. To be pleasant, a musical experience should not be intermittent but fluid (continuous) and to be easy to use, the programming language and protocol need to be rich but simple.

4) A detailed evaluation of WaveTune [Section VI]. We gather 24 users of different ages and genders to demonstrate that WaveTune can detect the intended gestures with 93.5% accuracy. We also provide a video at https://youtu.be/mvcjfo-Iffs, where we demonstrate WaveTune. We encourage the reader to watch the video at this point to attain a better understanding of the remainder of the paper. Our system will be made publicly available.

II. System Overview

A layered approach for music generation. In traditional musical settings, such as those in an orchestra, multiple instruments work together to produce a rich and harmonious composition. Instruments like violins often carry the melody, trumpets provide highlights, and percussion instruments establish the rhythm. Each instrument or group of instruments contributes to a unique layer in the composition. When these layers intertwine harmoniously, they together generate a *musical beat.* Digital Audio Workstations (DAWs) are essential tools in modern music production and they adopt this multi-layered approach. For example, in the case of Ableton Live – a widely-used DAW, users can assign separate tracks to different instruments or sounds, and then craftily, modify or merge these tracks to achieve the desired musical composition.

We adopt a similar approach of using layers for music beat generation. Our current implementation focuses on three



(a) Office scenario (b) Point cloud for the Stop Gesture.

Fig. 2: WaveTune operation: A person performs gestures in front of a mmWave radar that are transformed into a musical track.

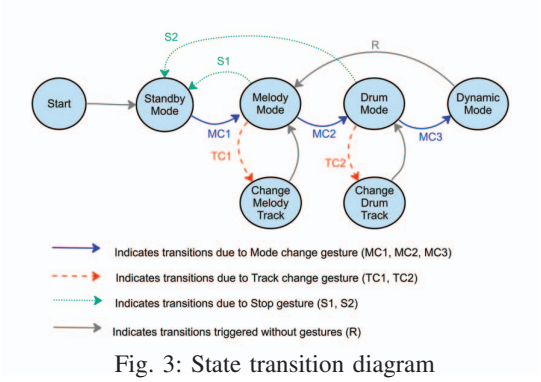


Fig. 3: State transition diagram

layers: Melody, Drums, and Dynamic. WaveTune is inherently flexible and can be expanded to include additional layers. Each layer in WaveTune operates as a distinct mode, with transitions between modes triggered through gestures. Next, we describe in detail how users compose music with WaveTune.

System setup and Music composition We evaluate WaveTune in two different environments, an office space and a living room, the office space is shown in Figure 2. The radar is mounted at a height of 1.2 m using a tripod with an elevation angle of 0 deg. On the floor, we place a small elliptic area, around 60 cm x 40 cm, where the user should stand. The elliptic area is at a distance of approximately 1.5 m. The user can stand on any part of the elliptic area but facing the radar.

The radar captures the movements as a point cloud and transmits it over a serial port to a computer. The computer runs the gesture recognition algorithm and the music controller. The gestures interpreted by the algorithm are relayed as commands to the music controller that creates the music layer-by-layer. An external speaker is used to play the composed music beat in real-time. The music composition process is captured by the state transition diagram in Figure 3 and it is explained next:

1) Standby mode. After starting the system, the radar begins gathering cloud points. The gesture recognition algorithm ignores all movements, except the *Mode change* (MC) gesture, shown in Figures 4a. Once this gesture is provided, the controller starts the music composition process.

2) Melody mode. Once the MC gesture is provided (transition MC1 in Figure 3), the controller starts playing the first melody from a list of preset melodies. The user can navigate through

(a) Mode Change gesture. (b) Track Change gesture. (c) Stop gesture.

Fig. 4: Gestures in WaveTune that act as commands to transition through the state machine.

this list using the *Track change* (TC1) gesture until s/he finds the desired melody.

3) Drum mode. Once the desired melody is found, the user can switch to the Drum Mode by performing again the *Mode change* gesture (transition MC2 in Figure 3). In Drum mode, the user can choose a specific drum rhythm, from a collection of preset drum tracks, using the *track change* gesture (transition TC2).

4) Dynamic mode. The dynamic mode is accessed by performing the *mode change* gesture (transition MC3). In this mode, gesture recognition is disabled for 15 s because the goal is to introduce musical effects. The frequency effect of the composed tracks (melody and drums) is modified based on how fast the user moves the arms. The frequency effect can make a sound warmer or more resonant, impacting the perception of music. The dynamic mode offers users the ability to enhance the expressiveness and feel of their compositions through random body motions.

5) Calibration mode [Optional]. Since the dynamic mode makes use of movement velocity, we provide an optional calibration mode before the standby mode. This optional mode helps WaveTune personalize the user experience with a more fine-tuned mapping between the frequency effect and the arm movements.

6) Stopping music composition. After the dynamic mode ends, WaveTune transitions back to the Melody mode (transition R). The user can then continue to transition between the Melody, Drum, and Dynamic modes through mode change gestures. If a user performs a *Stop gesture* (S) anytime in Melody mode or Drum mode, the system goes back to the standby mode (transitions S1 and S2).

Gestures act as commands to the music generation tool, and from the description of the state diagram, we observe that WaveTune requires three gestures: mode change, track change and stop, plus a dynamic mode. To define these three gestures, we select options that are intuitive and easy to perform. Figures 4a, 4b, and 4c describe the mode change, track change, and stop gestures respectively.

III. Sensor setup and data capture

In this section, we describe what frequency and radar parameters are better suited for WaveTune.

Operating frequency and off-board processing. For WaveTune, we choose the IWR1443 sensor from Texas Instruments (TI) which operates in the 77 GHz - 81 GHz range. The IWR1443 has 3 transmitters and 4 receivers and provides three important advantages for our application compared to other alternatives [9]: (i) fine angular resolution, enabling more detailed cloud points, (ii) lower latency, facilitating timely responses, and (iii) as it will be described later, two SoA platforms for gesture recognition utilize the same sensor, Pantomime [10] and Tesla-Rapture [11], validating the sensor's strengths for capturing gestures.

Parameter configuration. Radar sensors have multiple parameters that usually trade off range versus granularity. Our objective is to get a distinct and dense point cloud that aids gesture recognition. Table II provides the configuration parameters of our setup. Considering the radar's field of view, an approximate distance of 1.5 m from the sensor is sufficient to capture gestures from people of different heights. Also, from our initial experiments, we determined that the maximum gesture velocity is lower than 5 m/s. Accordingly, we set the maximum unambiguous range to 2.5 m and the maximum radial velocity to 6.9 m/s. With those values, the range and velocity resolutions are set to 0.047 m and 0.87 m/s.

After a scan, the radar embeds the point clouds into a single frame together with additional information such as frame number and checksum. Given that gestures take a few seconds, WaveTune needs to collect successive frames to capture the entire gesture and classify it. This process is presented in the next section.

Parameter	Value
Operating Frequency	77 GHz
Range Resolution	0.047 m
Maximum Unambiguous Range	2.5 m
Maximum Radial Velocity	6.9 m/s
Radial Velocity Resolution	0.87 m/s
Frame Rate	30 fps
Range Detection Threshold	15 dB

TABLE II: Key mmWave radar parameters.

IV. Gesture recognition algorithm

The transitions across various modes in WaveTune are enabled by gestures. Thus, the gesture recognition algorithm plays a critical role. WaveTune performs gesture recognition in two steps: first, by identifying the start and end of a gesture, and second, by determining the type of gesture. The first step is performed by the delimiter identification algorithm, whereas we use a neural network model to determine the gesture.

A. Delimiter identification algorithm

We design a method to automatically identify the beginning and end of gestures to maintain the fluidity of movements. A valid movement is detected by reading the velocity information in the point cloud. A frame having at least three points with non-zero velocity is considered a *valid* frame, whereas other frames are designated as *idle*.

Tracking a gesture starts after receiving a valid frame. The goal is to capture N valid frames while ignoring any

intermediate idle frames. In practice, this translates to using the frames with richer information, when the person is actually moving. The sensor is set to 30 fps and we estimate that the average time of a gesture is around 3 seconds. Note that in a 3-second window, we can have 90 frames, both valid and idle, but we only consider the first N valid frames. Our analysis shows that 50 frames is an optimal value, ensuring that the entire gesture is captured without introducing excessive redundancy or risking data loss. Figure 2b shows an example of 50 active frames related to the Stop gesture.

After 50 valid frames are detected, the data is sent to the algorithm for gesture recognition. While the gesture is being recorded, a counter keeps track of the number of valid and idle frames. An idle duration greater than twenty consecutive frames indicates that the event was not triggered by a gesture but by incidental movement. In this case, we discard the recorded frames and start all over again. The neural network model requires a uniform input structure, but the sampled frames generally contain varying numbers of points. The method to guarantee a uniform input is described next.

B. Data preprocessing

State-of-the-art studies in gesture recognition have reported that using 32 frames per gesture, with 32 points per frame, provides the best input to maximize the accuracy of a neural network model [11]. We, thus, follow these guidelines. To map 50 frames into 32 frames, we divide the original frames into 18 groups of two frames and 14 individual frames. The cloud points in the groups of two are merged into a single frame, having a total of 32 frames (18 merged frames and 14 original frames). After having 32 frames for each gesture, each frame undergoes point resampling to have a uniform number of points in each frame. For frames with fewer than 32 points, Agglomerative Hierarchical Clustering (AHC) is employed for upsampling. Conversely, for frames with more than 32 points, the K-means algorithm is used for downsampling.

C. Neural network model

After a valid gesture is detected, the neural network model takes over the identification step. WaveTune's goal is not to create a new gesture recognition algorithm but to identify the best candidate in the SoA and optimize it to work seamlessly with a music controller. Next, we describe our analysis of the SoA in point cloud processing to identify the most appropriate architecture for WaveTune.

Neural network models for point cloud processing: PointNet is a pioneering work in point cloud processing [12]. Instead of using standard images for object recognition, Pointnet uses point clouds, thus reducing the volume of data processing. PointNet however does not take advantage of the temporal evolution of frames and is more suitable for static object identification. PointLSTM [13] and Pantomime [10] further enhance point cloud recognition by exploiting the temporal evolution of events. However, both approaches require high computational resources, affecting real-time applicability in resource-constrained environments. Tesla-Rapture [11] includes temporal properties but also provides a lightweight model. Next, we describe the Tesla-Rapture model and the changes made to suit our application.

Metric/Aspect	**Tesla-Rapture**	**WaveTune**
Input Dimension	[1024, 3]	[1024, 3]
Initial Processing	STN (Spatial Transformer)	STN
MPNN Input	[n X 2 x 3]	[n X 2 x 3]
MPNN Output	n X 64	n X 32
MPNN MLP Sequence	[6, 64, 64, 64]	[6, 32, 32]
Attention Input Features	64	32
Number of Attention Heads	8	4
Linear Transformation (In/Out)	$64 \rightarrow 1024$	$32 \rightarrow 256$
Final Classification MLP Sequence	[1024, 512, 256]	[256, 128, 64]
Dropout in Final MLP	0.5	0.4
Trainable Parameters	**1.556M**	**0.859M**

TABLE III: Model comparison: Tesla-Rapture and WaveTune.

Tesla-Rapture and WaveTune adaptations: Figure 5 shows the Tesla-Rapture model and the blocks modified to create WaveTune. TeslaRapture takes a collection of point cloud samples referred to as `Batch`. As stated before, the SoA has identified that 32 (frames) x 32 (points) = 1024 (points) are the best `Batch` size, and hence, we use the same format. Each point in the `Batch` provides its corresponding 3D coordinates (x, y, z). The 1024 x 3 values represent the raw spatial data provided as input to the model. The TFNeT module performs standardization to account for differences in orientation or position. Until this point, there is no difference between TeslaRapture and WaveTune.

To exploit temporal correlation, TeslaRapture uses a Temporal Graph K-NN algorithm. This graph connects cloud points with their nearest neighbors in previous frames. The value of k is a critical factor that determines how far in time the point clouds would be linked. A value of one indicates that each point cloud is connected only to its next successive frame whereas larger values create more complex graphs for longer durations. Since, for our application, there is no performance benefit in adding complexity (bigger k) as shown in Figure 7a, we set k=2. Our high-performing low-complexity graph is obtained because we need fewer features, which increases the inter-gesture distance.

The remaining blocks of the original model need to be optimized to suit WaveTune. By means of MLP (Multi-Layer Perceptron), the network learns the representation of the data and extracts features from the graph. We retain the core architecture but simplify the MLP configuration from [3x2, 64, 64, 64] to [3x2, 32, 32]. We also reduce the number of attention heads from 8 to 4, which implies that we aggregate a lower number of features. At this stage, the graph generation is complete. In the subsequent linear transformation stage, which is used to represent the gesture as a fixed-size vector, an MLP followed by a max-pooling is performed. We modify the MLP size from 1024 to 256, thus reducing the feature size to 25% of the original design. Accordingly, the subsequent MLP layers that compute the class probabilities also change from [1024, 512, 256] to [256, 128, 64]. Our steps follow an approach of aggressively scaling down the feature size at each layer. Table III compares the configurations of WaveTune and Tesla-Rapture. Overall, our configuration reduces the number of trainable parameters by half. In Section VI, we analyze the

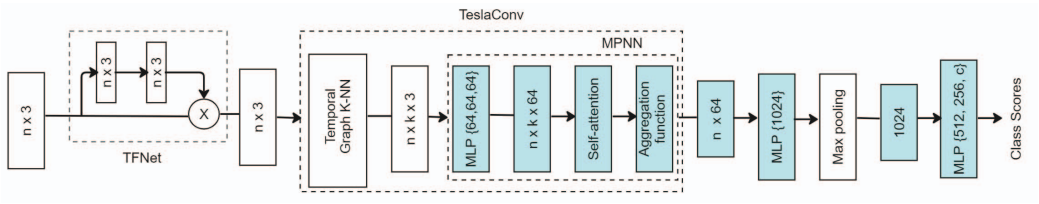


Fig. 5: Architecture of Tesla-Rapture. The shaded blocks had to be optimized for WaveTune.

impact that our configuration has on the model's accuracy.

V. Music Generation Block

While there are several studies on gesture recognition with mmWave radars, there are virtually no studies exploiting these sensors for music beat composition. The only study we are aware of is O Soli Mio [7], which as described before is limited to controlling a single parameter with a single gesture.

The reason why no gesture recognition algorithm has been mapped to beat composition is that deep expertise in musical elements is required to provide a rich and pleasant musical experience. Music is structured in bars and beats, which the gesture recognition algorithm can not comprehend. The music generation block must be intelligent enough to wait for the current bar or loop completion before acting on the commands. Otherwise, the composed music track would sound terrible. To develop our music generation block, we require (i) a music generation tool that embeds the concepts of bars and beats to provide a pleasant experience, (ii) an interface that connects seamlessly the gesture recognition algorithm and the music composing tool, and (iii) a musical protocol that can be efficiently used by both, the musical generation tool and the algorithm. These three components are essential for music generation.

A. Music generation tool

WaveTune's aim is to create a flexible and scalable interface that can be used for further enhancements. Thus, firstly, it must provide fine control programmability. This implies support for Application Programming Interfaces (APIs) to tune different musical parameters in real-time such as *reverb* and *filter cut-off frequency*. Secondly, it must offer a comprehensive list of features capable of creating complex compositions. This necessitates support for extensive soundbanks and knowledge of musical elements such as bars and beats.

We explore various tools, as detailed in Table IV. Algorithmic Composition Tools, despite their effectiveness in complex composition creation, fall short on integration support. On the other hand, Python-based Libraries offer a smoother integration with gesture algorithms but compromise real-time multitrack support, limiting the musical experience. Among the options, Digital Audio Workstations (DAWs) stand out, providing an industry-standard platform with extensive feature sets and robust integration capabilities. Specifically, we choose Ableton Live, a DAW commonly used by professional musicians. Choosing Ableton Live for music generation in WaveTune facilitates a smooth interaction with the gesture recognition system, a rich musical output with various options, and enables real-time music creation.

B. Music controller

The music controller has two primary responsibilities, (a) enabling a reliable layer-by-layer beat generation by implementing the state machine described in Figure 3, and (b) providing a low-latency interface especially during the dynamic mode when the gesture recognition is disabled and the movements need to be mapped to musical parameters instantaneously.

For a reliable state-machine implementation, we design a controller that maintains two-way communication with the music generation tool. A new command is issued to the tool only after confirming its current state of operation. For example, if the tool is currently playing Track-1 in the Drum mode and there is a command to switch to the next track, the music controller reads the current tracks for melody and drum, maintains the melody and increments the drum track. The music controller is also responsible for the interactive music generation during the Dynamic mode. Specifically, the controller extracts the velocity component from points within a frame. The average value of the velocity is computed and mapped to a parameter called 'Filter cutoff frequency'. Slow hand movements decrease the knob value, producing a more subdued effect, while faster movements increase it, intensifying the sound. The impact of the movements is felt instantaneously, promoting highly engaging interactions.

C. Protocol

The music controller interfaces with the music generation tool through an industry-standard protocol. To enable future WaveTune's enhancements and compatibility with the industry standard DAWs, our analysis of existing protocols is limited to two widely used alternatives: Musical Instrument Digital Interface (MIDI) and Open Sound Control (OSC). MIDI, which operates on a serial communication model, is a longstanding standard. OSC is a more recent protocol designed to address some of MIDI's limitations. First, unlike MIDI, OSC transmits data over networks enabling wireless communication from various devices. Second, it has low-latency communication, making it ideal for real-time performances [20]. Third, OSC offers a finer control resolution of musical parameters, which is advantageous in our design to map small changes in velocity into musical perception during the Dynamic mode. Due to the above reasons, we chose the OSC protocol.

In summary, the careful analysis done prior to selecting the best music generation tool and protocol, and the flexible and reliable design of our controller, are the key reasons why WaveTune can provide a melodious musical experience.

Choices	Examples	Features	Limitations
Algorithmic composition tools	SuperCollider [14], Sonic Pi [15], MAX [16]	Capable of generating complex and detailed compositions	Limited integration support, Limited soundbanks
Python based libraries	FluidSynth [17], Pyo [18], Mido [19]	Easy to integrate with gesture processing algorithm	Lack flexibility and control such as support for simultaneous multitrack playback and real-time parameter control
Digital audio workstations (DAWs)	Ableton Live	Industry standard tools, Wide options and Integration support	Complexity

TABLE IV: Alternatives for music generation

	Environment 1	Environment 2
Area	Office space: 5m x 3m	Residential room: 10m x 10m
Volunteers	14 adults (8 males, 6 females) + 2 children	5 adults (3 males, 2 females)
Height range	130 cm - 185 cm	150 cm - 170 cm

TABLE V: Characteristics of the evaluated scenarios.

VI. Evaluation

Data collection. We test WaveTune in two different environments. We got an Ethics approval from our institution to perform the experiments. Table V presents the characteristics of the environments and volunteers. Figure 2a shows the picture of Environment-1. We collected a total of 3350 gesture samples from 21 volunteers. The number of samples is balanced among the gestures. In addition to performing the three valid gestures, the participants also performed several undefined movements labeled as *random gesture*. Thus, we train the neural network model to classify four gestures, three classes derived from valid gestures and one from random movements. Contrary to most gesture recognition studies where the input is guaranteed to be a valid gesture, in our case, classifying (and discarding) random gestures is central to providing a pleasant experience.

Data augmentation. The point cloud data goes through the preprocessing stage presented in Section IV-B before the neural network model classifies it. Further, we perform a data augmentation step, which is customary in many studies. We generate additional *frame* samples by adding Gaussian noise with a standard deviation of 0.01; and performing random scaling to the collected data, from 0.8x to 1.25x with a uniform distribution. Data augmentation adds resilience to the model, helping it perform better under noise and minor variations. For every original frame, we create three new augmented frames using the added Gaussian noise and random scaling. We only perform data augmentation on the training set (not for validation or testing). Thus, our training dataset has in the end more than 10,000 data points including the original and augmented samples.

A. Model size, Complexity, Optimal parameter, and Accuracy

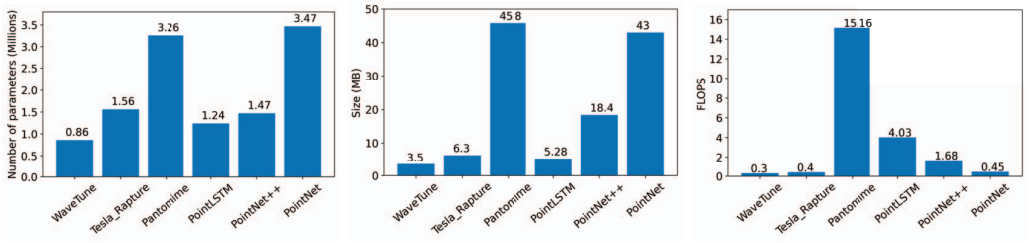
As described before, one of our main goals is to make the gesture recognition model as lightweight as possible, without sacrificing performance. First, we present the results about complexity and later about accuracy. Figure 6a and Figure 6b compare the number of parameters and the memory footprint of WaveTune with other SoA models. The performance data for models other than WaveTune is taken from what has been reported in prior studies [11]. We observe that the lower number of parameters in WaveTune –nearly 45% lower than Tesla-Rapture and 30% lower than PointLSTM– lead to smaller memory footprints. Similarly, Figure 6c confirms that WaveTune also has the lowest computational demands. The systematic scaling down of the architecture layers, to recognize only the gestures of interest for WaveTune, helps to significantly reduce the memory footprint and computing operations.

As described earlier, the value of k during graph generation is a critical parameter. The aim is to select the lowest values without compromising on the accuracy. Figure 7a plots the accuracy for varying values of k. We observe that the accuracy stays at 93.46% even when k is varied from 2 to 6. Since, in our case, increasing k increases complexity without improving accuracy, we conclude that k=2 is an optimum value for WaveTune.

Having validated the low complexity of WaveTune, we now benchmark its accuracy compared to the original architecture on the WaveTune dataset. We randomly split our data into training, validation, and testing in a 60:20:20 ratio. WaveTune achieves an accuracy of 93.46% against the 93.79% accuracy for Tesla-Rapture, showing comparable performance. Thus, the drastic reduction of model parameters does not cause a deterioration in accuracy for our application. Figure 8 shows the confusion matrix across different gestures. We note that the Track Change gesture, which we perceive to be one of the simple gestures, has the lowest probability of correct identification. We conjecture that random movements may have more resemblance with simple gestures, such as Track Change. Thus, the model has a slightly higher difficulty differentiating them.

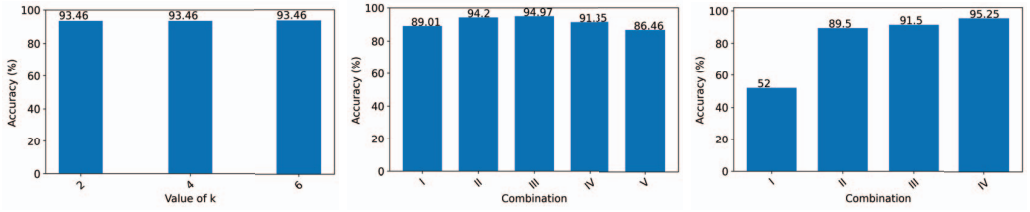
B. Performance with new participants

In a real scenario, WaveTune will face new users, who do not provide any training data. To test WaveTune's resilience to unseen participants, we split the participant's data into five groups: four groups of four and one group of five. Each group contains participants from both environments. We perform the training and validating phases of the WaveTune model with the data from four groups, and perform the testing phase with the data from the fifth group. We repeat this process for the five different combinations, each time keeping a different group for testing. Figure 7b summarizes the results from this experiment. We see that while the accuracy is between 85% and 95% for all combinations, for two of the combinations (labeled as I and V), the accuracy is slightly lower compared to others. Combinations I and V have a young child with lower height characteristics in their test group, which impacts the accuracy adversely. Thus, we conclude that while WaveTune can adapt to new users, it performs better if users with a similar age and height are considered in the training and validation stages.



(a) Number of parameters. (b) Memory footprint. (c) Computational complexity.

Fig. 6: Complexity comparison between WaveTune and state-of-the-art approaches.



(a) Different k values (b) Testing new participants. (c) Testing a new environment.

Fig. 7: WaveTune accuracy under different parameters, new users, and new scenarios.

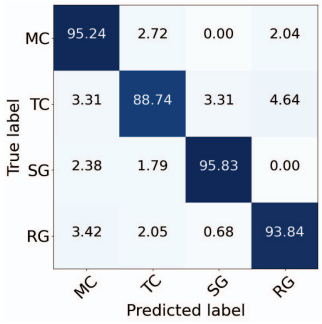


Fig. 8: Confusion matrix. MC, TC, SG, and RG correspond to Mode change, Track change, Stop, and Random gestures.

C. Performance in different environments

A new environment implies potential differences in the intensity of received signal, noise, and reflections due to stationary objects that the model is not aware of. To analyze this case, we initially train and validate the model with users coming only from Environment-1 and test it with users from Environment-2. After that, we add training data from Environment-2 but in an increasing manner. First, we split the users of Environment-2 into four groups, three groups with two users each and one group with the remaining user. Then, we add the first group to the training data and test the system with the remaining three groups. After that, we add the second and third groups to the training phase and test the system with the remaining groups. In this way, we gradually increase the number of training samples of Environment-2, simulating a scenario of improving levels of environment familiarity. Figure 7c summarizes the results. We observe that the accuracy is only 52% when the model does not have any training samples from the new environment. However, it improves considerably to 89.5% when the first group is added (combination II) and reaches 95.25% when the second and third groups are added (combination IV). This indicates that while the performance of WaveTune is adversely impacted when placed in a new environment, a small amount of training data can significantly improve its accuracy.

D. User experience

Until now, our evaluation focused on technical parameters. To evaluate the user experience, we sought feedback from new users. We invited three new volunteers, all young adults, who had not provided any type of data or had any sort of prior interaction with WaveTune. The evaluation of the new users was done in Environment-2. After explaining WaveTune to the new users, we provided a live demonstration, beginning with the start gesture, transitioning through various modes, performing track changes, and showing the interactions in the dynamic mode. The users were provided with the opportunity to clarify their understanding of gestures. Then, they interacted with WaveTune for three minutes each, browsing through different modes. We summarize the user experience findings in the following points:

- During the explanation, most of the questions were about the Mode Change gesture, which was more difficult to understand compared to the other gestures. We believe that this occurs because the Mode Change gesture captures the fundamental principle of composing music by layers with modern tools, which is a new concept for most people.
- However, after knowing the gestures and starting the interaction, they could independently control WaveTune without much help. The fact that WaveTune relies on a limited number of gestures helped users learn quickly.
- During the interactions, the Mode Change gesture was recognized better than the other gestures. In all, 81.48% of the total 54 gestures were correctly recognized. When a gesture was not recognized, the user performed it again.
- The mode that the users enjoyed the most was the live control during the dynamic mode and wanted to spend more time in it.

E. Lessons learned

Based on the studies we had with 24 users, we identified two main lessons, one regarding the musical experience and the other regarding gesture complexity.

Musical experience. Users enjoy playing with all of WaveTune's modes, but the most engaging mode is the dynamic one, which provides the most freedom of movement. Future designs could focus on providing a richer connection between free body movements and music composition. For example, two or more people could be in the field of view of the radar, and one person could control one layer with her body movements and the other user could control a different layer with his movements. In the extreme case, valid gestures may not be even needed. The radar could compose a song based on free movements.

Gesture complexity. By design, we minimized the number of gestures required, which users appreciated. The advantage of the layered approach is that we can provide a richer experience, without adding more gestures. For example, before the Melody mode, we can add an initial layer to select a genre (rock, pop, classical), and after the drum mode, we could add layers to highlight specific instruments (piano, guitar, violin). To increase the experience further, we could add one gesture to control the speed of each layer and another gesture to jump directly to the dynamic mode without going through all the composition layers. Adding more gestures to WaveTune would not be complex. Since many SoA studies are able to recognize more than 10 gestures, we would only need to optimize the model for the final number of gestures. However, our initial trials indicate that having too many gestures would reduce the quality of the experience. A richer yet enjoyable composition would not require adding more than a handful of gestures.

VII. Related Work

There has been a long-standing interest in blending technology with music, especially gesture-based music generation. 'The Hands' project developed in 1984 consists of two controllers attached to the hands [1]. Sensorband further developed the concept by using ultrasound, infrared, and bioelectric gestural controls to generate music [21]. More recent works in gesture-based music generation use ML-based techniques to learn gestures from a variety of sensors including EMG [22]. All these approaches use wearable sensors, which are valuable but restrictive tools. The other widely used technology is camera-based sensing. Togootogtokh et al. [23] introduce a method for 3D finger gesture tracking and recognition using depth sensors for real-time music playing. Kritsis et al. [24] investigate deep learning architectures, specifically LSTM networks, for real-time gesture recognition of 3D virtual music instruments with the Leap Motion Sensor. Leng et al. [25] presents the 'Virtual Kompang', a digital representation of the traditional Malay percussion instrument, Kompang. However, these works use camera-based devices. One of our objectives is to develop a privacy-friendly interface. O Soli Mio is the only other work we came across that uses mmWave for capturing gestures and using them for music generation, but as described in earlier sections, O Soli Mio provides a more limited experience.

VIII. Conclusions

In this study, we analyze and merge two different areas: gesture recognition with mmWave radars and modern music generation tools. Our approach optimizes and connects SoA methods to provide a novel system for beat composition that does not require wearables or cameras. We believe that WaveTune could be a valuable tool to create new musical experiences in our schools, museums, homes, or cities.

Acknowledgements

The authors gratefully acknowledge the help from Amsterdam Institute for Advanced Metropolitan Solutions (AMS). The research was partly supported by a grant from AMS.

References

[1] V. Krefeld and M. Waisvisz, "The hand in the web: An interview with michel waisvisz," *Computer music journal*, 1990.
[2] "Leap motion controller," http://www.leapmotion.com.
[3] "Mi.mu gloves," http://www.mimugloves.com.
[4] "SOMI-1 Turn your Movements into Sound," https://instrumentsofthings.com.
[5] B. K. Chakraborty *et al.*, "Review of constraints on vision-based gesture recognition for human–computer interaction," *IET Computer Vision*, 2018.
[6] W. Dai and L. Zhou, "Tinyml-enabled static hand gesture recognition system based on an ultra-low resolution infrared array sensor and a low-cost ai chip," *IEEE*, 2022.
[7] F. Bernardo *et al.*, "O soli mio: Exploring millimeter wave radar for musical interaction," in *New Interfaces for Musical Expression*, 2017.
[8] J. Lien *et al.*, "Soli: Ubiquitous gesture sensing with millimeter wave radar," *ACM Transactions on Graphics*, 2016.
[9] Industrial mmwave radar sensors. Accessed: 2023-07-31. [Online]. Available: https://www.ti.com/sensors/mmwave-radar/industrial
[10] S. Palipana *et al.*, "Pantomime: Mid-air gesture recognition with sparse millimeter-wave radar point clouds," *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2021.
[11] D. Salami *et al.*, "Tesla-rapture: A lightweight gesture recognition system from mmwave radar sparse point clouds," *IEEE Transactions on Mobile Computing*, 2022.
[12] C. R. Qi *et al.*, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proc. Computer Vision and Pattern Recognition (CVPR)*. IEEE, 2017.
[13] X. Min *et al.*, "An efficient pointlstm for point clouds based gesture recognition," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020.
[14] "Supercollider," https://supercollider.github.io/, accessed: 2023-08-04.
[15] "Sonic pi," https://sonic-pi.net/, accessed: 2023-08-04.
[16] "Max," https://cycling74.com/products/max, accessed: 2023-08-04.
[17] "Fluidsynth," http://www.fluidsynth.org/, accessed: 2023-08-04.
[18] "Pyo," http://ajaxsoundstudio.com/software/pyo/, accessed: 2023-08-04.
[19] "Mido," https://mido.readthedocs.io/en/latest/, accessed: 2023-08-04.
[20] N. Aoki and Y. Morimoto, "Introduction to sound programming in supercollider," in *Proceedings of the International Computer Music Conference*, 2013.
[21] B. Bongers, "An interview with sensorband," *Computer Music Journal*, 1998.
[22] C. Rhodes *et al.*, "New interfaces and approaches to machine learning when classifying gestures within music," *Entropy*, 2020.
[23] E. Togootogtokh *et al.*, "3d finger tracking and recognition image processing for real-time music playing with depth sensors," *Multimedia Tools and Applications*, 2017.
[24] K. Kritsis *et al.*, "Deployment of lstms for real-time hand gesture interaction of 3d virtual music instruments with a leap motion sensor," 2018. [Online]. Available: https://api.semanticscholar.org/CorpusID:250613404
[25] H. Y. Leng *et al.*, "Virtual kompang: Mapping in-air hand gestures for music interaction using gestural musical controller," *Journal of Fundamental and Applied Sciences*, 2018.