Pitch Documentation

September 2025

Goals

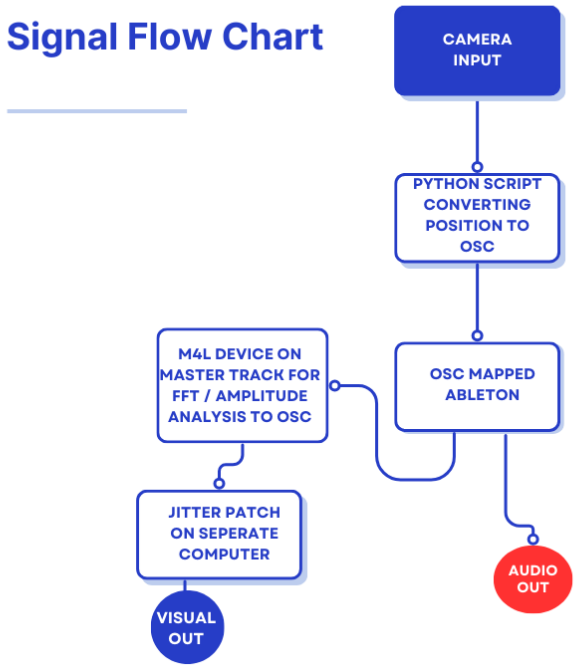
The artistic goal of this installation is to provide an environment in which people can make music with others and mix different stems in a more intuitive and interactive style than with a traditional DJ mixer. Rather than relying on a single performer behind a booth, the system is designed to open up the process of mixing and sequencing, allowing multiple participants to contribute simultaneously. This transforms the act of listening into a shared creative experience, where collaboration and spontaneity drive the music forward.

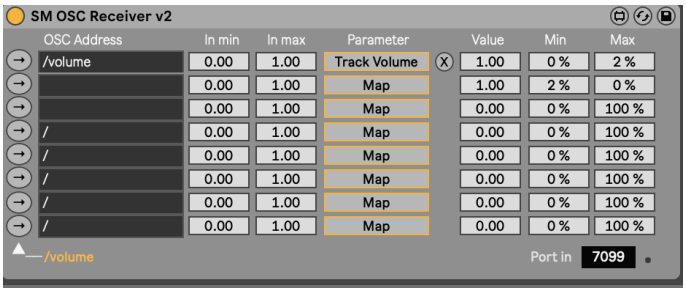
The styles of music chosen will be inspired by the nightlife culture of Underground Atlanta, reflecting its showcase of underground electronic, hip-hop, and experimental sounds. By embedding these stylistic references into our project, the installation not only becomes an interactive artwork but also a sonic reflection of the city's nightlife identity.

Audio

The audio for our installation is generated in Ableton. OSC is mapped to parameters such as clip trigger, wave position, filter frequency, volume, and spatial azimuth. Four voices will be controlled (Bass, Drums, Chords, Lead), and the timbre and sequence will vary, although all sequences should fit semi-neatly with each other to maintain a coherent musical environment. The goal is to balance variation and cohesion so that the system feels dynamic and responsive without devolving into randomness. Each voice retains its own sonic identity while contributing to an evolving collective texture, ensuring that the installation remains immersive, musically engaging, and adaptable to interaction.

Below is a signal flow chart as well as an image of the Max for Live device used to map OSC in Ableton.





Visual

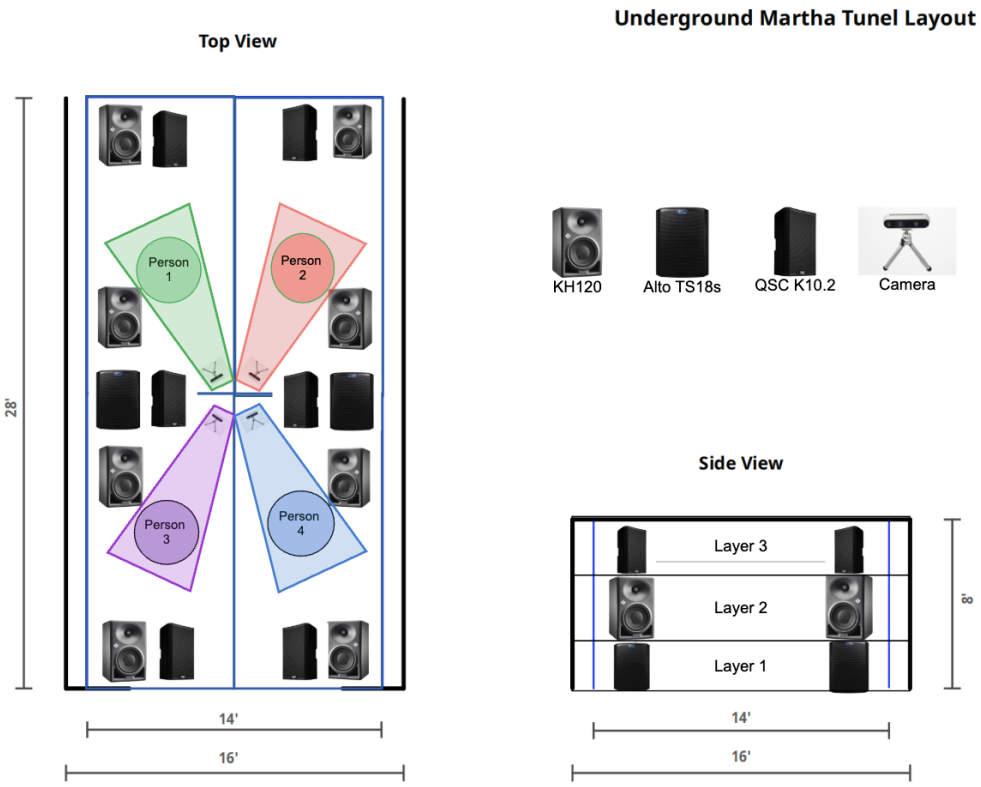
For this project, we are going to create simple visual content that is tightly synchronized with the music. Using Max/Jitter, the visuals will react to the BPM, effects, and volume of each audio track, which are sent from Ableton as OSC data. Based on signals captured from a camera to track audience movement, we can make the visuals more interactive. The final output will be shown on three projectors. My main task is to build a generative particle system in Jitter, possibly using some Vizzie modules as well.

Since our performance venue has four cameras positioned at the center facing four directions, We want to divide the space into four zones with visual setting. When people enter a new camera zone and are captured by that camera, the visuals will react accordingly. This approach is better than drawing lines on the floor or dividing the space manually.

Interaction

Camera and Speaker Set Up in the Venue

The system is arranged in a rectangular space that measures 16 feet by 28 feet. 16 loudspeakers of 3 different models are distributed along both sides of the room symmetrically in three vertical layers. The setup includes 2 Alto TS18s subwoofers in the bottom layer, 8 KH120 monitors in the middle layer, and 6 QSC K10.2 speakers in the top layer, ensuring coverage of the frequency throughout the listening area. Four participants stand in the center of the quadrants, each facing outward toward a distinct speaker array, with their listening zones indicated by colored cones. Four cameras are positioned at the central dividing line, directed toward the participants, enabling synchronized audio-visual capture of the performance space. This configuration provides immersive surround sound while maintaining balanced monitoring for all four individuals.



Input Tracking: Position, Distance, and Gesture

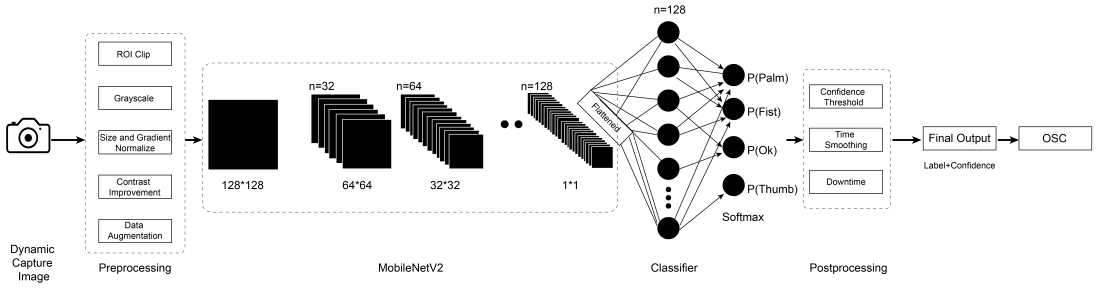
Based on the camera layout in the venue, during interactions with participants, we extract three inputs from the cameras: participants' spatial positions, mapped as azimuth angles for spatial audio to control sound source orientation; participants' distances, converted into the volume of the track; and participants' hand gestures, used as audio effects for corresponding tracks.

Regarding the location of the participants, we normalize the dynamic image along the horizontal axis and map it to the angle range $[-\pi, \pi]$, so the participants on the left have negative azimuths and those on the right have positive azimuths. Angle parameters are handled by *Max for Live (M4L)* and passed to a spatial audio plugin in *Ableton*, which acts as an ambisonic decoder. In the rendering environment, the audio is spatialized to create a sense of immersion.

For the distance between the camera and participants, we use the scale of participants within the camera frame to determine the camera's distance. This distance is normalized to $[0, 1]$ and is mapped to the track volume parameter. The closer participants are to the camera, the louder the volume. As they move farther away, the energy decreases.

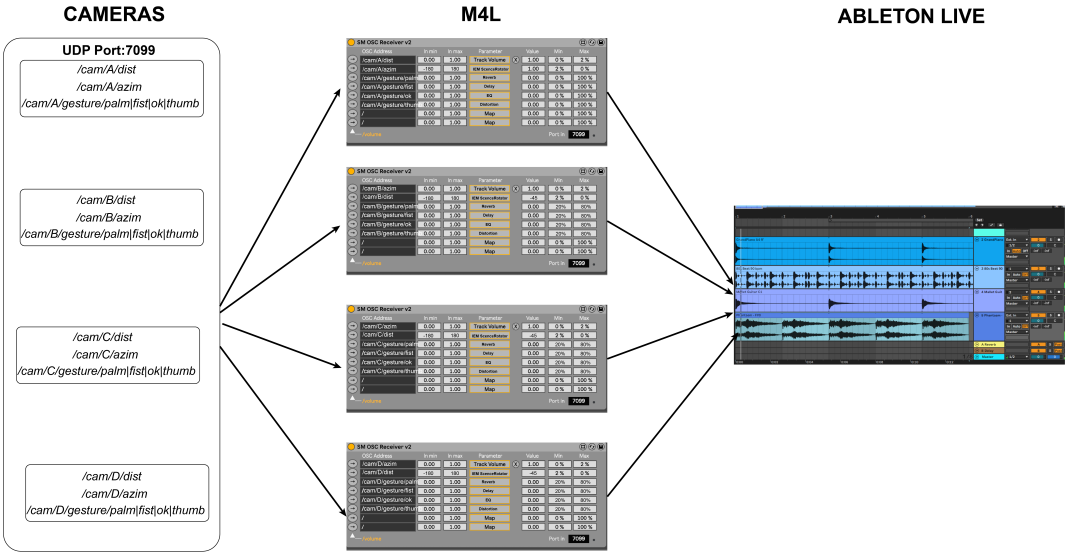
For gesture recognition to control sound effects, we collected four types of gestures: "palm," "fist," "OK," and "thumb" in low–light conditions. Each gesture category has 200 images, and each gesture controls a distinct sound effect, such as palm to reverb, fist to delay, OK to filter, and thumb to distortion. Furthermore, we included 800 "none" gesture images in the data set to prevent bugs. In data preprocessing, we cropped the hand region and converted the color images to a single-channel format, since gesture classification is based on shape contours rather than color. We normalized and resized all data to 128×128 pixels and improved contrast for more recognition stability in low–light conditions. Finally, we normalize and randomly apply rotation, scaling, and translation for data augmentation. In addition, for model training, we allocate the datasets to training, validation, and test sets. We use *MobileNetV2* as the training model, a convolutional

neural network that uses separable convolutions in depth and inverted residual blocks. This model reduces computational complexity while maintaining high classification accuracy, making it suitable for real-time scenarios (citation). The output is a softmax probability distribution across 4 classes, with the highest probability assigned to the correct classification. Model performance is evaluated using the Top-1 accuracy and confusion matrix. The trained model is then applied to camera-based interaction capture, followed by post-processing. Only softmax probabilities above 0.7 are considered valid gestures; otherwise, they are ignored. Also, a gesture is only activated when the predictions remain consistent in 3-5 consecutive frames. Repeated gestures within 200 to 500 ms after a trigger are ignored. The final output is sent as an OSC signal to the audio component, triggering different audio effects. The whole process is shown in the figure below.



Signal Processing and OSC Mapping for Camera

Under the same IP address, the connections between four cameras and four audio tracks are connected via Max 4 Live OSC receivers, and using OSC address naming conventions. For the four cameras, all cameras transmit signals through UDP port 7099. These signals are pre-processed through the region of interest (ROI), smoothing, dejittering, and normalization. After that, different OSC address naming conventions are used to classify and transmit signals for cameras, distance, azimuth, and gestures. The address controlling audio track volume based on distance is `/cam/X/dist`(X: A, B, C, D). The address controlling spatial audio azimuth based on human orientation is `/cam/X/azim`(X: A, B, C, D). These signals are transmitted to the Max 4 Live OSC receiver to control the orientation of the spatial azimuth of the *IEM*. In addition, addresses controlling different poses by gesture recognition are: `/cam/X/gesture/label` (X: A, B, C, D), with different poses corresponding to different sound effect gains. At runtime, the camera sends a signal when a palm gesture is confirmed, such as `/cam/A/gesture/palm`. In *Ableton Live*, there are four *Max for Live (M4L)* OSC receivers, one for each track, so each track only listens to its assigned camera addresses. Distance controls track volume, azimuth sets the spatial angle of the *IEM*, and gesture addresses adjust or trigger different effects. This setup keeps all 24 addresses organized and avoids crosstalk between cameras and tracks.The signal flow for connecting cameras and audio is shown below:


CAMERAS
M4L
ABLETON LIVE
UDP Port:7099
/cam/A/dist
/cam/A/azim
/cam/A/gesture/palm|fist|ok|thumb
/cam/B/dist
/cam/B/azim
/cam/B/gesture/palm|fist|ok|thumb
/cam/C/dist
/cam/C/azim
/cam/C/gesture/palm|fist|ok|thumb
/cam/D/dist
/cam/D/azim
/cam/D/gesture/palm|fist|ok|thumb