SharedSpace: Spatial Audio and Video Layouts for Videoconferencing in a Virtual Room

Matthew Wong and Ramani Duraiswami
Perceptual Interfaces and Reality Lab, Computer Science and UMIACS, University of Maryland College Park
College Park, MD 20742
mwong129@terpmail.umd.edu, ramanid@umd.edu

***Abstract*—Online video conferencing, a growing mode of meeting, has grown exponentially since the start of the coronavirus pandemic. Many cues available in in-person interactions are absent in the currently implemented systems. We present a novel web application for video conferencing which uses spatial video and audio to provide better online conferencing experiences. We used SharedSpace to have many different conference calls and from our results, we can conclude that the user ratings for spatialized video and audio conferencing far exceed the user ratings for non-spatialized video and audio calls.**

***Keywords*—Spatial Audio, Web Application, Video Conferencing.**

I. Introduction

A. Background

Online videoconferencing has become a huge part of everyone's lives this past year. School, work meetings, and family/friend gatherings have all become virtual and many have gotten used to this new environment. However, current video conferencing applications have their own set of problems. The term "Zoom fatigue" has been used a lot by students and media in the previous year and can be defined as "computer mediated communication exhaustion" [1]. One of the main problems that causes "Zoom fatigue" is the cognitive load that is places on users during a video call [2]. With face-to-face interaction, nonverbal communication is quite natural and tends to complement the words we say so our points are more clear and understandable. On the other hand, users in video conference calls are required to put in a lot more cognitive effort when trying to send or interpret nonverbal cues which can cause confusion or even conflict when the cue is interpreted incorrectly [2,3]. In addition to the difficulty of sending and receiving nonverbal cues, another problem is the lack of turn taking and speaker identification when the number of participants grow in a video conference call [2]–[5]. In a recently published study, Bailenson states, "Inferring the attention of others is nearly impossible once there are more than a handful of people on a conference call, and conversational moves such as turn-taking become difficult to manage" [2]. These problems can possibly be solved by improving the video and audio quality of the call as we will discuss in a later section. In a recent survey accessing the effectiveness and inclusiveness of video conferences at a large technology company, the researchers found that audio and video quality are paramount for effective and inclusive meetings [6]. They concluded that due to the quality of audio and video not being up to par with the quality participants would get in an in-person conference meeting and the reduced ability to decipher non-verbal communication, participants tend to participate less in video conference calls [6]. By changing the way video and audio is presented in a video conference call, we can solve the problems mentioned above like lack of speaker identification, sparse participation, and the absence of non-verbal cues that can make video conferencing really tiring and unexciting.

We present SharedSpace,[1] a video conferencing web application that utilizes spatial video and audio to make the video conferencing experience more rich and lively. With developments of robust technologies such as Angular, Node.js, and WebAssembly, web applications have evolved to be very powerful. In addition, exploiting the numerous benefits of spatialized video and audio, SharedSpace attempts to solve the problems that plague current video conferencing applications while at the same time minimizing content download.

II. Related Work

Many of the previous conferencing systems with spatial audio and video have involved additional hardware, have been mainly used as a tool for experiments, and have tried to preserve the gaze of the user which typical video conferencing systems fail to achieve [7]–[9].

In [7] the video conferencing system consisted of two cameras, two speakers, two microphones, and a widescreen monitor, and experiments were performed with this system on groups with only three users. In a three-way call, two users would be on opposite sides of the widescreen monitor and each one of those users would have a microphone, camera, and speaker assigned to them. This setup would preserve the gaze of the current user talking to the other two since when the user turned to face one of the two users, the corresponding camera to that user on the screen would detect the user's full face rather than the side of their face. The horizontal layout of the users on the screen was how they spatialized video. Furthermore, the assigned microphone and speaker allowed the audio spatialization is done using hardware rather than software. In [8] the videoconferencing system consisted of a single widescreen monitor, a single camera, a single

[1] A prototype SharedSpace server can be experienced at http://mediasoup.umiacs.umd.edu. At least three users, all wearing headphones, should be present in the call to experience the prototype.

microphone, and software that performed pose tracking of the head. Like the previous study, users in the call were laid out on the screen horizontally to spatialize the video and they only tested a call with three users. Whenever the user turned their head to face another on the screen, the pose tracking algorithm changed the audio spatialization to reflect the position and orientation of the user's head. This system uses software to spatialize the sound rather than hardware.

Another video conferencing system, Multiview, consisted of a viewing screen, multiple cameras, and projectors to simulate a life sized conversation at a table [9]. Multiview's main goal was to preserve gaze, nonverbal gestures, and spatial awareness used in communication since typical video conferencing applications fail to do this. In order to achieve this, the system used a custom viewing screen and specific camera placements where it would only show the video of users produced by the projectors to a specific person in a specific viewing zone when the users sat in a row of three. There was no attempt to implement audio in this system.

Our application is quite different than the previously developed systems. First off, it gives a bird's eye view of the users sitting around an elliptical table. Most previous systems had the user's video lined up horizontally across the screen to preserve the gaze of the users. In addition, SharedSpace does not require any additional hardware setup or software installation from the user, but only requires them to have an up to date Chrome browser, any sized monitor, a microphone, and a pair of headphones. Even though SharedSpace does not attempt to preserve the gaze of the user like previous applications, it utilizes spatial audio and a unique spatial video format to deliver a better video conferencing experience.

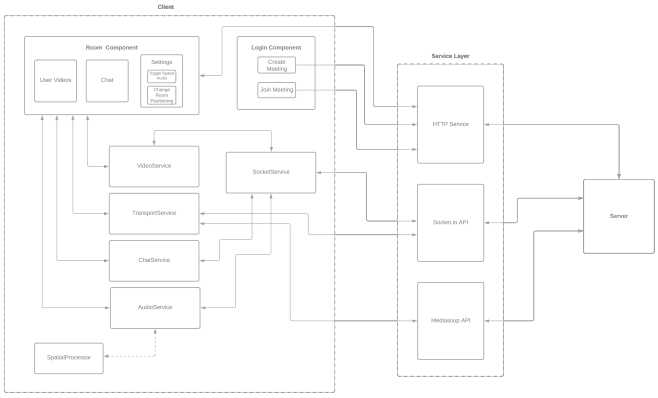

Fig. 1. The system architecture of SharedSpace.

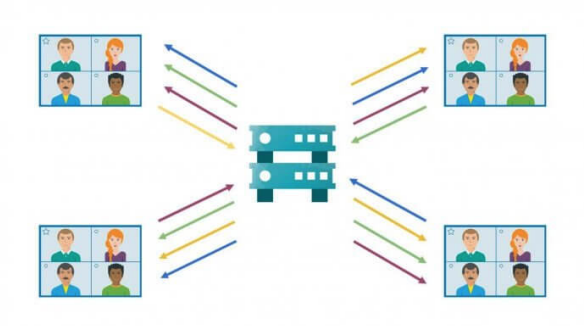

Fig. 2. A SFU architecture where each person sends one stream of data and receives $(N-1)$ streams of data where N is the total number of users in call (four in this case). Adapted from [16]

III. Benefits of Spatial Video and Audio

The main video conferencing systems today such as Zoom and Google Meet do not provide spatialized audio or video. Most of the time, the videos are positioned in a grid like format with the audio being monaural. Several studies have showed that spatial audio can help with better speaker identification and intelligibility when multiple people are speaking at the same time [10]–[12]. In a monaural video conference call, it is impossible to have multiple people speak at the same time and have the listening user discern what one of them is saying. In another study, researchers found that users had the highest accuracy in identifying different speakers in a call when the system had a visual representation of voice locations combined with spatialization of audio rather than just a system with spatial audio or monaural audio alone (without a visual interface of where other users were) [13]. A previous mentioned study has also shown that spatial audio has helped increase memory, comprehension, focal assurance, and speech intelligibility [12]. Speech intelligibility refers to the ability to focus on a single speaker and understand what they are saying when there are multiple competing speakers [12]. Focal assurance refers to the feeling of knowing who the participants are and knowing who is speaking at any given time [12]. Given the many benefits that spatial audio and video provide, the user has a better opportunity to be more engaged, lively, and delighted in a video conferencing environment that utilizes spatial audio and video.

IV. Implementation and Design

The application uses Angular, Typescript, and WebAssembly for the front end and Express.js and Node.js on the back end. There is currently no database being used in the application and everything is stored in memory. In order for real time transport of video and audio streams, the Mediasoup API was used [14]. The Socket.IO library was also used for real time event-based communication between the client and server [15]. In addition, Angular's HTTPClient module was used with authentication services with the HTTP protocol. Fig. 1 shows the system architecture of the application.

Mediasoup was the underlying library used for the transport of the video and audio streams. Its architecture follows that of a Selective Forwarding Unit (SFU) [14]. As a SFU, Mediasoup receives different media streams from different endpoints (clients) and relay them to all other clients using WebRTC (see Fig. 2) [14]. If there are N users in a call, then each user in the call transmits their own video and audio streams and receive $(N-1)$ video and audio streams ($2(N-1)$ total media streams) from all the other users. Some benefits of using a SFU include

a low server CPU load, absence of video and audio decoding and encoding, scalability, and a lot of control over the streams that are transmitted to the clients [17,18]. Mediasoup mainly works with the Javascript Web API MediaStream object which interfaces with a user's video and audio streams [14].

A. Spatial Audio

The spatial audio library RS3D (VisiSonics' RealSpace 3D) simulates a 3D real world listening environment in a room by taking into account different cues [19]. It is available commercially with a CC-BY-NC v4.0 license to academic users, as a javascript library which is called *RealSpace 3D Audio* (RS3D) from *VisiSonics*[2]. Some static cues that affect the spatialized sound include the anatomical scattering represented by the user's head-related transfer function (HRTF) and environmental scattering cues represented by a room impulse response (RIR) [19]. In addition, the library uses head position and orientation as dynamic cues to strengthen the localization of the audio [19]. The main environmental variables in the library that affect the spatialization of sound include the types of floor and wall material, room dimensions, head position and orientation, and the placement of audio sources in 3D space [20]. Currently, SharedSpace mainly utilizes all these factors except the head position and orientation, since we did not use a head tracker. The RS3D library was compiled into a WebAssembly (WASM) module using the Emscripten compiler. More on how the library was utilized in the application is in Section IV-C3.

B. Back End

The back end primarily uses the Express.js web framework and Node.js to create a robust server. It uses both the HTTP and WebSocket protocol to communicate with the client. The server uses the Socket.IO library which is a wrapper around the WebSocket API [15]. The HTTP protocol is mainly used for authentication purposes of a user creating or joining a room. On the other hand, the WebSocket protocol is mainly used for real time bidirectional communication with the client when the user is in a conference room. For example, when a new user joins the room, Socket.IO can help notify the rest of the users in the current call by emitting an event in real time so that all other user screens can update the meeting with the new user.

C. Front End

The front end primarily uses the Angular framework along with the client libraries of Mediasoup and Socket.IO. There were a lot of Angular components and services that were developed to work together to create the application. The main components and services include room, video, and audio. Additional features include a simple chat component and a settings page where users can control if they want spatialized audio or not as well as the ability to switch the layout of the room. A login page enables users to join different rooms or create new ones (Fig. 3). Some smaller features include controlling the volume of other users in the call and muting the current user's own video and audio streams.

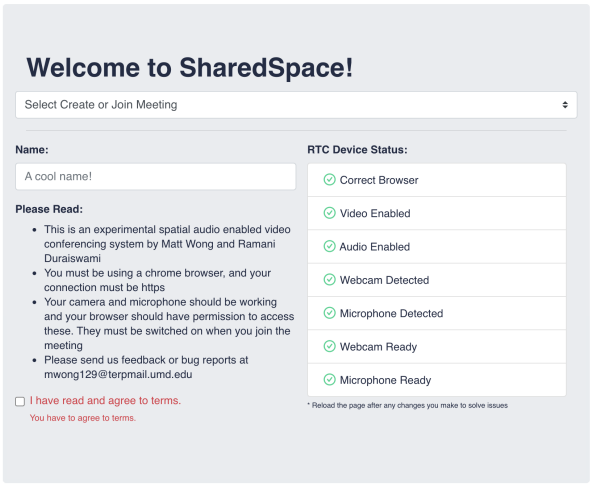

Fig. 3. A simple login page with some status checks that display whether the current user's machine and browser are compatible with the application. Users are required to type in a name and can provide an optional meeting ID depending on whether they want to create or join a room.

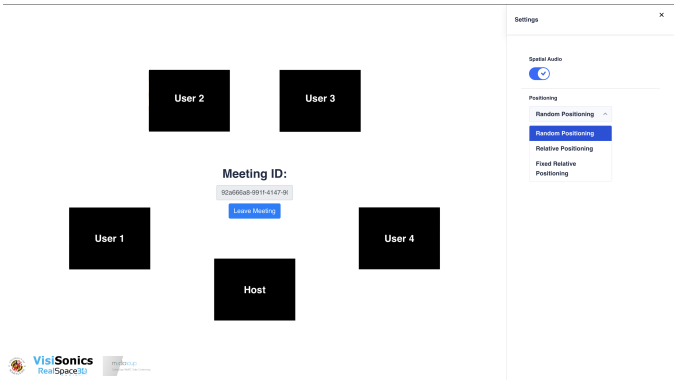

Fig. 4. A five person video call with spatial video and audio in a random positioning format. The settings page is also currently open.

1) Room Component: The room component renders the main page for the video conference and displays the videos of users who are currently in the room. In addition, the listening user can toggle between displaying a chat box or a settings bar on the side (Fig. 4). The settings sidebar has options that can change the room layout and toggle between spatialized and monaural audio. The room component utilizes the transport service which makes use of the Mediasoup API to manage and transport video and audio streams (Fig 1). The room component also utilizes the video and audio services to manage the video layout on the screen and the spatialization of the sound (Fig. 1). All the users are seated in an elliptical shape which simulates a bird's eye view of the meeting.

[2]*VisiSonics* is a company that specializes in spatial audio solutions and acoustic visualization hardware. Information on RS3D is available at https://www.visisonics.com/rs3d, including its use in Oculus VR.

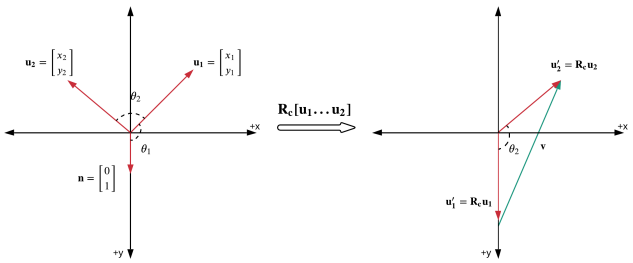

Fig. 5. We let $\mathbf{u_1}$ be the current position of the user A and $\mathbf{u_2}$ be the current position of another user B. Now to get the positioning with respect to user A's coordinate frame, we first perform a clockwise rotation $\mathbf{R_c}$ with respect to the origin of all vectors in the graph by θ_1 such that $\mathbf{u_1}$ has the same direction as $\mathbf{n}$. Then the vector being passed to RS3D would be $\mathbf{v} = R_c\mathbf{u_2} - R_c\mathbf{u_1}$

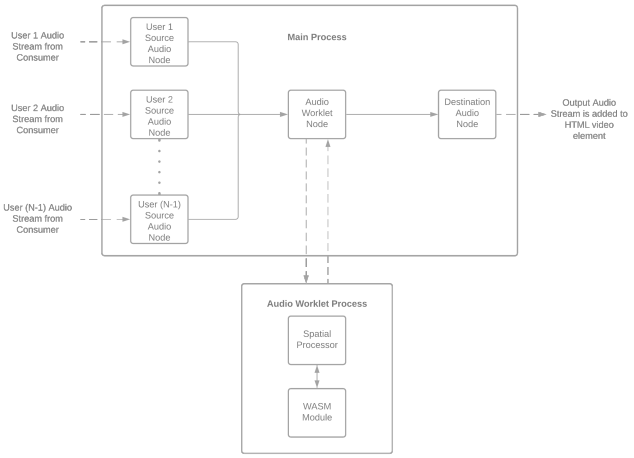

Fig. 6. User audio streams are fed into a pipeline which consists of Web API AudioNodes. Source audio nodes produce the raw input audio stream to an AudioWorklet node with a spatial processor script. Then a single output stereo stream of the spatialized audio is produced from the AudioWorklet node to the destination audio node and outputted to the user's headphones.

2) Video Service: The video service controls the creation and layout of the HTML video elements (user videos) in the call. Only the host of the meeting can change the room positioning. Users are situated in an ellipse no matter the positioning scheme and the angle of separation between each user is the same. For example, if there are 5 users, then each user is at 72 degrees from the next user.

The three positioning schemes are random, relative, and fixed relative. In the random positioning scheme, users are seated at random positions. However, the listening user is always at the bottom. In the relative positioning scheme, users are in the same order, with the the listening user at the bottom. In essence, each user's screen layout is just a rotation of the video placements of another user's arrangement. In the fixed relative positioning scheme, all users see the same positioning layout on their screen. The screen shows the absolute positioning of the users in the call. Since all users face the center no matter where they are positioned, we can represent each user as a vector from the origin to a user's video position on the screen. Performing some simple matrix rotations, we can get the positions of all the other users relative to the listening user's position on the screen and then use those coordinates in the audio spatialization calculations.

Given a fixed positioning scheme, in order to get the coordinates of the other users relative to the listening user, we perform a rotation with respect to the origin as the center of the screen. In the fixed positioning scheme, the listening user's coordinate frame is one where the user is the origin, the line that connects the user to the screen origin is the y axis and the x axis is perpendicular to the y axis and goes through the listening user origin. We can rotate all the users until the listening user's positive y axis lies on the screen's positive y axis. Let the angle of rotation θ_1 be the angle between the listening user's screen vector $\mathbf{u_1} = (x_1, y_1)$ and the vector $\mathbf{n} = (0, 1)$ (Fig. 5). After calculating θ_1, we need to determine whether we need to perform a clockwise ($\mathbf{R_c}$) or counter clockwise rotation ($\mathbf{R_{cc}}$). Using the correct rotation matrix, we can rotate all the vectors by θ_1. Assuming the listening user's absolute vector is $\mathbf{u_1}$ and it's rotated vector is $\mathbf{Ru_1}$ where $\mathbf{R} \in \{\mathbf{R_c}, \mathbf{R_{cc}}\}$, then the other users' positions relative to the listening user will be $\mathbf{Ru_i} - \mathbf{Ru_1}$ where $i = 2, 3, ..., N$ where N is the total number of users in the call. These relative vectors can then be used in the RS3D library.

3) Audio Service: The audio service controls the spatialization of the sound and the processing happens in real time on the client side of the application (Fig. 1). It takes in the positioning of all the users relative to the listening user and spatialize the audio streams accordingly. The audio service uses the AudioNodes interface in the Web Audio API library [21] to accomplish this (Fig. 6). A source audio node, containing a user's raw audio stream is created for every user in the call except the listening user. The outputs of these nodes feed into an AudioWorklet node which interfaces with the RS3D library. The RS3D library is compiled into a WASM module and instantiated when the AudioWorklet node is first created [22]. Upon being instantiated, the node creates an AudioWorklet sub-process [22] which performs the calculations for the audio spatialization. The AudioWorklet node then retrieves the output stream- a spatialized stereo audio stream- from the sub-process and passes it along to the destination node for outputting on the user's headphones.

The main transformation of the audio streams happens in the AudioWorklet sub-process (Fig. 7). In this process, the spatial processor script retrieves audio samples as a 2 dimensional array of $(N - 1)$ by 128 32-bit floating point numbers [21] where N is the total number of users in call and then passes these streams into the RS3D WASM module for processing. The spatial processor first creates a shared buffer between its script and the WASM module. The spatial processor script has a *process* function which takes in the input audio samples, copies them over into the shared buffer, and then the WASM module's *process* function is called. The *process* function in the WASM module takes the audio samples from the shared

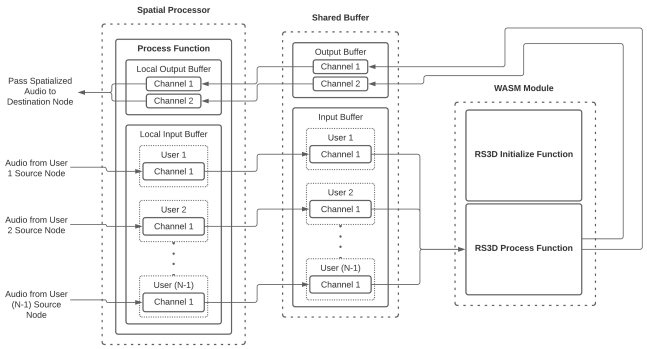

Fig. 7. The internals of how the AudioWorklet sub-process utilizes the the spatial processor script and RS3D WASM module to spatialize the sound.

buffer and feeds it into the RS3D library which then output a stereo stream of binaural audio samples. The WASM module copies over the output samples into another shared buffer that the spatial processor script can access. The spatial processor script takes these output samples and pass them back to the AudioWorklet node. Whenever, there is an update to the placement of the user videos in call or other setting changes, the main process and AudioWorklet process communicates through a dedicated channel [21] where the main process can send updates to change positioning vectors, wall coefficients, and other global variables if necessary.

V. User Experience

SharedSpace has been tested with many users and everyone has had positive ratings about the spatial video and audio formats as well as the design of the application. Most of the group calls consisted between 2 to 8 people and there was even one call with 16 people. A lot of the users who were using SharedSpace for the first time were astonished by the sound quality and video layout. The SharedSpace experience was totally different from the experience they were used to with typical video conferencing applications like Zoom that used monaural audio and grid-like video formats. Even with the positive reviews, a more in-depth study of how SharedSpace can benefit meetings needs to be done in the future. Experiments that involve collaboration and task completion on the platform could help discover crucial features that should be added and flaws in SharedSpace.

VI. Limitations and Future Work

This application is definitely a prototype for now and has some limitations, but has a lot of potential to become an application that could be widely used. One of the current limitations of SharedSpace is the ability to have very large conference calls. Currently, Mediasoup can process around 15 users' media streams in a single C++ worker process [14] on our sever. Furthermore, when there are a lot of users in the call, it will be really hard to maintain a circular positioning format. Currently, SharedSpace shrinks the user videos every time an additional four users join the call to maintain the positioning, and user videos can become really small.

Even though there are some limitations, there are some solutions to these limitations. To get past the limitation of the number of media streams the Mediasoup library can process, we can distribute a single conference room across several servers or create more Mediasoup C++ worker processes on a single machine that manage a single conference room [14]. In order to get past the RS3D library limitation, we can use a more powerful library that does not have a limit on the number of input audio sources it can process. Lastly, in order to preserve the circular positioning as the number of users grow in the call, several circles could be made where users are seated in different circles rather than a single one. These circles could be within each other or be completely different ones.

There are numerous features that can be implemented in SharedSpace. One feature that would be interesting to see is having users upload their own personal HRTF so that each user can have a unique hearing experience during the conference call. Another feature could be adding the ability to move users around the room and having breakout rooms. Adding graphs that track the amount of time each user has spoken can be quite helpful to researchers. Also adding gaze tracking or eye tracking could help in producing better audio localization as mentioned before [19]. Lastly, adding an option to change between a bird's eye view layout and a horizontal view layout (mentioned in the related works section) could be useful in helping researchers determine which viewing format could be more effective.

VII. Conclusion

In this paper, we provide a working prototype of a video conferencing web application which utilizes the power of spatial audio and video formats along with relevant web technologies. The application provides the user with a bird's eye view of the conference call with a circular layout which is different from the grid-like view of user videos in typical video conferencing applications like Zoom. In addition, with the inclusion of spatial audio and video, previous research mentioned above has shown that users can have better speaker identification, retain more information, and have increased comprehension from video conference meetings. Although there are some limitations, there are a lot of improvements that can be made and features that can be added to make SharedSpace better.

Acknowledgment

We would love to thank the developers at Mediasoup for making such an amazing library. We would also like to thank Matthew Goupell, Rochelle Newman, and Shevaun Lewis in the UMD Hearing and Speech Sciences department for guiding us with the features in this project. We would like to thank David Gadzinski of VisiSonics for his help with RS3D.

REFERENCES

[1] R. Nadler, "Understanding "zoom fatigue": Theorizing spatial dynamics as third skins in computer-mediated communication," *Computers and Composition*, vol. 58, p. 102613, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S8755461520300748

[2] J. N. Bailenson, "Nonverbal overload: A theoretical argument for the causes of zoom fatigue," *Technology, Mind, and Behavior*, vol. 2, no. 1, 2 2021, https://tmb.apaopen.org/pub/nonverbal-overload. [Online]. Available: https://tmb.apaopen.org/pub/nonverbal-overload

[3] K. Fischer and T. Tenbrink, "Video conferencing in a transregional research cooperation: Turn-taking in a new medium," in *Connecting Perspectives. Videokonferenz: Beiträge zu ihrer Erforschung und Anwendung*, J. Döring, H. W. Schmitz, and O. Schulte, Eds. Shaker, 2003.

[4] E. A. Isaacs and J. C. Tang, "What video can and can't do for collaboration: A case study," in *Proceedings of the First ACM International Conference on Multimedia*, ser. MULTIMEDIA '93. New York, NY, USA: Association for Computing Machinery, 1993, p. 199–206. [Online]. Available: https://doi.org/10.1145/166266.166289

[5] L. M. Seuren, J. Wherton, T. Greenhalgh, and S. E. Shaw, "Whose turn is it anyway? latency and the organization of turn-taking in video-mediated interaction," *Journal of Pragmatics*, vol. 172, pp. 63–78, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0378216620302782

[6] R. Cutler, Y. Hosseinkashi, J. Pool, S. Filipi, R. Aichner, Y. Tu, and J. Gehrke, "Meeting effectiveness and inclusiveness in remote collaboration," *Proc. ACM Hum.-Comput. Interact.*, vol. 5, no. CSCW1, Apr. 2021. [Online]. Available: https://doi.org/10.1145/3449247

[7] K. Inkpen, R. Hegde, M. Czerwinski, and Z. Zhang, "Exploring spatialized audio & video for distributed conversations," 01 2010, pp. 95–98.

[8] S. Mehrotra, W. ge Chen, Z. Zhang, and P. A. Chou, "Realistic audio in immersive video conferencing," in *2011 IEEE International Conference on Multimedia and Expo*, 2011, pp. 1–4.

[9] D. Nguyen and J. Canny, "Multiview: Spatially faithful group video conferencing," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 799–808. [Online]. Available: https://doi.org/10.1145/1054972.1055084

[10] M. Hyder, M. Haun, and C. Hoene, "Placing the participants of a spatial audio conference call," 02 2010, pp. 1 – 7.

[11] R. Drullman and A. W. Bronkhorst, "Multichannel speech intelligibility and talker recognition using monaural, binaural, and three-dimensional auditory presentation," *The Journal of the Acoustical Society of America*, vol. 107, no. 4, pp. 2224–2235, 2000.

[12] J. J. Baldis, "Effects of spatial audio on memory, comprehension, and preference during desktop conferences," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '01. New York, NY, USA: Association for Computing Machinery, 2001, p. 166–173. [Online]. Available: https://doi.org/10.1145/365024.365092

[13] R. Kilgore and M. Chignell, "Listening to unfamiliar voices in spatial audio: Does visualization of spatial position enhance voice identification," 01 2006.

[14] I. Castillo *et al.*, "Mediasoup." [Online]. Available: https://mediasoup.org/

[15] Automattic, "Socket.io." [Online]. Available: https://socket.io/

[16] TrueConf, "Sfu (selective forwarding unit)." [Online]. Available: https://trueconf.com/blog/wiki/sfu

[17] Dialogic. Conquering scalable webrtc conferencing. [Online]. Available: https://cdn2.hubspot.net/hubfs/292250/Collateral/14726-scalable-webrtc-conf-wp.pdf?hsCtaTracking

[18] E. André, N. Le Breton, A. Lemesle, L. Roux, and A. Gouaillard, "Comparative study of webrtc open source sfus for video conferencing," in *2018 Principles, Systems and Applications of IP Telecommunications (IPTComm)*, 2018, pp. 1–8.

[19] D. Zotkin, R. Duraiswami, and L. Davis, "Rendering localized spatial audio in a virtual auditory space," *IEEE Transactions on Multimedia*, vol. 6, no. 4, pp. 553–564, 2004.

[20] Visisonics. [Online]. Available: https://www.visisonics.com/

[21] MDN, "Web audio api." [Online]. Available: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API

[22] H. Choi, "Audio worklet design pattern." [Online]. Available: https://developers.google.com/web/updates/2018/06/audio-worklet-design-pattern