Real-Time Sound Mapping of Object Rotation and Position in Augmented Reality Using Web Browser Technologies

Victor Vlad[a] and Sabin Corneliu Buraga[b]
Faculty of Computer Science, Alexandru Ioan Cuza University of Iaşi, Romania

Keywords: Spatial Audio, Augmented Reality, Web Browser, Object Tracking, Sound Localization, Web Audio API.

Abstract: Growing focus on immersive media within the browser has been driven by recent advances in technologies such as WebXR for augmented reality (AR), Web Audio API for spatial sound rendering and object tracking libraries such as TensorFlow.js. This research presents a real-time system for spatial audio mapping of physical object motion within a browser based augmented reality environment. By leveraging native Web technologies, the system captures the rotation and position of real world objects and translates these parameters into dynamic 3D soundscapes rendered directly in the user browser. In contrast to conventional AR applications that necessitate native platforms, the proposed solution operates exclusively within standard Web browsers, eliminating the requirement for additional installations. Performance evaluations demonstrate the system's proficiency in delivering low-latency, directionally precise sound localization in real time. These findings suggest promising applications within the interactive media domain and underscore the burgeoning potential of the Web platform for advanced multimedia processing.

1 INTRODUCTION

Nowadays, there is a growing interest in enabling real-time spatial interaction within Web browsers, particularly in the context of augmented reality (AR) and audio processing. Our paper explores the integration of physical object tracking with spatial sound rendering (Sodnik et al., 2006; Montero et al., 2019) using native Web technologies, evaluating the feasibility and performance of a fully browser-based system that maps real-world object motion to dynamic audio cues in real time. This work builds upon recent advancements in *WebXR* (McArthur et al., 2021), the *Web Audio API* (Matuszewski and Rottier, 2023), and *TensorFlow.js* library (Smilkov et al., 2019), which together enable low-latency, high-fidelity spatial experiences directly in modern (mobile) Web browsers such as Google Chrome and Mozilla Firefox. Moreover, the depth estimation for sound position tracking is achieved through the MiDaS model (Ranftl et al., 2022), adapted for in-browser execution. AR visualization is implemented using the WebXR API via the *A-Frame* library (Mozilla VR Team, 2025).

To evaluate the system's performance and reliability, we conducted a series of benchmark tests measuring computational latency, rendering frame rates, and audio spatialization accuracy across various browsers running on mobile devices.

This research holds significant value in academic and applied domains, including auditory training, human-computer interaction studies, assistive technologies for the visually impaired, interactive sound installations, and browser-based AR applications for educational, commercial and/or entertainment purposes.

These experiments were primarily conducted on Google Chrome version 136.0.7103.87 running on a Samsung Galaxy S24 smartphone with Android 14. We also performed several tests using Firefox for Android version 138.0.2. The device used for our study is equipped with a Snapdragon 8 Generation 3 processor and 12 GB of RAM. Naturally, execution performance may vary depending on the (mobile) Web browser, the device hardware, and the available system resources.

Paper Organization: The object tagging method is described in Section 2, followed by the sound mapping based on rotation (Section 3). Additionally, we present the depth estimation in Section 4. Section 5 details the position of a physical object in the AR context. We close with related approaches (Section 6),

[a] https://orcid.org/0009-0005-8504-1964
[b] https://orcid.org/0000-0001-9308-0262

Vlad, V. and Buraga, S. C.
Real-Time Sound Mapping of Object Rotation and Position in Augmented Reality Using Web Browser Technologies.
DOI: 10.5220/0013672800003985

In *Proceedings of the 21st International Conference on Web Information Systems and Technologies (WEBIST 2025)*, pages 39-46
ISBN: 978-989-758-772-6; ISSN: 2184-3252

conclusions, and further research work.

2 OBJECT TAGGING AND MODEL TRAINING WITH YOLOV11 AND TENSORFLOW

To perform the proposed research studies, a meticulously curated image corpus was assembled, encompassing a wide range of instances representing the target object. For experimental evaluation, the model was trained on our dataset comprising 100 images depicting a plastic toy – *Stitch*, a fictional figure from Disney's animated franchise *Lilo & Stitch* (Walt Disney Animation Studios, 2002). We personally captured these images from multiple viewpoints to ensure variability in pose and orientation. Each image underwent accurate annotation, including the precise bounding box that delineates the character and the corresponding viewing angle.

Figure 1: Sample images from the curated dataset showing the plastic toy Stitch captured from various viewpoints.

Using *Label Studio*[1], we carefully configured a labeling interface that allowed for the precise annotation of bounding boxes around each toy figure (see also Figure 1). This process was repeated for every image in the dataset.

After finalizing the annotations, we exported them in a format compatible with the *TensorFlow* (Smilkov et al., 2019) object detection pipeline.

The model was initially trained using YOLOv11s[2], followed by YOLOv11m, in order to evaluate and compare their effectiveness in detecting the *Stitch* figurine across the annotated image dataset. While YOLOv11m exhibited superior detection accuracy and recall, its increased architectural complexity and capacity resulted in overfitting, manifesting as a higher incidence of false positives. Conversely, YOLOv11s achieved a more balanced performance, characterized by a lower false positive rate and faster inference times (consult Table 1).

Table 1: Comparative performance of YOLOv11m and YOLOv11s in terms of precision, recall, and false positives.

Model	Precision	Recall	False Positives
YOLOv11m	85.1%	93.6%	High
YOLOv11s	**92.3**%	85.2%	**Low**

Table 2: Frame processing time and object counts for the first six frames on a mobile device.

Frame	#Objects Found	Processing Time (ms)
1	1	64.20
2	1	62.85
3	1	58.67
4	1	59.44
5	1	62.18
6	1	64.73

As shown in the Figure 2 and Table 2, the results confirm that the system can provide fast and reliable output. This is a significant factor when developing Web applications that handle video content in real-time.

YOLOv11s exhibits strong results with very few incorrect alerts and fast processing, making it a great candidate for mobile systems that need quick video processing. When used in Web applications running inside a browser, as shown in this paper, its speed helps keep things smooth for users while still finding objects with high trustworthiness.

Acquiring precise performance data from a Web browser running on a mobile phone presents significantly greater challenges compared to data processing on desktop platforms. This is due to the constrained access to low-level diagnostic tools and the influence of background processes and power-saving mechanisms that are more prominent in the mobile environments.

At a high level, the system initiates with a video source and continuously captures individual frames in a loop (Algorithm 1). For each successfully retrieved frame, the YOLO architecture is executed to perform precise object detection. In a concurrent manner, the MiDaS model is utilized to generate a corresponding depth map (see Section 4), providing spatial context

[1]*Label Studio*, a data labeling platform to fine-tune Large Language Models (LLMs), prepare training data or validate AI models – *labelstud.io* (Last accessed: May, 10th, 2025).

[2]Ultralytics YOLO11 – *docs.ultralytics.com/models/yolo11/* (Last accessed: May 14th, 2025).

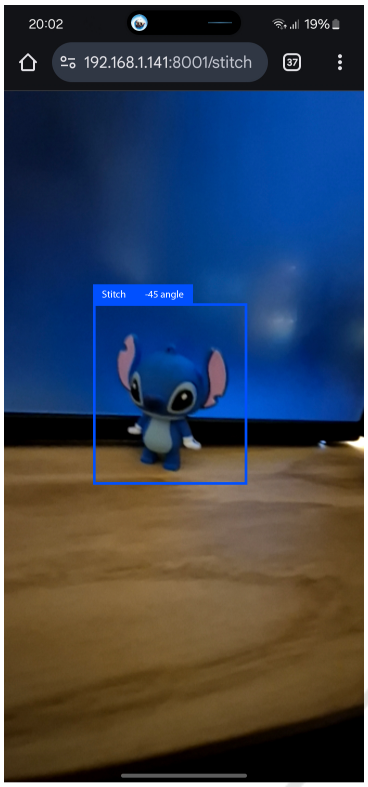

Figure 2: Precise detection on the mobile device of the orientation and bounding box of the Stitch toy.

Algorithm 1: Pseudo-code of the main loop: YOLO object detection and MiDaS depth estimation.

Data: Video source
Result: Processed frames with object detection and depth estimation
while *video stream is active* **do**
  Read the next frame from the video source
  if *frame could not be read* **then**
    Stop processing.
  end
  Perform object detection on the frame using YOLO.
  Estimate depth for the detected objects using MiDaS.
  Compute the spatial sound.
  (see Section 3 and Section 5)
end

and depth information for the detected objects. The iterative process continues until frame capture is completed.

3 SOUND MAPPING ROTATION

Based on the detected object, we propose an algorithm that modifies a sound signal based on the orientation of a 3D object, assuming the listener is stationary and continuously facing the object. The motivation behind this approach is to enhance spatial realism in audio rendering by dynamically adapting the sound based on the orientation of virtual objects. The current approach is designed to work with standard stereo MP3 aural sources. While limited to stereo for now, future work will focus on extending the system to support artificial multichannel formats such as 4.0 or 5.1, enabling richer and more immersive audio experiences. As such, the algorithm accounts for the angle between the object's facing direction and the listener's position, and adjusts the signal accordingly using a cardioid-based gain function and optional filtering.

To model directional sound radiation, we apply a cardioid gain function (Blauert, 1997):

$$g(\theta) = \frac{1+\cos(\theta)}{2} \tag{1}$$

where:

- $g(\theta)$ represents the gain based on direction,
- $\theta \in [0,\pi]$ is the angle between the object's forward direction and the position of the listener, expressed in radians.

This function produces the highest gain when the object is fully oriented toward the listener ($\theta = 0$), and the lowest gain when the object is turned away ($\theta = \pi$). This modulation can be interpreted as a simplified spatial effect similar to the Doppler phenomenon where a reduction in amplitude suggests that the object is turning away. The gain value scales the amplitude of the emitted sound as follows:

$$\hat{s}(t) = g(\theta) \cdot s(t) \tag{2}$$

where:

- $s(t)$ is the original sound signal produced by the object,
- $\hat{s}(t)$ is the resulting signal after gain adjustment, as perceived by the listener.

To improve the sense of realism, particularly by simulating the reduction of high-frequency content due to acoustic obstruction, we apply an optional low-pass filter. The cutoff frequency of this filter decreases as the object rotates away from the listener. Also can be used when object moves farther:

$$f_c(\theta) = f_{\max} - (f_{\max} - f_{\min}) \cdot \left(\frac{\theta}{\pi}\right) \tag{3}$$

where:

- $f_c(\theta)$ is the cutoff frequency of the filter based on angle,
- $f_{\max}$ is the highest cutoff frequency when the object is directed toward the listener,
- $f_{\min}$ is the lowest cutoff frequency when the object faces away,
- θ is again the angle between the object's main direction and the position of the listener.

This algorithm provides a effective solution for simulating rotational sound variation in AR environments – consult the Algorithm 2.

Algorithm 2: Rotation Based Audio Filter in JavaScript.

Data: `detectedAngle` (in degrees), `fMin`, `fMax`, `soundAmplitude`
Result: Computed gain, filtered amplitude, and cutoff frequency

Convert angle to radians:

```
const thetaRad = detectedAngle *
 (Math.PI / 180);
```

Define *cardioid* gain function:

```
function
 computeCardioidGain(theta) {
   return (1 + Math.cos(theta))
    / 2;
}
```

Define *cutoff* frequency function:

```
function
 computeCutoffFrequency(theta,
 fMin, fMax) {
   return fMax - (fMax - fMin) *
    (theta / Math.PI);
}
```

Compute gain:

```
const gain =
 computeCardioidGain(thetaRad);
```

Apply gain to sound:

```
const filteredAmplitude = gain *
 soundAmplitude;
```

Compute low-pass cutoff frequency:

```
const cutoffFrequency =
 computeCutoffFrequency(thetaRad,
 fMin, fMax);
```

Output `gain`, `filteredAmplitude`, `cutoffFrequency`;

The computational complexity of Algorithm 2 is also analyzed, which comprises a sequence of arithmetic operations:

- Conversion of the angle from degrees to radians is $O(1)$.
- Computation of the cardioid-based gain function.
- Computation of the cutoff frequency for a low-pass filter also a constant-time operation.
- Application of the gain to the input sound amplitude a single multiplication, again $O(1)$.

Since all steps use constant-time operations, the overall time complexity for processing a single object is:

$$T(n = 1) = O(1) \tag{4}$$

4 DEPTH ESTIMATION

Depth estimation can be performed with different levels of detail, from generating a full map of the scene to analyzing only the area around a detected object. To improve efficiency, we focus the depth calculation within the boundaries of a selected region, allowing the system to concentrate resources on the most relevant part of the computed image.

Table 3: Frame processing times for YOLO object detection and MiDaS depth estimation on a mobile device.

Frame Number	Objects Found	YOLO Time (ms)	MiDaS Time (ms)
1	1	62.35	113.52
2	1	64.10	115.88
3	1	60.78	111.94
4	1	63.56	114.25
5	1	61.90	112.80
6	1	65.00	116.30

To achieve this objective, we used *MiDaS*, a convolutional neural network trained on a diverse set of depth estimation datasets (Birkl et al., 2023). The model was first converted to the Open Neural Network Exchange (ONNX) format[3], which allows it to run directly in a Web browser. This approach enables the use of the client's local CPU, eliminating the need for a dedicated server for inference. After capturing a frame, the image is processed by transforming the pixel values into tensor representations that match the input structure required by the MiDaS model.

To evaluate the computational performance of the proposed pipeline, we provided in Table 3 a detailed breakdown of frame inference times for both the object detection and depth estimation components.

The YOLO model, responsible for real-time object detection, consistently performs inference within a range of approximately 60—65 milliseconds per

[3]Open Neural Network Exchange – *onnx.ai/onnx/intro/* (Last accessed: July 29th, 2025).

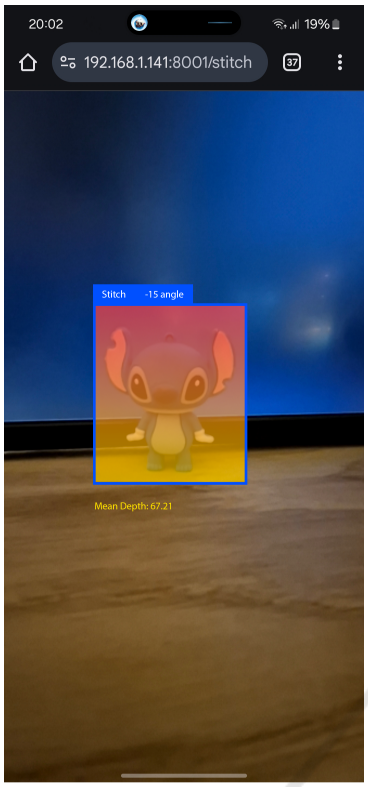

Figure 3: A video frame showing YOLO-detected objects with labeled bounding boxes, depth heatmaps overlaying the detections.

frame. In contrast, the MiDaS model, which estimates dense depth information for the corresponding frames, exhibits slightly higher latency, averaging between 111 and 116 milliseconds.

Despite this overhead, both models maintain response times suitable for near real-time Web applications.

5 SOUND MAPPING POSITIONING

To simulate spatial sound in AR environments, accurate object positioning is essential. The goal is to recreate how humans naturally perceive sound in the real world where direction, distance, elevation, and movement of sound sources all affect what we hear. We use monocular depth estimation from the MiDaS network (Ranftl et al., 2022) to infer 3D positions of objects from 2D camera input. This estimation allows us to derive the relative distance between the listener and sound source, which in turn informs amplitude attenuation and spatial filtering.

Assuming the listener remains at the origin $\mathbf{L} = (0,0,0)$ in 3D space, and an object is detected at $\mathbf{O} = (x,y,z)$, we compute the linear distance:

$$d = \sqrt{x^2 + y^2 + z^2} \tag{5}$$

We adopt a linear attenuation function capped at a maximum perceptual distance $d_{\max}$ according to (Tsingos et al., 2004). This function gradually decreases the gain with distance but ensures a floor of zero when $d \geq d_{\max}$. It maintains clarity for nearby sounds and reduces abrupt drop-offs:

$$g(d) = 1 - \min\left(\frac{d}{d_{\max}}, 1\right) \tag{6}$$

where:

- d is the distance between the sound-emitting object and the listener,
- $d_{\max}$ is the maximum perceptual distance beyond which the sound is fully attenuated,
- $\frac{d}{d_{\max}}$ normalizes the actual distance into the range $[0,1]$ relative to $d_{\max}$,
- $\min\left(\frac{d}{d_{\max}}, 1\right)$ ensures the normalized distance does not exceed 1, preventing negative gain values,
- $g(d)$ is the resulting gain, which decreases linearly from 1 to 0 as the object moves from the listener to the distance threshold.

This function behaves as follows:

- If $d = 0$, then $g(d) = 1$, then maximum volume.
- If $0 < d < d_{\max}$, then $g(d)$ decreases linearly.
- If $d \geq d_{\max}$, then no audible output $g(d) = 0$.

The sound signal perceived by the listener becomes:

$$\hat{s}(t) = g(d) \cdot s(t) \tag{7}$$

where:

- $s(t)$ is the original emitted signal,
- $g(d)$ is the distance-based linear gain,
- $\hat{s}(t)$ is the resulting signal as perceived by the listener.

The pseudo-code from Algorithm 3 outlines the real-time computation of this gain.

This approach enables sound levels to respond to the relative distance between objects and the listener in real-time. It uses a scaling method that is both computationally efficient and perceptually meaningful.

Also, we can analyze the time complexity of the position-based audio attenuation algorithm. The process of accessing values from the depth map and extracting 3D coordinates is assumed to take constant

Algorithm 3: Position Audio Attenuation Algorithm.

Data: `depthMap`, `soundAmplitude`, `dMax`
Result: Computed gain and perceived amplitude
Extract 3D position from MiDaS depth map:

```
const pos3D =
 get3DCoordinates(depthMap);
```

Compute distance from listener:

```
const d = Math.sqrt(
 Math.pow(pos3D.x, 2) +
  Math.pow(pos3D.y, 2) +
   Math.pow(pos3D.z, 2));
```

Compute linear attenuation gain:

```
const gain = 1 - Math.min(d /
 dMax, 1);
```

Apply gain to original sound amplitude:

```
const perceivedAmplitude = gain *
 soundAmplitude;
```

Output `gain`, `perceivedAmplitude`;

time, that is, $O(1)$. In addition, mathematical functions such as square root (`sqrt`), power (`pow`), and minimum (`min`) are treated as basic operations with constant time complexity.

Therefore, the overall time complexity of the algorithm for a single object is:

$$T(n=1) \equiv O(1) \tag{8}$$

When the algorithm is applied to n individual objects, each one is processed separately. In this case, the overall time complexity becomes:

$$T(n) \equiv O(n) \tag{9}$$

This result confirms that the method is well suited for use in real time environments where multiple sound sources are present.

Table 4: Performance results of position audio attenuation algorithm on an Android smartphone.

Frame Number	Time Taken (ms)	Average Attenuation (dB)	Memory Usage (MB)
1	1	-3	2.0
2	1	-6	2.0
3	1	-10	2.0
4	1	-15	2.0
5	1	-18	2.0

Additionally, we conducted several tests using actual MP3 files. The performance results are summarized in the Table 4 and illustrate the phone device's gradual movement away from the Stitch object, which serves as the primary sound source.

Throughout this motion, both the processing time and memory consumption exhibit a consistent pattern, underscoring the algorithm's noteworthy efficiency and reliability.

6 RELATED WORK

Recent advances in AR have enabled increasingly sophisticated interactions between physical and digital environments, particularly in the domain of audio-visual alignment and spatial sound rendering. This paper builds on prior work in spatial audio localization and physical object tracking in AR.

Spatial sound reproduction in virtual and augmented environments has been extensively studied. Techniques such as binaural rendering allow simulation of three dimensional sound fields with perceptually accurate directionality cues in the context of conventional applications (Wenzel et al., 1993; Zotkin et al., 2004) and Web (Fotopoulou et al., 2024).

Alternative approaches (Chelladurai et al., 2024) (Wald et al., 2025) concentrate on spatial haptic feedback for accessible sounds in virtual reality environments for individuals with hearing impairments. Subsequent work has focused on improving perceptual realism and reducing latency in rendering pipelines (Vazquez-Alvarez et al., 2012; Hirway et al., 2024), often within native (desktop or mobile) platforms and/or proprietary audio engines. Although these systems deliver high fidelity, they are not designed for use in Web browser environments.

The WebXR group of standards enable access to spatial sensors and camera input directly through the Web browser. Applications created with WebXR offer augmented experiences while avoiding the need for native app installation, enhancing both accessibility and portability (McArthur et al., 2021). Open source tools like A-Frame demonstrate that augmented reality in the browser is practical, although maintaining smooth, responsive interaction continues to be a challenge.

The latest advancements in spatial audio rendering – exemplified by different works such as Sonify-AR (Su et al., 2024), Auptimize (Cho et al., 2024), and enhanced sound event localization and detection in 360-degree soundscapes (Roman et al., 2024; Santini, 2024) – highlight the increasing sophistication of this technology in extended and augmented reality settings.

Also, various APIs provided by the modern (mobile) Web browsers[4] (e.g., Web Audio API) allow for

[4]The Web Platform: Browser technologies – *html-now.*

directional sound placement within browser environments. However, integration of these tools with dynamic object tracking from physical space is still limited. Early experimental systems demonstrate feasibility in combining WebXR, Web Audio, and WASM (Web Assembly) to synchronize object pose with real-time sound field updates (Tomasetti et al., 2023) (Boem et al., 2024).

These proposals point to promising directions, but lack the performance guarantees and hardware compatibility required for widespread adoption.

7 CONCLUSION

Clearly, spatial audio is an important element in immersive AR, allowing users to experience sound that responds to real-world motion and location. This work introduced a Web browser-based system for converting the rotation and position of a real object into spatial audio feedback within an AR setting.

By combining object tracking with three-dimensional sound rendering, built using JavaScript, we showed that interactive audio features can run efficiently on modern Web platforms. Performance tests on mobile Web browsers confirmed that the system delivers low latency and smooth execution with efficient audio processing.

To accomplish the research goals, we described object tagging in Section 2, sound mapping for rotation in Section 3. Also, depth estimation was detailed in Section 4 and location mapping for position in Section 5. Our obtained results confirm that *responsive, real-time audio interaction is achievable directly in the Web browser without the need for external plug-ins or native code*.

Future studies will build on this foundation by integrating dynamic environmental soundscapes and investigating innovative approaches in perceptual audio design (Schiller et al., 2024) (Batat, 2024). Drawing inspiration from previous studies like (Bhowmik, 2024), (Munoz, 2025), and (Peng et al., 2025), we also aim to incorporate richer physical object representations and more diverse interaction modalities to further extend the potential of the Web Audio API in various virtual/augmented/mixed reality experiences.

In addition, another research perspective may focus on usability evaluation with blind participants to validate system effectiveness in real contexts, and exploration of HRTF (head-related transfer function) (Cheng and Wakefield, 2001) binaural audio to enhance spatial perception.

github.io/ (Last accessed: July 29th, 2025).

REFERENCES

Batat, W. (2024). Phygital customer experience in the metaverse: A study of consumer sensory perception of sight, touch, sound, scent, and taste. *Journal of Retailing and Consumer Services*, 78:103786.

Bhowmik, A. K. (2024). Virtual and augmented reality: Human sensory-perceptual requirements and trends for immersive spatial computing experiences. *Journal of the Society for Information Display*, 32(8):605–646.

Birkl, R., Ranftl, R., and Koltun, V. (2023). Boosting monocular depth estimation models to high-resolution via content-aware upsampling. *arXiv preprint arXiv:2306.05423*.

Blauert, J. (1997). *Spatial hearing: The psychophysics of human sound localization*. MIT Press.

Boem, A., Dziwis, D., Tomasetti, M., Etezazi, S., and Turchet, L. (2024). "It Takes Two"—Shared and Collaborative Virtual Musical Instruments in the Musical Metaverse. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–10. IEEE.

Chelladurai, P. K., Li, Z., Weber, M., Oh, T., and Peiris, R. L. (2024). SoundHapticVR: head-based spatial haptic feedback for accessible sounds in virtual reality for deaf and hard of hearing users. In *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility*, pages 1–17.

Cheng, C. I. and Wakefield, G. H. (2001). Introduction to head-related transfer functions (hrtfs): Representations of hrtfs in time, frequency, and space. *Journal of the Audio Engineering Society*, 49(4):231–249.

Cho, H., Wang, A., Kartik, D., Xie, E. L., Yan, Y., and Lindlbauer, D. (2024). Auptimize: Optimal Placement of Spatial Audio Cues for Extended Reality. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–14.

Fotopoulou, E., Sagnowski, K., Prebeck, K., Chakraborty, M., Medicherla, S., and Döhla, S. (2024). Use-Cases of the new 3GPP Immersive Voice and Audio Services (IVAS) Codec and a Web Demo Implementation. In *2024 IEEE 5th International Symposium on the Internet of Sounds (IS2)*, pages 1–6. IEEE.

Hirway, A., Qiao, Y., and Murray, N. (2024). A Quality of Experience and Visual Attention Evaluation for 360 videos with non-spatial and spatial audio. *ACM Transactions on Multimedia Computing, Communications and Applications*, 20(9):1–20.

Matuszewski, B. and Rottier, O. (2023). The Web Audio API as a standardized interface beyond Web browsers. *Journal of the Audio Engineering Society*, 71(11):790–801.

McArthur, A., Van Tonder, C., Gaston-Bird, L., and Knight-Hill, A. (2021). A survey of 3d audio through the browser: practitioner perspectives. In *2021 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, pages 1–10. IEEE.

Montero, A., Zarraonandia, T., Diaz, P., and Aedo, I. (2019). Designing and implementing interactive and

realistic augmented reality experiences. *Universal Access in the Information Society*, 18:49–61.

Mozilla VR Team (2025). A-frame: A web framework for building virtual reality experiences. https://aframe.io/. Accessed: 2025-05-11.

Munoz, D. R. (2025). Soundholo: Sonically augmenting everyday objects and the space around them. In *Proceedings of the Nineteenth International Conference on Tangible, Embedded, and Embodied Interaction*, pages 1–14.

Peng, X., Chen, K., Roman, I., Bello, J. P., Sun, Q., and Chakravarthula, P. (2025). Perceptually-guided acoustic "foveation". In *2025 IEEE Conference Virtual Reality and 3D User Interfaces (VR)*, pages 450–460. IEEE.

Ranftl, R., Bochkovskiy, A., and Koltun, V. (2022). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Roman, A. S., Balamurugan, B., and Pothuganti, R. (2024). Enhanced sound event localization and detection in real 360-degree audio-visual soundscapes. *arXiv preprint arXiv:2401.17129*.

Santini, G. (2024). A case study in xr live performance. In *International Conference on Extended Reality*, pages 286–300. Springer.

Schiller, I. S., Breuer, C., Aspöck, L., Ehret, J., Bönsch, A., Kuhlen, T. W., Fels, J., and Schlittmeier, S. J. (2024). A lecturer's voice quality and its effect on memory, listening effort, and perception in a vr environment. *Scientific Reports*, 14(1):12407.

Smilkov, D., Thorat, N., Assogba, Y., Nicholson, C., Kreeger, N., Yu, P., Cai, S., Nielsen, E., Soegel, D., Bileschi, S., et al. (2019). Tensorflow.js: Machine learning for the Web and beyond. *Proceedings of Machine Learning and Systems*, 1:309–321.

Sodnik, J., Tomazic, S., Grasset, R., Duenser, A., and Billinghurst, M. (2006). Spatial sound localization in an augmented reality environment. In *Proceedings of the 18th Australia conference on computer-human interaction: design: activities, artefacts and environments*, pages 111–118.

Su, X., Froehlich, J. E., Koh, E., and Xiao, C. (2024). SonifyAR: Context-Aware Sound Generation in Augmented Reality. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, pages 1–13.

Tomasetti, M., Boem, A., and Turchet, L. (2023). How to Spatial Audio with the WebXR API: a comparison of the tools and techniques for creating immersive sonic experiences on the browser. In *2023 Immersive and 3D Audio: from Architecture to Automotive (I3DA)*, pages 1–9. IEEE.

Tsingos, N., Carlbom, I., Elko, G., Funkhouser, T., and Pelzer, S. (2004). Validating auralization of architectural spaces using visual simulation and acoustic measurements. In *IEEE Virtual Reality 2004*, pages 28–36. IEEE.

Vazquez-Alvarez, Y., Oakley, I., and Brewster, S. A. (2012). Auditory display design for exploration in mobile audio-augmented reality. *Personal and Ubiquitous computing*, 16:987–999.

Wald, I. Y., Degraen∗, D., Maimon∗, A., Keppel, J., Schneegass, S., and Malaka, R. (2025). Spatial Haptics: A Sensory Substitution Method for Distal Object Detection Using Tactile Cues. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–12.

Walt Disney Animation Studios (2002). Lilo & stitch. Directed by Dean DeBlois and Chris Sanders.

Wenzel, E. M., Arruda, M., Kistler, D. J., and Wightman, F. L. (1993). Localization using nonindividualized head-related transfer functions. *The Journal of the Acoustical Society of America*, 94(1):111–123.

Zotkin, D. N., Duraiswami, R., and Davis, L. S. (2004). Rendering localized spatial audio in a virtual auditory space. *IEEE Transactions on multimedia*, 6(4):553–564.