Head tracker using webcam for auralization

Conference Paper · August 2021

DOI: 10.3397/IN-2021-2956

CITATIONS
3

READS
745

5 authors, including:

Davi Rocha Carvalho
Universidade Federal de Santa Maria
5 PUBLICATIONS **10** CITATIONS
SEE PROFILE

William D'Andrea Fonseca
Universidade Federal de Santa Maria
116 PUBLICATIONS **151** CITATIONS
SEE PROFILE

Jacob Hollebon
University of Southampton
7 PUBLICATIONS **13** CITATIONS
SEE PROFILE

Paulo Mareze
Universidade Federal de Santa Maria
80 PUBLICATIONS **178** CITATIONS
SEE PROFILE

inter·noise 2021
1–5 August | Washington, DC

Head tracker using webcam for auralization

Davi Rocha Carvalho[1]
Acoustical Engineering, Federal University of Santa Maria (UFSM)
Av. Roraima nº 1000, Cidade Universitária, Bairro Camobi, 97105-900, Santa Maria, RS, Brazil

William D'Andrea Fonseca[2]
Acoustical Engineering, Federal University of Santa Maria (UFSM)
Av. Roraima nº 1000, Cidade Universitária, Bairro Camobi, 97105-900, Santa Maria, RS, Brazil

Jacob Hollebon[3]
Institute of Sound and Vibration Research (ISVR), University of Southampton
Highfield Southampton, SO17 1BJ, United Kingdom

Paulo Henrique Mareze[4]
Acoustical Engineering, Federal University of Santa Maria (UFSM)
Av. Roraima nº 1000, Cidade Universitária, Bairro Camobi, 97105-900, Santa Maria, RS, Brazil

Filippo Maria Fazi[5]
Institute of Sound and Vibration Research (ISVR), University of Southampton
Highfield Southampton, SO17 1BJ, United Kingdom

Abstract

Binaural rendering is a technique that seeks to generate virtual auditory environments that replicate the natural listening experience, including the three-dimensional perception of spatialized sound sources. As such, real-time knowledge of the listener's position, or more specifically, their head and ear orientations allow the transfer of movement from the real world to virtual spaces, which consequently enables a richer immersion and interaction with the virtual scene. This study presents the use of a simple laptop integrated camera (webcam) as a head tracker sensor, disregarding the necessity to mount any hardware to the listener's head. The software was built on top of a state-of-the-art face landmark detection model, from Google's MediaPipe library for Python. Manipulations to the coordinate system are performed, in order to translate the origin from the camera to the center of the subject's head and adequately extract rotation matrices and Euler angles. Low-latency communication is enabled via User Datagram Protocol (UDP), allowing the head tracker to run in parallel and asynchronous with the main application. Empirical experiments have demonstrated reasonable accuracy and quick response, indicating suitability to real-time applications that do not necessarily require methodical precision. Furthermore, cross-validation with existing hardware head trackers revealed an adequate agreement on measured head orientation, confirming its potential as a contactless head tracking device.

Keywords: Head tracker, Webcam, Binaural, Auralization, Python, Matlab, Real-time.

PACS: 43.66.Pn, 43.58.Ta, 43.60.Gk, 43.66.Qp, 43.60.Jn, 42.30.Tz.

[1]davi.carvalho@eac.ufsm.br, [2]will.fonseca@eac.ufsm.br, [3]j.hollebon@soton.ac.uk, [4]paulo.mareze@eac.ufsm.br, [5]filippo.fazi@soton.ac.uk.

1. INTRODUCTION

The binaural auralization of virtual scenes refers to the process of reproducing an acoustic field with the intention of representing a real or virtual environment [1] including the listener's perception of three dimensional sound source positioning. In general, the synthesis of spatialized audio perception is given by the re-creation of spectral cues and interaural time and level differences, generated by the interaction of the sound source at a given position in space and the listener's ear canals. Mathematically these spatialization mechanisms can be represented in the frequency domain by Head-Related Transfer Functions (HRTFs) or their time counterpart Head-Related Impulse Responses (HRIRs) [2, 3], when a free-field condition is considered. For non-free-field conditions, e.g. a classroom, Binaural Room Impulse Responses (BRIRs) are the most common nomenclature — while in the frequency domain some authors refer to Binaural Room Transfer Functions (BRTFs).

The process to actually reproduce the three-dimensional acoustic field may vary according to the adopted reproduction hardware. Whether choosing between different loudspeaker configurations or the use of headphones, each one has its intrinsic advantages and limitations. For the sake of simplicity in implementation, headphone reproduction is mostly referenced in this work. But one should be aware that any system that can accurately reproduce auditory fields, whether recorded or synthesized, can benefit from the knowledge of the listener's actual head orientation.

Considering that headphone auralizations can be reasonably achieved with the application of HRTFs [1] from the desired source position [2], without the information of the listener's current head orientation, the relative source coordinates would remain static independently of the subject's movement. In order to account for the real-time influence of the listener's pose, head tracking systems can be utilized [4]. It has been shown that this method impacts sound source realism, externalization, and reduces localization confusion when low-latency and stable sensors are used [5–7].

Many head tracking solutions rely on a physical device mounted to the listener's head [8–11], which for most applications presents the drawbacks of convenience, aesthetics, or even comfort. Convenience is affected as cables or batteries are needed; aesthetics, when the subject's own appearance is relevant during the head tracker use; and comfort, as extra weight must be carried, sometimes over long periods of use. With that in mind, this paper presents the initial investigations for implementing a head tracker in Python that relies primarily on webcam images and machine learning techniques. The following sections outline fundamental concepts regarding coordinate conventions and communication protocols, as well as the face detection libraries adopted for detection and tracking, as well as their limitations. The processes to connect to the head tracker from another application via network communication and a basic auralization pipeline are also presented.

2. AURALIZATION

Auralization is analogous to visualization, but instead of making something visible (for humans), it is the process of *making things audible*. Although a relatively simple concept, its contemporary meaning embraces techniques for recording, generating, processing, and reproducing sounds for binaural listeners [12]. Moreover, *binaural* signifies *having two ears*, in which the geometries involved help to localize sound sources in space — via cues, time, and level differences aforementioned.

In general, auralization is the process involving auditory-related steps when creating *virtual worlds*. That is, a virtual system may also include stereoscopic images and haptics, for example. Thus, for a subject to be convinced of authenticity in a virtual scene, the biology-physiology and processing of sensory signals must be understood (by the creator of the scene) [13, 14]. With that in mind, it is possible to take advantage of the fact that not all information is used from the acoustic event to the brain processing it. Therefore, the details can be narrowed down to only those needed to persuade the subject perceptions. Computationally speaking, these boundary conditions are crucial to achieving real-time virtualization, since computer power is limited.

[1] References to HRTFs or BRIRs can be freely interchanged within this paper according to the reader's preference.

3. COORDINATE SYSTEM

Besides translational orientation (relative to the subject's position in the room), the most common movements associated with head tracking, regard the subject's rotational degrees of freedom (turning-like motion, nodding-like motion, and tilting-like motion). To represent the rotation over each axis (named *x*, *y*, and *z*, usually) navigational nomenclature is adopted, where the rotations are respectively denominated pitch, yaw, and roll, as shown in Figure 1.

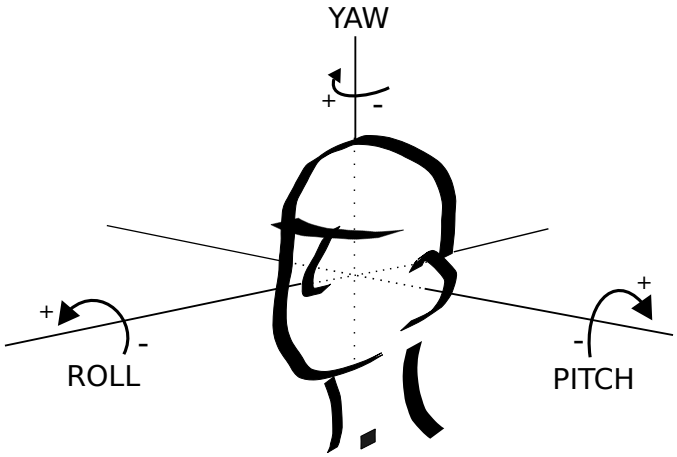


Figure 1: Yaw, pitch, and roll of the human head (coordinate system).

By definition, negative angles denote downwards pitch, counterclockwise roll, counterclockwise yaw, such that full rotation is bound between -180° and 180°. In this manner, the origin is defined as if the subject were facing straight ahead, in which yaw, pitch, and roll are equal to zero.

4. USER DATAGRAM PROTOCOL

The developed head tracking system is thought to be an independent application. While it can be called upon from any other master process, it necessarily needs to run in parallel for better overall performance. For that reason, together with the necessity to expand the use of the head tracker outside of the Python environment, a low-latency data transmission protocol is necessary.

The User Datagram Protocol (UDP) is a connectionless communication model, meaning that no handshaking is performed (between applications) in order to check if there is a receiver on the other end or to confirm if the potential listener has completely received the message sent [15, 16]. These properties may seem like a drawback at first glance. However, the lack of a robust error-checking protocol (like the three-way handshake in the Transmission Control Protocol (TCP) [17]) avoids the processing overhead of such mechanisms, making the UDP an efficient and low-latency implementation.

Considering that the head tracker is a system where only the latest values matter, having a protocol that ensures every sent packet will be received, would actually be detrimental to the system's functioning and latency. Given that both processes will start at different times and there is considerable discrepancy between the camera and audio sampling rates, both processes will most likely run asynchronously. If the head tracker tried to deliver every packet at any cost, eventually a queue of past head orientations could be accumulated at some point, even though only the current values taken would be relevant. For example, if the head tracker were turned on and the auralization engine were idling, it would try to deliver all the accumulated past data the moment the auralization engine would be turned back on if TCP were used instead of UDP.

5. THE FACE TRACKING MODEL

The Mediapipe library is a set of solutions for detection, tracking, and segmentation problems focused on virtual and augmented reality, that is approached using machine learning (ML) [18]. In this project, the Face Mesh module is used to track and estimate head orientation using a single camera input. This ML pipeline comprises two simultaneous working neural network models, where the first detects the face's location in the image with a very lightweight implementation [19], while the other fits a three-dimensional surface geometry onto the subject's face inside that area, based on relevant face landmarks.

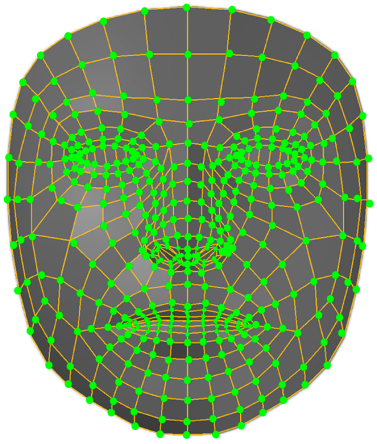


Figure 2: Face landmarks calculated with Face Mesh as default (adapted from [20]).

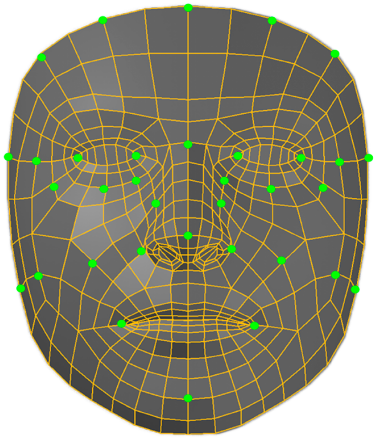


Figure 3: Selected face landmarks to estimate head pose (adapted from [20]).

As the main objective of the Face Mesh module is to be an augmented reality solution where virtual objects would interact with the subject's face, the landmark model estimates the location of 468 vertices [20], in order to capture a great range of facial expressions, as illustrated in Figure 2. However, in the attempt to further reduce processing requirements and more importantly to minimize the noisy influence of facial expressions in relation to the actual head pose, only a select few landmarks were used to calculate the rotation matrices. Figure 3 presents the final face landmarks used to estimate the subject's head orientation in this study.

5.1. Computer vision: the perspectives problem

The process of converting face landmarks into virtual space coordinates to the corresponding head pose observed in the real world is also the process of estimating the orientation of a 3D object onto a 2D image. This computer vision problem is known as "perspective-*n*-point" (PnP) [21]. In order to solve this PnP problem, the location of the landmarks in the 2D image, the location of these same points in a 3D space, and the intrinsic parameters of the camera such as focal length, optical center, and radial distortion are needed .

Ideally, the camera parameters are distinct for each individual hardware. Thus, their precise measurement can be very tricky. However, a reasonable approximation for webcams and smartphones is to use the optical center as the image's center, focal length as the image's width, and radial distortion as non-existent [22]. Even though the 3D coordinates are also an approximation (considering the subject's own 3D scan is not measured in this pipeline), in this way the use of an accurate neural network model to fit the landmarks to the subject's face ensures the quality of the data derived from them, since these coordinate values are directly extracted from the Face Mesh module.

In order to actually solve the PnP problem, the OpenCV [23] function `cv2.solvePnP()` is used. In it, the outputs, rotation, and translation vectors allow for the derivation of Euler angles (yaw, pitch,

and roll) relative to the rotation over the axis with the origin at the subject's head. The rotation vector is converted into a rotation matrix with Rodrigues' rotation formula [24, 25]. Finally, Euler angles are obtained from the projection matrix [26] (the column-wise stacking of the rotation matrix and translation vector) using the function `cv2.decomposeProjectionMatrix()`, such as shown in Code 1[2].

Code 1: Solve PnP and extract Euler angles (Python).

```
import cv2
import numpy as np
_, rvec, tvec = cv2.solvePnP(model_points, image_points, cam_matrix, dist_coeff)
rmat = cv2.Rodrigues(rvec)[0]
P = np.hstack((rmat,tvec))
euler_angles = cv2.decomposeProjectionMatrix(P)[6]
pitch = -euler_angles.item(0)
yaw = -euler_angles.item(1)
roll = euler_angles.item(2)
```

6. EXPERIMENTS: PERFORMANCE EVALUATION

A head tracker can be described in terms of at least three main characteristics; its resolution, the range of its coverage angles, and its sensor's accuracy. Note that different applications may have distinct requirements for each of these properties, which should be taken into consideration in order to select the most appropriate system for each use.

In this paper, the implemented head tracker has its resolution[3] truncated to 1° at every axis in order to account for noisy values as a consequence of intrinsic uncertainty in the process, such as poor lighting conditions, low-resolution cameras, or facial expressions. As for the system's accuracy, it is worth noting the Face Mesh Module's performance as an indirect indicator, in which the normalized mean absolute error distance was evaluated between predicted and target landmarks over multiple images yielded less than 3% [20]. As such, comparisons against hardware sensors were carried out, they are described in Section 6.1.

As for range of motion, the system is limited to the camera view of the subject's face, which may constrain the user's motion in some cases. Another consequence of this head tracking approach is that full torso rotation is not directly supported, as the camera would lose sight of the face image. Considering a static camera, the full range of motion contemplated by this approach would ideally be 180° for yaw, 180° for pitch, and 360° for roll. However, the results of a simple experiment to evaluate the range of angles that can be tracked are observed in Figure 4. The subject was asked to fully rotate his head on each axis individually while keeping a static torso. For the pitch and yaw, notice a sudden angle drop right at the peaks, which represents the exact moment the partial face occlusion from the camera's point of view restrained the algorithm from estimating the actual head pose. For the roll, on the other hand, as face occlusion is never a problem, the amplitude observed in the graph is only a matter of the subject's own neck flexibility.

The presented measurements for one subject indicate the maximum amplitude of motion for pitch and yaw to be 80° and 120°, respectively. There is no reason for the roll not to contemplate the full 360°, as the face would still be visible but upside down.

[2]Note that when considering only this code snippet, angle coordinates still need to be converted in order to match the standards in Section 3.

[3]It is also interesting to notice that the spatial resolution threshold of the human auditory system is around 1°∼ 2° in the horizontal plane (yaw) [27].

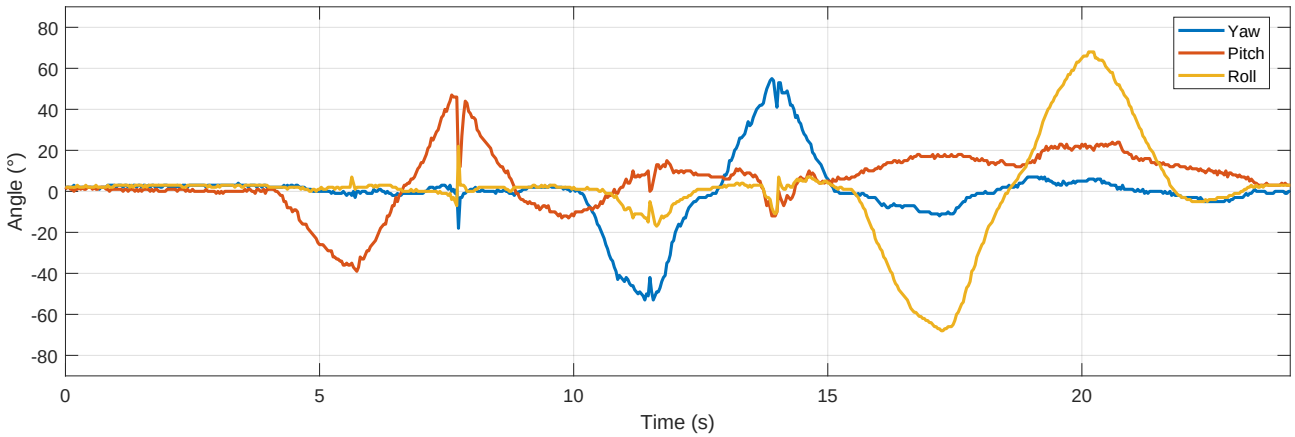


Figure 4: Head tracker motion extension measurement.

6.1. Hardware cross-validation

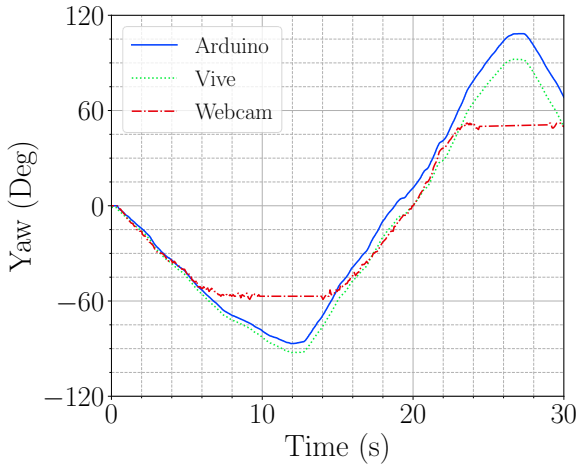
In order to test the accuracy of the developed head tracker, simultaneous measurements with two other existing hardware head trackers were carried out. The measurements took place in the audio laboratories at the Institute of Sound and Vibration Research (ISVR), University of Southampton. The first head tracker was a micro Arduino combined with a BNO055 orientation sensor [9, 11, 28], which interfaced with a measurement laptop via USB. This is a relatively cheap solution and is fairly non-invasive if mounted on a pair of headphones, however requires programming of the Arduino before use as well as a wired connection. The second head tracker was a HTC Vive Tracker [29], which required a separate laptop running SteamVR, a Python script utilising the Triad OpenVR library [30] to pull the tracking information into the Python environment, then UDP messaging to send the tracking data to the separate measurement laptop. This head tracker is wireless and also allows for absolute position tracking of the listener as opposed to just the head orientation. However, it requires more expensive hardware and the SteamVR software can be resource intensive, hence the use of a second laptop for running the tracker. Finally, the webcam head tracker was set up on a third laptop and transmitted the tracking data through UDP messaging. Where UDP communication was used, the computers were linked via hardwired connections through a single network switch to minimise latency. The head trackers were set up on/viewing a Neumann KU-100 Binaural Dummy Head [31], as seen in Figure 5. Due to the lack of facial features on the dummy head, to improve the facial detection a human-like mask was used.

The KU-100 was rotated manually through a yaw rotation between approximately $-90°$ to $+90°$ and the output of all trackers was recorded. The measurement was repeated 4 times, and the results are shown in Figure 6. It is clear that the webcam head tracker performs suitably well in comparison to the two existing head trackers with respect to the accuracy of the measured yaw angle. Indeed, all three head trackers closely map a similar measured yaw position throughout the measurements. The only exception is the already mentioned limit to the webcam yaw rotation, due to the occlusion of the face from the camera's point of view. For this experimental setup this limit was observed at approximately $\pm 60°$. Overall, all head trackers agreed on average within $\pm 3°$ of each other, excluding the results in this limited region of the webcam. The comparison might be better improved by measurements of the absolute latency of the head trackers, however a more suitable measurement rig must first be designed to investigate this issue. Despite this, it was informally observed that the webcam head tracker has a slightly increased latency, most likely due to the larger amount of processing required due to the image rendering. Furthermore, the refresh rate of the webcam tracker was lower than both other options. Despite this, the webcam head tracker is advantageous over the two compared head trackers due to its lack of further hardware required, zero additional cost and lack of requiring the user to wear an orientation sensor.

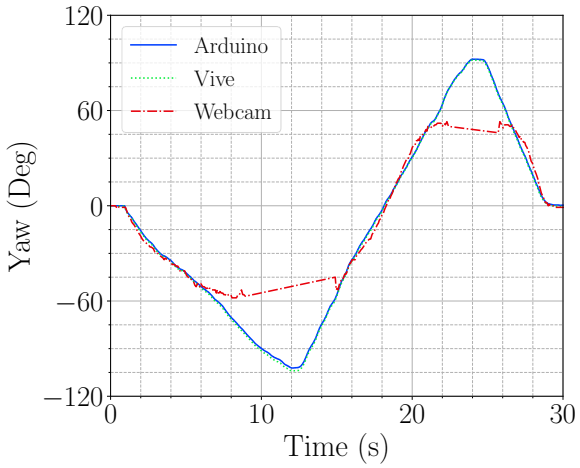
(a) Front view of the setup.

(b) Rear view of the setup.

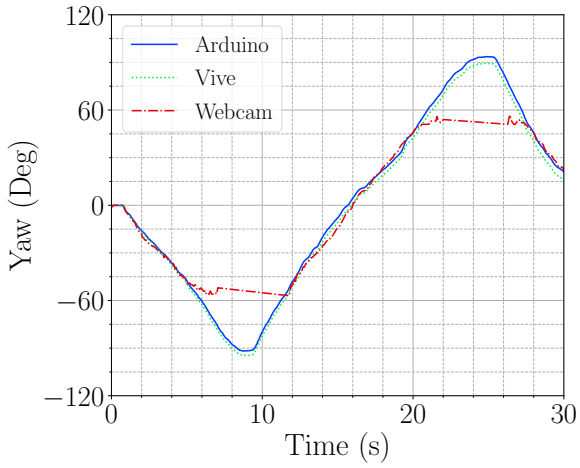
Figure 5: Experimental measurement setup. Neumann KU-100 with the three head trackers — the Arduino (attached to a pair of headphones), Vive (attached to the top of the head), and webcam (on the computer in front of the head).



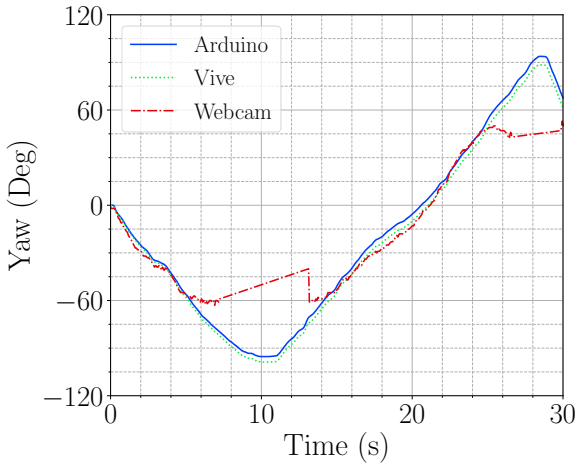
(a) Yaw Measurement 1.



(b) Yaw Measurement 2.



(c) Yaw Measurement 3.



(d) Yaw Measurement 4.

Figure 6: Head tracker comparison for a yaw rotation, repeated measurements.

7. APPLICATION: USING THE HEAD TRACKER

As soon as the software receives images from the webcam it starts sending its calculated yaw, pitch, and roll to the UDP socket via the IP address `127.0.0.1` and host port `50050`, so that any other applications connected to this address will receive the subject's motion information. The data itself are Python strings encoded into bytes, such as `b'-5,-2,10'`, where the numbers represent yaw, pitch, and roll, respectively. The coordinate system respects the standard previously elaborated in Section 3.

Code 2 illustrates in Python how the head tracker opens a UDP port and sends bytes containing single yaw, pitch, and roll orientation to the specified address. Code 3, on the other hand, shows in Matlab, as an example of cross-platform communication, how to connect to the same socket address opened in Python, convert the bytes to Matlab matrix and display the values in the command window.

The project repository on GitHub [32] contains further details about the developed codes, as well as a Matlab tutorial code to guide new users. Moreover, for Windows users, an executable file was released (`HeadTracker.exe`), so no further files are required to run the head tracker. Together with the example code `Auralization_ht.m`, it is possible to achieve real-time auralization.

Code 2: How to send data via UDP in Python.

```python
import socket
IP = '127.0.0.1'
PORT = 50050 # Arbitrary non-privileged (dynamic) port
s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
coord = '{yaw},{pitch},{roll}'.format(yaw=-5,pitch=-2,roll=10)
s.sendto(coord.encode(), (IP,PORT)) #send message
```

Code 3: How to connect the Head Tracker to Matlab.

```
open('HeadTracker.exe')   % Open the HeadTracker application
udpr = dsp.UDPReceiver('RemoteIPAddress', '127.0.0.1',...
                       'LocalIPPort',50050); % Connect to the local server
while true
    py_output = step(udpr); % Read data from server
    if ~isempty(py_output)
        data = str2num(convertCharsToStrings(char(py_output)));
        disp([' yaw:', num2str(data(1)),...
              ' pitch:', num2str(data(2)),...
              ' roll:', num2str(data(3))])
    end
end
```

7.1. Auralization pipeline

In order to build a binaural virtual environment that takes full advantage of real-time head pose information, one would ideally have HRTFs for multiple combinations of source positions and Head Above Torso Orientations (HATOs) [33]. However the process to precisely measure or simulate such a dense data structure is prohibitive for most laboratories, when a single HATO is already very time consuming or computationally expensive. Considering generic HRTF profiles[4], to the best of our knowledge the most complex model publicly released to date is the Fabian dataset [34]. It explored lateral range of motion relative to yaw with a head movement resolution of 10° and bound between -50° and 50°.

[4] Any HRTF different from the listener's own model.

Even though HRTF sets that comprise the full range of head motion are mostly limited to a fixed head orientation, and HATO interpolation or extrapolation is also a very recent research topic, it is still possible to take advantage of the listener's real-time head pose by estimating and correcting the auralization for the apparent source position, considering the difference between the source location and the relative head orientation. Figure 7 shows a basic signal flow regarding the binaural auralization process when taking into account the influence of a head tracker. Initially the apparent sound source position is calculated by the estimation of relative orientation between the subject's ears and the source in the virtual space. Then a pair of HRTFs is selected or interpolated based on HATO and the apparent source position. These HRTFs are then convolved with the current audio frame using Finite Impulse Response (FIR) filters to allow real-time reproduction and then written to the audio device buffer.

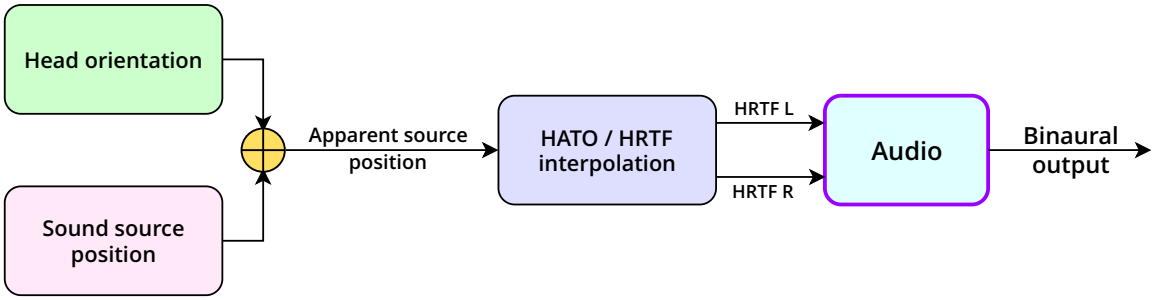


Figure 7: Simple auralization pipeline for binaural rendering with a head tracker.

Notice that other common auralization systems were omitted in this pipeline for the sake of simplicity. However, one is aware that systems such as headphones or room impulse responses are crucial to synthesize highly immersive auditory scenarios and externalized sound source perception [35]. Additionally, where a given parameter is also a function of space, e.g. Directional Room Impulse Responses, in some cases the head pose information can be equally integrated to evaluate the directional attributes of such variables.

The attached audio file [5] presents an auralization example of the above-mentioned pipeline. Using the head tracker and the Fabian HRTF dataset, the sound source was placed at a fixed position at 0° azimuth and elevation, while the subject rotated their head over the yaw axis (see the video in Figure 8 or in YouTube). Taking into account the auralization produced in this simple experiment considered head movements to correct for apparent source position, a third listener will instead perceive a moving source, unless they could precisely imitate the head position employed during that recording. This example also considers the subject's HATO, which is matched to its corresponding value in the Fabian set using the nearest neighbor approach. Only then is the corresponding HRTF selected, similar to the process described in Gomes et. al [36].

8. CONCLUSION

This paper has presented the implementational challenges for a head tracker based on face landmark identification. Beginning with a basic overview of the communication protocol adopted to connect the head tracker to any other application in the local network, the paper has also discussed a pipeline to generate auditory virtual scenes considering the integration with the head tracker presented.

Empirical experiments have demonstrated a limited range of coverage angles for yaw and pitch, as face occlusion may make the accurate estimation of face landmarks difficult after a certain threshold. However, the evaluated range may still be considered acceptable for many applications, such as when the subject needs to keep looking at the screen at all times (in games, for instance). On the other hand,

[5] Using Acrobat Reader, click on the icon or look for the attached files. The example audios of this paper were converted to mp3 320 kbps to save space, while the wave files can be accessed on the GitHub repositories eac-ufsm/internoise2021-headtracker/audios or eac-ufsm/webcam-headtracker.

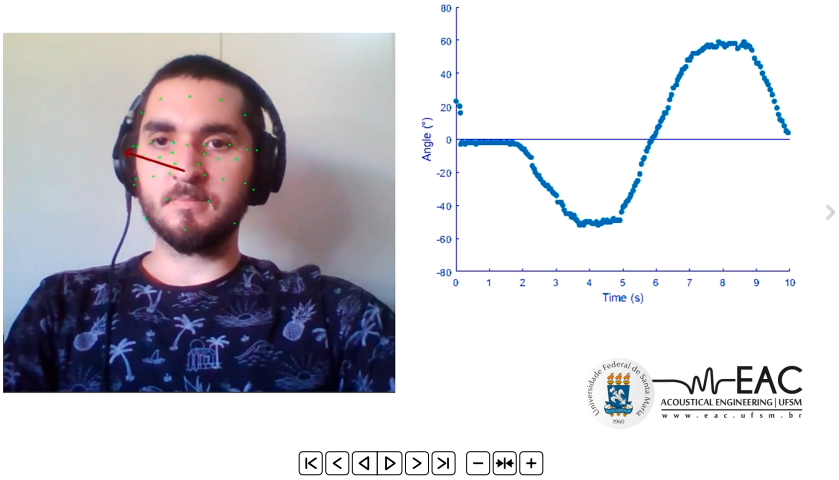


Figure 8: Video[6] of the audio attached to this paper (also available in YouTube). Depicted on the right side is the yaw tracked positions over time (the subject is the first author).

the viability of expansion to a stereo camera system could also be investigated, as a way of addressing full body rotation.

The accuracy of the head tracker has been validated through reported face tracking performance [20] and comparison to existing hardware head trackers through measuring yaw rotations of a binaural dummy head microphone. The results demonstrate good agreement between the newly developed head tracker and the existing solutions.

Unlike the usual hardware approaches, this implementation offers two distinct advantages. The first is that it is more comfortable for the listener, as no device needs to be mounted to their head. The second is that it is a low-cost solution that will not physically degrade over time and does not require extra hardware. On the other hand, the tracking is limited to the webcam/camera angle of view. This may come at the cost of a slightly increased latency due to the image processing that is required. However, a formal study into the latency of the tracker is suggested for future work.

The current user interface only shows the captured camera image using face landmarks and current orientation. Future versions shall include tweakable parameters, such as the options to select UDP host port and IP, to have a user-defined output data structure, and to specify the input camera device.

Support files of this project, including source code, binary files, and audio examples, can be accessed on the GitHub repository eac-ufsm/internoise2021-headtracker .

ACKNOWLEDGEMENTS

The authors want to thank all the support from the Acoustical Engineering Program at the Federal University of Santa Maria (UFSM, Brazil), as well as its scholarship programs FIPE and FIEX, which assisted this project. A special thanks also to Joseph Lacey for the text proofreading.

REFERENCES

[1] M. Vorländer. *Auralization, Fundamentals of Acoustics, Modelling, Simulation, Algorithms and Acoustic Virtual Reality*. RWTHedition. Springer, 2008. ISBN 978-3540488293. doi: 10.1007/978-3-540-48830-9.

[2] J. Blauert. *Spatial Hearing: The psychophysics of Human Sound Localization*. The MIT Press, Revised edition, 1996. ISBN 978-0262024136.

[6] Acrobat PDF Reader has capabilities to play the video and/or use the control bar in Figure 8.

[3] V. Pulkki and M. Karjalainen. *Communication Acoustics: An Introduction to Speech, Audio and Psychoacoustics*. John Wiley & Sons, Jan. 2015. ISBN 978-1118866542.

[4] F. Pausch and J. Fels. Localization Performance in a Binaural Real-Time Auralization System Extended to Research Hearing Aids. *Trends in Hearing*, 24:1–18, 2020. ISSN 23312165. doi: 10.1177/2331216520908704. PMID: 32324491.

[5] P. Stitt, E. Hendrickx, J.-C. Messonnier, and B. F. G. Katz. The Role of Head Tracking in Binaural Rendering. In *Proceedings of the 29th Tonmeistertagung - VDT*, pages 350–355, Nov. 2016. ISBN 987-3981283075.

[6] D. R. Begault, E. M. Wenzel, and M. R. Anderson. Direct Comparison of the Impact of Head Tracking, Reverberation, and Individualized Head-Related Transfer Functions on the Spatial Perception of a Virtual Speech Source. *Journal of the Audio Engineering Society*, 49(10):904–916, Oct. 2001. URL http://www.aes.org/e-lib/browse.cfm?elib=10175.

[7] P. Stitt, E. Hendrickx, J.-C. Messonnier, and B. Katz. The Influence of Head Tracking Latency on Binaural Rendering in Simple and Complex Sound Scenes. In *140th AES Convention*, pages 1–8, Paris, France, Jun. 2016. URL http://www.aes.org/e-lib/browse.cfm?elib=18289. Paper number: 9591.

[8] W. Hess. Head-Tracking Techniques forVirtual Acoustics Applications. In *133rd AES Convention*, pages 1–15, San Francisco, CA, USA, Oct. 2012. URL http://www.aes.org/e-lib/browse.cfm?elib=16524. Paper number: 8782.

[9] E. Bom, W. D'A. Fonseca, E. Brandao, and P. Mareze. Arduino-based head tracker: construction and use in acoustics (original: *Dispositivo rastreador de movimentos da cabeça baseado em Arduino: construção e utilização em acústica). Acústica & Vibrações*, 34(50):5–24, Dec. 2018. ISSN 1983-442X. URL https://bit.ly/head-tracker-ufsm.

[10] E. Bom. Development of binaural measurement and reproduction chains using head tracking device (original: *Desenvolvimento de cadeia de medição e reprodução biauricular utilizando dispositivo de rastreamento da cabeça*). Bachelor thesis, Acoustical Engineering, Federal University of Santa Maria, Santa Maria, RS, Brazil, Dec. 2018. doi: 10.13140/RG.2.2.16211.71205.

[11] M. Romanov, P. Berghold, M. Frank, D. Rudrich, M. Zaunschirm, and F. Zotter. Implementation and Evaluation of a Low-Cost Headtracker for Binaural Synthesis. In *142nd AES Convention*, pages 1–6, Berlin, Germany, May 2017. URL https://secure.aes.org/forum/pubs/conventions/?elib=18567. Paper number: 9689.

[12] H. Møller. Fundamentals of binaural technology. *Applied Acoustics*, 36(3):171–218, 1992. ISSN 0003-682X. doi: 10.1016/0003-682X(92)90046-U.

[13] J. Blauert, editor. *Communication Acoustics*. Springer-Verlag Berlin Heidelberg, 2005. ISBN 978-3540274377. doi: 10.1007/b139075.

[14] J. Blauert and J. Braasch, editors. *The Technology of Binaural Understanding*. Modern Acoustics and Signal Processing. Springer International Publishing, 2020. ISBN 978-3030003852. doi: 10.1007/978-3-030-00386-9.

[15] User Datagram Protocol. (STD 6, RFC 768), Aug. 1980. doi: 10.17487/RFC0768.

[16] L. Eggert, G. Fairhurst, and G. Shepherd. UDP Usage Guidelines. (BCP 145, RFC 8085), Mar. 2017. doi: 10.17487/RFC8085.

[17] V. Cerf, Y. Dalal, and C. Sunshine. Specification of Internet Transmission Control Program (TCP). (RFC 675), Dec. 1974. doi: 10.17487/RFC0675.

[18] C. Lugaresi, J. Tang, H. Nash, C. McClanahan, E. Uboweja, M. Hays, F. Zhang, C. Chang, M. G. Yong, J. Lee, W. Chang, W. Hua, M. Georg, and M. Grundmann. MediaPipe: A Framework for Building Perception Pipelines. *CoRR*, abs/1906.08172:1–9, 2019. URL http://arxiv.org/abs/1906.08172 or https://github.com/google/mediapipe.

[19] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann. BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs. *CoRR*, abs/1907.05047:1–4, 2019. URL http://arxiv.org/abs/1907.05047.

[20] Y. Kartynnik, A. Ablavatski, I. Grishchenko, and M. Grundmann. Real-time Facial Surface Geometry from Monocular Video on Mobile GPUs. *CoRR*, abs/1907.06724:1–4, 2019. URL http://arxiv.org/abs/1907.06724.

[21] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM*, 24(6):381–395, June 1981. ISSN 0001-0782. doi: 10.1145/358669.358692.

[22] S. Mallick. (2021, May 08) *Approximate Focal Length for Webcams and Cell Phone Cameras*. Learn OpenCV. URL https://learnopencv.com/approximate-focal-length-for-webcams-and-cell-phone-cameras/.

[23] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 25:120–125, 2000.

[24] O. Rodrigues. Des lois géométriques qui régissent les déplacements d'un système solide dans l'espace, et de la variation des coordonnées provenant de ces déplacements considérés indépendamment des causes qui peuvent les produire. *Journal de mathématiques pures et appliquées (1re série)*, 5:380–440, 1840. ISSN 0021-7874. URL http://sites.mathdoc.fr/JMPA/PDF/JMPA_1840_1_5_A39_0.pdf.

[25] G. Gallego and A. Yezzi. A Compact Formula for the Derivative of a 3-D Rotation in Exponential Coordinates. *Journal of Mathematical Imaging and Vision*, 51(3):378–384, August 2014. doi: 10.1007/s10851-014-0528-x.

[26] G. G. Slabaugh. Computing Euler angles from a rotation matrix. pages 1–7. URL http://eecs.qmul.ac.uk/~gslabaugh/publications/euler.pdf.

[27] D. W. Grantham, B. W. Y. Hornsby, and E. A. Erpenbeck. Auditory spatial resolution in horizontal, vertical, and diagonal planes. *The Journal of the Acoustical Society of America*, 114(2):1009–1022, 2003. doi: 10.1121/1.1590970.

[28] K. Townsend (Adafruit). (2021, May 28) *BNO055 Absolute Orientation Sensor*. Learn Adafruit. URL https://learn.adafruit.com/adafruit-bno055-absolute-orientation-sensor.

[29] HTC Corporation (Vive). (2021, May 28) *HTC Vive Orientation and Position Tracker*. URL https://www.vive.com/uk/accessory/vive-tracker.

[30] L. Beno. (2021, May 29) *GitHub repository for Triad OpenVR library (Python functions for SteamVR tracked systems)*. URL https://github.com/TriadSemi/triad_openvr.

[31] Neumann (Georg Neumann GmbH). (2021, May 29) *KU 100 Binaural Dummy Head Microphone*. URL https://en-de.neumann.com/ku-100.

[32] D. R. Carvalho, W. D'A. Fonseca, and J. Hollebon. (2021, May 25) GitHub repository for *Head tracker using webcam for auralization* paper. URL https://github.com/eac-ufsm/webcam-headtracker or https://github.com/eac-ufsm/internoise2021-headtracker.

[33] F. Brinkmann, R. Roden, A. Lindau, and S. Weinzier. Audibility and interpolation of head above-torso orientation in binaural technology. *IEEE Journal of Selected Topics in Signal Processing*, 9 (5):931–942, Aug. 2015. doi: 10.14279/depositonce-9004. doi: 10.1109/JSTSP.2015.2414905.

[34] F. Brinkmannand, A. Lindau, S. Weinzierl, G. Geissler, S. van de Par, M. Müller-Trapet, R. Opdam, and M. Vorländer. The FABIAN Head-Related Transfer Function data base. Feb. 2017. doi: 10.14279/depositonce-5718.

[35] V. Best, R. Baumgartner, M. Lavandier, P. Majdak, and N. Kopčo. Sound externalization: A review of recent research. *Trends in Hearing*, 24:1–14, Jan. 2020. doi: 10.1177/2331216520948390.

[36] L. M. Gomes, W. D'A. Fonseca, D. R. Carvalho, and P. Mareze. Rendering binaural signals for moving sources. In *Proc. 36th Reproduced Sound*, volume 42, Pt. 3, pages 1–12, 2020. ISBN 978-1906913380. doi: 10.25144/13386. URL https://bit.ly/rp2020-binaural.