Audio Engineering Society

Convention Paper 9689

Presented at the 142nd Convention
2017 May 20–23, Berlin, Germany

Implementation and Evaluation of a Low-cost Head-tracker for Binaural Synthesis

Michael Romanov[1,2], Paul Berghold[1,2], Daniel Rudrich[2,3], Markus Zaunschirm[2,3], Matthias Frank[2,3], and Franz Zotter[2,3]

[1] *Graz University of Technology*
[2] *University of Music and Performing Arts Graz*
[3] *Institute of Electronic Music and Acoustics, IEM*

Correspondence should be addressed to Michael Romanov (mromanov@gmx.de)

ABSTRACT

Human auditory localization strongly relies on head movements. Thus, for plausible perception of virtual acoustic scenes the incorporation of head movements is mandatory. This is achieved via loudspeaker playback as a listener can move the head relatively to the scene. When using binaural synthesis the headmovements need to be tracked and the scene needs to be rotated accordingly to achieve a stable perception of the acoustic scene. We present a low-cost, plug-and-play device (*MrHeadTracker*) to facilitate head-tracking based on the Arduino platform and the BNO055 sensor. Its performance is compared against another low-cost device (GY-85) and an optical tracking system (Optitrack Flex 13). The proposed *MrHeadTracker* outperforms the GY-85 device in terms of accuracy and latency and yields comparable results to the optical tracking system.

1 Introduction

The auditory perception of an acoustic event is based on the binaural signals received at the ear drums. Thus, an acoustic event can be reproduced by generating the corresponding binaural signals. This can be either achieved by binaural recording, e.g. with a dummy head, or binaural synthesis of arbitrary acoustic scenes using binaural impulse responses. These impulse responses can either resemble the directional information only (head-related impulse responses, HRIR) or contain additional room information (binaural room impulse response, BRIR). Convolution of an arbitrary source signal with binaural impulse responses can recreate the exact binaural signals that are indistinguishable from those caused by a real source in the measured room [1, 2]. Exactness is only achieved if both the recording and playback chains are acoustically transparent and if measurement and playback positions are identical (e.g. at the ear drum).

As a plausible perception of virtual acoustic scenes requires a playback that incorporates a listener's head movements, dynamic binaural synthesis is preferred over a static one. Dynamic synthesis is achieved by measuring binaural impulse responses for various source positions and head orientations followed by online head-tracking of the listener and inaudible interpolation of the binaural impulse responses accordingly.

The required interpolation between measured binaural impulse responses is often problematic due to the number of synthesized sources, the low-latency requirement and the desired high audio quality. In order to simplify the implementation, one could use a virtual *Ambisonics* approach as outlined in [3].

A typical setup of dynamic binaural *Ambisonics* rendering is depicted in Fig. 1. The $(N+1)^2$ *Ambisonics* channels (where N is the maximum order) are either real recordings of compact microphone arrays (EM32 or tetrahedral array) or source signals that are encoded according to a desired direction.

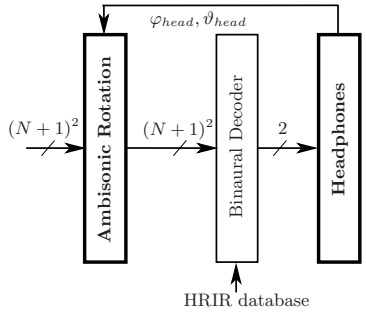

Fig. 1: Dynamic binaural rendering with *Ambisonics*.

Out of a measured set of individual or non-individual binaural impulse responses the ones that correspond to locations of a virtual loudspeaker setup are chosen for decoding. The actual frequency-independent decoder matrix is preferably calculated according to the All-Round Ambisonic Decoding (AllRAD) strategy [4]. For dynamic playback the *Ambisonics* channels are rotated, as stated in [5], contrary to the head movements of the listeners. The advantage of this approach is that no interpolation of binaural impulse responses is needed as the rotated scene is synthesized using the fixed set of filters corresponding to the number of selected virtual loudspeakers.

The plausibility and naturalness of binaurally rendered acoustic scene is highly dependent on the quality of the used head-tracking system, i.e. resolution and latency. In [6] is stated that a resolution of 2° is needed to obtain a synthesis that is free from artefacts, while Stitt [7] found that the stability of an acoustic scene degraded for latencies above 30 ms. In order to allow for mobile use, the tracker should be reasonably small to facilitate mounting direct on the headphones.

We present a compact, low-cost head-tracker with specifications that are in line with the recommendations and compare its performance against an optical tracking system and another low-cost device. An overview of the used components, the tracker calibration and the achieved latency is given in Sec. 2. Results comparing the accuracy of the proposed head-tracker against a large-scale optical tracking system and another low-cost device are presented in Sec. 3.

2 MrHeadTracker

2.1 Overview

The proposed small low-cost DIY head-tracker incorparates the *Bosch BNO055* 9-DOF absolute orientation sensor from *Adafruit* and an *Arduino Mini/Nano* (see Fig. 2).

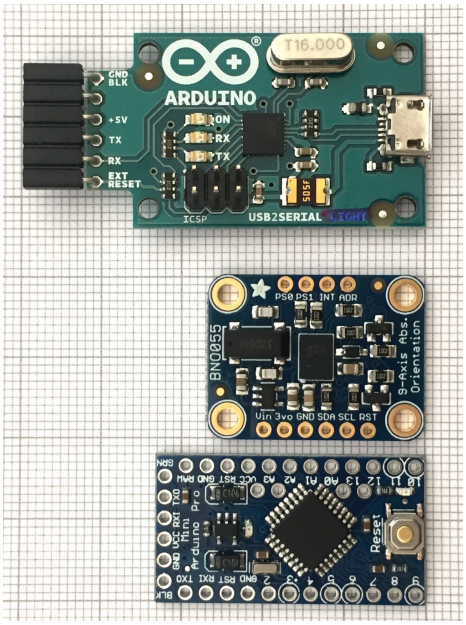

Fig. 2: Used components of the *MrHeadTracker*. Top-down: Arduino USB2Serial (24x42 mm, without connectors), Adafruit BNO055 (20x27 mm), Arduino Mini (18x33 mm).

The **BNO055** sensor comes with an advanced tri-axial 16 bit gyroscope, a versatile tri-axial 14 bit accelerometer and a full performance geomagnetic sensor with a range of 2000 degrees per second. The sensor-fusion of gyroscope and accelerometer provides relative orientation and gravity data with an update rate of 100 Hz.

The **Arduino Nano/Mini** board holding an ATmega328 micro-controller is connected to the sensor via I^2C and is used for data processing, i.e. calibration, conversion from quaternion to Euler format (Yaw, Pitch, Roll) and serial transmission. With its on-board USB connector the Arduino Nano is natively able to transmit the orientation data over a serial stream. However, additional software is needed for usage in a digital audio workstation (DAW) (e.g. *Hairless MIDI*[1] for conversion to MIDI).

To enable plug-and-play support without any additional software, the **Arduino USB2Serial** converter is used. Its Atmel 16u2 micro-controller can be programmed to appear as a class-compliant USB-MIDI device using the HIDUINO library[2]. As a consequence, the orientation data can be sent using MIDI CC messages, which can be interpreted by a variety of DAWs (e.g. *Reaper*). With this configuration, also the *Arduino Mini* board with smaller dimensions can be used, as the on-board USB connector of the Nano becomes redundant.

However when using MIDI, the resolution of a single CC value (7 bit) is not sufficient to reach the human JND for directional hearing (1°)[8]. For that reason, the proposed head-tracker increases this resolution to 0.02° with the usage of 14 bit MIDI (see Sec. 2.3).

For both operational modes (serial and MIDI) the update rate is limited by the BNO055 sensor, as the data processing on the Arduino Nano/Mini takes about only 2 ms. Although, the low baudrate of 31 250 bit/s for the MIDI transmission introduces a delay of about 1 ms for each sent CC value[3].

The minimal configuration also consists of a push button, which is needed reset the orientation and perform the calibration.

2.2 Calibration

The BNO055 sensor provides gravity data as vector $\mathbf{g}$ and the raw orientation data in quaternion format as $\mathbf{q}_{raw}$ for each sampling step. The latter one stands in reference to an internal reference system, which is - in general - unknown[4]. However, with an easy and quick calibration process, this data can be transformed into a calibrated reference system with the x-axis pointing into the user's *front* direction, the y-axis to the *left* and the z-axis to the *top*. The advantage using the proposed calibration procedure and raw data processing is the possibility to mount the sensor to the headphones with arbitrary position and orientation.

The calibration procedure goes down as follows: With pressing and holding the calibration button for longer than 1 s the $\mathbf{q}_{Idle}$ and the gravity vector $\mathbf{g}_{Idle}$ is stored. The user then nods and presses the button again while looking downward. The gravity vector in this position is stored as $\mathbf{g}_{Nod}$. This data is sufficient to calculate the new reference system, with predefining the z-axis pointing upwards (negative gravity vector). Therefore, the internal z-axis unit vector $\mathbf{z}$ is defined as

$$\mathbf{z} = -\frac{\mathbf{g}_{Idle}}{\|\mathbf{g}_{Idle}\|}. \tag{1}$$

With defining the nodding gesture being a rotation around the positive y-axis, the corresponding unit vector can be calculated with the normalized cross-product of both stored gravity vectors

$$\mathbf{y} = \mathbf{g}_{Nod} \times \mathbf{g}_{Idle} \tag{2}$$

$$\mathbf{y} \leftarrow \frac{\mathbf{y}}{\|\mathbf{y}\|}. \tag{3}$$

The cross-product of the y- and z-axis yields the x-axis

$$\mathbf{x} = \mathbf{y} \times \mathbf{z}. \tag{4}$$

The rotation matrix $\mathbf{R}$ defines the rotation of the calibrated coordinate system in reference to the sensor's internal one and is built by the three unit vectors of the three axes

$$\mathbf{R} = [\mathbf{x}, \mathbf{y}, \mathbf{z}]. \tag{5}$$

The conversion of the rotation matrix into the quaternion format yields the calibration quaternion $\mathbf{q}_R$.

During head movement after the calibration process, the steering quaternion $\mathbf{q}_{steering}$ represents the orientation in reference to the already rotated reference system by $\mathbf{q}_{Idle}$

$$\mathbf{q}_{steering} = \mathbf{q}^*_{Idle} \cdot \mathbf{q}_{Raw}, \tag{6}$$

with $\cdot$ denoting the quaternion multiplication and $*$ the complex conjugate.

[1] `http://projectgus.github.io/hairless-midiserial/`

[2] `http://github.com/ddiakopoulos/hiduino`

[3] A MIDI CC consists of three 7 bit data values. For each value a start, stop and identifier bit is added resulting in 30 bits per CC message. $t = \frac{30\,\text{bit}}{31\,250\,\text{bit/s}} = 0.96\,\text{ms}$.

[4] Actually, by default, the sensor calibrates its internal reference system with the aid of the earth's magnetic field and gravity.

With a quaternion consisting of an angle and a rotation-axis, the latter one has to be transformed into the calibrated reference system. This can be done by a two-sided quaternion multiplication yielding the orientation of the user within the calibrated reference system

$$\mathbf{q}_{Cal} = \mathbf{q}_R^* \cdot \mathbf{q}_{steering} \cdot \mathbf{q}_R. \tag{7}$$

As long as the sensor's mounting position is not subject to changes only one full calibration is necessary. As a consequence, the calibration data $\mathbf{q}_R$ can be stored in a non-volatile memory (e.g. EEPROM) and loaded at every startup. With updating $\mathbf{q}_{Idle}$ (e.g. by pressing the button shortly) the idle orientation can be reseted without influencing the calibration. Hence, the *front* direction can be changed that way.

2.3 14 bit MIDI resolution

With the standard MIDI resolution of 7 bits, the achievable angular resolution is 2.83°, which exceeds the JND of 1°. 14 bit Midi can be used, which is compatible with most DAWs (e.g. Reaper).

The 14 bit MIDI technique is based on the idea that a 14 bit word can be split into two 7 bit words, the LSB and MSB. In the MIDI protocol, the CC numbers 0 to 31 can be used for 14 bit MIDI. Therefore, they represent the MSB part, with the LSB part being located with an offset of 32 (CC numbers 32 to 63) [12]. The LSB (e.g. CC 20) has to be transmitted first, followed by the MSB (e.g. CC 52) in rapid succession to increase the resolution to 0.02°.

As a consequence, the data to be transmitted (angular data in radians) has to be transformed into two 7 bit integer values. This can be done by scaling the angle data by a scaling factor of $2^{14}/2\pi$ and converting the result to integer. The higher 7 bit then represent the MSB, the lower the LSB, respectively. Shifting and masking yields the individual parts, which can be transmitted as 7 bit integer data

$$MSB_{7bit} = (v_{14bit} \gg 7)\ \&\ 01111111_2 \tag{8}$$

and

$$LSB_{7bit} = v_{14bit}\ \&\ 01111111_2, \tag{9}$$

with v_{14bit} representing the already scaled value in 14 bit integer format and $\gg$ denoting the bit shifting operation.

3 Evaluation

3.1 Methodology and data processing

For the evaluation process the *MrHeadTracker* is compared against the GY-85 based tracking device [13] and the *Optitrack Flex 13* as a reference system. The two low-cost tracking devices and some reflectors for the optical tracking system were mounted on top of a AKG 272 headphone.

A PureData patch was used to receive data from the three different tracking systems via MIDI (*MrHeadTracker*), serial input (GY-85) and UDP (Optitrack). The data was sampled in time-steps of 1 ms and stored in a text file for further analysis with MATLAB.

Three movement trajectories were tested: A rotation around the *Yaw*, *Pitch* and *Roll* axis, each with five continuous iterations. In Fig. 3 the axes and rotation specification is presented. A test person was wearing the head-tracker and turned to and fro two times, separately for *Yaw*, *Pitch* and *Roll* head movements. The idle orientation was reset for all devices before each test case. For the first iteration the absolute angle offset to the null position was compensated, such that all three sensors started the tracking at an angular value of 0. This offset was also applied to the four consecutive iterations. To see if errors propagate over time no further calibration was done after the first iteration. In order to validate the test movements, the individual traces of *Yaw*, *Pitch* and *Roll* were confirmed not to contain contra-rotatory segments.

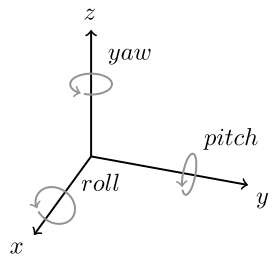

Fig. 3: *Yaw*, *Pitch* and *Roll* movements [5].

Fig. 3 exemplary shows the pitch movement as well as the deviations between the reference system and the time aligned tracking result of the tested devices for one trial.

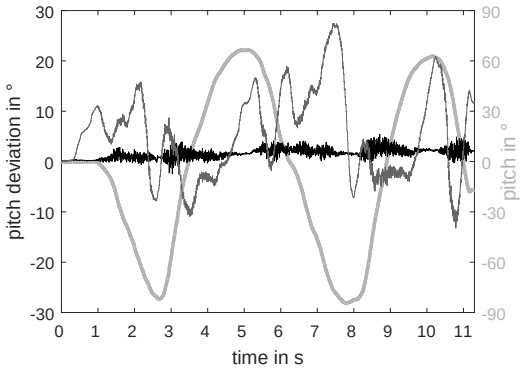

Fig. 4: *Pitch* deviations of the BNO055 based head-tracker (black line) and GY-85 based head-tracker (thin grey line) compared to the trace of the optical reference system (thick grey line).

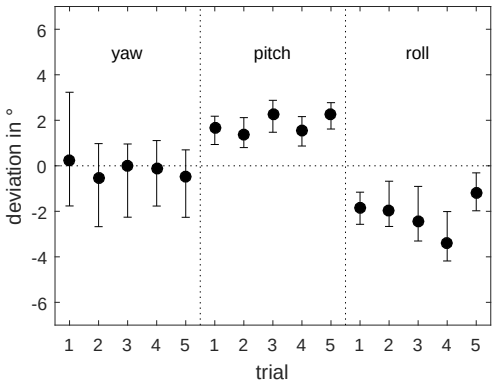

Fig. 5: Median and interquartile range of the error propagation for the BNO055 based tracker for *Yaw*, *Pitch* and *Roll* rotation in five continuous trials.

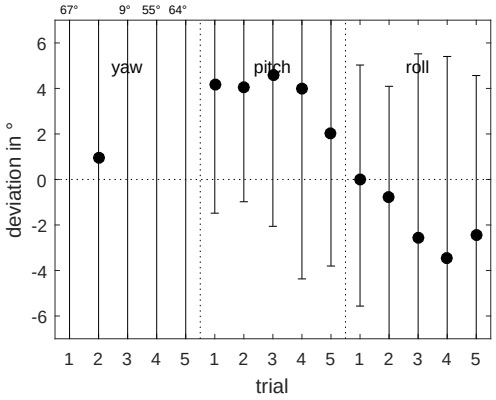

Fig. 6: Median and interquartile range of the error propagation for the GY-85 based tracker for *Yaw*, *Pitch* and *Roll* rotation in five continuous trials.

3.2 Results

Fig. 4 shows the deviation along the *Pitch* axis with respect to the Optitrack system. As the results of the test cases for the *Roll* and *Yaw* rotations look similar we use the *Pitch* case as an representing example.

The thick grey line represents the head movement for the optical reference system, with the corresponding axis scaling on the right side of the figure. The high fluctuating black line represents the angular deviation between *MrHeadTracker*and the reference system. The axis scaling on the left side of the figure corresponds to pitch deviation. The angular deviation between the GY-85 based tracker and the Optitrack system is represented by the darker grey line.

One can observe, that the *MrHeadTracker*shows up the greatest fluctuation of deviation when the slope of the movement is high. In contrast to this the GY-85 based tracker overestimates the movements and has the highest error when the direction of movement changes. In general one can see that the *MrHeadTracker*reaches with angular errors of $-7^\circ < \alpha_{diff} < 6^\circ$ and performs better compared to the GY-85 sensor making angular errors in a range of $-12^\circ < \alpha_{diff} < 27^\circ$.

In Fig. 5 and 6 a box-plot is presented for both portable tracking systems. Each of them represents the statistical error propagation for *Yaw*, *Pitch* and *Roll* rotation for five iterations. There are slight changes from one to another iteration noticeable, but no significant drift can be detected. We guess that this slight changes appear due to the experimental setup, as we had a real person performing the head movements. The GY-85 based tracker performed worst for movement along the *Yaw* axis, with a median angular error up to $\alpha_{diff} = 67^\circ$. *MrHeadTracker*shows the greatest error propagation at *Yaw* rotation but lies in a acceptable range of $\pm 1,5^\circ$.

3.3 Discussion

*MrHeadTracker*is able to track rotations very fast and does not exceed a maximum delay of $\Delta_{time} = 26\,\text{ms}$. Its response time of about 10 ms and angular standard deviation between 0.5° and 2.5° serves the requirements remarkably well for its target application, given its low pricing.

The GY-85 based tracker shows a general delay of about $\Delta_{time} = 0.47\,\text{s}$. Such a delay is audible and not sufficient for plausible binaural synthesis. Also the sensor has a slow response time and overestimates movements with every motion change.

During the experiments, we observed that the magnetic field of some headphone loudspeakers affect the GY-85 based head-tracker so much that it looses its orientation and tracks back to the starting position after a few seconds. This effect did not show up with the BNO055.

4 Summary and outlook

In this paper we presented the *MrHeadTracker*, a reliable plug-and-play head-tracking solution with small assembly effort at a reasonable price. Our evaluation showed that its performance for tracking of rotations can compete with a state-of-the-art optical tracking system. Its update rate and precision is sufficient enough to enable a smooth binaural synthesis.

A detailed description of the project, including a list of needed parts, instructions for assembly, source code and housing blueprints is available on `http://git.iem.at/DIY/MrHeadTracker`. There is also an extended version of the *MrHeadTracker* consisting of additional mode switches to select between different output data formats (quaternions or Euler angles) and different application modes (standard or reversed) as well as a small buzzer to indicate the calibration state.

Furthermore, we plan to work on an all-in-one solution with all the necessary parts on one circuit board resulting in a even smaller packaging size. Also a low-energy bluetooth transmission is one of our future goals.

References

[1] Møller, H., "Fundamentals of binaural technology," *Applied Acoustics*, 36(3-4):171–218, 1992.

[2] Bernschütz, B., "Microphone Arrays and Sound Field Decomposition for Dynamic Binaural Recording," page 264, 2016.

[3] Noisternig, M., Sontacchi, A., Musil, T., and Höldrich, R., "A 3D Ambisonic Based Binaural Sound Reproduction System," *Audio Engineering Society Conference: 24th International Conference: Multichannel Audio, The New Reality*, pages 1–5, 2003.

[4] Zotter, F. and Frank, M., "All-round ambisonic panning and decoding," *AES: Journal of the Audio Engineering Society*, 60(10):807–820, 2012.

[5] Kronlachner, M., and Zotter, F., "Spatial transformations for the enhancement of Ambisonic recordings," *2nd International Conference on Spatial Audio*, (2):1–5, 2014.

[6] Lindau, A., Maempel, H.J., and Weinzierl, S., "Minimum BRIR grid resolution for dynamic binaural synthesis," *The Journal of the Acoustical Society of America*, 123(5), 2008.

[7] Stitt, P., Hendrickx E., Messonnier J.C., and Katz, B., "The Influence of Head Tracking Latency on Binaural Rendering in Simple and Complex Sound Scenes," *Audio Engineering Society Convention 140*, 2016.

[8] David, R., Perrott, D.R., and Saberi, K., "Minimum audible angle thresholds for sources varying in both elevation and azimuth," *The Journal of the Acoustical Society of America*, 1990.

[9] Frank, M., Zotter F., and Sontacchi, A., "Localization experiments using different 2D ambisonics Decoders," *25th Tonmeistertagung – VDT International Convention*, 2008.

[10] Spors, S., Wierstorf, H., Raake, A., Melchior, F., Frank, M., and Zotter, F., "Spatial sound with loudspeakers and its perception: A review of the current state," *Proceedings of the IEEE*, 101(9):1920–1938, 2013.

[11] Avni, A., and Rafaely, B., "Sound localization in a sound field represented by spherical harmonics," *2nd Ambisonics Symposium conference*, 28:2–6, 2010.

[12] MIDI 1.0 Detailed Specification, Midi Manufacturers Association, Los Angeles CA, 1995.

[13] DIY Headtracker, http://www.rcgroups.com/forums/showthread.php?t=1677559, Accessed on 09.01.2017.