Principles of Echolocation
in Bats and Machines

Note 1

Why We Learn an HRTF With a Small MLP

Fourier features, smoothing on the sphere and smoothing across frequency

Jan Steckel
Cosys-Lab, University of Antwerp, and Flanders Make, Belgium jan.steckel@uantwerpen.be

Version 2026-10-31 15:23

1 What Is an HRTF?

A sound that reaches an ear has been filtered by the head on its way in. The pinna, the head and, for a bat, the noseleaf scatter the incoming wave, and the way they do this depends on both the frequency of the sound and the direction it comes from. The head-related transfer function (HRTF) captures exactly this filtering. For one ear, we write it as

$$H(f, \psi), \qquad \psi = (\theta, \varphi) \in \mathbb{S}^2, \tag{1}$$

with f the frequency, θ the azimuth (positive to the left) and φ the elevation (positive upward) in the frame of the head (Fig. 1). The direction ψ lives on the unit sphere $\mathbb{S}^2$, and we will often use its unit vector

$$\boldsymbol{u}(\psi) = [\cos\varphi\cos\theta,\ \cos\varphi\sin\theta,\ \sin\varphi]^T. \tag{2}$$

The HRTF is therefore a spectrospatial filter: for every direction it is a filter over frequency, and for every frequency it is a directivity pattern over the sphere. In what follows, we work with its magnitude in decibels, which is what our cochlear models (and the bat's cochlea) are sensitive to. For an echolocating bat, the emitter has a directivity as well, and what an echo carries is the product of the emitter and ear directivities, the echo-related transfer function (ERTF). Everything in this note holds for the ERTF in the same way.

Figure 2 shows the left-ear HRTF of this bat, computed with the boundary element method on the mesh of Fig. 1 [2]. Two things stand out. Firstly, the pattern narrows with frequency: at 30 kHz the ear hears a broad part of the left hemisphere, at 75 kHz only a small lobe remains, surrounded by structure. Secondly, in a fixed direction, the spectrum is a smooth curve with a few broad peaks and notches. Both observations will matter later on.

2 Why Not Learn Every Number?

Usually, an HRTF is measured or computed, and then simply used. In some models, however, the HRTF is not known beforehand and has to be *learned* as part of the model. An example is our self-supervised model of how a bat could learn its own ears [1] (Fig. 3). There, a simulated sonar head drives around, encodes the echoes of one call into a three-dimensional scene, moves that scene with its own self-motion, and predicts the echoes of a later call. To make that prediction, it needs the ERTF of its own head, and it starts from nothing more than "the left ear points left and the right ear points right". The prediction error is the only teacher, so the

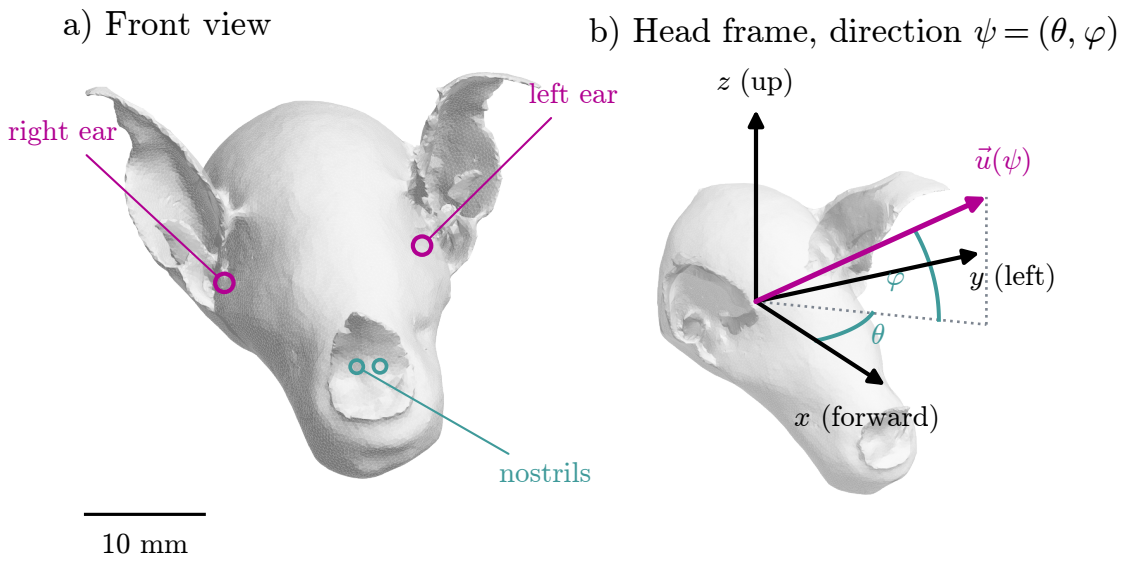


Figure 1: The head of *Phyllostomus discolor*, a CT scan of a preserved specimen. a) Front view, with the ears (where the HRTF is evaluated) and the nostrils (the emitter of this species). b) The head frame, with the origin between the ears, and a direction $\psi = (\theta, \varphi)$ with its unit vector $\boldsymbol{u}(\psi)$.

ERTF is learned together with everything else, and the way it is represented decides how well that works.

The most direct way to represent an HRTF that has to be learned is a table: sample the frontal hemisphere on a grid of 2.5° in azimuth and elevation (73 × 73 directions) and store the gain at each of 141 frequencies between 25 and 95 kHz. That is 751,389 numbers for one ear. An azimuth-elevation grid oversamples the poles, and sampling the hemisphere uniformly at the same spacing (for example on an icosphere) would need about 3,300 directions, but that still leaves about 465,000 numbers. Every one of them becomes a free parameter.

This is a bad idea for two reasons. The first is the sheer number of parameters, which all have to be pinned down by data. The second, and more important, reason is that the table has no structure. Each entry is independent of its neighbors, so an observation from one direction tells the table nothing about the direction 2.5° next to it, and an observation at 50 kHz tells it nothing about 50.5 kHz. Physically, this is absurd: the HRTF of a head a few centimeters across cannot change abruptly between neighboring directions or frequencies. A table cannot know this, so it can only memorize what it has seen, including the noise, and it has nothing to say about what it has not seen.

3 The HRTF as a Small Network

Instead of storing the HRTF, we can represent it as a function, and let a small neural network compute it. This is the idea behind neural radiance fields (NeRF) [3], in which a multilayer perceptron (MLP) maps a position in space to a color and a density. In our case, the MLP $g_{\boldsymbol{w}}$ with weights $\boldsymbol{w}$ maps a direction to the HRTF at all frequencies at once:

$$\hat{\boldsymbol{H}}(\psi) = g_{\boldsymbol{w}}\big(\boldsymbol{\gamma}(\psi)\big) \in \mathbb{R}^F, \qquad \hat{\boldsymbol{H}}(\psi) = [\hat{H}(f_1, \psi), \ldots, \hat{H}(f_F, \psi)]^T, \tag{3}$$

where $\boldsymbol{\gamma}(\psi)$ is an encoding of the direction that we will discuss in a moment. We use an MLP with two hidden layers of 128 units, as in our HRTF learning work [1]. Two properties make this a good representation. The first is that the network is a continuous function of the direction, so it can be evaluated in any direction, not only on a grid. The second is that every weight influences many directions and all frequencies at once. A sample from one direction therefore

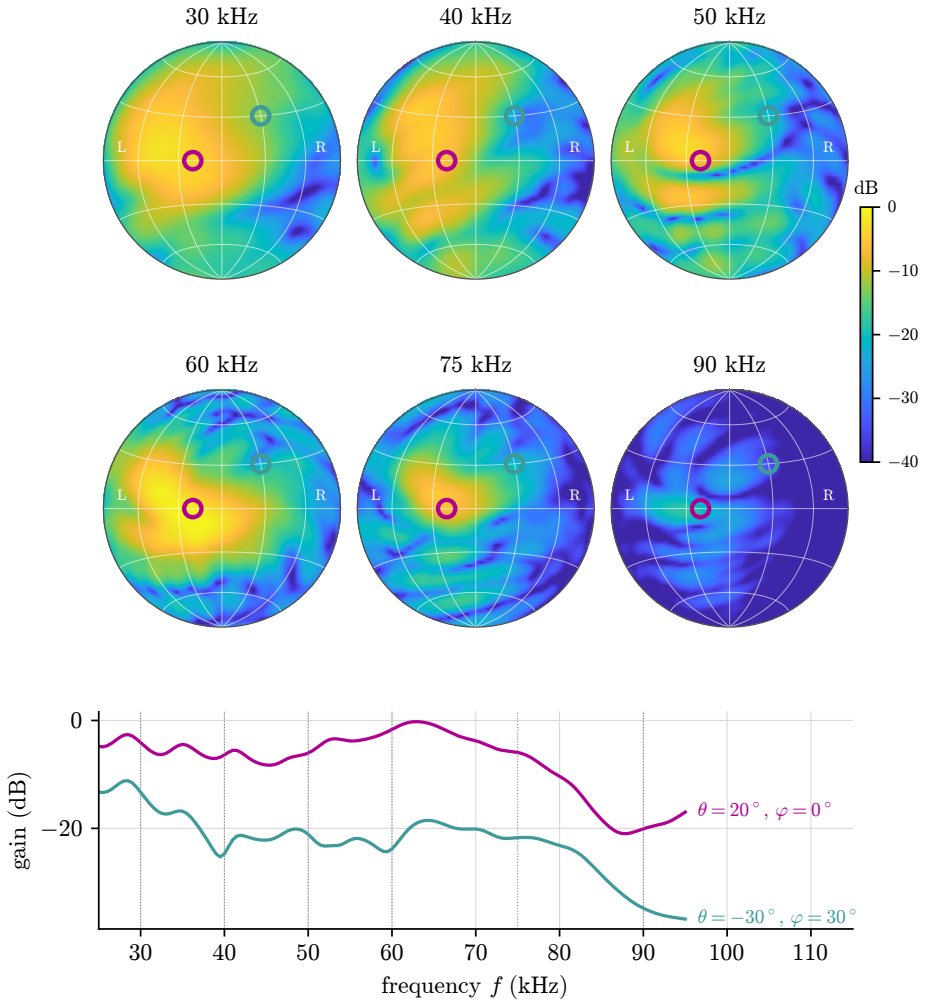


Figure 2: The left-ear HRTF of *P. discolor* at six frequencies (top), and its spectrum in the two directions marked by the circles (bottom, dotted lines at the frequencies of the discs). The discs show the frontal hemisphere in a Lambert azimuthal equal-area projection: the centre is straight ahead, the rim is 90° away, the bat is seen from behind (L and R mark its left and right) and the grid lines are 30° apart. Equal areas on the disc are equal solid angles.

also updates the neighboring directions, and the structure that the frequencies share (a lobe that narrows with frequency, a notch that moves) only has to be learned once. Figure 4 shows the complete model, including the two filters that we will add in the next sections.

The simplest choice for the input is the unit vector itself, $\boldsymbol{\gamma}(\psi) = \boldsymbol{u}(\psi)$. This works, but it gives a surprisingly blurry HRTF. Neural networks have a strong preference for smooth functions, called the spectral bias [5]: they learn the slowly varying parts of a function first, and the fine detail only very slowly, if at all. With only three smooth inputs, the 75 kHz lobe of Fig. 2 and the structure around it are out of reach (Fig. 7, top row).

4 Fourier Features on the Sphere

The fix that NeRF made popular is to feed the network Fourier features: instead of the raw coordinates, the network gets sines and cosines of the coordinates at a range of frequencies [4]. The raw direction changes only slowly over the sphere. To turn it into an output with fine angular detail, a network would need large weights and sharply bent activations, which are hard to train and numerically fragile: a small change of a weight then changes the output everywhere.

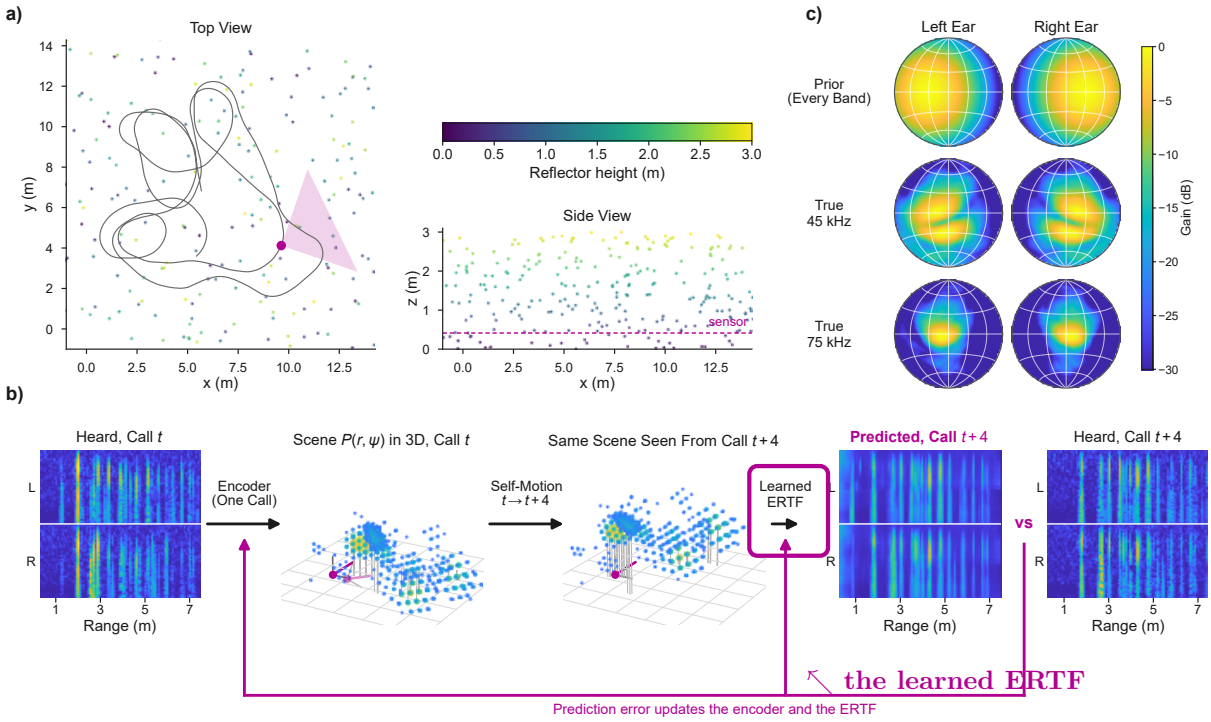


Figure 3: The self-supervised model of [1], in which the ERTF is learned. a) The simulated agent drives among point reflectors. b) The echoes of call t are encoded into a scene, the scene is moved with the agent's self-motion, and a renderer turns the moved scene into the echoes predicted for call $t+4$. The learned ERTF, the subject of this note, sits in that renderer (magenta box): it decides how loud every part of the scene is heard in each ear and frequency band. The prediction error updates the encoder and the ERTF. c) The ERTF at the start (a generic prior) and the true ERTF it has to find.

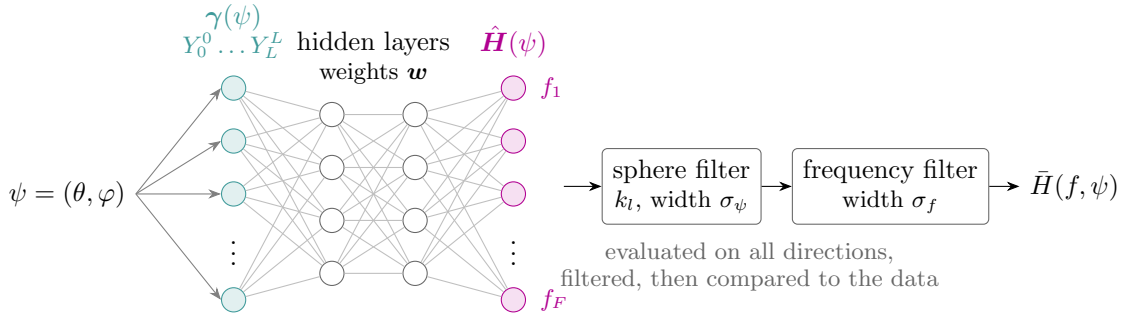


Figure 4: The HRTF as a small network. The direction is encoded into Fourier features on the sphere, an MLP maps them to the HRTF at all F frequencies at once, and two fixed filters smooth the result over directions and over frequency. Only the weights $\boldsymbol{w}$ are learned.

With Fourier features, the fine detail is already present in the input, and the network only has to combine the features with moderate weights, which trains smoothly and stably. On the sphere, the natural Fourier basis consists of the real spherical harmonics $Y_l^m(\psi)$, with degree $l = 0, 1, \ldots, L$ and order $m = -l, \ldots, l$ (Fig. 6a). We therefore use

$$\boldsymbol{\gamma}(\psi) = \sqrt{4\pi}\,\left[Y_0^0(\psi),\ Y_1^{-1}(\psi),\ Y_1^0(\psi),\ \ldots,\ Y_L^L(\psi)\right]^T, \tag{4}$$

which has $(L+1)^2$ entries. The degree l plays the role of the frequency of a Fourier series: Y_l^m varies over angles of roughly $180°/l$. With $L = 20$, as in our work, that is 441 features and an angular detail of about 9°.

How much angular detail does an HRTF need? A useful rule of thumb from acoustics is that an object of radius a scatters a sound of wavenumber $k = 2\pi f/c$ into spherical harmonics up to a degree of about ka. For the bat head of Fig. 1, with $a \approx 1$ cm, ka rises from about 5 at 25 kHz to about 17 at 95 kHz. Figure 5 checks this on the true HRTF, over the full sphere: the degree that holds 90% of the variance of the pattern indeed follows ka. The last few percent, the edges of the notches and of the deepest minima, reach higher, up to degree 47, and at most 9% of the variance lies above degree 20. Two conclusions follow. With $L = 20$, the features cover the bulk of the HRTF at every frequency. And the HRTF is (nearly) band limited on the sphere: strong high-degree content in a learned HRTF is not acoustics, but noise.

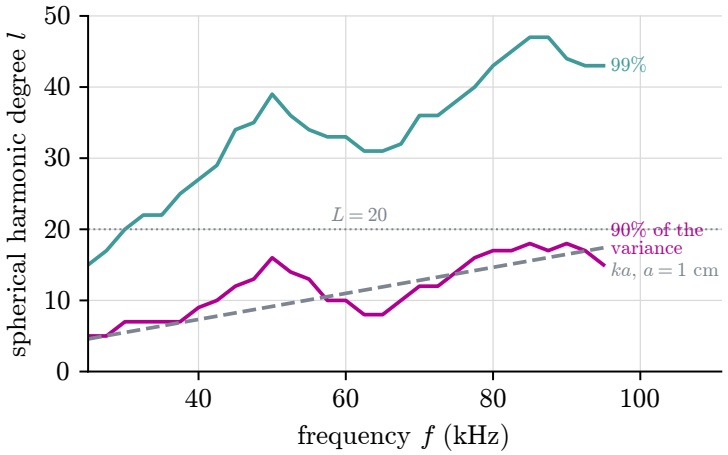


Figure 5: How many spherical harmonic degrees the true HRTF needs, per frequency: the degree below which 90% and 99% of the variance of the pattern (in dB, over the full sphere) lie, against the rule of thumb ka.

Note that with $\boldsymbol{\gamma}(\psi)$ as input, a network without hidden layers is simply a truncated spherical harmonic expansion, with one set of 441 coefficients per frequency (62,322 numbers in total). This is a classical way to represent an HRTF [6], and we include it in the comparison below. The MLP adds the sharing across frequencies.

Key idea 1

With Fourier features (spherical harmonics up to degree L), an output with fine angular detail can be learned from inputs that each vary in a simple, known way, with moderate weights and in a numerically stable way. From the raw direction alone, a network cannot learn that detail reliably.

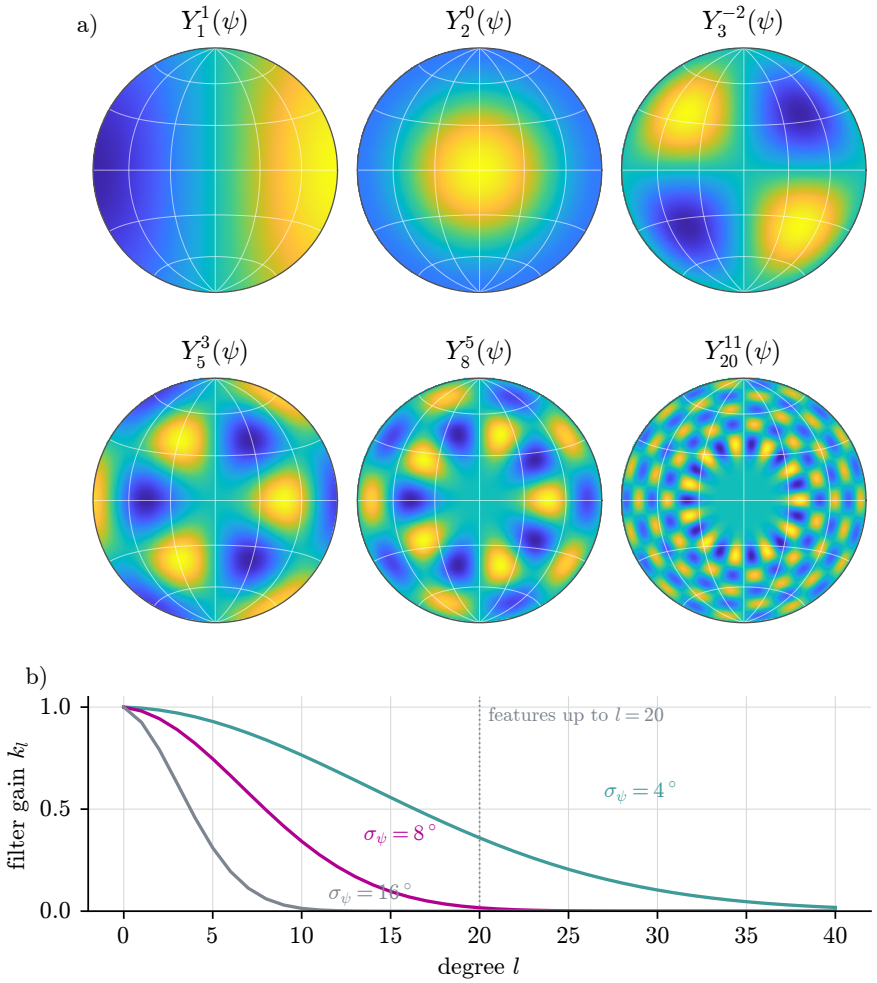


Figure 6: a) Six of the 441 Fourier features on the sphere, real spherical harmonics of increasing degree, over the frontal hemisphere (same projection as Fig. 2). b) The gain of the smoothing filter on the sphere for three widths. The filter leaves low degrees alone and suppresses the high ones.

5 Smoothing on the Sphere

Fourier features solve one problem and create another. A network that *can* represent fine detail will also happily represent detail that is not there. When the data are sparse or noisy, which is always the case in practice, the network fits the noise and produces a speckled HRTF (Fig. 7, bottom row). We know from the physics that this speckle cannot be real, so we build that knowledge into the model, by filtering the output of the network on the sphere.

How do we filter a function on a sphere? Recall how we would smooth a signal $x(t)$ in time. We take its Fourier transform, multiply every frequency component by the gain of a low-pass filter (for a Gaussian blur, a Gaussian that falls off with frequency), and transform back. On the sphere we do exactly the same, with the spherical harmonics in the role of the complex exponentials. Their degree l plays the role of the frequency: low degrees describe broad lobes, high degrees fine detail and speckle.

The first step is the Fourier transform on the sphere, the spherical harmonic transform. For every frequency f of the HRTF, we compute how much of each spherical harmonic the network output contains,

$$a_l^m(f) = \int_{\mathbb{S}^2} \hat{H}(f,\psi)\, Y_l^m(\psi)\, d\Omega \approx \sum_j w_j\, \hat{H}(f,\psi_j)\, Y_l^m(\psi_j), \tag{5}$$

which is the projection of the output onto Y_l^m, just like a Fourier coefficient is the projection of a signal onto a complex exponential. In practice, the integral becomes a sum over a grid of directions ψ_j with weights w_j (for a Gauss-Legendre grid, the sum is exact up to a known degree). The second step is the filter itself: we multiply every coefficient by a gain k_l that depends on the degree, and the third step is the inverse transform, summing the series again:

$$\tilde{H}(f,\psi) = \sum_{l=0}^{L'} k_l \sum_{m=-l}^{l} a_l^m(f)\, Y_l^m(\psi), \qquad k_l = \exp\Big(-\tfrac{1}{2}\, l(l+1)\, \sigma_\psi^2\Big). \tag{6}$$

This is the heat kernel on the sphere. Because the gain depends only on l and not on m, the filter treats all directions in the same way, and it is equivalent to convolving the HRTF with a bell-shaped kernel of width σ_ψ on the sphere, the spherical equivalent of a Gaussian blur. Figure 6b shows its gain. With $\sigma_\psi = 8°$, the gain drops to one half around degree 8, which suppresses the speckle while it keeps the lobes. The filter is a fixed, linear operation, so gradients flow through it during training, and the network learns an HRTF that *after* filtering explains the data.

6 Smoothing Across Frequency

The same argument holds across frequency. In a fixed direction, the spectrum of the HRTF is shaped by interference between waves that travel paths of slightly different lengths around the head and the pinna. A path difference Δr creates spectral peaks and notches spaced $c/\Delta r$ apart, and for path differences of a few centimeters, that is more than about 10 kHz. A learned spectrum that wiggles from one 0.5 kHz bin to the next is therefore noise as well. In every direction, we therefore smooth the output across frequency by convolving it with a Gaussian $g_{\sigma_f}(f)$ of width $\sigma_f = 1.5$ kHz (normalized to unit sum):

$$\bar{H}(f,\psi) = \big(g_{\sigma_f} * \tilde{H}\big)(f,\psi). \tag{7}$$

Both filters are linear and act on different axes, so their order does not matter.

Key idea 2

Real acoustics are smooth over direction and over frequency. Filtering the output of the network on the sphere and across frequency builds this into the model, so that the network can only explain the data with physically plausible HRTFs.

7 What Each Component Does

To see the effect of every component, we fitted the HRTF of Fig. 2 directly, in two situations. In the first, the learner sees 4,000 directions without noise (dense and clean), which tests whether a model can hold the detail of the HRTF at all. In the second, it sees only 600 directions with 3 dB of noise on every value (sparse and noisy), which is much closer to learning from real data. All models were trained with Adam for 4,000 steps on the mean squared error in dB, and scored on all 5,329 directions against the true HRTF (solid-angle weighted). Figures 7, 8 and 9 show the results.

The table. With dense, clean samples, the table is exact where it has data and knows nothing elsewhere, which still gives an error of 3.57 dB over all directions. With 600 noisy samples, it is the worst of all models (9.75 dB): it reproduces the noise in the sampled cells, and the rest of the sphere stays empty.

Fourier features. On clean data, the MLP with Fourier features is by far the most accurate model (0.39 dB), against 1.85 dB for the MLP on the raw direction and 1.30 dB for the plain

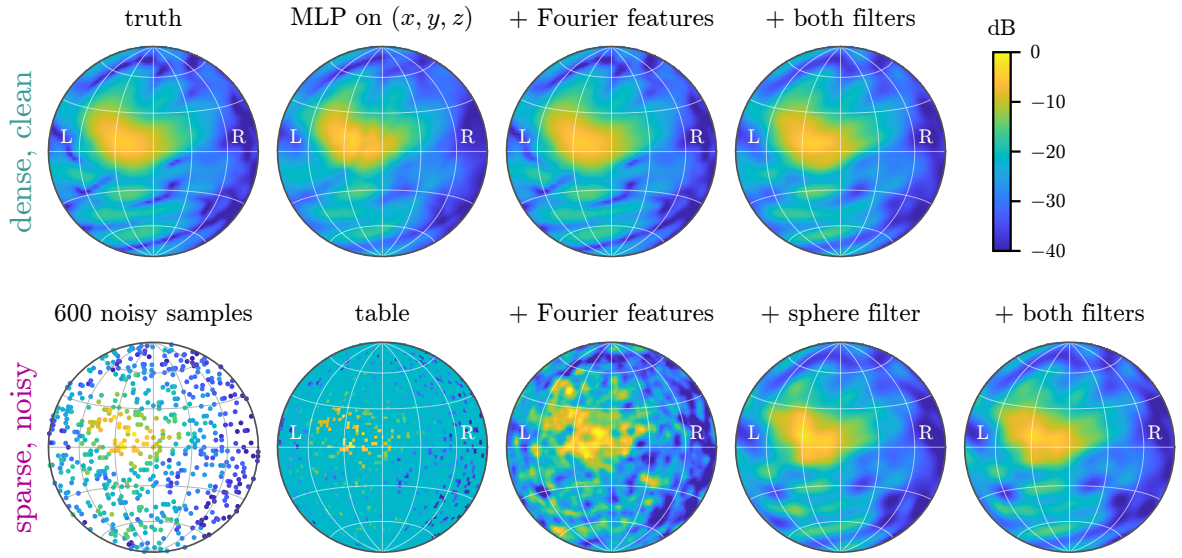


Figure 7: The learned HRTF at 75 kHz. Top row: from dense, clean samples. The MLP on the raw direction is too blurry, and Fourier features recover the detail. Bottom row: from 600 noisy samples. The table only knows the sampled cells, the MLP with Fourier features fits the noise, and the two filters remove it.

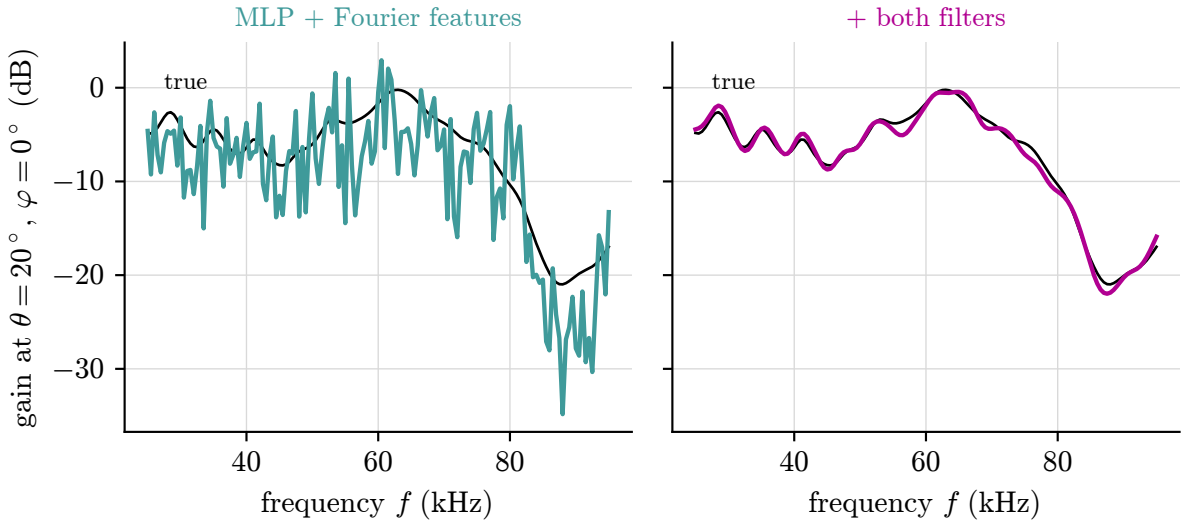


Figure 8: The learned spectrum in one direction, from the sparse, noisy samples, against the true spectrum (black).

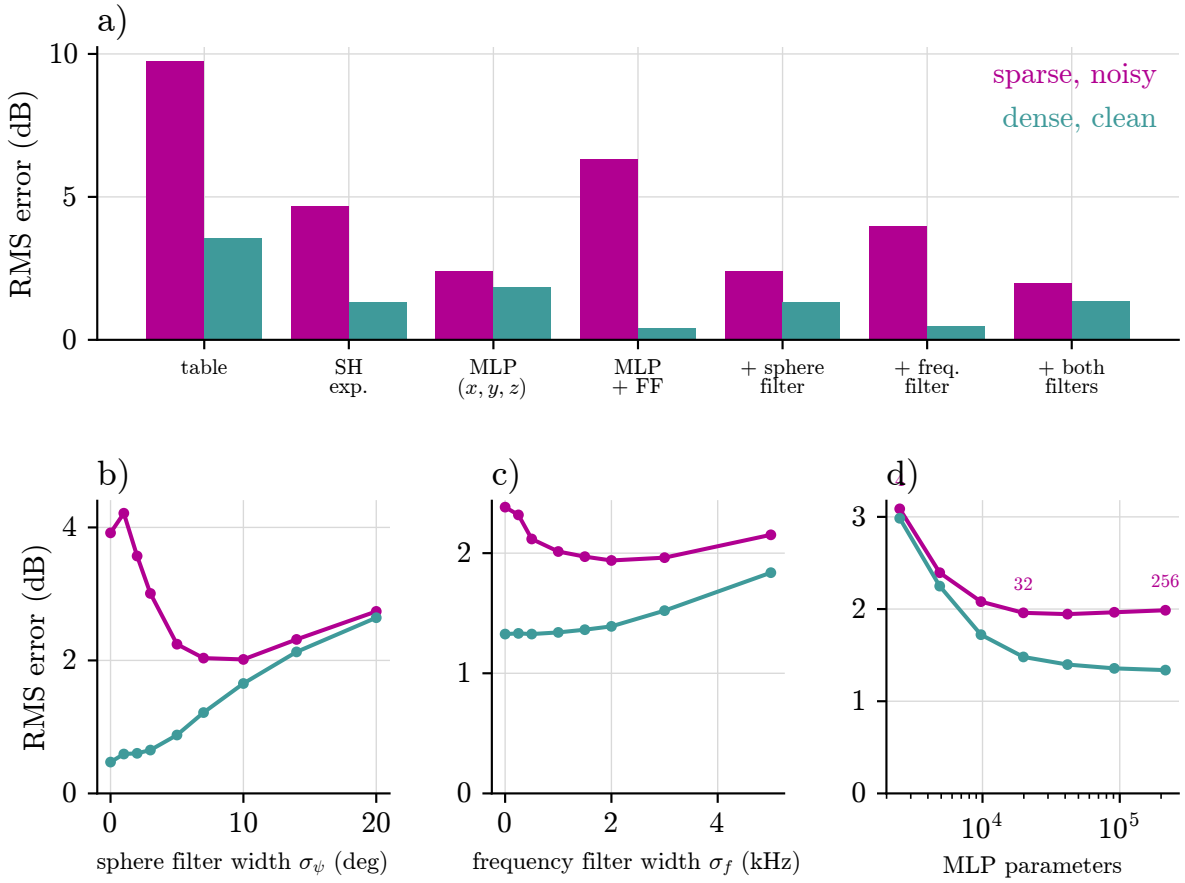


Figure 9: a) RMS error of every model in both situations. b, c) The error as a function of the width of the sphere filter and of the frequency filter (the other filter at its default width). d) The error as a function of the size of the MLP (both filters on), against the number of parameters.

spherical harmonic expansion. The last comparison is instructive. The expansion and the MLP see the same 441 features, but the MLP combines them nonlinearly, which lets it build detail above degree 20 and share structure across frequencies. On noisy data, however, the same model is the second worst (6.33 dB), as it spends its capacity on the noise (Fig. 7).

The filters. On noisy data, the sphere filter brings the error from 6.33 to 2.39 dB, and the frequency filter on top of it to 1.97 dB, the best of all models. The frequency filter alone helps less (3.98 dB), since the speckle over directions is the larger problem. Figure 8 shows what the filters do to a spectrum. On clean data, the frequency filter is almost free (0.47 dB), whereas the sphere filter costs detail (1.33 dB), because the few percent of the HRTF above the filter's band (Fig. 5) are removed as well. Figure 9b shows this trade-off. On noisy data, the error is lowest for a width of about 10°, while on clean data every bit of smoothing costs accuracy. The width of the filter is therefore a statement about how much we trust the data, and 8° is a compromise.

The MLP without Fourier features. The MLP on the raw direction does surprisingly well on noisy data (2.39 dB), because its spectral bias acts as a smoothing filter as well. However, that smoothing is implicit. We cannot choose its width, and it does not go away when the data get better, which is why this model falls behind on clean data. With Fourier features and filters, the smoothing is explicit, and its width has a physical meaning: an angle on the sphere and a bandwidth in kilohertz.

How small can the network be? Figure 9d varies the number of hidden units, with both filters on. With 32 hidden units (19,853 parameters), the MLP is as accurate as with 128 or 256 on noisy data (1.96 against 1.97 dB), and only about 0.1 dB worse on clean data. Even 16 units

(9,741 parameters) give 2.08 dB. Most of these parameters sit in the first layer, which maps the 441 features to the hidden units. A network of 32 hidden units stores the HRTF of one ear in about 38 times fewer numbers than the table, and about three times fewer than the spherical harmonic expansion, while it is more accurate than both.

Key idea 3

The result is a compact representation. Twenty thousand numbers describe an HRTF that a table stores in three quarters of a million, and they describe it better from the same data, because the structure of the model (Fourier features, a shared network and two filters) matches the structure of the acoustics.

8 In Our HRTF Learning Work

In our self-supervised HRTF learning work [1], the bat never observes its HRTF directly. It only hears echoes, and the HRTF is learned because it is the only way to predict how the echoes change when the head moves. That makes the signal much weaker and noisier than in the experiment above, so the structure of the model matters even more. The ERTF network there is the one of equation (3): spherical harmonic features up to degree 20, two hidden layers of 128 units, and 128 outputs (two ears and 64 cochlear bands), 89,856 parameters in total. One difference is how the smoothness is imposed. Instead of fixed filters, the paper adds two penalties to the loss, a total variation over directions and one over bands, which penalize steep edges only linearly so that genuine notches can survive. The idea is the same: the data decide the HRTF, within the set of HRTFs that a real head can produce. The experiments above suggest that this network could be smaller still, although we have not tested that in the self-supervised setting.

Exercise

1. In this note, we only considered functions on the sphere that are smooth, as an HRTF is. What would happen if we wanted to represent a discontinuous function on the sphere, for example a map that is 1 inside the field of view of a sensor and 0 outside? Would the two filtering steps still be useful, or should they be removed? And what would happen to the Fourier features?

2. Assume we have a large data set of HRTFs, of many bats of many species, and we fit one small MLP to every HRTF. What could we do with the fitted MLP parameters?

References

[1] J. Steckel, “Bats could learn three-dimensional echolocation using echoacoustic flow,” preprint, 2026.

[2] J. Steckel, “Full-sphere emission and reception directivities of seven echolocating bat species,” submitted, 2026. Data: https://doi.org/10.5281/zenodo.23729269.

[3] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi and R. Ng, “NeRF: Representing scenes as neural radiance fields for view synthesis,” ECCV, 2020.

[4] M. Tancik, P. P. Srinivasan, B. Mildenhall et al., “Fourier features let networks learn high frequency functions in low dimensional domains,” NeurIPS, 2020.

[5] N. Rahaman, A. Baratin, D. Arpit et al., “On the spectral bias of neural networks,” ICML, 2019.

[6] M. J. Evans, J. A. S. Angus and A. I. Tew, “Analyzing head-related transfer function measurements using surface spherical harmonics,” J. Acoust. Soc. Am., 104(4), 2400–2411, 1998.