FAST TIME-DOMAIN SPHERICAL MICROPHONE ARRAY BEAMFORMING

Zhiyun Li and Ramani Duraiswami

Perceptual Interfaces and Reality Lab., CS and UMIACS, Univ. of MD, College Park, MD 20742

ABSTRACT
Capturing and reproducing 3D audio has several applications. Spherical array based beamforming is an important technique in this field. However, such a beamformer is very expensive to implement, especially for interactive applications. In this paper, we propose a fast spherical beamforming algorithm in the time domain. It uses pre-computed data and can be steered to arbitrary 3D directions. It is ideal for interactive audio applications such as 3D games, etc. Simulation and experimental results are demonstrated to verify our algorithm.

1. INTRODUCTION
Spherical microphone arrays are a promising technology in 3D audio applications. They are the natural choice to sample the sound field at a point spatially. The beamformer based on a spherical array decomposes the captured sound field into a converging sequence of orthogonal basis functions, i.e., the spherical harmonics, and then recombines them to form the desired beampattern. The major challenge with the use of a spherical microphone array lies in the expensive beamformer, especially for the array mounted on a sound-hard sphere. It is widely known that a sound-hard sphere acts as a frequency-dependent scatterer for the incident sound wave. Therefore, to faithfully reconstruct the sound, this process has to be reversed at some stage of the beamforming, which clearly means that to arrive back at a time domain signal, the filter applied on each recording channel has to be frequency-dependent instead of a single weight[?, ?]. In other words, there is no simple weight-and-sum spherical beamformer in time domain, unless there is some pre-processing.

Therefore, to achieve a simple weight-and-sum beamformer in the time domain we have to cancel this frequency-dependency for each recording channel (microphone) before beamforming, but the frequency-dependency is unknown unless this soundfield has already been decomposed in advance. Our proposed algorithm is based on this approach.

2. SPHERICAL BEAMFORMER
The basic principle of the spherical beamformer is to make use of the orthonormality of spherical harmonics to decompose the soundfield arriving at a spherical array. Then the orthogonal components of the soundfield are linearly combined to approximate a desired beampattern [?]. For a unit magnitude plane wave $k$, incident from direction $\Theta_k = (\theta_k, \varphi_k)$, the complex pressure field on the surface $(\Theta_s, r_s = a)$ of the rigid sphere is [?]:

$$ p_s = 4\pi \sum_{n=0}^{\infty} i^n b_n(ka) \sum_{m=-n}^{n} Y_n^m(\Theta_k) Y_n^{m*}(\Theta_s), \quad (1) $$

$$ b_n(ka) = j_n(ka) - \frac{j_n'(ka)}{h_n'(ka)} h_n(ka), \quad (2) $$

where $j_n$ is the spherical Bessel function of order $n$, $Y_n^m$ the spherical harmonics of order $n$ and degree $m$, * denotes the complex conjugation, and $h_n$ is the spherical Hankel function of the first kind.

The ideal beampattern looking at the direction $\Theta_0$ can be modeled as a delta function:

$$ F(\Theta_0, \Theta) = \delta(\Theta - \Theta_0). \quad (3) $$

This can be expanded into an infinite series of spherical harmonics [?]:

$$ F(\Theta_0, \Theta) = 2\pi \sum_{n=0}^{\infty} \sum_{m=-n}^{n} Y_n^{m*}(\Theta_0) Y_n^m(\Theta). \quad (4) $$

The weight at each point $\Theta_s$ to achieve this beampattern is:

$$ w_s = \frac{1}{2^n b_n(ka)} \sum_{m=-n}^{n} Y_n^{m*}(\Theta_0) Y_n^m(\Theta_s). \quad (5) $$

This is easy to prove using the orthonormality of spherical harmonics on unit spherical surface $\Omega_s$:

$$ \int_{\Omega_s} Y_n^{m*}(\Theta_s) Y_n^{m'}(\Theta_s) d\Omega_s = \delta_{nm'} \delta_{mm'}. \quad (6) $$

This is for an ideal continuous pressure sensitive surface. For discrete arrays with finite number of microphones, the practical beampattern is a truncated version of (??) to some limited order $N$:

$$ F_N(\Theta_0, \Theta) = 2\pi \sum_{n=0}^{N} \sum_{m=-n}^{n} Y_n^{m*}(\Theta_0) Y_n^m(\Theta). \quad (7) $$

This work was partially supported by NSF Award 0205271. Email: zli@cs.umd.edu; ramani@umiacs.umd.edu
Using the addition theorem of spherical harmonics, we get

\[ F_N(\Theta_0, \Theta) = \sum_{n=0}^{N} \frac{2n + 1}{2} P_n(\cos \gamma), \quad (8) \]

where \( \gamma \) is the angle between \( \Theta_0 \) and \( \Theta \), and \( P_n \) is the \( n \)th degree Legendre polynomial. Obviously, \( F_N(\Theta_0, \Theta) \) is a real number, which will be an important property in our algorithm in later sections.

3. FAST BEAMFORMING USING PRE-COMPUTED DATA

Suppose we perform \( N \)-order spherical beamforming toward \( L \) uniform 3D directions \( \{\Theta_i, i = 1, ..., L\} \). Then, for the \( i \)th beamformed result, the incident wave from \( \Theta_k \) has the gain of \( F_N(\Theta_k, \Theta) \).

Now we want a new beamformed result at the direction \( \Theta_j \), where \( \Theta_j \notin \{\Theta_i, i = 1, ..., L\} \). Apparently, the straightforward approach is to make a regular spherical beamforming at the direction \( \Theta_j \). However, since we already have \( L \) beamformed results, we can design an efficient way to achieve this, and in fact we can do it in time domain.

We noticed the following identity (see Appendix ?? for proof):

\[ \int_{\Omega_s} F_N(\Theta_k, \Theta) F_N(\Theta, \Theta_j) d\Omega_s = F_N(\Theta_k, \Theta_j). \quad (9) \]

Rewrite (9) into its discrete approximation:

\[ \sum_{i=1}^{L} F_N(\Theta_k, \Theta_i) F_N(\Theta_i, \Theta_j) = F_N(\Theta_k, \Theta_j) + \epsilon(\Theta_k, \Theta_j), \quad (10) \]

with constant coefficient omitted for simplicity. This clearly means if we apply the weight

\[ W_{ij} = F_N(\Theta_i, \Theta_j) \quad (11) \]

to the \( i \)th beamformed result, and sum all \( L \) results up, we will get the new beamformed result as if we steer the beamformer to \( \Theta_j \), plus a small spatial distortion \( \epsilon(\Theta_k, \Theta_j) \) due to quadrature errors in beampattern. Since \( F_N(\Theta_k, \Theta_j) \) is a real number and frequency-independent, we can apply \( W_{ij} \) in time domain. In addition, from (9), \( \epsilon(\Theta_k, \Theta_j) \) is also a real number, which means there is no additional phase distortion.

Therefore, the fast spherical beamforming algorithm is:

Step 1 Precompute the beamformed results in \( L \) uniform directions \( \{\Theta_i, i = 1, ..., L\} \), with the maximum beamforming order \( N \).

Step 2 To beamform in a new direction \( \Theta_j \), simply multiply \( W_{ij} \) to each precomputed channel in time domain and then sum them up.

In Step 1, the order of beamformer is usually frequency-dependent to make it robust to noise[?]. Following Appendix ??, it is easy to generalize (9) as:

\[ \int_{\Omega_s} F_N(\Theta_k, \Theta) F_N(\Theta, \Theta_j) d\Omega_s = F_N(\Theta_k, \Theta_j), N' \leq N. \quad (12) \]

Therefore, in Step 2, \( W_{ij} \) is still a single real value computed with the maximum beamforming order \( N \).

Since the order of beampattern depends on the number of microphones on spherical surface, that means to achieve maximum performance, we need \( L \) to be at least the same as the number of microphones. If we prefer to not consume more storage than the original recordings, we can just use the \( L \) directions corresponding to the microphone angular locations if those are uniform, or can choose any other \( L \) uniform directions to satisfy the quadrature requirement. If we want more accurate performance, i.e. to achieve smaller spatial distortion \( \epsilon \), we can make \( L \) a larger number. Please note that even when \( L = \infty \), we still have \( \epsilon \neq 0 \) which is caused by the finite number of microphones.
4. SIMULATION AND EXPERIMENT

We simulate using the 128-node layout as shown in Fig. ??.

With equal quadrature weight, this layout shows high accuracy in quadrature of spherical harmonics [2]. In our experiment, we use the hemispherical array as shown in Fig. ??.

It is equivalent to a 128-microphone array with mirrored sound sources. For more details about this system, please refer to [?].

4.1. Uniform Layout

In Fig. ??, we compare (?1) with the exact results for \( \Theta_k = (\pi/2, 0), \Theta_j \) sampling \( \{(\theta, \varphi) | \theta \in [0, \pi], \varphi \in [-\pi, \pi]\} \).

Fig. ?? shows the amplitude and phase errors. As discussed in the previous section, the phase error should be zero, except those caused by numerical precision, and this is observed.

In the experiment, we place a music sound source at (1.075, 2.7594), and a speech source at (1.1456, -0.24085), both about two meters away from the array. Fig. ?? shows the beamformed result using the fast algorithm with precomputed data, facing towards the speech source. Also in this figure, the error caused by spatial distortion is plotted in light color.

4.2. Non-uniform Layout

We use the hemispherical array to verify our optimal design algorithm in [?]. Instead of building another spherical microphone array with a non-uniform layout, we simply block 20 channels in our 64-node hemispherical array. The resulting 44-node (88-node if including images) layout is shown in Fig. ??(a).

As expected, the straightforward implementation of beamformer using equal quadrature weights has distorted beampatterns such as the one shown in Fig. ??(b). Using the optimal beamformer designed in [?], we can achieve optimal beampattern very close to (??) and still robust. Fig. ?? plots the 3D localization results using optimal beamformers with different robustness.

Figure 3: Steer the beamformers to 3D directions. Plot shows the gains using the regular (left) and fast (right) beamforming algorithms. \( n = 6 \).

Figure 4: The amplitude and phase errors.

Figure 5: The beamformed result using fast algorithm. Spatial noise is plotted in lighter color. \( SNR \approx 41\text{dB} \).

Figure 6: (a) The 88-node layout is generated by removing 40 symmetric nodes in the 128-node layout in Fig. ??(a). (b) The order 7 beampattern using equal quadrature weights at 2.5KHz.
as white noise gain (WNG). The sound source is placed at $(\pi/4, -\pi/2)$ and the optimal beamformer is steered into all 3D directions to localize it. The ideal expected result should be two regular beampatterns (??) combined together. According to this figure, we use minimum 10dB WNG to pre-compute the optimal beamforming result in 64 (equivalent to 128 with image) uniform directions. Fig. ?? shows the beamformed signal using the fast beamformer steering at the speech source. Although the optimal beamformer has maximum similarity with the regular beampattern (??), it still contains more spatial distortions than the one with uniform layout. Comparing the signals resulted from fast beamformer and from optimal beamformer, there is a slightly smaller $SNR$ than as seen in Fig. ??.

5. CONCLUSIONS

We proposed a fast beamforming algorithm. Using pre-computed data, the beamforming becomes a simple weight and sum in time domain. Simulation and experiment are used to verify the algorithm. This algorithm is useful in interactive audio applications, where the data is recorded and processed in advance. In addition, this algorithm is especially efficient when the soundfield is captured by spherical microphone arrays with non-uniform layout or microphone failure, in which cases, the frequency-dependent adaptive beamformer can be computed first, then the recordings from the non-uniformly positioned microphones are beamformed into uniform directions. After that, we can use the simple weight-and-sum algorithm to beamform to arbitrary directions easily.

6. REFERENCES


