移動を許容する BoSC システムの開発

Development of BoSC system that allows movement.

関西大学　環境都市工学部　建築学科
建築環境工学第Ⅰ研究室
建 15-120 山本　拓弥
指導教官　豊田　政弘

目次

1 序論

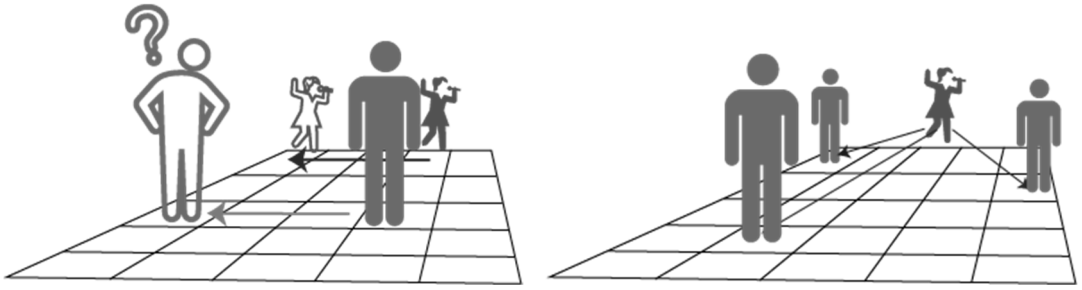
映像処理技術の向上やヴァーチャル・リアリティ（Virtual Reality: VR）を体験できるヘッドマウントディスプレイの普及に伴い、VR コンテンツの発展が著しく、空間的な情報を正しく再現できるだけではなく、より現実味に迫った高い臨場感のある VR 体験を実現し得る音響システムの必要性が高まっている。しかし、ヘッドホンやヘッドマウントディスプレイなどのウェアラブルデバイスによる肉体的制限は VR 体験における没入感を阻害する一因となっており、これらに頼らない様々な手法が提案されている。

音場再現において、ヘッドホンを用いずに臨場感を得る再現手法として、複数スピーカを用いてバイノーラル収録された信号を再現するトランスオーラルシステム [1]や、伊勢により提案された境界音場制御の原理に基づく三次元音場再現（Boundary Surface Control: BoSC）システム [2]に代表される境界音場制御方式などがある。いずれも受聴者がその場で立ち止まって、あるいは、座って音を受聴することを前提としており、受聴者の移動を考慮しておらず、受聴者が音場再現領域から出てしまうと音像定位性が損なわれてしまう。しかしながら、人の移動を許容する音場再現システムや、人の移動が音像定位性能に与える影響に関する研究は少ない。Thurlow W. R. [3]らによると首を振ることによって音像定位性能が向上することが報告されており、伊勢 [4]らは受聴者の頭部運動を許容する音場再現システムを提案しているが、人の移動を許容できるようになれば、さらなる音像定位性能の向上が期待される。

2 研究方法

2.1 開発概要

本研究では受聴者が受聴領域内で動き回っても音像位置が変わらない音場再現システムの開発とその音像定位性能評価を行う。従来の音場再現システムでは特定の音場再現領域内での受聴を前提としているため、領域を出ると音場が再現できず音像定位しない（図 1 - 左)。 そこで、今回は各辺 3 チャンネル、計 1 2 チャンネルのスピーカアレイに囲まれた 180 cm 四方の正方形の受聴領域を 30 cm 四方の小さな正方形のグリッドで分割し、受聴者の移動に応じて、各グリッドで 1 0 チャンネルのマイクロホンアレイによって収録され計算された BoSC 音源を連続的に再生することによって、擬似的に移動を許容する音場再現を実現する（図 1-右、図 2)。受聴者の位置は受聴領域の上部に取り付けた web カメラによって取得する。



従来の音場再現システム　　移動を許容する音場再現システム

図 1　移動を許容する BoSC システムの開発イメージ

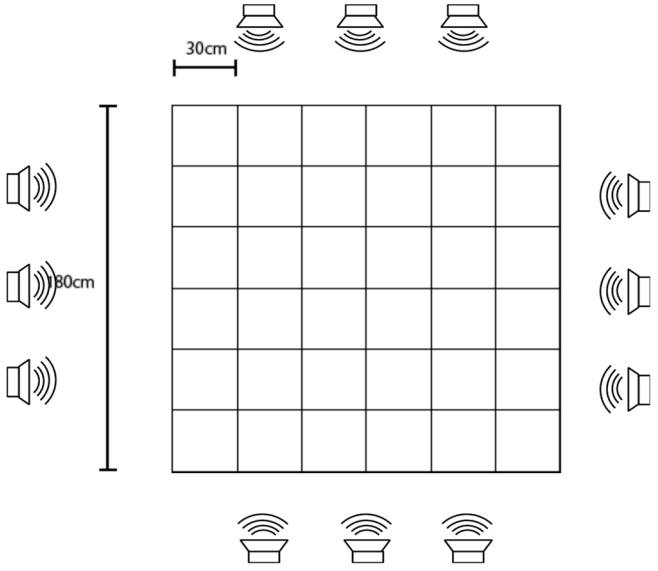


図 2　スピーカ配置と受聴領域グリッドのレイアウト

2.2 BoSC システム

境界音場制御の原理 [2]によると、原音場（Primary Field）に置ける制御領域 V と再現音場（Secondary Field）における制御領域 V'、原音場における領域 V を囲む境界面 S と再現音場における制御領域 V'を囲む境界面 S'がそれぞれ合同であるとき、境界面 S 上の離散化された制御点における音圧 $p(s)$および粒子速度と、境界面 S'上の制御点における音圧 $p(s)'$ および粒子速度を等しくすることができれば、領域 V 内の音場を領域 V'内に再現することができる（図３）。 BoSC システムでは原音場の境界面 S 上にマイクロホンを設置し、ここで得られたマイクロホン信号と同じ出力信号が再現音場の境界面 S'上で得られるように、再生空間の周囲に設置したスピーカの入力信号を制御する。

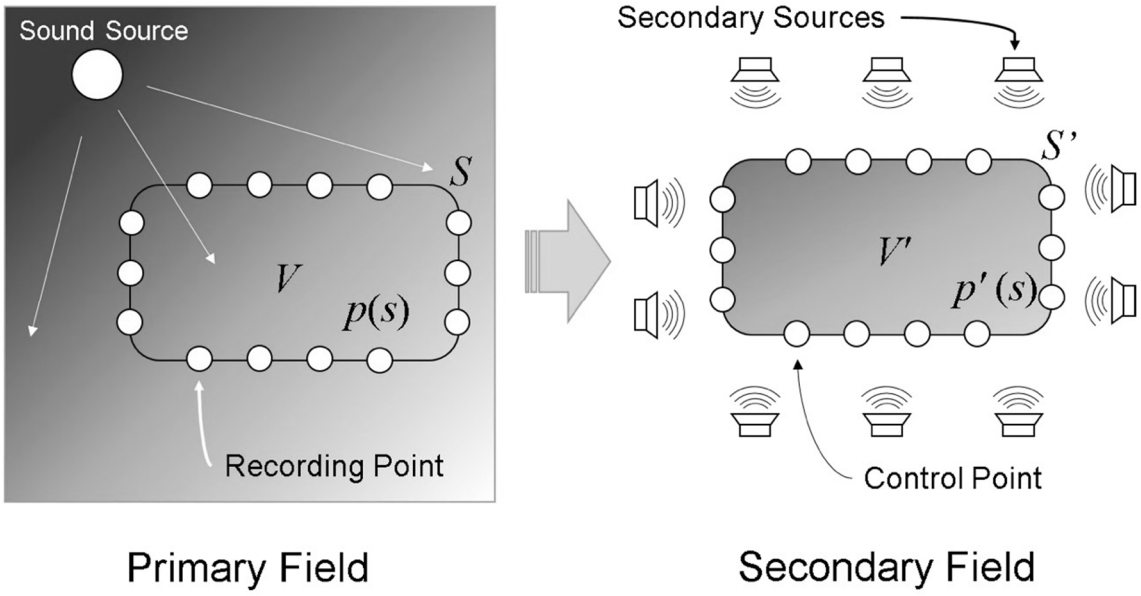


図３　BoSC システムの概要

2.2.1 逆フィルタの理論

原音場の観測点で音圧などの物理量を観測し、再現音場の制御点で同じ物理量が観測されるようなスピーカ（二次音源）から出力する信号を計算する必要がある。この問題を解決するフィルタを逆フィルタ［5]と呼び、これの設計手法を以下に示す。逆フィルタの設計の手法として時間領域で設計する手法［6]があるが、多数の音源を用いる場合、逆行列の計算に膨大な計算量が必要となるため、フーリエ変換によって時間領域から周波数領域に変換して逆フィルタの計算を行う［7]。

原音場での境界面 S での観測信号を $x(\omega)$、再現音場での境界面 S'での観測信号を $y(\omega)$、スピーカへの入力信号を $u(\omega)$、スピーカから制御点までの伝達関数を $G(\omega)$、逆フィルタを $H(\omega)$とする。伝達関数とは再現室内の伝達特性が数値化されたものである。音場再生では

$$y(\omega) = G(\omega)u(\omega) \qquad (1.1)$$

$$u(\omega) = H(\omega)x(\omega) \qquad (1.2)$$

という関係が成り立つ。すなわち、ここでは $y(\omega)=x(\omega)$を満たす $u(\omega)$を計算する必要があるため、マイクロホンとスピーカの数が等しい場合、逆フィルタ $H(\omega)$は

$$H(\omega) = G^{-1}(\omega) \qquad (1.3)$$

で与えられ、スピーカへの入力信号は

$$u(\omega) = G^{-1}(\omega)x(\omega) \qquad (1.4)$$

となる。BoSC システムではしばしばマイクロホンより多くのスピーカを用いる。その際、式(1.3)は劣決定系となるため、逆フィルタの解は無数に存在するが、その中で最小ノルム解（least-norm solution: LNS）を選択する。これは Moore-Penrose 型一般逆行列を用いることによって与えられ、逆フィルタは

$$H(\omega) = G^H(G^HG)^{-1} \qquad (1.5)$$

で与えることができる。この時 G^H は G の共役転置行列である。したがって、各スピーカに入力する信号のベクトルは

$$u(\omega) = G^H(G^HG)^{-1}x(\omega) \qquad (1.6)$$

で与えられ、フーリエ逆変換で時間領域に戻した $u(t)$を二次音源への入力信号とする。

2.2.2 正則化法

前述の逆フィルタの計算に用いる伝達関数は室内のインパルス応答を測定することによって得られるが、測定値に微弱な雑音が含まれる場合や温度や湿度などの環境変動により再現音場の室内伝達特性が変動する場合、逆フィルタ設計時に用いた伝達関数が十分に正確でない場合がある。その場合、これは完成した BoSC システムの再現性低下の一因となる。この不安定性の緩和手法として正則化法が提案されている [8]。正則化パラメータを $\beta(\omega)$、単位行列を I として逆フィルタ式(1.6)を

$$H(\omega) = G^H(G^H G + \beta(\omega)I)^{-1} \qquad (2.1)$$

とすることで、逆行列としては不正確ではあるが、不安定さを緩和することができる。

逆フィルタの計算は周波数領域で行うため、本来は周波数毎に正則化パラメータを決定する必要がある。しかしながら、正則化パラメータは 0 から無限大までの値を取り得るため実際に耳で聞きながら最適な値を設定することは不可能である。また、機械学習により正則化パラメータを設計する手法 [9]が提案されているが膨大な計算量を必要とするため、今回は全周波数で一義的に等しい正則化パラメータとし、耳で聞くことによりその最適値を探ることとする。

2.2.3 室内伝達特性の測定

伝達関数は室内のインパルス応答の測定によって得られる。インパルス応答には音源位置から受音点位置間の線形・時不変性をもつ伝搬系の物理特性情報が全て含まれており、残響時間などの物理特性や聴覚的印象などの心理特性が求められる。インパルス応答の測定には様々な手法が提案されており、継続時間の短いパルスを用いる直接法、ホワイトノイズ・ピンクノイズなどの広域帯ノイズを用いて相互相関関数を求める相互相関法、またインパルス信号を時間伸長した Swept-Sine（SS）信号 [10]を用いた Swept-Sine 法などがある。直接法は実際にパルスを使用するため室の応答が直接的に求められる利点があるが、ノイズが入りやすく、精度を高めるためには極めて多数回の同期加算が必要になる。相互相関法は連続的な音源信号を使用するため高い S/N 比を確保できるが、室の応答を耳で確かめながら測定を行うことには不向きである。Swept-Sine 法はこの中間的な性質を持ち、SS 信号は連続的な音源信号であるため暗騒音の状況を判断しながら測定を行えるほか、室の応答を実際に耳で聞くことができる。本研究では Swept-Sine 法でインパルス応答を測定する。

SS 信号は周波数範囲の下限の周波数をf_1、上限の周波数をf_2とし、サンプリング周波数をf_s、データ点数を N として、時間インデクスが一つ進むごとに角周波数を位相に加算することによって、f_1からf_2まで周波数が時間変化するサイン波として作成できる。また SS 信号を作成する Python プログラムを付録 1 に示す。得られた SS 信号を図 4 に示す。

次に、畳み込みの原理を用いたインパルス応答測定法の手順について説明する。使用するスピーカから上述した SS 信号を再生し、受音点で観測された信号を収録する。精度を上げるため連続で数回 SS 信号を再生・収録し同期加算を行う。以降、これを応答信号と呼ぶ。3 回分の再生・収録に対して同期加算を行った信号を図 5 に示す。続いて、SS 信号を時間軸上で逆にした信号（逆特性信号）（図 6）を用意し、応答信号と逆特性信号をいずれも離散フーリエ変換により時間領域から周波数領域に変換し、周波数軸上で畳み込む。これをフーリエ逆変換し周波数領域から時間領域に戻し、実部を取り出したものがインパルス応答となる（図 7）。12 個のスピーカから 10 個のマイクロホンへのインパルス応答を測定する場合、付録 2 に添付した Python プログラムであれば２３０秒で測定が完了する。

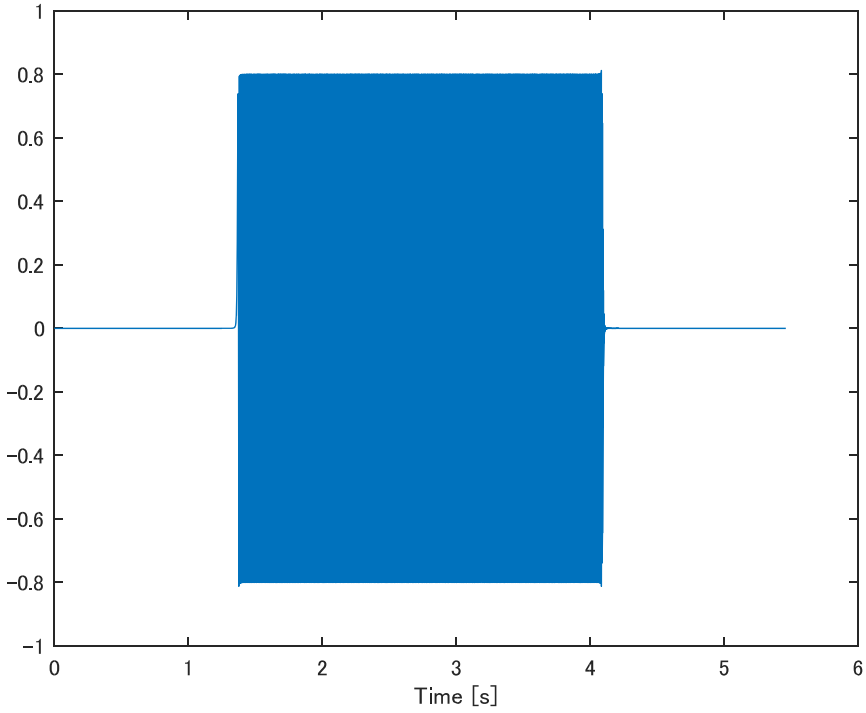


図 4　作成した SS 信号

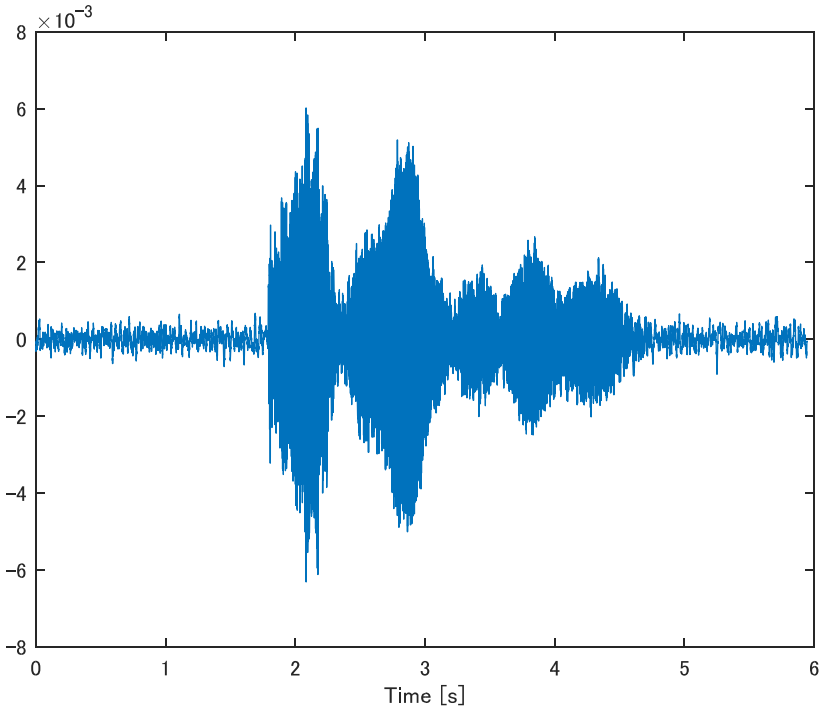


図 5　同期加算した応答信号

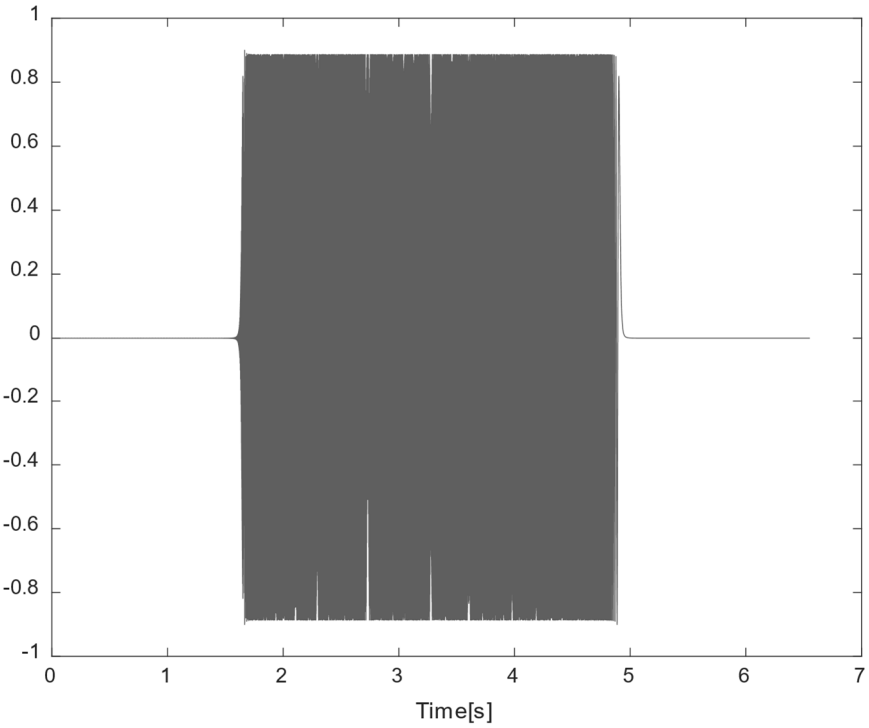


図 6　逆特性信号

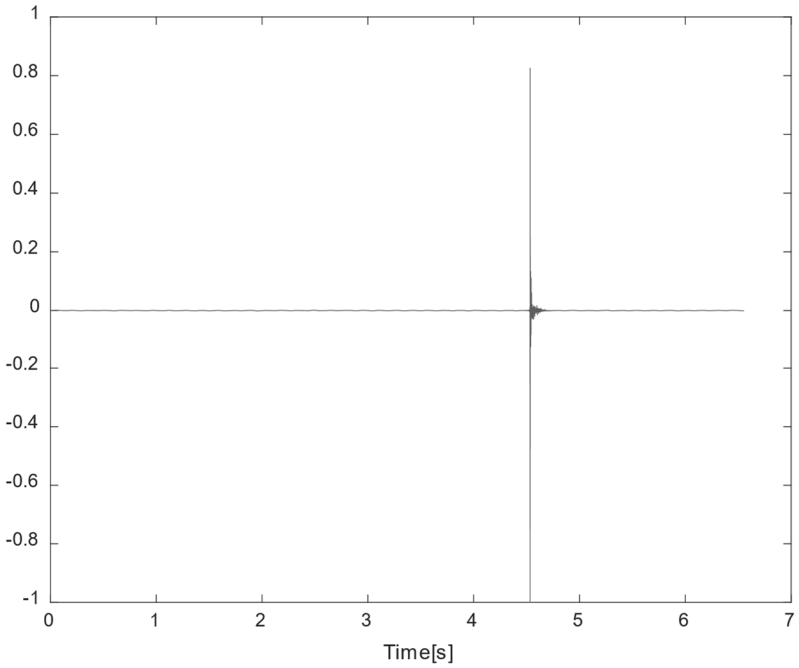


図 7　インパルス応答

2.2.4 本研究における BoSC システムの実装

本研究では収録に１０チャンネルのマイクロホンアレイ、そして二次音源に１２チャンネルのスピーカを用いて BoSC システムを実装した。実装に用いた機材を表１に、実装環境を表２に示す。

表１　BoSC システム実装のための機材一覧

用途	名称	メーカー	備考
マイクロホン	Miniature Omnidirection Microphone, Hi-Sens, Black [4060-BM]	DPA Microphones	
マイクロホンアダプタ	Adapter: MicroDot to 3-pin XLR(P48) [DAD6001]	DPA Microphones	マイクロホンとオーディオインターフェースに接続
スピーカ	TIME DOMAIN mini	Time domain Corporation	定格周波数特性 80Hz~18kHz
USB オーディオインターフェース	16A	MOTU	
マイクラインアンプリファ	MLA 8	YAMAHA	ヘッドアンプ
ステレオミニジャック-ステレオ RCA ピンプラグ変換アダプタ	DH-MFWR03	ELECOM	スピーカとオーディオインターフェースを接続
ステレオミニプラグ延長ケーブル	DH-MPJN30	ELECOM	
マイクロホンアレイ土台	プラスチックボウル 30cm	ダイソー	

表２　BoSC システム実装環境

場所	関西大学　建築環境第 1 研究室　無響室
プログラム言語	収録・再生　Python3.6(Spider) 計算　MATLAB R2018b – academic use
OS	macOS Mojave Ver 10.14.2
コンピュータ	MacBook Pro (13-inch, 2018), 2.7 GHz Intel Core i7, RAM 8GB

マイクロホンとプラスチックボウルで１０チャンネルマイクロホンアレイを作成した（画像 1）。また、図１のレイアウトで受聴者の耳の高さ（170 cm）でスピーカを配置した（画像 2）。

画像１　作成した１０チャンネルマイクロホンアレイ

画像２　受聴領域を囲む１２チャンネルスピーカ

続いて、収録音源を BoSC 音源に変換するためのプロセスについて説明する。スピーカ i からマイクロホン j までの伝達関数を h_{ij} とする（図 8）。伝達関数は 2.2.3 で説明した方法で測定する。ここまでの処理を行う Python プログラムを付録 2 に示す。次に 2.2.1 及び 2.2.2 で説明した逆フィルタの理論と正則化法から BoSC 音源を設計する。式（2.1）での伝達関数 $G(\omega)$を

$$G = \begin{pmatrix} h_{11} & \cdots & h_{121} \\ \vdots & \ddots & \vdots \\ h_{112} & \cdots & h_{1212} \end{pmatrix} \qquad (4.1)$$

と置き、各マイクロホンで収録された音源を S_j として式(1.6)での $x(\omega)$を

$$x = \begin{pmatrix} S_1 \\ \vdots \\ S_{10} \end{pmatrix} \qquad (4.2)$$

と置いて、計算を行う。これを実行する MATLAB プログラムを付録 3 に示す。

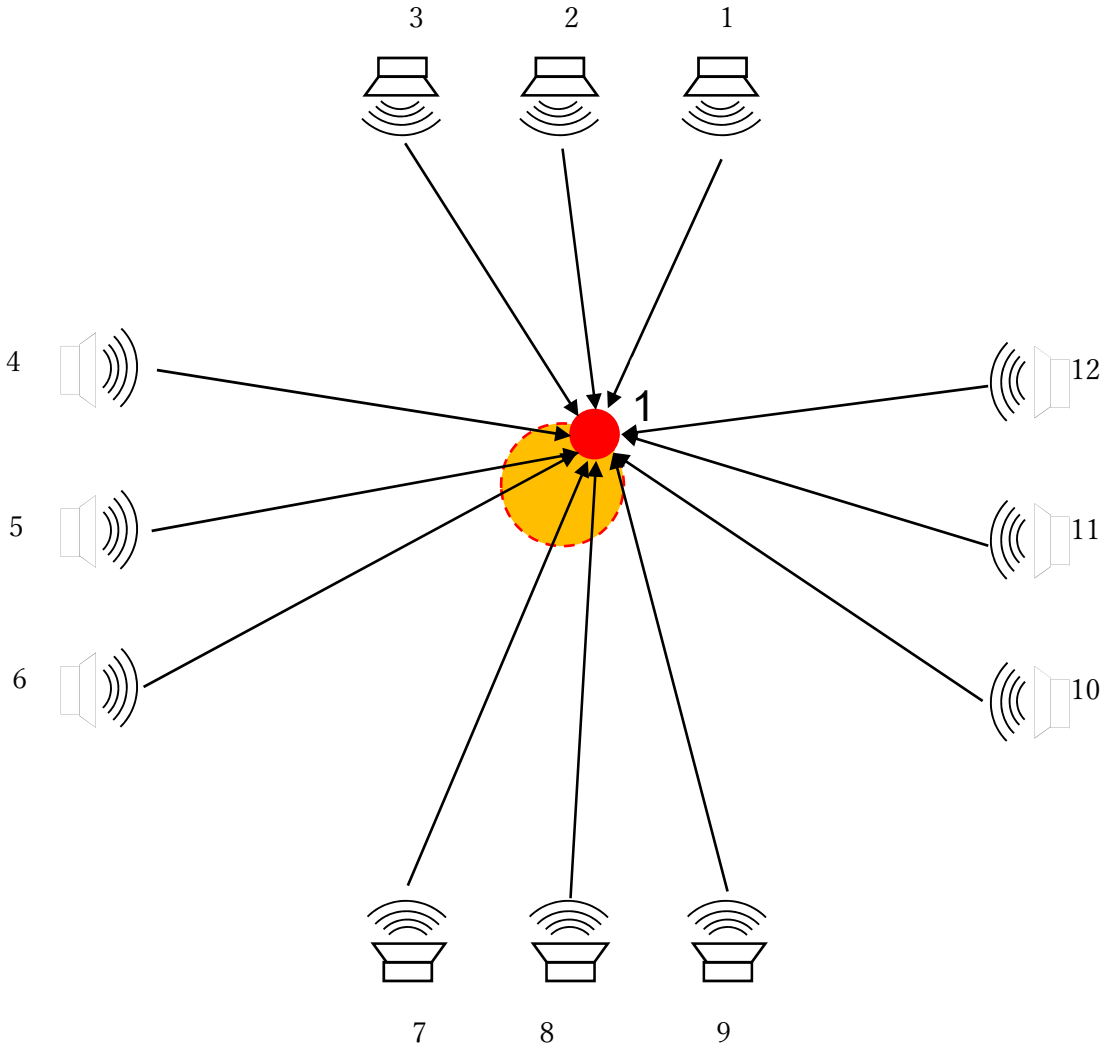


図 8　スピーカからマイクロホンまでの伝達関数

2.3 映像処理系

本研究において受聴者の位置の特定には web カメラを使用する。実装に用いた機材を表 3 に、実装環境を表 4 に示す。

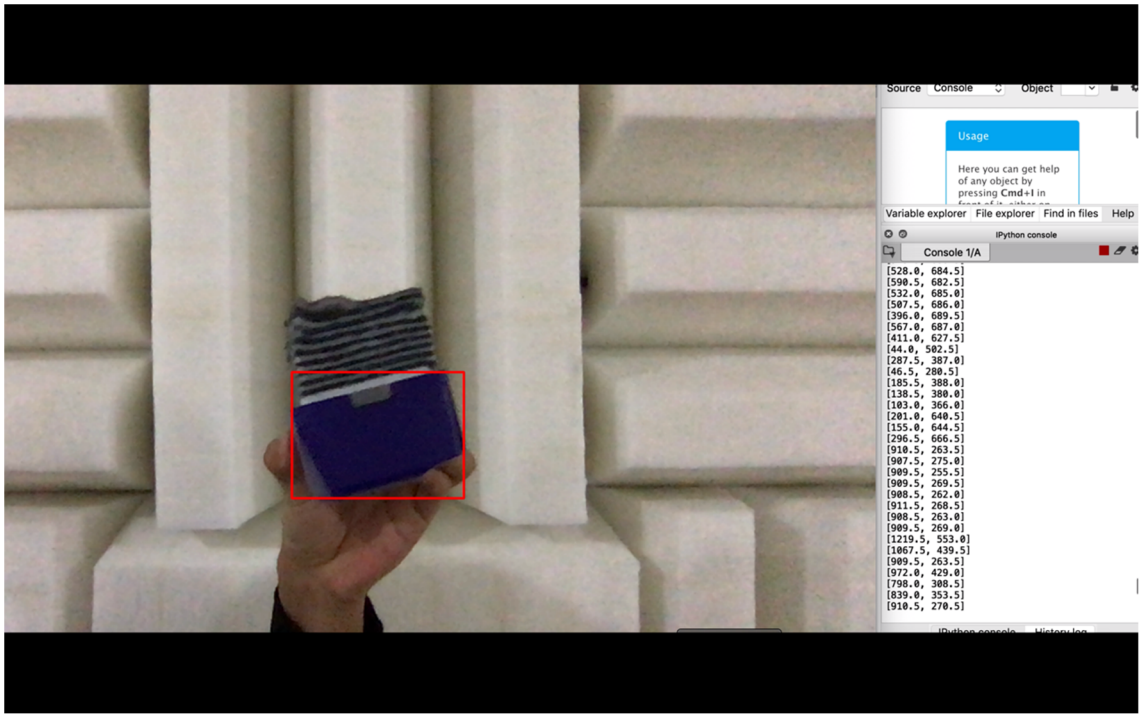
表 3　映像処理系のための使用機材

用途	名称	メーカー
カメラ	KM00-0467-01	BUFFALO

表 4　映像処理系実装環境

プログラム言語	Python3.6(Spider/OpenCV)
OS	macOS Mojave Ver 10.14.2
コンピュータ	MacBook Pro (13-inchi,2018), 2.7 GHz Intel Corei7, RAM 8GB

本研究では色によって受聴者の位置座標を推定するため、受聴領域上部から web カメラで取得した映像に対し、そのフレーム画像を HSV 変換し、指定した HSV 範囲領域で二値化してマスク画像を作成する。その後、最大の検出領域の中心座標を取得することで受聴者の位置座標を特定する（画像 3)。また、これを実行する Python プログラムを付録 4 に示す。



画像 3　OpenCV と web カメラによる受聴者の位置情報の取得

3 評価方法・条件

本研究では実際に移動しながら音を聞くことに先立ち、マイクロホンアレイで囲まれた受聴領域中心だけでなく、中心から外れた位置において BoSC 音源を生成した時の音像定位性能を評価する。前処理として図 9 で示すように受聴領域を 3 6 のグリッドに分け、グリッドの格子点の 9 箇所に音源を設置した場合それぞれについて、指定した 3 箇所のグリッドでの BoSC 音源を生成する。被験者には 3 箇所のグリッドで各 9 種類の音源をランダムに聴いてもらい、どこから音が聞こえてきたかを回答させる。また、BoSC 音源に変換する前の音源を聴かせ、音の聞こえの自然さ、明瞭さを 5 段階で答えさせ、音質性能を評価する。

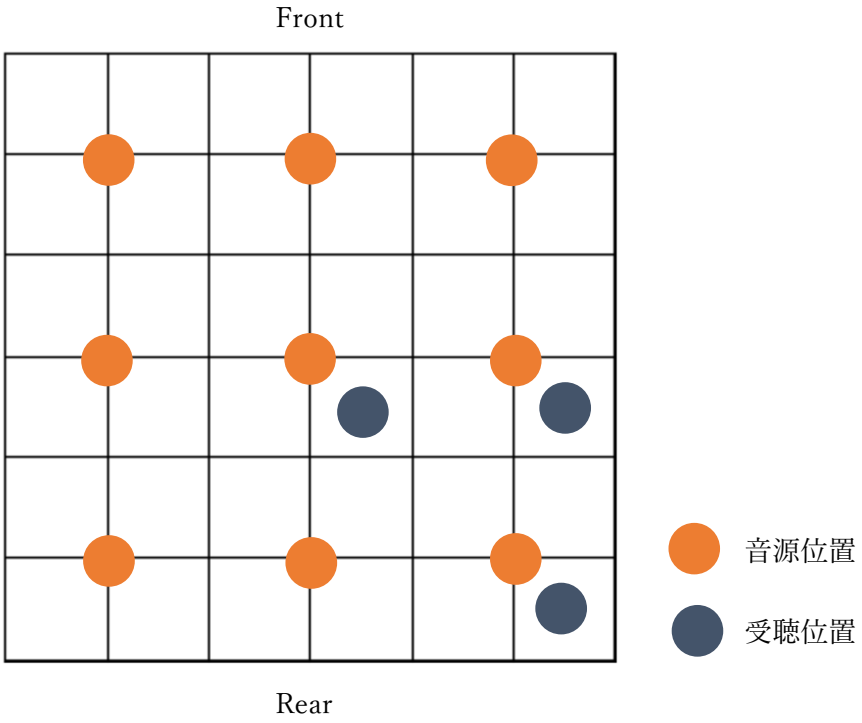


図 9　静止時の被験者の受聴位置

4 結果

作成された BoSC 音源は高域に継続的なノイズが入り、それに伴う音質の低下から音像定位位置の把握は困難であった（図 10－13)。

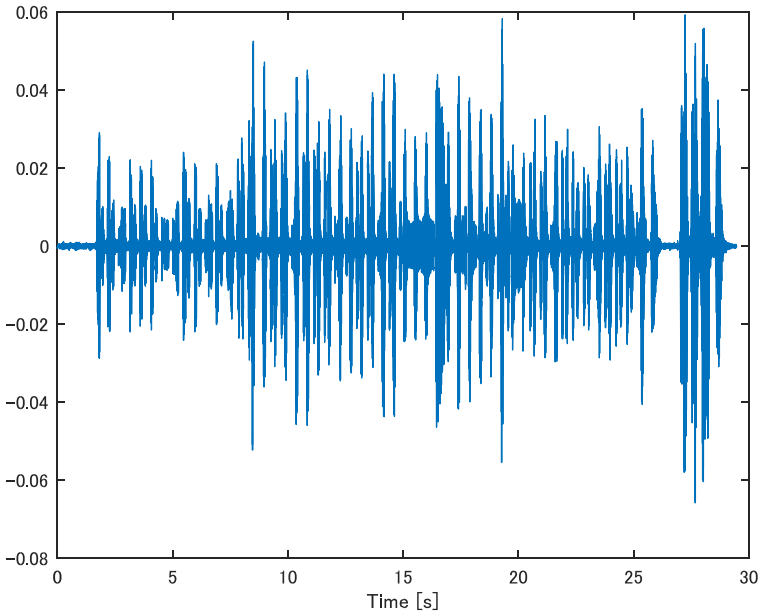


図 10　音源の波形

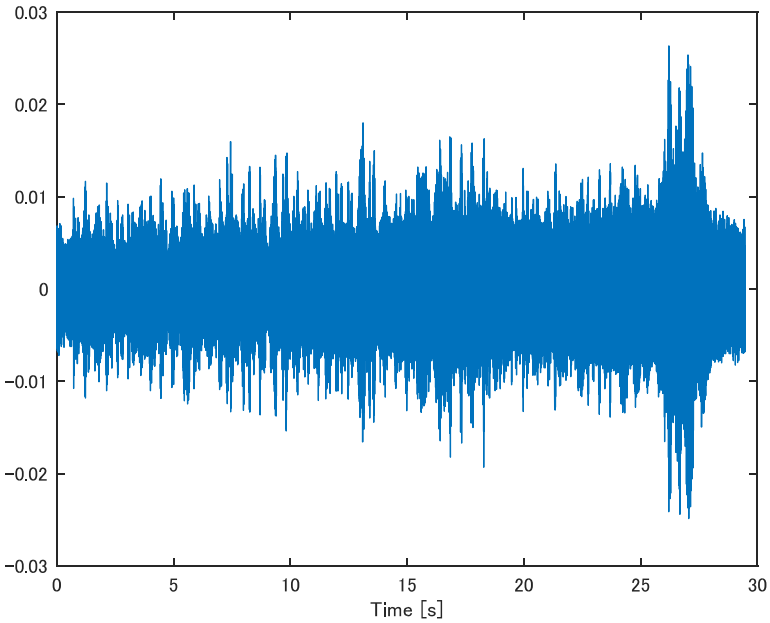


図 11　BoSC 音源の波形

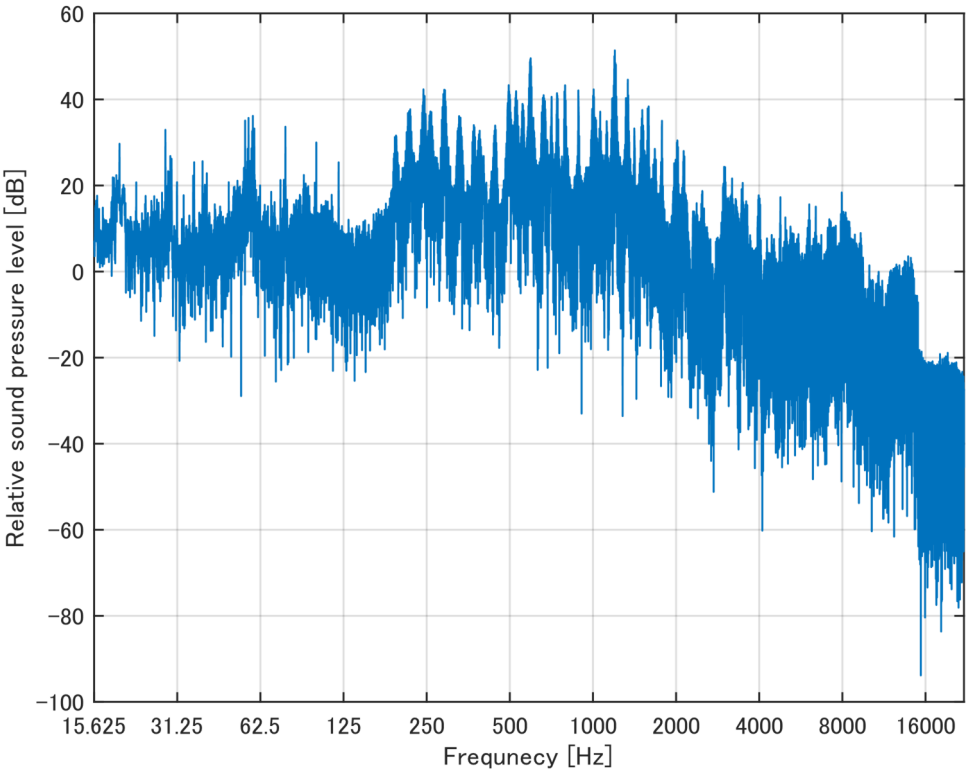


図 12　音源の音圧スペクトル

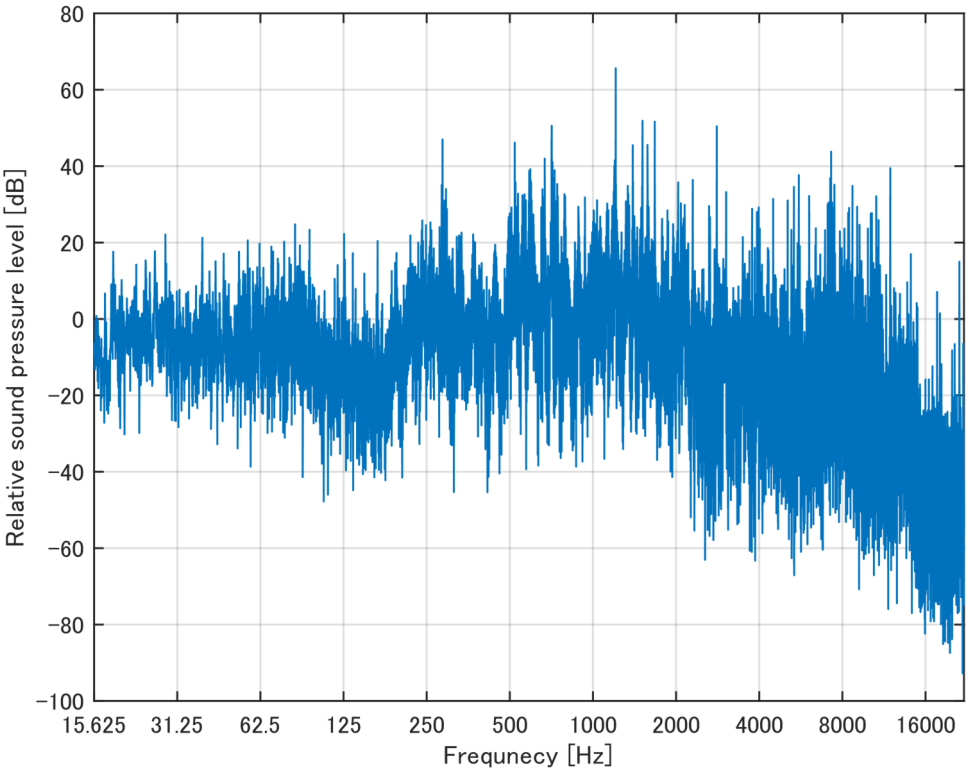


図 13　BoSC 音源の音圧スペクトル

5 考察

ノイズが乗る原因について考えられることとして、インパルス応答測定に用いた SS 信号及び収録モジュールの不具合が考えられる。SS 信号の上限周波数と下限周波数、あるいはその継続時間なども用いるデバイスに応じて最適化する必要があろう。また、本来 BoSC は三次元音場で BoSC 音源を作成することが多いが、本研究では二次元音場で BoSC 音源を作成したため、これが再現性低下の原因がある可能性も考えられる。

6 結論

二次元音場における BoSC システムによる音場再現は高域の周波数帯においてノイズが入り、これは正則化によっても除去できなかった。しかし、使用する SS 信号によって完成する BoSC 音源の音質やノイズの大きさに変化が生じることが確認されたため、SS 信号の上限周波数や下限周波数、またその継続時間の最適化によって音質の改善が見込まれるものと考えられる。

付録

（付録１）Swept-Sine 信号を作成する Python プログラム
makeSweptSine.py [11] [12]

```
# -*- coding: utf-8 -*-
"""
Created on Mon Sep 24 15:43:56 2018

@author: csl1
"""

import wave
import numpy as np
from matplotlib import pylab as plt
import struct
from scipy import arange, cumsum, sin, linspace
from scipy import pi as mpi

#パラメータの入力
A = 1        #振幅
fs = 44100 #サンプリング周波数
start_freq = 20   #下限の周波数
end_freq = 20000 #上限の周波数
sec = 5     #秒

def make_sweepsound(A, fs, start_freq, end_freq, sec):

    freqs = linspace(start_freq, end_freq, num = int(round(fs * sec)))
    #  角周波数の変化量
    phazes_diff = 2. * mpi * freqs / fs
    #  位相
    phazes = cumsum(phazes_diff)
    #  サイン波合成
    ret = A * sin(phazes)
```

```
    return ret


def sweepsound():
    sweepsound = make_sweepsound(A, fs, start_freq, end_freq, sec)
    sweepsound = [int(x * 32672.0) for x in sweepsound]

    binwave = struct.pack("h" * len(sweepsound), *sweepsound)
    w =
wave.Wave_write("sweepsound_"+str(start_freq)+"hz_to_"+str(end_freq)+"hz_"+str(sec
)+"sec"+".wav")
    p = (1, 2, fs, len(binwave), 'NONE', 'not compressed')
    w.setparams(p)
    w.writeframes(binwave)
    w.close()

    plt.plot(sweepsound[0:200])
    plt.show()



if __name__ == "__main__":
    sweepsound()
```

（付録２）Multi Input/Multi Output のためのインパルス応答測定 Python プログラム
[13]

```
#!/usr/bin/env python3
# -*- coding: utf-8 -*-
"""
Created on Thu Dec 20 12:48:16 2018

@author: csl
"""

import numpy as np
import scipy as sp
#from scipy.io.wavfile import write
#import matplotlib.pyplot as plt
import sounddevice as sd
import wave
import struct

#Setting
Fs=44100#サンプリング周波数の設定
small=1000000#small 分の一の音量に調節　スピーカからの出力音量を制御する
#sounddevise Setting
sd.default.samplerate=Fs
print(sd.query_devices())
deviceNo = input("Please enter the ID of the device to use　:" )
sd.default.device = int(deviceNo)


#mapping 使用するスピーカの出力設定を行う
out1 = sd.CoreAudioSettings(channel_map=[0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1])
out2 = sd.CoreAudioSettings(channel_map=[-1,0,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1])
out3 = sd.CoreAudioSettings(channel_map=[-1,-1,0,-1,-1,-1,-1,-1,-1,-1,-1,-1])
out4 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,0,-1,-1,-1,-1,-1,-1,-1,-1])
out5 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,-1,0,-1,-1,-1,-1,-1,-1,-1])
out6 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,-1,-1,0,-1,-1,-1,-1,-1,-1])
out7 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,-1,-1,-1,0,-1,-1,-1,-1,-1])
```

```
out8 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,-1,-1,-1,-1,0,-1,-1,-1,-1])
out9 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,-1,-1,-1,-1,-1,0,-1,-1,-1])
out10 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,-1,-1,-1,-1,-1,-1,0,-1,-1])
out11 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,0,-1])
out12 = sd.CoreAudioSettings(channel_map=[-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,-1,0])

ca_in = sd.CoreAudioSettings(channel_map=[0,1,2,3,4,5,6,7,8,9])

def wavread(filename):#Reading 16bit WAVE files
    wf = wave.open(filename,'rb')
    buf = wf.readframes(wf.getnframes())
    data = np.frombuffer(buf, dtype = "int16")
    return data

TSP=wavread("tsp.wav")/small #使用する信号の名前を入力する



#Convolution
def IR(rTSPdata,outputfilename):
    rTSPdata=rTSPdata.T #逆特性を与える SS 信号の定義
    ipls=np.real(sp.ifft(sp.fft(rTSPdata)*sp.fft(np.flipud(TSP),rTSPdata.size)))
    #高速フーリエ変換で音源を時間領域から周波数領域に変換して畳み込みを行う
    c=np.fft.fftshift(ipls/max(ipls)) #逆フーリエ変換で時間領域に戻す
    int16amp=32768 / int(c.max())
    y2 = np.array([c * int16amp],dtype = "int16")[0]
    y4 = struct.pack("h" * len(y2), *y2)
    w = wave.Wave_write(outputfilename)
    w.setparams((
            1,                          # channel
            2,                          # byte width
            44100,                       # sampling rate
            len(y4),                    # number of frames
            "NONE", "NONE"    # no compression
        ))
    w.writeframesraw(y4)
    w.close()
```

```
#Playrec　3 回収録を行い和をとることで精度を上げる
def PlayRec(outmap,IDnumber):
    #sd.play(TSP,Fs,extra_settings=outmap,blocking=True)
    TSP1=sd.playrec(TSP,Fs,channels=10,extra_settings=(ca_in,outmap),blocking=True)
    TSP2=sd.playrec(TSP,Fs,channels=10,extra_settings=(ca_in,outmap),blocking=True)
    TSP3=sd.playrec(TSP,Fs,channels=10,extra_settings=(ca_in,outmap),blocking=True)

    rTSP = TSP1 + TSP2 + TSP3 # 3 回収録を行い和をとることで精度を上げる

    rTSP1 = rTSP[:,0]
    rTSP2 = rTSP[:,1]
    rTSP3 = rTSP[:,2]
    rTSP4 = rTSP[:,3]
    rTSP5 = rTSP[:,4]
    rTSP6 = rTSP[:,5]
    rTSP7 = rTSP[:,6]
    rTSP8 = rTSP[:,7]
    rTSP9 = rTSP[:,8]
    rTSP10 = rTSP[:,9]

    #完成したインパルス応答は IRij.wav として出力される
    IR(rTSP1,"IR"+str(IDnumber)+"1.wav")
    IR(rTSP2,"IR"+str(IDnumber)+"2.wav")
    IR(rTSP3,"IR"+str(IDnumber)+"3.wav")
    IR(rTSP4,"IR"+str(IDnumber)+"4.wav")
    IR(rTSP5,"IR"+str(IDnumber)+"5.wav")
    IR(rTSP6,"IR"+str(IDnumber)+"6.wav")
    IR(rTSP7,"IR"+str(IDnumber)+"7.wav")
    IR(rTSP8,"IR"+str(IDnumber)+"8.wav")
    IR(rTSP9,"IR"+str(IDnumber)+"9.wav")
    IR(rTSP10,"IR"+str(IDnumber)+"10.wav")

print("測定開始")
PlayRec(out1,1)
PlayRec(out2,2)
```

```
PlayRec(out3,3)
PlayRec(out4,4)
PlayRec(out5,5)
PlayRec(out6,6)
PlayRec(out7,7)
PlayRec(out8,8)
PlayRec(out9,9)
PlayRec(out10,10)
PlayRec(out11,11)
PlayRec(out12,12)
print("測定終了")
```

（付録３）逆フィルタ計算を行い BoSC 音源を生成する MATLAB プログラム

```
name = 'sd_BoSCvoice_0.01A'; %作成したBoSC音源を入れるフォルダ名
mkdir test;
foldername = 'ss_voice'; %使う音源の入ったファイル名
reg = 0.01; %正則化パラメータの設定

movefile(foldername,'rec');
ss1  = audioread('rec/rec1.wav');
ss2  = audioread('rec/rec2.wav');
ss3  = audioread('rec/rec3.wav');
ss4  = audioread('rec/rec4.wav');
ss5  = audioread('rec/rec5.wav');
ss6  = audioread('rec/rec6.wav');
ss7  = audioread('rec/rec7.wav');
ss8  = audioread('rec/rec8.wav');
ss9  = audioread('rec/rec9.wav');
ss10 = audioread('rec/rec10.wav');

s1   = fft(ss1);
s2   = fft(ss2);
s3   = fft(ss3);
s4   = fft(ss4);
s5   = fft(ss5);
s6   = fft(ss6);
s7   = fft(ss7);
s8   = fft(ss8);
s9   = fft(ss9);
s10  = fft(ss10);
 %インパルス応答を読み込む
ir11      = fft(audioread('IR/IR11.wav'),length(s1));
ir12      = fft(audioread('IR/IR12.wav'),length(s1));
ir13      = fft(audioread('IR/IR13.wav'),length(s1));
ir14      = fft(audioread('IR/IR14.wav'),length(s1));
ir15      = fft(audioread('IR/IR15.wav'),length(s1));
ir16      = fft(audioread('IR/IR16.wav'),length(s1));
```

```
ir17      = fft(audioread('IR/IR17.wav'),length(s1));
ir18      = fft(audioread('IR/IR18.wav'),length(s1));
ir19      = fft(audioread('IR/IR19.wav'),length(s1));
ir110     = fft(audioread('IR/IR110.wav'),length(s1));

ir21      = fft(audioread('IR/IR21.wav'),length(s1));
ir22      = fft(audioread('IR/IR22.wav'),length(s1));
ir23      = fft(audioread('IR/IR23.wav'),length(s1));
ir24      = fft(audioread('IR/IR24.wav'),length(s1));
ir25      = fft(audioread('IR/IR25.wav'),length(s1));
ir26      = fft(audioread('IR/IR26.wav'),length(s1));
ir27      = fft(audioread('IR/IR27.wav'),length(s1));
ir28      = fft(audioread('IR/IR28.wav'),length(s1));
ir29      = fft(audioread('IR/IR29.wav'),length(s1));
ir210     = fft(audioread('IR/IR210.wav'),length(s1));

ir31      = fft(audioread('IR/IR31.wav'),length(s1));
ir32      = fft(audioread('IR/IR32.wav'),length(s1));
ir33      = fft(audioread('IR/IR33.wav'),length(s1));
ir34      = fft(audioread('IR/IR34.wav'),length(s1));
ir35      = fft(audioread('IR/IR35.wav'),length(s1));
ir36      = fft(audioread('IR/IR36.wav'),length(s1));
ir37      = fft(audioread('IR/IR37.wav'),length(s1));
ir38      = fft(audioread('IR/IR38.wav'),length(s1));
ir39      = fft(audioread('IR/IR39.wav'),length(s1));
ir310     = fft(audioread('IR/IR310.wav'),length(s1));

ir41      = fft(audioread('IR/IR41.wav'),length(s1));
ir42      = fft(audioread('IR/IR42.wav'),length(s1));
ir43      = fft(audioread('IR/IR43.wav'),length(s1));
ir44      = fft(audioread('IR/IR44.wav'),length(s1));
ir45      = fft(audioread('IR/IR45.wav'),length(s1));
ir46      = fft(audioread('IR/IR46.wav'),length(s1));
ir47      = fft(audioread('IR/IR47.wav'),length(s1));
ir48      = fft(audioread('IR/IR48.wav'),length(s1));
ir49      = fft(audioread('IR/IR49.wav'),length(s1));
```

```
ir410      = fft(audioread('IR/IR410.wav'),length(s1));

ir51       = fft(audioread('IR/IR51.wav'),length(s1));
ir52       = fft(audioread('IR/IR52.wav'),length(s1));
ir53       = fft(audioread('IR/IR53.wav'),length(s1));
ir54       = fft(audioread('IR/IR54.wav'),length(s1));
ir55       = fft(audioread('IR/IR55.wav'),length(s1));
ir56       = fft(audioread('IR/IR56.wav'),length(s1));
ir57       = fft(audioread('IR/IR57.wav'),length(s1));
ir58       = fft(audioread('IR/IR58.wav'),length(s1));
ir59       = fft(audioread('IR/IR59.wav'),length(s1));
ir510      = fft(audioread('IR/IR510.wav'),length(s1));

ir61       = fft(audioread('IR/IR61.wav'),length(s1));
ir62       = fft(audioread('IR/IR62.wav'),length(s1));
ir63       = fft(audioread('IR/IR63.wav'),length(s1));
ir64       = fft(audioread('IR/IR64.wav'),length(s1));
ir65       = fft(audioread('IR/IR65.wav'),length(s1));
ir66       = fft(audioread('IR/IR66.wav'),length(s1));
ir67       = fft(audioread('IR/IR67.wav'),length(s1));
ir68       = fft(audioread('IR/IR68.wav'),length(s1));
ir69       = fft(audioread('IR/IR69.wav'),length(s1));
ir610      = fft(audioread('IR/IR610.wav'),length(s1));

ir71       = fft(audioread('IR/IR71.wav'),length(s1));
ir72       = fft(audioread('IR/IR72.wav'),length(s1));
ir73       = fft(audioread('IR/IR73.wav'),length(s1));
ir74       = fft(audioread('IR/IR74.wav'),length(s1));
ir75       = fft(audioread('IR/IR75.wav'),length(s1));
ir76       = fft(audioread('IR/IR76.wav'),length(s1));
ir77       = fft(audioread('IR/IR77.wav'),length(s1));
ir78       = fft(audioread('IR/IR78.wav'),length(s1));
ir79       = fft(audioread('IR/IR79.wav'),length(s1));
ir710      = fft(audioread('IR/IR710.wav'),length(s1));

ir81       = fft(audioread('IR/IR81.wav'),length(s1));
```

```
ir82      = fft(audioread('IR/IR82.wav'),length(s1));
ir83      = fft(audioread('IR/IR83.wav'),length(s1));
ir84      = fft(audioread('IR/IR84.wav'),length(s1));
ir85      = fft(audioread('IR/IR85.wav'),length(s1));
ir86      = fft(audioread('IR/IR86.wav'),length(s1));
ir87      = fft(audioread('IR/IR87.wav'),length(s1));
ir88      = fft(audioread('IR/IR88.wav'),length(s1));
ir89      = fft(audioread('IR/IR89.wav'),length(s1));
ir810     = fft(audioread('IR/IR810.wav'),length(s1));

ir91      = fft(audioread('IR/IR91.wav'),length(s1));
ir92      = fft(audioread('IR/IR92.wav'),length(s1));
ir93      = fft(audioread('IR/IR93.wav'),length(s1));
ir94      = fft(audioread('IR/IR94.wav'),length(s1));
ir95      = fft(audioread('IR/IR95.wav'),length(s1));
ir96      = fft(audioread('IR/IR96.wav'),length(s1));
ir97      = fft(audioread('IR/IR97.wav'),length(s1));
ir98      = fft(audioread('IR/IR98.wav'),length(s1));
ir99      = fft(audioread('IR/IR99.wav'),length(s1));
ir910     = fft(audioread('IR/IR910.wav'),length(s1));

ir101     = fft(audioread('IR/IR101.wav'),length(s1));
ir102     = fft(audioread('IR/IR102.wav'),length(s1));
ir103     = fft(audioread('IR/IR103.wav'),length(s1));
ir104     = fft(audioread('IR/IR104.wav'),length(s1));
ir105     = fft(audioread('IR/IR105.wav'),length(s1));
ir106     = fft(audioread('IR/IR106.wav'),length(s1));
ir107     = fft(audioread('IR/IR107.wav'),length(s1));
ir108     = fft(audioread('IR/IR108.wav'),length(s1));
ir109     = fft(audioread('IR/IR109.wav'),length(s1));
ir1010    = fft(audioread('IR/IR1010.wav'),length(s1));

ir111     = fft(audioread('IR/IR111.wav'),length(s1));
ir112     = fft(audioread('IR/IR112.wav'),length(s1));
ir113     = fft(audioread('IR/IR113.wav'),length(s1));
ir114     = fft(audioread('IR/IR114.wav'),length(s1));
```

```
ir115      = fft(audioread('IR/IR115.wav'),length(s1));
ir116      = fft(audioread('IR/IR116.wav'),length(s1));
ir117      = fft(audioread('IR/IR117.wav'),length(s1));
ir118      = fft(audioread('IR/IR118.wav'),length(s1));
ir119      = fft(audioread('IR/IR119.wav'),length(s1));
ir1110     = fft(audioread('IR/IR1110.wav'),length(s1));

ir121      = fft(audioread('IR/IR121.wav'),length(s1));
ir122      = fft(audioread('IR/IR122.wav'),length(s1));
ir123      = fft(audioread('IR/IR123.wav'),length(s1));
ir124      = fft(audioread('IR/IR124.wav'),length(s1));
ir125      = fft(audioread('IR/IR125.wav'),length(s1));
ir126      = fft(audioread('IR/IR126.wav'),length(s1));
ir127      = fft(audioread('IR/IR127.wav'),length(s1));
ir128      = fft(audioread('IR/IR128.wav'),length(s1));
ir129      = fft(audioread('IR/IR129.wav'),length(s1));
ir1210     = fft(audioread('IR/IR1210.wav'),length(s1));

N1     = zeros(length(s1),1);
N2     = zeros(length(s1),1);
N3     = zeros(length(s1),1);
N4     = zeros(length(s1),1);
N5     = zeros(length(s1),1);
N6     = zeros(length(s1),1);
N7     = zeros(length(s1),1);
N8     = zeros(length(s1),1);
N9     = zeros(length(s1),1);
N10    = zeros(length(s1),1);
N11    = zeros(length(s1),1);
N12    = zeros(length(s1),1);

for i=1:length(s1)
    A    =    [ir11(i) ir21(i) ir31(i) ir41(i) ir51(i) ir61(i) ir71(i) ir81(i) ir91(i) ir101(i) ir111(i) ir121(i);
              ir12(i) ir22(i) ir32(i) ir42(i) ir52(i) ir62(i) ir72(i) ir82(i) ir92(i) ir102(i) ir112(i) ir122(i);
              ir13(i) ir23(i) ir33(i) ir43(i) ir53(i) ir63(i) ir73(i) ir83(i) ir93(i) ir103(i) ir113(i) ir123(i);
              ir14(i) ir24(i) ir34(i) ir44(i) ir54(i) ir64(i) ir74(i) ir84(i) ir94(i) ir104(i) ir114(i) ir124(i);
```

```
                ir15(i) ir25(i) ir35(i) ir45(i) ir55(i) ir65(i) ir75(i) ir85(i) ir95(i) ir105(i) ir115(i) ir125(i);
                ir16(i) ir26(i) ir36(i) ir46(i) ir56(i) ir66(i) ir76(i) ir86(i) ir96(i) ir106(i) ir116(i) ir126(i);
                ir17(i) ir27(i) ir37(i) ir47(i) ir57(i) ir67(i) ir77(i) ir87(i) ir97(i) ir107(i) ir117(i) ir127(i);
                ir18(i) ir28(i) ir38(i) ir48(i) ir58(i) ir68(i) ir78(i) ir88(i) ir98(i) ir108(i) ir118(i) ir128(i);
                ir19(i) ir29(i) ir39(i) ir49(i) ir59(i) ir69(i) ir79(i) ir89(i) ir99(i) ir109(i) ir119(i) ir129(i);
                ir110(i) ir210(i) ir310(i) ir410(i) ir510(i) ir610(i) ir710(i) ir810(i) ir910(i) ir1010(i)
ir1110(i) ir1210(i)];     %10*12
    B    =    [s1(i); s2(i); s3(i); s4(i); s5(i); s6(i); s7(i); s8(i); s9(i); s10(i)];      %1*10
    At   =    conj(A');     %12*10
    AAt =     A * At;       %10*10
    E    =    eye(10);      %10*10
    SP   =    E * reg;      %10*10
    ASP =     AAt + SP;     %10*10
    IAS =     inv(ASP);     %10*10
    H    =    At * IAS;     %12*10
    N    =    H * B;        %1*12

    N1(i)    = N(1);
    N2(i)    = N(2);
    N3(i)    = N(3);
    N4(i)    = N(4);
    N5(i)    = N(5);
    N6(i)    = N(6);
    N7(i)    = N(7);
    N8(i)    = N(8);
    N9(i)    = N(9);
    N10(i)   = N(10);
    N11(i)   = N(11);
    N12(i)   = N(12);
end

N1   = real(ifft(N1));
N2   = real(ifft(N2));
N3   = real(ifft(N3));
N4   = real(ifft(N4));
N5   = real(ifft(N5));
```

```
N6  = real(ifft(N6));
N7  = real(ifft(N7));
N8  = real(ifft(N8));
N9  = real(ifft(N9));
N10 = real(ifft(N10));
N11 = real(ifft(N11));
N12 = real(ifft(N12));

audiowrite('test/test1.wav',N1,44100);
audiowrite('test/test2.wav',N2,44100);
audiowrite('test/test3.wav',N3,44100);
audiowrite('test/test4.wav',N4,44100);
audiowrite('test/test5.wav',N5,44100);
audiowrite('test/test6.wav',N6,44100);
audiowrite('test/test7.wav',N7,44100);
audiowrite('test/test8.wav',N8,44100);
audiowrite('test/test9.wav',N9,44100);
audiowrite('test/test10.wav',N10,44100);
audiowrite('test/test11.wav',N11,44100);
audiowrite('test/test12.wav',N12,44100);
movefile('test',name);
movefile('rec',foldername);

figure; plot(ss1); hold on; plot(ss10);
figure; plot(N1); hold on; plot(N10);
```

参考文献

[1] D. H. Cooper , J. L. Bauck, “Prospects for Transaural Recording,” *JAES,* 37, pp. 3-19, 1989.

[2] 伊勢史郎, “キルヒホッフ-ヘルムホルツ積分方程式と逆システム理論に基づく音場制御の原理,” *日本音響学会誌,* 第 53 巻, 第 9 号, pp. 706-713, 1997.

[3] R. P. Thurlow WR, “Effect of induced head movements on localization of direction of

sounds.,” *The Journal of the Acoustical Society of America,* 第 42 巻, pp. 480-488, 1967.

[4] 神沼充伸, 伊勢史郎 , 鹿野清宏, “受聴者の頭部の動きを考慮した多チャンネル音場再現システム,” *TVRSJ,* 第 5 巻, 第 3 号, pp. 957-964, 2000.

[5] 猿渡洋, 立蔵洋介 , 鹿野清宏, “M 出力 N 入力の一般的な制御問題とその解法,” *日本音響学会誌,* 第 61 巻, 第 7 号, pp. 380-385, 2005.

[6] Y. M.Miyoshi, “Inverse filtering of room acoustics,” *IEEE Transactions on Acoustics, Speech, and Signal Processing,* 第 36 巻, 第 2 号, pp. 145-152, 1988.

[7] 神沼 充伸, 伊勢 史郎 , 鹿野 清宏, “周波数領域における最小ノルム解を利用した多チャンネル音場再現システムにおける逆フィルタの設計,” *日本音響学会誌,* 第 57 巻, 第 3 号, pp. 175-183, 2001.

[8] A.N.Tihonov, “Solution of incorrectly formulated problems and the regularization method,” *Sov.Math,* 第 4 巻, pp. 1035-1038, 1963.

[9] 李 容子 , 伊勢 史郎, “境界音場制御の原理に基づく三次元音場再現システムとその逆フィルタ設計時の正則化パラメータ決定方法,” *IEICE Technical Report,* 第 49 巻, pp. 39-44, 2011.

[10] 橘 秀樹 , 日高 新人, “実物及び模型ホールのインパルス応答の測定,” *日本音響学会誌,* 第 4 8 巻, 第 4 号, pp. 244-249, 1992.

[11] mouse, “Python でサイン波(正弦波)をつくる,” 17 8 2015. [オンライン]. Available: http://www.non-fiction.jp/2015/08/17/sin_wave/. [アクセス日: 23 1 2019].

[12] yukara_13, “【Python】時間変化するサイン波を作る,” 23 12 2013. [オンライン]. Available: http://yukara-13.hatenablog.com/entry/2013/12/23/053957. [アクセス日: 23 2 2019].

[13] M. Geier, “Play and Record Sound with Python,” 2018. [オンライン]. Available: https://python-sounddevice.readthedocs.io/en/0.3.12/. [アクセス日: 9 2018].