RealSense と音響信号を用いた視覚障害者支援デバイス のシステム改善と性能評価

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 17-0091　水﨑 景太
指導教官 豊田政弘

目次

1　序論

1.1 研究背景

現在、視覚障害者が外出した場合、自転車や自動車、段差、用水路などの障害を回避し、歩行するために、白杖や盲導犬の利用、点字ブロックの設置などの工夫がなされている。これらは視覚障害者が周囲の状況を把握するのに必要不可欠なものとなっているが、白杖や盲導犬を利用するためには訓練が必要であり、点字ブロックは自転車などの障害物が置かれてしまうといった問題点が存在する。また、白杖や盲導犬の利用時には“片手はふさがった状態”である。

そこで、視覚障害者自身が音の再生装置を装着し、「音」によって常に、感覚的に、障害物の距離や位置を把握することが出来れば、両手が空いた状態で活動することができ、これまで以上に行動の幅を広げることができる。

本研究では、上記実現のために RealSense と音響信号を用いた視覚障害者支援のためのデバイスを開発し、その有用性を検証する。

1.2 既往研究

以下に深度カメラを用いた視覚障害者向けの支援装置に関する既往研究を紹介する。

●コンスタンツ大学での研究[1]

ヘルメットに取り付けた写真 1 の Kinect によって、壁や障害物への距離を測定し、腰に巻きつけた写真 2 振動装置で着用者に警告するというシステム。また、壁やドアに貼られたシンボルマークを検出し、距離などを音声で伝えることも可能である。

写真 1　Kinect を取り付けたヘルメット

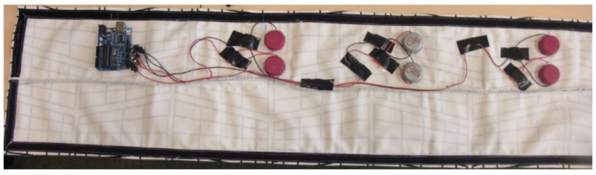

写真 2　振動装置

●高知大学での白状型歩行支援デバイスの開発[2]

腰に取り付けた写真 3 の Xtion によって、壁や障害物への距離を測定し、両手首に装着した振動装置で着用者に警告するというシステム。また、白状に右、左、中央の 3 つのボタンを取り付け、ボタンが押されると、割りあてられた領域の探索を行い、障害物の有無と障害物までの歩数を音声によって通知する。

写真 3　Xtion

●久保の研究[3]、古川の研究[4]

Kinect のドライバ、MATLAB、puredata、音の出力ドライバを用い、図 1 のように障害物を検出し音を提示する。

音の鳴り方に関しては、図 2 のようにサブセットの位置の方向からの HRTF を畳み込んだ音を提示し、音量は深度が大きくなるにつれて、小さくなるように設定する。 また、距離によって提示する音のテンポを変え、サブセットの高さによって提示する音の周波数を変化させる。

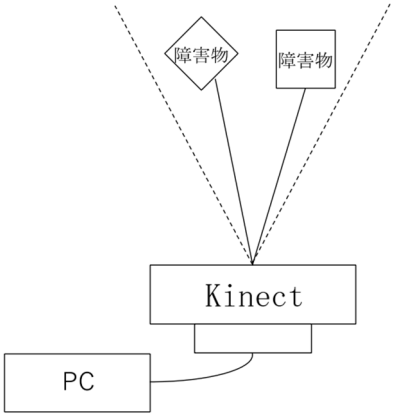

図 1 システム概要図

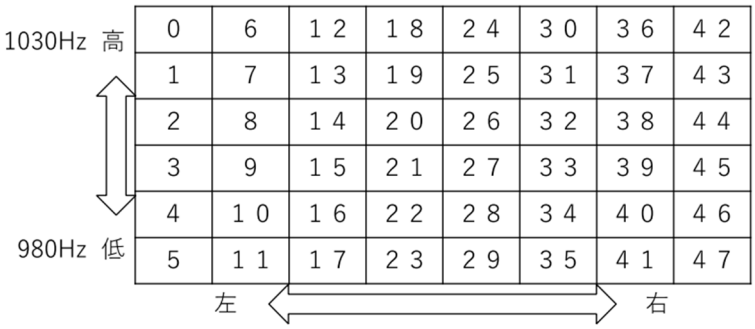

図 2 サブセット番号による提示音

結果としては、障害物の位置認識は、左右の判別は比較的容易で、正解率が高いが、上下方向の判別は難しく正解率はかなり低い結果となった。また、初めて使う状態では難しいが、少しデバイスに慣れる練習を重ねると、ゆっくりと音が鳴る方向を確認しながら進むことで、目隠しをしてデバイスを装着した状態で簡易な迷路を接触せずに通り抜けることも可能であった。

●萩原の研究[5]

Kinect によるデバイスから、新しい機器である RealSense によるデバイスへとシステムを移行し、その他の大体のシステム概要はそのままで、[4]の既往研究で確認された問題点に対する改善案を提案した。

1.3 研究目的

[5]と同様、 [3][4]の既往研究で作成された Kinect によるデバイスから、新しい機器である RealSense によるデバイスへとシステムを移行し、またさらなるシステムの安定性やコンパクトさを確保するためその他の機器もすべて新しい機器に変更し、それらにシステムを更新することを目的とする。また、新たに設定したシステムの性能の評価を行う。

最終的には RealSense と音響信号を用い、「音」によって視覚障害者が、より直感的に、また正確に障害物の位置を認識し、行動できるようになるためのデバイスを開発する。

1.4 問題点と改善案

1.4.1 問題点①

既往研究で使用されていた機器では、実際に装着して生活を送ることを考慮した時、コンパクトさや性能面において不十分な点が多かった。例えば、解像度が低い、測定範囲域が狭い、デバイスサイズが大きい、バッテリーが必要、有線接続が必要なことなどが挙げられる。

1.4.2 提案①

Kinect から RealSense、MATLAB から PYTHON、PureData から MAX、骨伝導ヘッドフォンから FRAMEALTO へ変更し、システムを更新する。

1.4.3 問題点②

今回、立体音響のように「右に障害物があれば右から聞こえるように」するために、右に音源がある場合の HRTF を畳み込むよう設定するのだが、あくまで右耳には右耳用の音のみを、左耳には左耳用の音のみを聞かせないとそのようには聞こえない。

本来これはヘッドフォンやイヤフォンなら実現するものだが、本研究ではオープンイヤー型オーディオサングラス FRAMEALTO を用いるため、右耳用の音が右耳だけでなく、左耳にも一部聞こえてしまう(これをクロスオーバーという)可能性がある。

1.4.4 提案②

クロスオーバーがどの程度起こり、音の定位を邪魔しているのかを実験により検証する。

2　研究内容

2.1 開発環境

本研究では、システムの処理には PC を用いる。PC に RealSense、FRAMEALTO を接続し、障害物の検知、使用者への通知を行う。また、RealSense との互換性とプログラム作成の簡易性を考慮して、デバイスのためのシステム作成には PYTHON を使用する。図 4 に各機器の接続を示し、また以下に使用機器を示す。

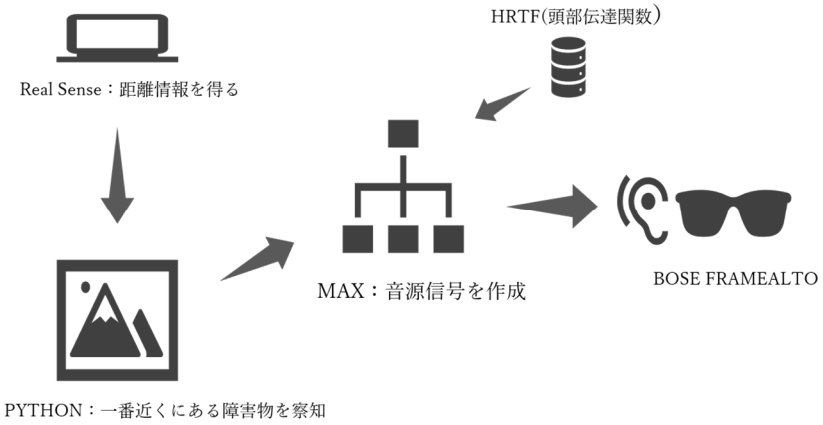

図 4 各機器の接続

●更新した機器とその効果

・Kinect→Real Sense：小型化、解像度向上、測定範囲広域化、PC からの電源供給

・MATLAB→PYTHON：Real Sense が MATLAB に対応してないため

・PureData→MAX：動作安定化

・骨伝導ヘッドフォン→FRAMEALTO：出力の増強、ワイヤレス接続、バッテリー駆動

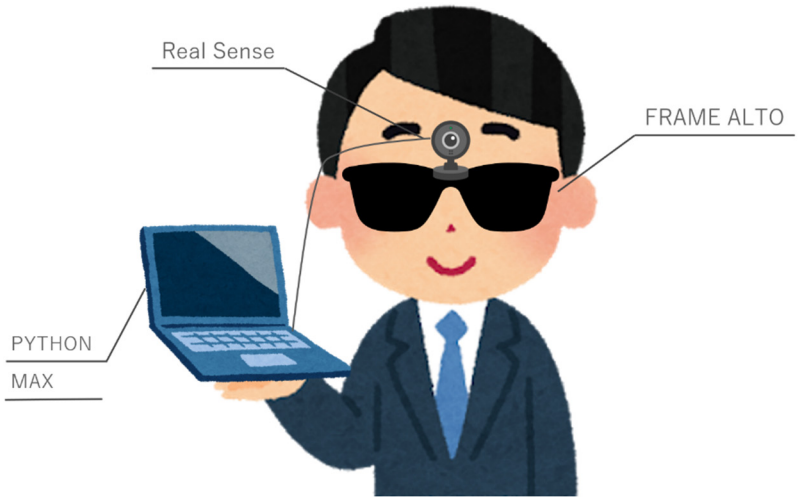

図 5 見た目のビジュアル

写真 4 RealSense

写真 4 が Intel 社の RealSense モデルD435 である。RGB センサと赤外線を用いた深度センサが 2 つ搭載されており、障害物との距離を測ることができる。尚、深度センサの解像度は以下に最高値を記している。

仕様

・解像度深度センサ／1280×720pixel
・ＲＧＢセンサ／1920×1080pixel
・角度横／85° 縦／58°
・距離計測範囲 0.2～10m
・サイズ 90×25×25mm

写真 5 FRAMEALTO

写真 5 が BOSE 社の FRAMEALTO である。これは、耳をふさがずに音が聞けるオープンイヤー”オーディオサングラス”で、Bluetooth が搭載されており完全ワイヤレス接続ができ、また耳をふさがないため安全性も確保できる。

2.2 システム

❶RealSense によって 640×480 の深度データを取得し、PYTHON で 8×6 のサブセットに分割する。 そして最も深度の小さいサブセットの番号と深度を検出し、MAX に送る。

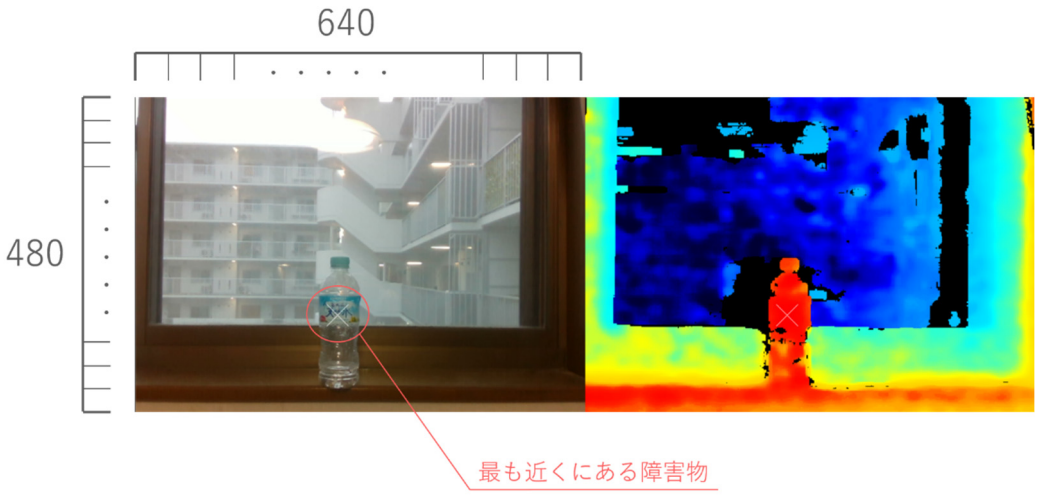

図 6 RealSense により取り込んだ深度データ

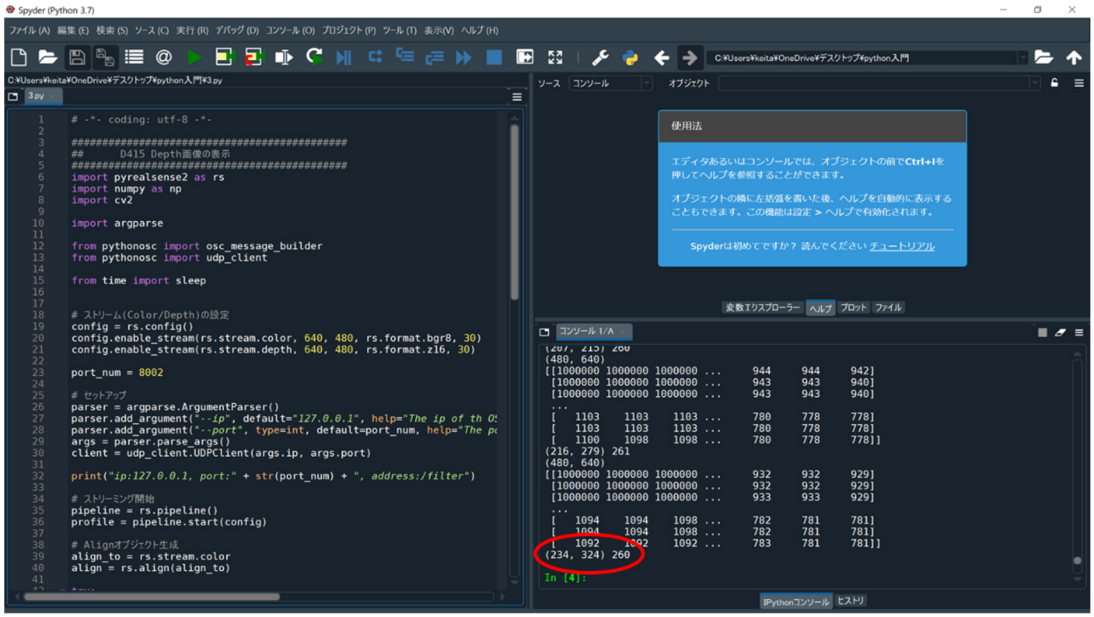

図 7 最も深度の小さいサブセットの番号と深度を検出した PYTHON の画面

❷ PYTHON で得たデータから MAX で障害物までの方向を算出し、それに対応した HRTF を取得。また障害物距離に応じた音源信号を用意し、この音源信号と HRTF を畳み込んだ信号を作成する。

HRTF：立体音響、右に障害物があれば右から聞こえるようにする

音源信号：距離によって提示する音のテンポが変わる。

近い(0.2～0.4m)　ピピピピピ

中間(0.5～0.7m)　ピピピ

遠い(0.8m～)　　ピピ

サブセットの高さによって提示する音の周波数を変化させる。

上方：高い音

下方：低い音

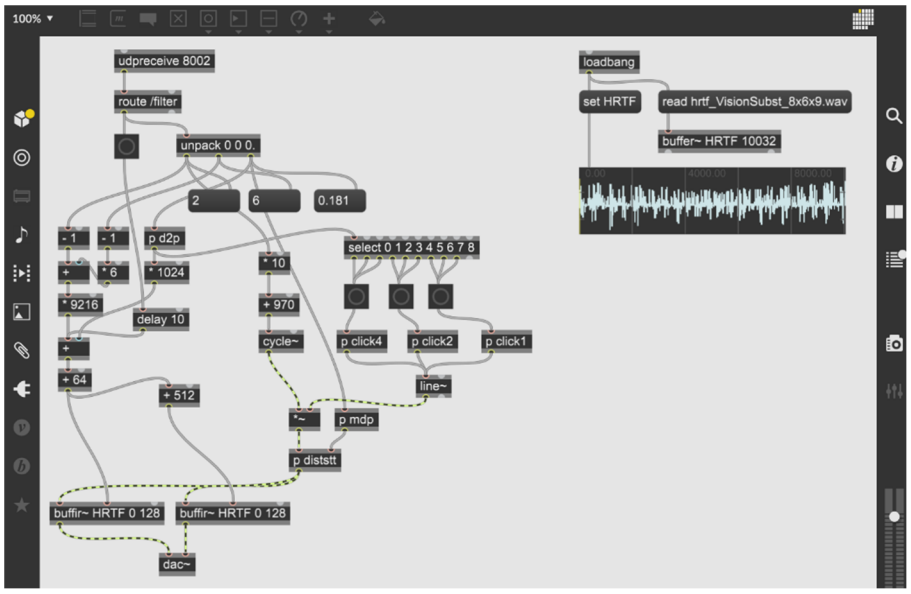

図 8 MAX の画面

❸MAX で作成した音源信号を Bluetooth 接続した FRAMEALTO から鳴らす。

3　評価実験とデバイスの改良

3.1 実験①　FRAMEALTO による立体音響再現性能の評価

機器を更新したことで新たな課題となったクロスオーバーによりどの程度再現性が失われてしまうのかを調べる。本来、右下に障害物があれば右下から音が鳴っているように聞こえるはずだが、実際 FRAMEALTO から自分の耳に伝わってくるおとによってどの程度正確に場所を特定できるのかを調べる。

実験方法

図 9 のように自分の視野を縦 6 横 8 に区切ったマス目に置き換え、音が鳴った場所に最も近いと思われるマスにチェックを入れてもらう。

図 9　縦 6 横 8 のチェックシート

以上の行程を 5 人(一人 20 回)に行い、合計 100 データを得た。

3.2 結果①

図 10 に実験結果を示す。一番上の横列は試行回数、2.3 番目の横列は縦方向のズレと横方向のズレ、一番下の横列は縦横のズレから算出したズレの絶対距離を示す。なお、「1」は「1マス」のズレを表す。

24歳男性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	1	1	0	0	0	0	0	0	0	1	1	1	1	0	2	1	1	1	0	0.55
横	0	1	1	0	1	0	0	0	0	1	1	0	0	1	1	1	1	1	1	1	0.6
距離	0	1.4	1.4	0	1	0	0	0	0	1	1.4	1	1	1.4	1	2.2	1.4	1.4	1.4	1	0.9

23歳男性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	0	1	1	0	1	1	2	2	1	1	1	0	0	1	2	1	0	0	0.75
横	1	1	0	0	0	1	1	2	0	0	1	1	0	0	1	1	2	1	1	0	0.7
距離	1	1	0	1	1	1	1.4	2.2	2	2	1.4	1.4	1	0	1	1.4	2.8	1.4	1	0	1.2

22歳男性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	2	1	0	1	2	0	0	1	3	0	2	0	3	0	0	2	0	2	0	0	0.95
横	0	1	0	0	0	0	1	0	2	1	1	1	0	1	0	0	0	1	1	1	0.55
距離	2	1.4	0	1	2	0	1	1	3.6	1	2.2	1	3	1	0	2	0	2.2	1	1	1.32

22歳女性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	0	1	1	1	0	1	0	2	3	1	0	1	3	0	1	4	3	3	1.25
横	0	1	0	1	1	0	2	0	0	2	1	2	2	0	0	2	1	1	2	1	0.95
距離	0	1	0	1.4	1.4	1	2	1	0	2.8	3.1	2.2	2	1	3	2	1.4	4.1	3.6	3.1	1.805

21歳女性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	2	2	2	0	1	0	1	1	1	1	3	2	0	1	1	0	1	1	3	1	1.2
横	0	2	0	0	0	0	1	0	1	0	0	1	0	1	0	0	0	0	0	2	0.4
距離	2	2.8	2	0	1	0	1.4	1	1.4	1	3	2.2	0	1.4	1	0	1	1	3	2.2	1.37

図 10-1 実験結果

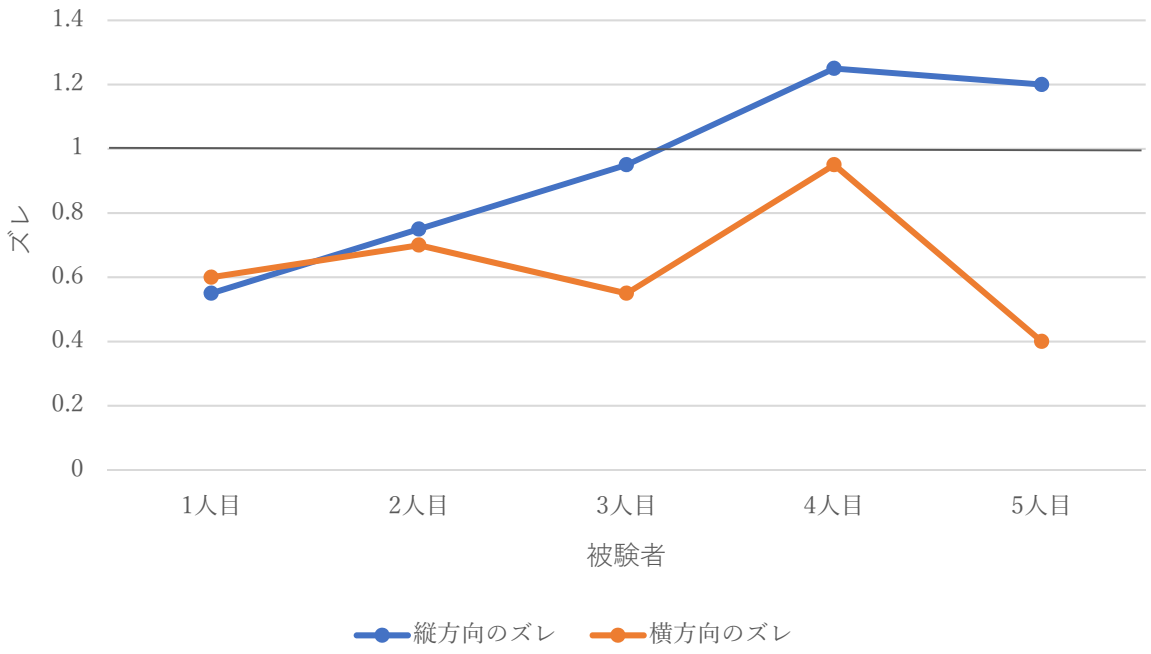

図 10-2 実験結果

縦平均	0.94
横平均	0.64
距離平均	1.319

図 10-3 実験結果

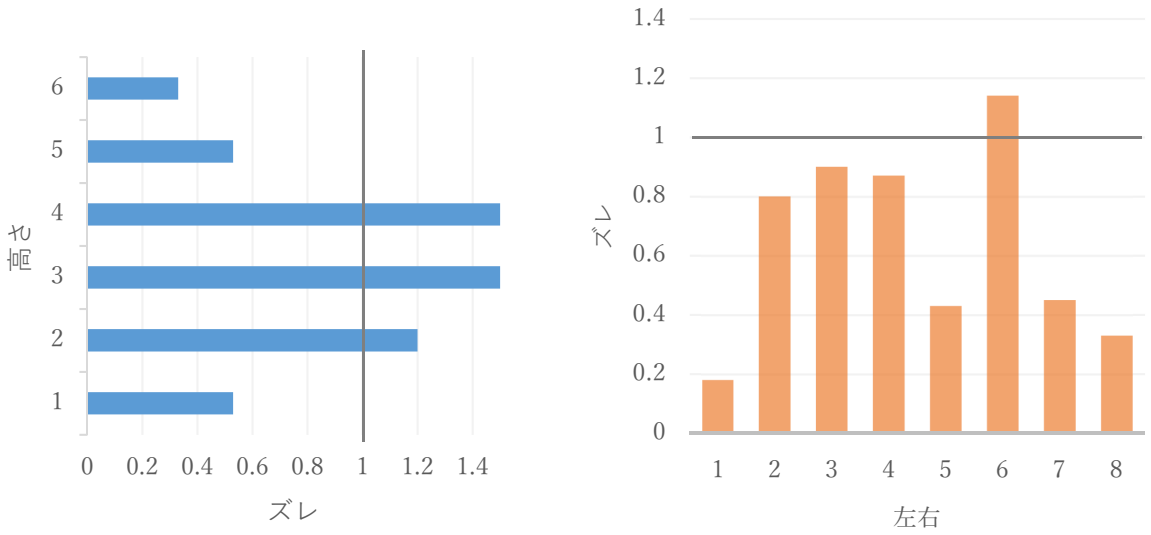

図10-4 実験結果

3.3 考察①

横方向のズレの平均が 1 以下であることから、クロスオーバーによる立体音響再現性能に問題はほとんどないと読み取れる。このことから機器を新たに更新したことにより、コンパクトさが上昇しただけでなく、性能面でも高い評価を得られたといえる。

ただ、縦方向のズレの平均も 1 以下ではあるが横方向のズレと比べると大きい値になっており、被験者からも「少し分かりにくい」という声が上がっていたことから見直すべきであると判断した。また、このデバイスは 1 回の使用だと上下左右の感覚を覚えられず判別することが難しいため、複数回使用することで精度が上がってくるのではないかと考えた。

。

3.4 実験② 縦方向のズレを改善

実験①では縦方向のズレが目立ったのだが、それは下から上にかけて 10Hz ずつ変化させ、だんだん音が高くなるように設定していたが、適当な周波数を割り当てていたことでこの音の高さがどの位置を示しているのか分かりにくくなっていたからであると考えた。したがって鳴らす音をある普遍的で絶対的な量にすることで認識しやすく、また慣れやすくなるのではないかと考え、下から「ドレミファソラ」という音階を使用することにした。
（図 11）

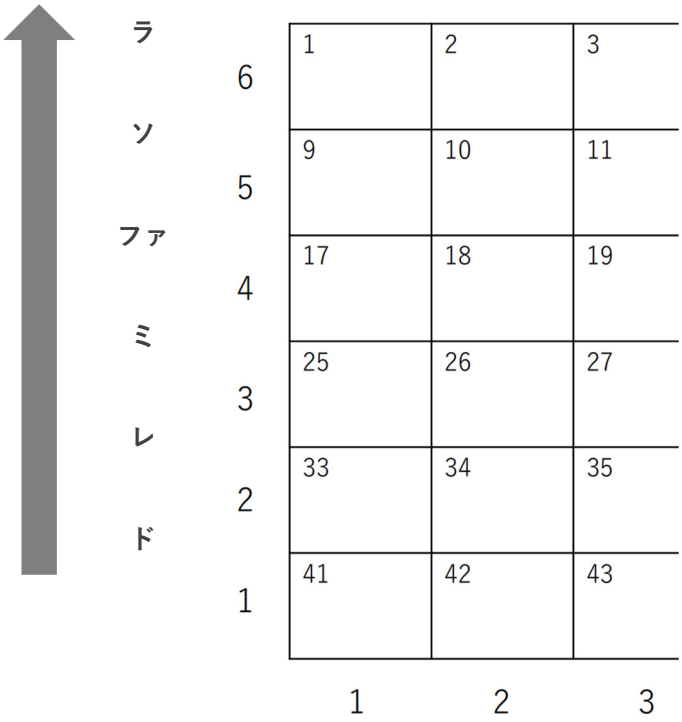

図 11

3.5 結果②

図 12 に実験結果を示す。

22歳女性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
横	1	0	1	1	2	0	1	0	1	1	0	0	0	0	1	0	0	0	1	2	0.6
距離	2.2	0	1	1	2	0	1	0	1	1	0	0	0	0	1	0	0	0	1	2	0.66

21歳女性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0.05
横	2	1	0	0	1	0	0	0	0	1	1	0	1	1	0	1	1	0	0	0	0.5
距離	2	1	0	0	1	0	0	0	0	1	1	0	1	1	1	1	1	0	0	0	0.55

図

21歳男性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	1	0	0	0	0	1	0	1	0	1	1	1	0	1	1	0	0	0	0	0	0.4
横	0	0	1	1	0	1	1	1	1	0	0	1	0	0	2	0	0	1	1	2	0.65
距離	0	0	1	1	0	1.4	1	1.4	1	1	1	1.4	0	1	2.2	0	0	1	1	2	0.87

22歳男性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	1	0	0	0	0	1	0	0	1	1	1	0	2	1	0	0	1	0	0.45
横	1	0	1	0	0	0	2	1	0	1	3	2	2	1	1	0	0	2	0	1	0.9
距離	1	0	1.4	0	0	0	2	1.4	0	1	3.1	2.2	2.2	1	2.2	1	0	2	0	1	1.075

23歳女性

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	1	0	0	0	0	2	0	0	0	0	0	0	0	0	0	2	2	2	0.45
横	0	1	1	0	2	3	2	2	1	0	1	1	1	0	0	2	1	0	1	0	0.95
距離	0	1	1.4	0	2	3	2	2.88	1	0	1	1	1	0	0	2	1	2	2.2	2	1.274

図 12-1 実験結果

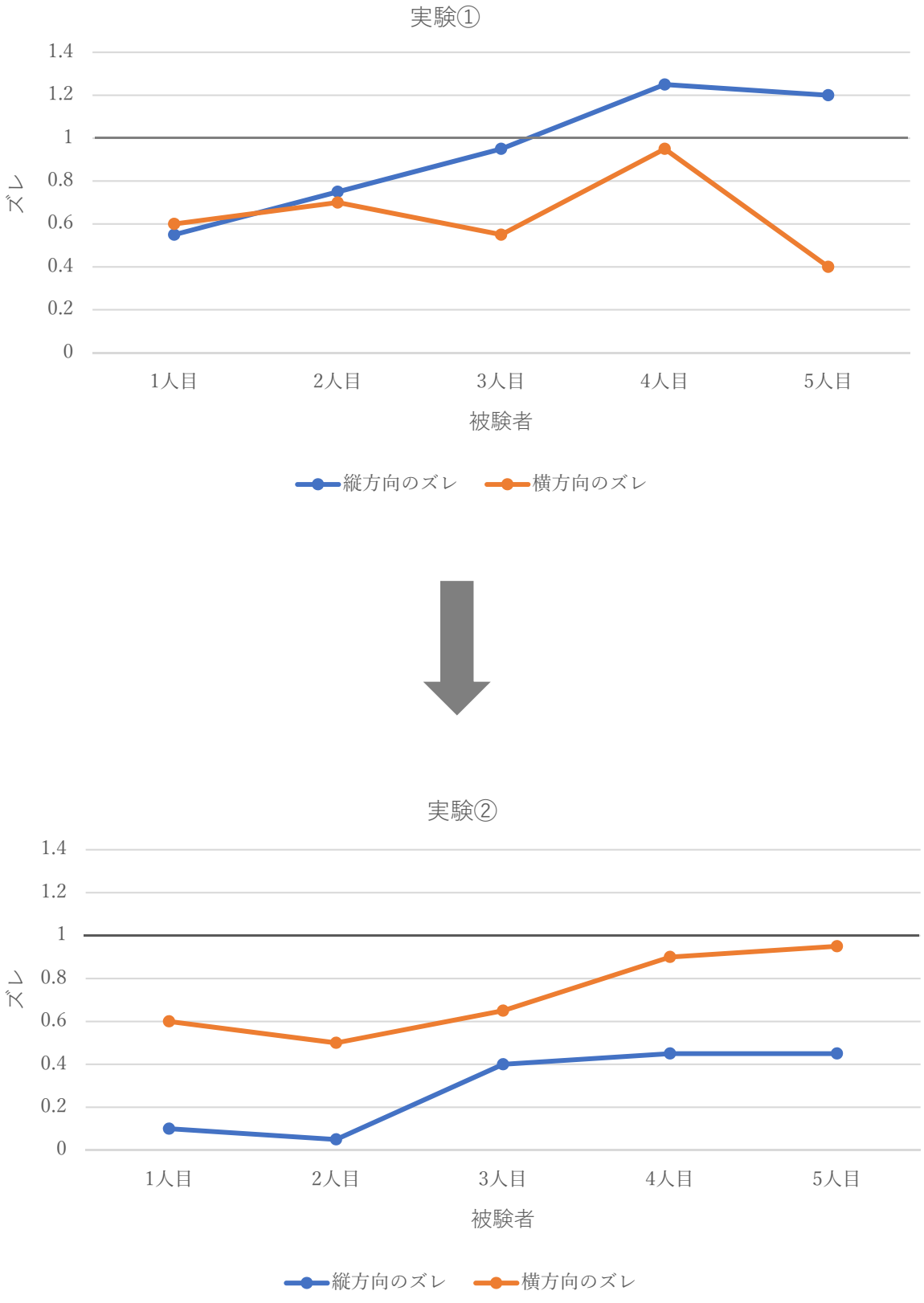
実験①
1.4
1.2
1
0.8
0.6
0.4
0.2
0
ズレ
1人目
2人目
3人目
4人目
5人目
被験者
縦方向のズレ
横方向のズレ
実験②
1.4
1.2
1
0.8
0.6
0.4
0.2
0
ズレ
1人目
2人目
3人目
4人目
5人目
被験者
縦方向のズレ
横方向のズレ

図 12-2 実験結果

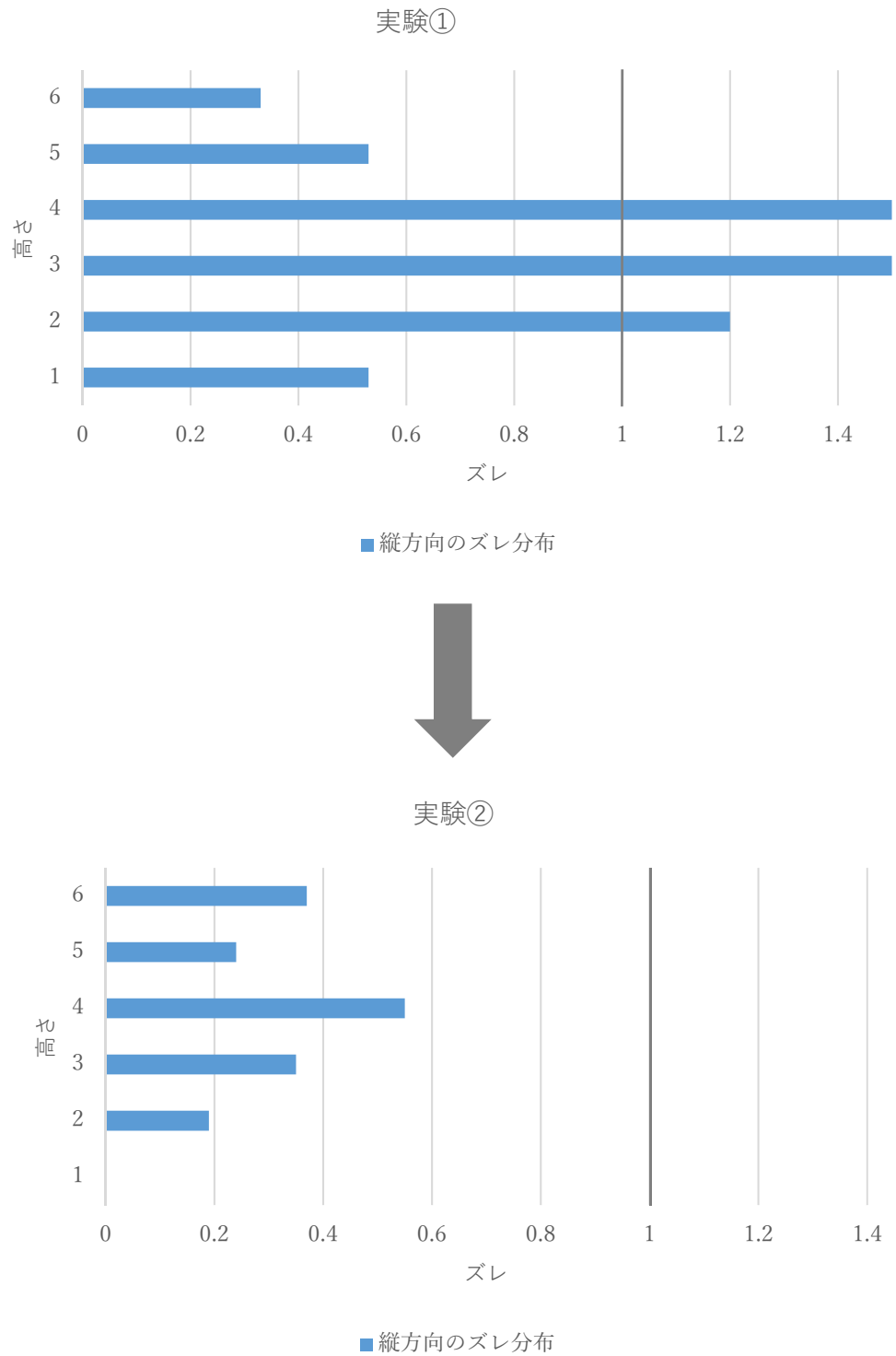

図12-2 実験結果

縦平均	0.29
横平均	0.72
距離平均	0.886

図12-3 実験結果

3.6 考察②

高さを表すのに音階を使用したことで、縦方向のズレの平均は 0.94 から 0.29 にまで減少し、かなり判別の精度が上がったといえる。

3.7 実験③　複数回使用することで精度が上がるのか

高さを音階で表すことで縦方向の精度が上がることが分かったが、このデバイスは 1 回の使用だと上下左右の間隔を覚えられず判別することが難しいため、複数回使用することで精度が上がってくるのではないかと考えた。

そこで、実験②で使用した改善版を使用し 1 人に 5 回連続で試行してもらい、どのように精度が変化するのか調べた。

3.8　結果③

図 13 に実験結果を示す。図 13－2 の直線はズレの平均値の線形近似曲線を表す。

1回目

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0.1
横	1	0	1	1	2	0	1	0	1	1	0	0	0	0	1	0	0	0	1	2	0.6
距離	2.2	0	1	1	2	0	1	0	1	1	0	0	0	0	1	0	0	0	1	2	0.66

2回目

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
横	0	0	1	0	1	0	0	1	0	1	0	1	2	1	0	0	0	1	1	0	0.5
距離	0	0	1	0	1	0	0	1	0	1	0	1	2	1	0	0	0	1	1	0	0.5

3回目

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
横	0	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0.4
距離	0	1	0	1	1	1	0	0	0	1	0	0	0	0	0	0	1	0	1	1	0.4

4 回目

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0.15
横	0	1	1	0	0	1	1	1	0	1	2	0	0	0	1	1	1	0	1	1	0.65
距離	0	1	1	0	0	2.2	1	1	0	1	2	0	0	0	1	1	1	0	1	1.4	0.73

5 回目

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	平均
縦	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
横	1	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0.3
距離	1	1	1	0	0	0	0	0	0	0	1	1	0	0	1	0	0	0	0	0	0.3

図 13-1 実験結果

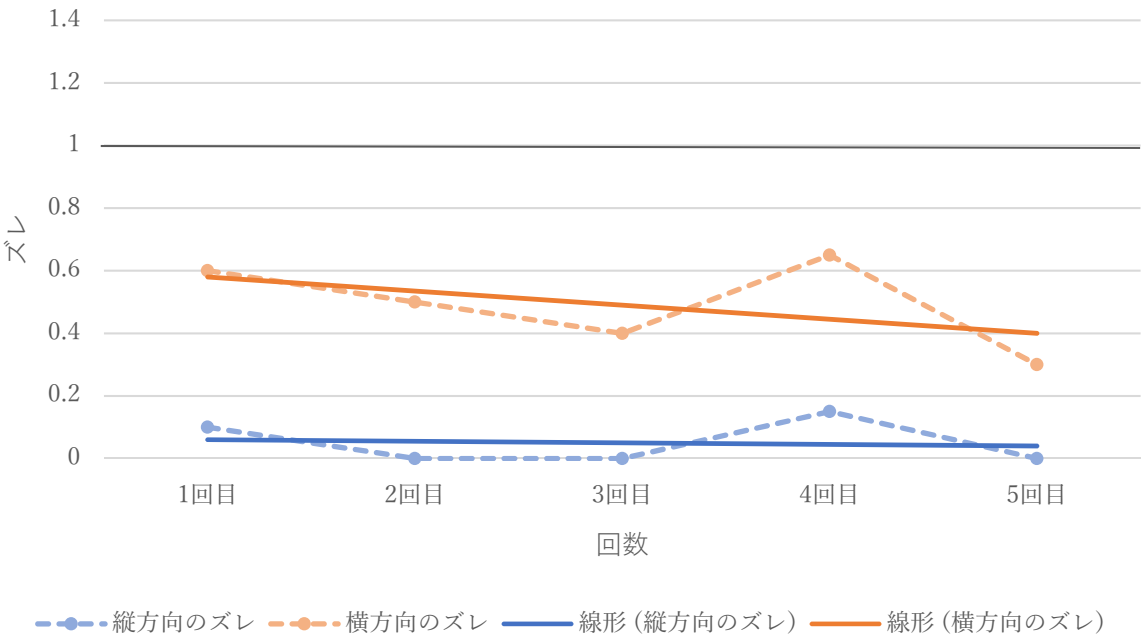
1.4
1.2
1
0.8
0.6
0.4
0.2
0
ズレ
1回目
2回目
3回目
4回目
5回目
回数
縦方向のズレ
横方向のズレ
線形（縦方向のズレ）
線形（横方向のズレ）

図 13-2 実験結果

3.9 考察③

グラフにあるように 1 回目と比べ 5 回目の方が全体的にズレが少なく、精度が上がったことが分かった。被験者自身も段々慣れてくることでやりやすさを感じていた。

4　総括

4.1 まとめ

RealSense と音響信号を用いた視覚障害者支援デバイスを実用化させていくには、障害物の認識の精度をより向上させていくことが求められており、まだまだ改良すべき点が多々ある。

本研究では、過去に行われていた研究のシステムを参考にし、使用機器やソフトを新しく物に移行することにより、コンパクトさや性能向上をはかった。

実際、バッテリー搭載による電源不要や機器同士のワイヤレス接続によりシンプルな見た目になり実用性が増した。また、立体音響再現性能をはかる実験では、認識のズレを 1 以下に抑え、さらに高さを音階で示すことにより益々認識の精度を上げることができた。

実験③では回数を重ねるごとに精度が上がることが分かったため、長期的な使用を行うことで今以上の視覚補助が期待できる。

4.2 今後の展開

本研究では実際に装着し歩行する実験を行えていないため、この新しくなったデバイスでどの程度歩行ができるのかを試したいと思った。そして、平面上の簡単な歩行のみでなく段差や動きのある障害物にも対応できるようにする必要がある。また、使用者に通知する音声の種類の増加、音声による伝達なども検討すべきである。

参考文献

[1] ウェアラブル Kinect ナビ、URL(https://japanese.engadget.com/2011/03/28/kinect/)

[2] 篠原克麻、森雄一郎、“視覚障害者のための白杖型歩行支援デバイスの開発”、高知大学 卒業論文、2016 年、URL(http://trick.is.kochi-u.ac.jp/Vol08/article01.html)

[3] 久保卓真、“kinect と音響信号を用いた視覚支援デバイスの開発”、関西大学卒業論文、2013 年

[4] 古川晃司、“kinect と音響信号を用いた視覚支援デバイスの性能向上”、関西大学卒業論文、2014 年

[5] 萩原洋平、“RealSense と音響信号を用いた視覚障害者支援デバイスの開発”、関西大学卒業論文、 2019 年

付録：Python コード

```
# -*- coding: utf-8 -*-

##############################################
##          D415 Depth 画像の表示
##############################################
import pyrealsense2 as rs
import numpy as np
import cv2

import argparse

from pythonosc import osc_message_builder
from pythonosc import udp_client

from time import sleep


# ストリーム(Color/Depth)の設定
config = rs.config()
config.enable_stream(rs.stream.color, 640, 480, rs.format.bgr8, 30)
config.enable_stream(rs.stream.depth, 640, 480, rs.format.z16, 30)

port_num = 8002

# セットアップ
parser = argparse.ArgumentParser()
parser.add_argument("--ip", default="127.0.0.1", help="The ip of th OSC Server")
parser.add_argument("--port", type=int, default=port_num, help="The port the OSC server is listening on")
args = parser.parse_args()
client = udp_client.UDPClient(args.ip, args.port)

print("ip:127.0.0.1, port:" + str(port_num) + ", address:/filter")
```

```
# ストリーミング開始
pipeline = rs.pipeline()
vfprofile = pipeline.start(config)

# Align オブジェクト生成
align_to = rs.stream.color
align = rs.align(align_to)

try:
    while True:
        # フレーム待ち(Color & Depth)
        frames = pipeline.wait_for_frames()
        aligned_frames=align.process(frames)
        color_frame = aligned_frames.get_color_frame()
        depth_frame = aligned_frames.get_depth_frame()
        if not depth_frame or not color_frame:
            continue
        color_image = np.asanyarray(color_frame.get_data())
        # Depth 画像
        depth_color_frame = rs.colorizer().colorize(depth_frame)
        depth_color_image3 = np.asanyarray(depth_color_frame.get_data())
        depth_color_image= np.asanyarray(depth_frame.get_data())

depth_color_image2=np.where(depth_color_image==0,1000000,depth_color_image)
        print(depth_color_image2.shape)
        print(depth_color_image2)
        idx = np.unravel_index(np.argmin(depth_color_image2),
depth_color_image2.shape)
        print(idx, np.amin(depth_color_image2))

        osc_msg = osc_message_builder.OscMessageBuilder(address= "/filter")
        osc_msg.add_arg(6-int(idx[0]/80))
        osc_msg.add_arg(int(idx[1]/80)+1)
        osc_msg.add_arg(float(np.amin(depth_color_image2)/1000))
        osc_msg = osc_msg.build()
        client.send(osc_msg)
```

```
#def main():
  #print("type int:")
  #input_str = input()
  #osc_msg, osc_list = make_osc(input_str)
  #print(osc_list)
  #client.send(osc_msg)

#def make_osc(input_str):
  #msg = osc_message_builder.OscMessageBuilder(address= "/filter")
  #input_list = list(input_str)
  #output_list = []
  #for i in range(len(input_list)):
    #output_list.append(int(input_list[i]))
    #msg.add_arg(output_list[i])
  #msg = msg.build()
  #return msg, output_list

#if __name__ == "__main__":
  #while True:
    #main()

        # 表示
        cv2.drawMarker(depth_color_image3, (idx[1],idx[0]), (255, 255, 255),
markerType=cv2.MARKER_TILTED_CROSS, markerSize=30)
        cv2.drawMarker(color_image, (idx[1],idx[0]), (255, 255, 255),
markerType=cv2.MARKER_TILTED_CROSS, markerSize=30)
        images = np.hstack((color_image, depth_color_image3))
        cv2.namedWindow('RealSense', cv2.WINDOW_AUTOSIZE)
        cv2.imshow('RealSense', images)
        if cv2.waitKey(1) & 0xff == 27:
            break

        sleep(1.0)
```

```
finally:
    # ストリーミング停止
    pipeline.stop()
    cv2.destroyAllWindows()
```