両目で異なる方向への視野の提示手法

森山有理名[1,a)]　魚井宏高[2]

概要：人間は通常両目が前を向き，その視野は大きく被っており，両目の視差情報から距離感を得ている．草食動物では捕食者から逃げる為に広い視野を確保するため，両目の視野は大きく離れており，被っている視野は人間や肉食動物よりも少なくなっている．本研究では人間に左右で全く異なる視野を提示することで，より周囲の環境の変化に早く気付くことを目的とする．

1. 背景

草食動物は捕食者から逃げる為にいち早く周囲の変化に気づく必要があり，両目が左右に大きく離れてついている．これは左右の視野を被せることによって視差情報から獲物までの距離を測る肉食動物に対して，距離感よりも広い視野角を必要としているからである．人間は狩猟を行うため，肉食動物と同様に両目が前面を向くようについているが，近年では狩猟を行う人間は減っており，対して事故や事件に巻き込まれる可能性の方が高いため，人間も左右で大きく離れた視野を持つことが出来れば，周囲の環境の変化により早く気付くことができると考えた．

2. 目的

左右の目に異なる方向の視野を提示し，その結果周囲の環境の変化への反応がより早く正確になることを目的とする．これにより周囲の環境を素早く認識できる視野を導き出す．

3. 試作

水野・早坂・山口 [1] は，任意の方向の独立した視野を両眼に提示する装置を，バーチャルカメレオンとして示した．

これを参考に，任意の左右の視野の角度を変更できるヘルメットを制作した．(図 1)

ヘルメットの頭上に Web カメラ 2 基をカメラマウントで固定した．このカメラ 2 基は自由に方向を変えることが可能である．

また，ヘルメット上の Web カメラをつないだ PC に，VR ヘッドマウントディスプレイを接続し，ヘッドマウントディスプレイの左右それぞれの映像が各 Web カメラの映像を映し出すソフトウェアを Unity で制作した．(図 2)

図 1　実験用ヘルメット

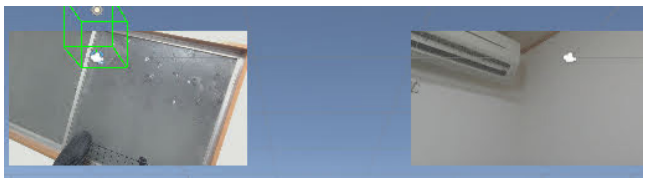

図 2　ヘルメット上の 2 基の Web カメラの映像をヘッドマウントディスプレイに表示するソフトウェア

このヘルメットを用いて被験者の周りに記号を提示し，回答させる実験を行おうとしたが，実環境での記号の提示が困難であったため，VR 空間上に記号を提示する実験を行った．当初想定した実験内容は以下のとおりである．

被験者に実験用ヘルメットを装着してもらい，Web カメラ 2 基を正面に向け通常の視野と同等の角度に設定する．被験者の周囲 360 度のうち無作為に選出した方向にトランプのカードを提示し，被験者にカードの名前を回答させる．

1　大阪電気通信大学大学院 総合情報学研究科
2　大阪電気通信大学 総合情報学部
a)　mw20a009@oecu.jp

提示から回答までの時間を計測する．10 回同様の手順を行った後，片方の Web カメラを回転させ，視野を変更して同様の手順を続ける．

4. VR 空間上での実験

4.1 準備

Unity を用い実験用ソフトウェアを制作した．(図 3)

5〜10 秒のランダムな時間が経過すると，VR 空間内で被験者の周囲のランダムな方向に 4 方向の矢印を記載したパネルを提示する．VR ゴーグルの左右の映像はそれぞれ独立して方向を変化させることができるようになっている．

図 3 実験用ソフトウェア

4.2 手順

被験者はパネルを発見し，可能な限り素早く矢印の方向へスティックを入力する．約 1 分おきに VR ゴーグルの左右の映像がそれぞれランダムな方向へ切り替わる．パネルが提示された時の左右の視線の方向および，被験者が回答を行った時の左右の視線の方向，そしてパネルが提示されてから被験者が回答を行うまでの時間を計測する．

5. 結果と考察

以下のコード (図 4) を用いて集計を行った．

```
1  import pandas as pd
2  import matplotlib.pyplot as plt
3  import numpy as np
4  import seaborn as sns
5  df = pd.read_csv("all.csv", header=0)
6  df['distance'] = df['rightEnd']-df['leftEnd
       ']
7  df['distance'] = np.where(df['distance'
       ]>180, df['distance']-360, df['distance'
       ])
8  df['distance'] = np.where(df['distance'
       ]<-180, df['distance']+360, df['distance
       '])
9  df = df.query('correct␣==␣True')
10 df.drop(['name', 'timestamp', 'correct'],
       axis='columns', inplace=True)
11 df.head()
```

図 4 回答を読み込んで左右の視線の角度の開きを計算するコード

回答した矢印の方向が間違っていたものについては，認識ができていなかったとして除外する．集計結果から 5 件抜き出したサンプルが (図 5) である．

	sec	targetDirection	direction	targetAngle	leftStart	rightStart	leftEnd	rightEnd	distance
0	3.133004	right	right	26.0	215.35730	83.35727	27.39023	255.39020	-132.00003
1	1.029540	down	down	209.0	24.65765	252.65770	328.51440	196.51440	-132.00000
2	1.977723	down	down	342.0	319.40560	187.40560	10.31140	238.31140	-132.00000
3	1.670750	right	right	36.0	165.37170	33.37170	165.09920	33.09924	-131.99996
4	0.585473	left	left	34.0	73.61475	301.61470	32.21796	260.21800	-131.99996

図 5 集計結果のサンプル

計測時に記録した値は以下のとおりである．

- sec：矢印パネルが提示されてから回答までの時間
- targetDirection：パネルの矢印の方向
- direction：被験者の回答した矢印の方向
- targetAngle：矢印パネルが提示された方向
- leftStart：矢印パネルが提示された瞬間の左目の方向
- rightStart：矢印パネルが提示された瞬間の右目の方向
- leftEnd：被験者が回答した瞬間の左目の方向
- rightEnd：被験者が回答した瞬間の右目の方向
- distance：左右の視線の角度の開き (-180〜180, 値が負の場合右目と左目の左右方向が入れ替わっている)

5.1 左右の視線の角度の開きと回答時間

著しい回答の遅れを除外した結果が以下のグラフ (図 6) である．

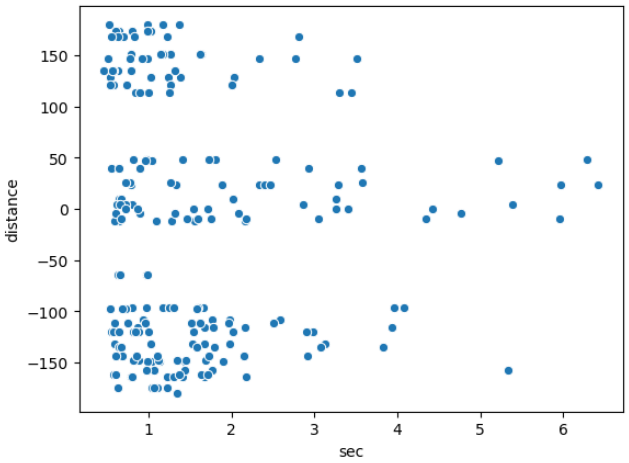

図 6 著しい回答の遅れを除外した左右の視線の角度の開きと回答時間

Y 軸が 0 に近いほど両目が同じ方向を向いており平常時の視線に近くなる．X 軸は矢印パネル提示から発見して回答までにかかった時間である．

左右の視線の角度の開きが大きくなるほど回答時間が短くなり，distance が 0 に近いほど，すなわち人類の通常の視野では矢印パネルの発見が遅れる傾向が見られる．

また，視線の角度が開いている状態であっても，distance

がプラスの場合，即ち右目と左目の左右方向が正常である方が回答時間が短くなる傾向が見られる．

5.2 パネル提示時点でのパネルの角度と視線の角度の差と回答時間

さらに，以下のコード (図 7) を用いて提示時点でのパネルの角度と視線の角度のずれを計算し，グラフ (図 8) にした．

```
df['startLeftDistance'] = (df['leftStart']-
    df['targetAngle']).abs()
df['startRightDistance'] = (df['rightStart'
    ]-df['targetAngle']).abs()
df['absDistance'] = df['distance'].abs()
df['startLeftDistance'] = np.where(df['
    startLeftDistance']>180, (df['
    startLeftDistance']-360)*-1, df['
    startLeftDistance'])
df['startLeftDistance'] = np.where(df['
    startLeftDistance']<-180, (df['
    startLeftDistance']+360)*-1, df['
    startLeftDistance'])
df['startRightDistance'] = np.where(df['
    startRightDistance']>180, (df['
    startRightDistance']-360)*-1, df['
    startRightDistance'])
df['startRightDistance'] = np.where(df['
    startRightDistance']<-180, (df['
    startRightDistance']+360)*-1, df['
    startRightDistance'])
df['startDistance'] = np.where(df['
    startRightDistance']<df['
    startLeftDistance'], df['
    startRightDistance'], df['
    startLeftDistance'])
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
```

図 7 提示時点でのパネルの角度と視線の角度のずれを計算するコード

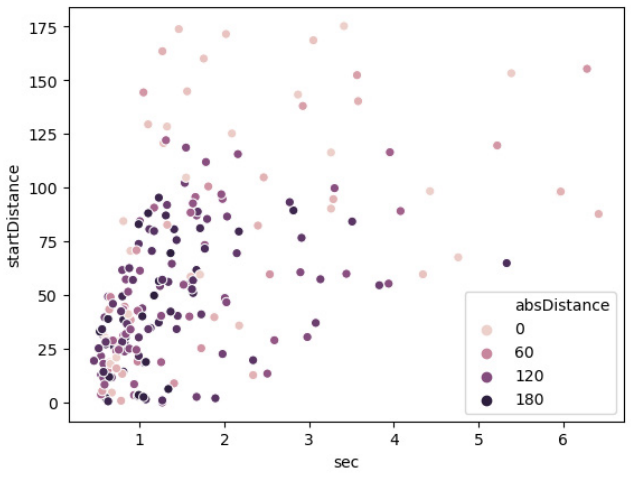

図 8 パネル提示時点でのパネルの角度と視線の角度の差と回答時間

計測時に記録した値は以下のとおりである．

- startLeftDistance：パネル提示時点でのパネルの角度と左目の角度の差の絶対値
- startRightDistance：パネル提示時点でのパネルの角度と右目の角度の差の絶対値
- startDistance： startLeftDistance と startRightDistance のうち，値が小さいもの
- absDistance：左右の視線の角度の開きの絶対値

Y 軸が 0 に近いほど，パネル提示時点で視線がパネルに近い方向を向いている．X 軸は矢印パネル提示から発見して回答までにかかった時間である．

また，点の色が濃くなるほど左右の視線の角度の開きが大きくなる．平常時に近いほど色が薄くなる．当然ながら，パネルの提示された方角が元から視線が向いていた方角に近いほど回答までの時間が短くなる傾向にある．

そして，色の薄い点がグラフ上部に多く見られることから，左右の視線の角度の開きが小さいほど，離れた方角にパネルが提示された時，回答に時間がかかる傾向にある．

6. 結論

以上の結果より，左右の視線の角度が 100 度〜180 度開いている場合に，周囲の環境の変化に素早く気づくことが可能である．

7. 課題

先に挙げたヘルメットを利用した実験を行えていないため，実環境での実験手法を考慮する必要がある．矢印パネルの提示には，AR ゴーグルを用い，背景は実環境を提示するといった方法が考えられる．

VR ゴーグルでは人間の片目で捉えられる全ての視界に映像を提示できないため，VR ゴーグルを使用しない環境での検証を行いたい．

また，今回は被験者の回転のみで実験を行ったため，被験者が前後左右に移動できる等の運動を伴う状態ではまた違った結果が出る可能性もある．

参考文献

[1] 水野 文雄, 早坂 智明, 山口 隆美：バーチャルカメレオン-任意の方向の独立した視野を両眼に呈示するシステムに関する基礎的研究-, 情報科学技術フォーラム講演論文集, Vol. 8, No. 3, p. 631 (2009).