視覚探索での眼球運動における探索方策に関する研究
Research on search strategy in eye movement in visual search

コース　　物質科学コース
学籍番号　　１４０５８３
氏名　　前川　黎
指導教員　　ミケレット・ルジェロ

In current modern society the flow of information is huge. It is very important to study how the brain searches for information. Visual search is the prominent example of information elaboration and in this research, we use an eye tracking device to investigate the eye movement during original targets visual search. We found that searching strategies varies between subject and we identified for the first time four different eye movements strategies. Also, we found a clear trend in the visual center of mass. In this thesis I will describe all the details of these new findings.

●背景・目的

視覚探索とは、ディスプレイに複数の刺激項目(Search Panel)を提示し、その中にあらかじめ決められた目標項目があるか否かを被験者が判断する課題である。視覚探索は、目を持つ動物が日常的に行う行動である。常に視覚から得られる膨大な情報を脳で処理し、必要な情報を選択するメカニズムの解明は今後の脳科学の研究において重要な課題である。このメカニズムを解明するため本研究では、近年発展を遂げている VR を用いて、視覚探索課題における眼球運動を定量的に測定した。その中でも、視覚探索において目標項目を探索する視線の動き方、すなわち探索方略に個人差があると仮定し解析を行った。

●実験方法

被験者は、VR(FOVE0)を装着し VR 空間内でパネル(Search Panel)に提示された計 24 個の刺激項目を１０秒間記憶し、その後切り替わったパネル(Target Panel)に提示される目標項目の有無を 3 秒間以内に口頭で答える。これに Break Panel 2 秒間を加えるセットを計 10 セット行い、視線の位置情報と時間を測定した。

●実験結果

実験の結果から、視線重心、クラスター解析から被験者１３人に共通して視線分布率は右上領域が高く、左下領域が低いことがわかった。また、視線のばらつきを表すエントロピーと正答率の相関関係はみられないが、エントロピーと視線重心の角度または傾きは正の相関を持つことがわかった。

さらに、視線の位置座標と時間からベクトル解析、眼球運動速度解析、座標速度分布解析を行った結果、探索方略が 4 つのモデルに収束することがわかった。人間の眼球は 1 点を見つめている間にも微小運動を行なっており、振幅と最大速度の関係が通常のサッカードと類似している運動をマイクロサッカードと呼ぶ。本研究の眼球運動速度解析からもマイクロサッカードを見ることができ、測定データの正確性とモデルによってマイクロサッカードの出現頻度にばらつきがあることが確認できた。

●考察

視線分布率に偏りがある結果から、エントロピーと正答率の関連性は見られなかったが、新たなアプローチとして周辺視野能力と呼ばれる広い視野を持ちながら情報を脳で処理することができる能力と正答率に関連性があるのではないかと考える。注視する点を中心視と呼ぶが、注視点となる重要情報を初めから捉えることはまずあり得ない。よって周辺視によって情報を選択し、重要情報である注視点を選択するメカニズムから、本実験における正答率とエントロピーの関連性を得る新たな知覚実験を行う必要性があると考える。

また、4つに収束したモデルは被験者数を増やすこと、CPUの処理能力を上げることで測定から得られる情報量を増やし、各モデルにおけるマイクロサッケードの出現頻度とは別に本研究で明らかにならなかった規則性が見られるのではないかと考える。

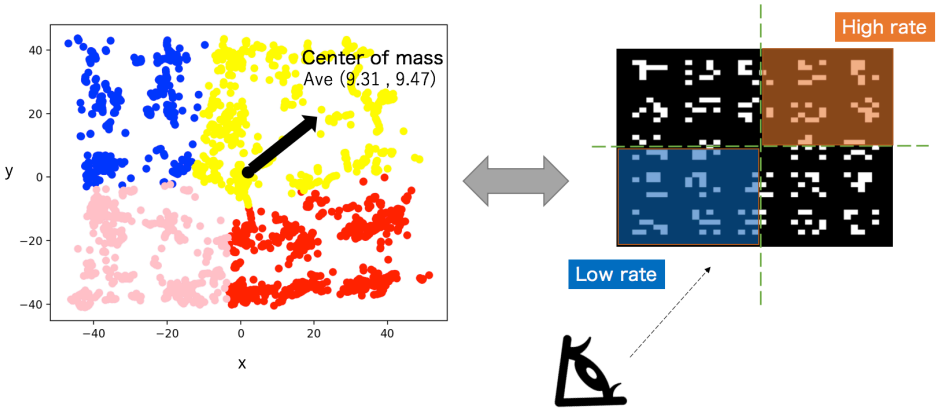


図1.クラスター、視線重心解析から得た全被験者に共通する視線傾向
Center of mass=ave(9.31,9.47)

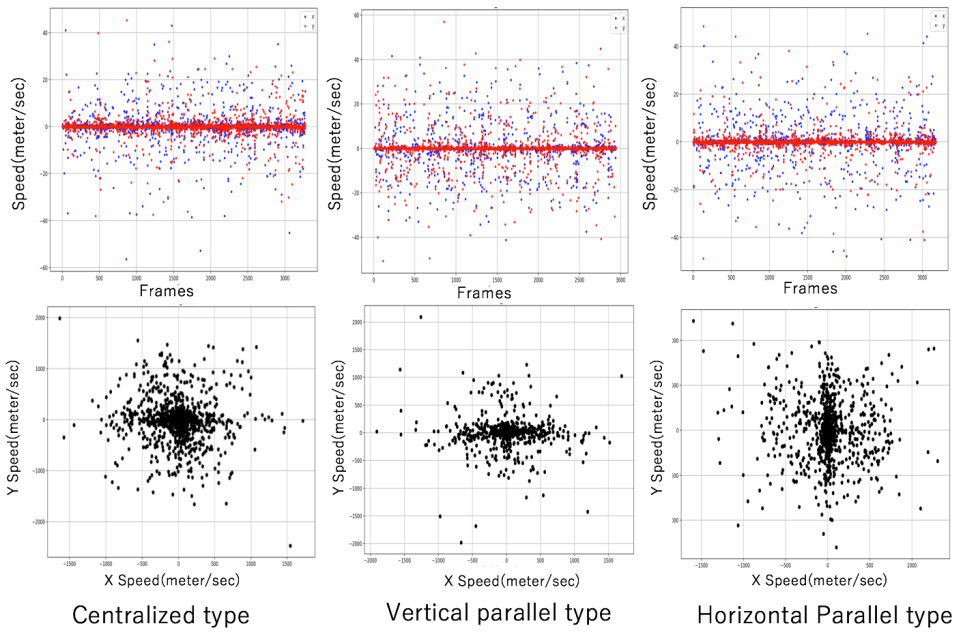


図2.上　眼球運動速度解析図
下　座標速度分布解析図

2019 年度　卒業論文

視覚探索での眼球運動における探索方策に関する研究

所属　国際総合科学部国際総合科学科理学系
コース　物質科学コース
指導教員指名　ミケレット・ルジェロ
学籍番号　140583
氏名　前川　黎

目次

第1章　　はじめに

1.1　　研究背景

普段私たちのように目を持っている人間、動物は寝ている時を除いて視覚に依存している。外界から刺激の多い私たちは身の危険を避けるため、障害物を避ける時や遠近の状況を把握するために視覚を用いている。目の見えない人または動物は、周りの状況を把握できずに身の危険に晒されることが簡単に想像できる。そもそも視覚からの情報が遮断された世界では、“注意”をすることが困難である。

しかし、そのような当たり前に感じる事実であっても視覚の重要性に対して具体的数値を挙げる一般書を始めとした根拠となる文献、専門書はほとんどない。また、科学的根拠に基づき記述されるべき教科書にも明示されない記述が多い。この事実からわかることは、視覚について明らかにする以前に情報を処理する脳について未だに明らかになっていない部分が多いことに由来する。そこで視覚について科学的根拠を解明する研究をするには、学生では時間、費用が圧倒的に不足している。しかし、視覚の実験結果に限定した法則性または、規則性を見つけることは可能ではないだろうか。そういった学問は心理実験として過去の研究者たちが結果を残している。近年では、スマートフォンのアプリ、PC の画面を一つとっても過去の研究の結果が用いられていることから視覚に関して心理学を学んでいく重要性に気づくことができる。

日々の生活の中で、“見落とし”をすることは誰にでも起こりうることから原因が視覚的注意不足であると考えられる。注意をある特定の領域に向けると、その領域の知覚は向上するが、他の領域の知覚は減少することがわかっている。このことから、システムに同じ機構を持つ VRHMD の研究が注目されている。

その一環として、私は人間が生活の中で視覚探索(visual search)を行っていることに注目した。視覚探索とは、視野内にある複数の対象から、色、形など特定の視覚的特徴を持つ対象を探し出すことを指す。わかりやすい例を挙げると、家を出る前に鍵を探すことも視覚探索の一例である。しかし、先程の例ではまず鍵を探す際に前日にどこに置いたかの記憶を元に探すのではないだろうか。
要するに、視覚探索の研究をする際に記憶について述べることが必須である。

我々人間を取り巻く環境は膨大な情報を持っており、全てを処理することは不可能である。よって人間が適切に行動をするためには膨大な情報の中から状況に応じて、目的の行動に関する情報を選択し、処理するメカニズムが必須となる。このメカニズムは”注意”という用語で表現される。

近年の研究では、人間の五感で情報を知覚する割合は視覚が 87%を占めると言われている。しかし、この視覚8割説[1]については、40年以上前の神経生理学の教科書に由来することがわかった。近年では、五感全てを活用、融合しながらリアリティーを再現するバーチャルリアルティー、感覚センサーもしくは感覚デバイスの開発が著しい。よって“注意”に関して視覚探索を行う際に視覚が大きく関係していることから、近年の開発によって有用になった視線追跡機能付き VR を用いてより定量的に注視点の研究を行った。

[1] 視覚8割説　加藤 宏『「視覚は人間の情報入力の 80%」説の来し方と行方』から引用

1.1.1 視覚探索

視覚探索とは、ディスプレイに複数の刺激項目（Search Panel）を提示し、その中にあらかじめ決められた目標項目があるか否かを被験者が判断する課題である。この視覚探索において一般的にSearch Panel が提示されてから、被験者が反応する探索時間が計測される。この課題をShiffrin(1998)の定義により厳密に述べると、あらかじめ被験者に、目標項目を1つ以上覚えてもらう。被験者は、その後提示される刺激項目（被験者に何らかの判断を求めるために提示される刺激）について、または標的の有無やその位置に関する判断を求められる。検出する目標項目の集合を記憶セットと呼び、その中の目標項目の数を記憶セットの大きさという。目標項目以外の刺激を妨害項目と呼ぶ。

視覚探索課題では、どのように刺激を実験的操作で与えるかにより記憶探索課題と視的探索課題に分類される。また、本研究では視的探索課題について着目している。視的探索課題について身近な例を挙げると”ウォーリーを探せ”[2]のような視覚探索を娯楽として作られたものなども存在する。某図書では、結合探索といった操作が必要となり図1のように探す特徴が多く目標項目を発見するまで時間がかかる。一方で目標項目を即座に見つけることができる視覚探索を特徴探索という。図2のように目標項目がポップアウトされ探索時間は短いことがわかる。

以上の2つの例の違いは、刺激項目をより増やした時に顕著に探索時間に差が生まれる点である。このメカニズムを特徴統合理論という。特徴探索では、刺激項目を増やしても探索時間は変わらないことから並列処理を行っている。一方で、結合探索では目標項目に対して刺激項目に情報が多いことから、刺激項目の増加に伴って探索時間が増加する逐次処理を行っている。

しかし、この理論では比較的単純な例でのみ適用が可能であり、対称性や目標項目の形、色、大きさなどの変化を加えると説明できないことが多く、今後のより発展した研究が必要不可欠である。

[2] https://ja.wikipedia.org/wiki/ウォーリーをさがせ!

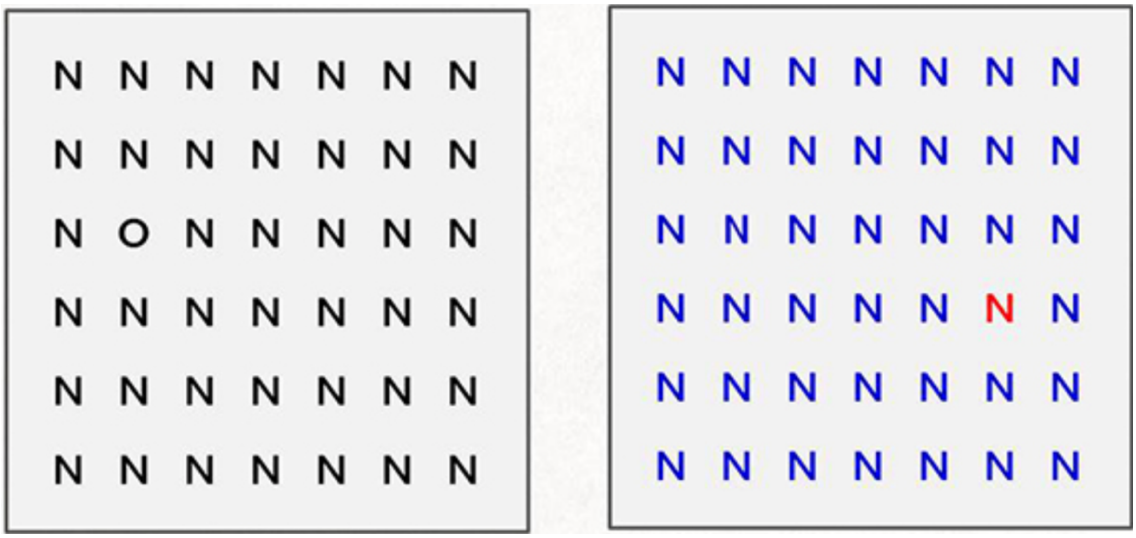


図 1　結合探索の例　https://psychmuseum.jp/show_room/visual_search/ から引用

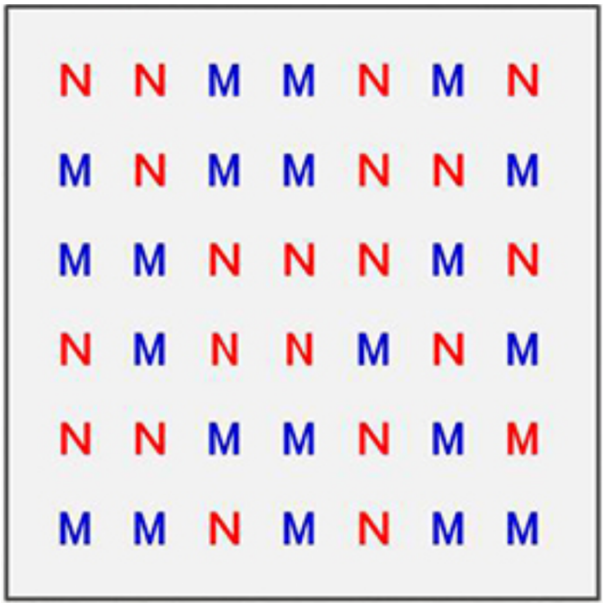


図 2　特徴探索の例（赤の M を探す）　https://psychmuseum.jp/show_room/visual_search/ から引用

1.2　　視覚探索における探索方略

1.2.1　　探索方略についての先行研究

日常生活において仮説を生成し、事例を生成するような行動には規則性が見られる。仮に、インターネットで検索をする際にキーワードを打ち込み、検索をする時キーワードを入れた段階で検索結果はある程度の予測がつく。同様のことが視覚でも言える。例えば目的の店に向かう際、看板を探すためにビルの前や上をみるように、今までの個人における経験に基づき行動を決定する。よってこのような探索方略は私たちにとって重要なものであることがわかる。

1.2.2　探索方略の先行研究内容

先行研究にて、本研究に近い実験操作から‘眼球運動を用いた仮説生成における探索方略の検討’について取り上げる。この研究では、図3のような試行を行う。この試行では、矢印，方位，数字の3つのオブジェクトの規則性を見つけることが目的である。各課題は、17 インチ、1280×1024 の解像度のモニタに表示された。参加者とモニタの距離は約 60cm に設定され、各パネルの大きさは 7.9°×7.9°であり、眼球運動は Tobii T60 eye tracker を用いて取得された。5 回の課題を通して、各課題を行う前に、眼球運動測定のためのキャリブレーションを行った。

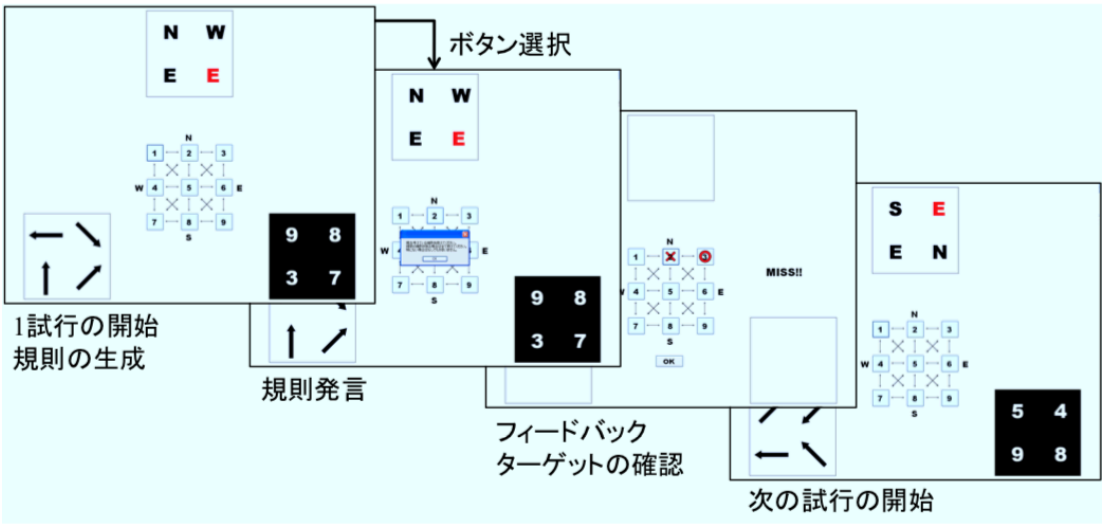


図 3　先行研究での試行の流れの一例

1.2.2.1 探索方略における先行研究解析

各試行における参加者の注視点の拡散を算出した。用いたデータは、各参加者の眼球運動から各パネルにおける注視時間の割合(p_i)を求め、エントロピーを算出した。

$$adjusted - H = -\frac{\sum p_i \cdot log p_i}{log 3} \cdots\cdots \text{式 } 1$$

この値から、エントロピーが大きいほど、複数のパネルに注視があったと言える。また、adjusted-H の値は 3 つのパネルに均等に注視点が拡散されているときに最も大きい値である 1 をとり、いずれか 1 つのパネルに注視点が集中しているときに最も小さい値 0 をとる。この結果は、表1のような相関をとり、被験者はある一定の探索方略を持っていることがわかった。

表 1　平均 adjusted-H の課題間の相関

https://www.jcss.gr.jp/meetings/JCSS2011/proceedings/pdf/JCSS2011_P3-16.pdf から引用

	easy 1	2	3	difficult 1	2
1	1.000				
2	.729***	1.000			
3	.838***	.661**	1.000		
1	.752***	.620**	.889***	1.000	
2	.636**	.668**	.809***	.896***	1.000

** : $p < .01$, *** : $p < .001$

続いて、規則発見率を比較した。注意拡散グループと注意集中グループの二つに視覚探索パターンをグループ分けした結果、注意集中グループの方が規則発見率は高かった。しかし、difficult phaseでは2つのグループに優位性は見られなかった。試行を通して、2つのグループの規則発見率を表2にまとめると以下のようになり、総合的に考えると、1 つの要因を仮定して，規則性を検討する注意集中グループの参加者の方がより規則を発見したことが支持される。

表 2 各グループにおける規則発見率

https://www.jcss.gr.jp/meetings/JCSS2011/proceedings/pdf/JCSS2011_P3-16.pdf から引用

	注意拡散				注意集中			
	矢印	方位	数字	全体	矢印	方位	数字	全体
easy	0.17 (2)	0.42 (5)	0.92 (11)	0.50 (18)	0.70 (7)	0.90 (9)	0.70 (7)	0.77 (23)
difficult	0.13 (1)	0.11 (1)	0.86 (6)	0.33 (8)	0.17 (1)	0.67 (4)	0.75 (6)	0.55 (11)
全体	0.15 (3)	0.29 (6)	0.89(17)	0.43 (26)	0.50 (8)	0.81 (13)	0.72 (13)	0.68 (34)

()内は規則発見者数.

よって，注意集中方略は一度に仮説空間の狭い範囲を探索している一方、注意拡散方略は一度に仮説空間の広い範囲を探索していると言える。先行研究では、一回の仮説生成で時間をかけて広い範囲を探索した注意拡散グループの参加者のほうが、規則発見率が低かった。この結果から、仮説生成の際に時間をかけて検討する際に、広い範囲を探索するより、ある程度の範囲に絞って探索する必要がある可能性を示している。

1.3　　研究目的

視線の注視点を理解、解明することは将来的な技術、日常生活、個々人のパーソナリティーの把握に貢献できる。仮に、スマートフォン、近未来型のフロントガラスに情報が出るような AR の技術が一般的になった際、普段見ることがない視界にあまり重要でない情報を提示することなどが可能になる。逆に、頻繁に視覚する領域には重要な情報を表示することで事故の危険性を除くような“見落とし”を防ぐなど有用性が増すなどのメリットが挙げられる。

現在視線追跡による研究は、調査実験などに用いる解析方法の一例としてあげられことが多いが、視覚探索課題を用いる実験操作をすることにより科学的根拠を述べることができる。本研究では、先行研究で行われた記憶セットを用いる中で、視覚探索中の視線モデルを作成し、より定量的な視覚情報を得ることを目的とした。

先行研究では、注視点の解析結果から探索方略における優位性を示すことはできていない。これは、目標項目であるオブジェクトが関連性を持たないことから思考パターンの解析としてデータを用いることは可能だが、注視点の規則性の解析には向いていない。よって本研究では、先行研究にて探索方略を注意集中方略、注意拡散方略に分けたことから、より定量的な視覚情報を用いることで探索方略のモデルを作成する。なお、解析の際にはフィードバックとして正答率を用いた。

第2章　　実験概要

2.1　　Unity について

本実験では、被験者が実際に体感する VR 空間を、ゲームエンジンとして凡用性の高い Unity3D 用いて作成した。Unity3D は主に PC ゲーム、スマートフォンアプリなどの開発をするために用いられることが多く、アセットが豊富に用意されていることから、プログラミング初心者であっても容易に VR 空間を作成することが可能である。Unity3D においてスクリプトに用いるプログラミング言語の特徴としてマイクロソフトが提供している C#または、JavaScript 等が用いられている。本実験では、VR 環境を C#言語のみを用いてプログラミングを行った。

Unity3D は本実験で用いる FOVE0と互換性があり、視覚探索を行っている被験者の視線データをプログラミングで管理することが可能である。他の互換性のある VR 装置としては Oculus Rift 等を挙げることができる。また、環境開発の中でターゲットになるデータを自動化し保存することができるプログラミングをすることが可能であり、解析を行う際の労力を軽減することができる点も Unity3D を用いる大きなメリットである。

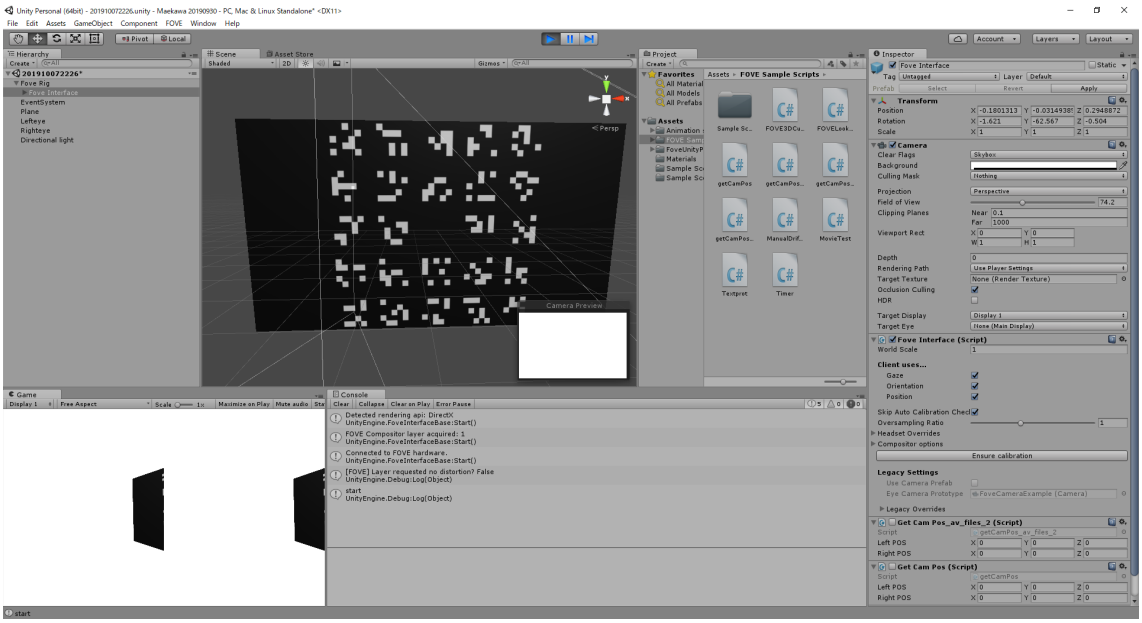


図 4 Unity 制御画面

Imported Object

getCamPos_av_files_2

```
using System.Collections;
using System.IO;
using System.Collections.Generic;
using UnityEngine;

public class getCamPos_av_files_2: MonoBehaviour
{

   string FILE_NAME = "posCam_file_oikawa";
   //int time = 0;
   int interval = 1;
   // public Vector3 objPOS; public Vector3 objROT;
   public Vector3 leftPOS;
   public Vector3 rightPOS;
   private float elapsedTime;
   private bool counter_flag = false;
   // private float tzero;
   private float avEyeX, avEyeY;
   private int caseF=-99;
   /*private StreamWriter sw0;
   private StreamWriter sw1;
   private StreamWriter sw2;
   private StreamWriter sw3;
   private StreamWriter sw4;
   private StreamWriter sw5;
   private StreamWriter sw6;
   private StreamWriter sw7;
   private StreamWriter sw8;
   private StreamWriter sw9;
   private StreamWriter sw10;
   private StreamWriter sw11;
   private StreamWriter sw12;
   private StreamWriter sw13;
   private StreamWriter sw14;
   private StreamWriter sw15;
   private StreamWriter sw16;
   private StreamWriter sw17;
   private StreamWriter sw18;
   private StreamWriter sw19;
   private StreamWriter sw20;
   */
   // Use this for initialization
   void Start()
   {
      Debug.Log("Start Camera script");
      string dateTime = System.DateTime.Now.ToString();
       StreamWriter sw0 = File.AppendText(FILE_NAME + "0.txt");
       /*StreamWriter sw1 = File.AppendText(FILE_NAME + "1.txt");
       StreamWriter sw2 = File.AppendText(FILE_NAME + "2.txt");
       StreamWriter sw3 = File.AppendText(FILE_NAME + "3.txt");
       StreamWriter sw4 = File.AppendText(FILE_NAME + "4.txt");
       StreamWriter sw5 = File.AppendText(FILE_NAME + "5.txt");
       StreamWriter sw6 = File.AppendText(FILE_NAME + "6.txt");
       StreamWriter sw7 = File.AppendText(FILE_NAME + "7.txt");
       StreamWriter sw8 = File.AppendText(FILE_NAME + "8.txt");
       StreamWriter sw9 = File.AppendText(FILE_NAME + "9.txt");
       StreamWriter sw10 = File.AppendText(FILE_NAME + "10.txt");
       StreamWriter sw11 = File.AppendText(FILE_NAME + "11.txt");
       StreamWriter sw12 = File.AppendText(FILE_NAME + "12.txt");
       StreamWriter sw13 = File.AppendText(FILE_NAME + "13.txt");
       StreamWriter sw14 = File.AppendText(FILE_NAME + "14.txt");
       StreamWriter sw15 = File.AppendText(FILE_NAME + "15.txt");
       StreamWriter sw16 = File.AppendText(FILE_NAME + "16.txt");
       StreamWriter sw17 = File.AppendText(FILE_NAME + "17.txt");
```

図 5 Search Panel データ保存自動化プログラム 1

```
Inspector
      StreamWriter sw17 = File.AppendText(FILE_NAME + "17.txt");
      StreamWriter sw18 = File.AppendText(FILE_NAME + "18.txt");
      StreamWriter sw19 = File.AppendText(FILE_NAME + "19.txt");
      */
     sw0.WriteLine(" **** New Trial: " + dateTime + " ****");
     sw0.WriteLine("SUBJECT NAME:");
     sw0.WriteLine("SCRIPT TYPE:");
     sw0.Close ();

     //heads = GameObject.FindWithTag
(["MainCamera"]);
   }

   // Update is called once per frame
   void Update()
   {
      caseF = -99;
      var lEye = GameObject.Find("Lefteye");
      var rEye = GameObject.Find("Righteye");
      //StreamWriter sw1 = File.AppendText(FILE_NAME);
      //sw1.WriteLine(" **** UPDATE: ");
      //sw1.Close();

      if (Input.GetKey(KeyCode.S))
      {
         counter_flag = !counter_flag;
         elapsedTime = 0;

      }

      if (counter_flag == true)

      //(time > interval)
      {// "this" の意味はこのもの!

         elapsedTime += Time.deltaTime;

         if (elapsedTime > 3 && elapsedTime < 13)
         { caseF = 0; }
         if (elapsedTime >= 18 && elapsedTime < 28)
         { caseF = 1; }
         if (elapsedTime >= 33 && elapsedTime < 43)
         { caseF = 2; }
         if (elapsedTime >= 48 && elapsedTime < 58)
         { caseF = 3; }
         if (elapsedTime >= 63 && elapsedTime < 73)
         { caseF = 4; }
         if (elapsedTime >= 78 && elapsedTime < 88)
         { caseF = 5; }
         if (elapsedTime >= 93 && elapsedTime < 103)
         { caseF = 6; }
         if (elapsedTime >= 108 && elapsedTime < 118)
         { caseF = 7; }
         if (elapsedTime >= 123 && elapsedTime < 133)
         { caseF = 8; }
         if (elapsedTime >= 138 && elapsedTime < 148)
         { caseF = 9; }
         //if (elapsedTime >= 54 && elapsedTime < 57)
         //{ caseF = 10; }
         //if (elapsedTime >= 57 && elapsedTime < 62)
         //{ caseF = 11; }
         //if (elapsedTime >= 64 && elapsedTime < 67)
         //{ caseF = 12; }
         //if (elapsedTime >= 67 && elapsedTime < 72)
         //{ caseF = 13; }
         //if (elapsedTime >= 74 && elapsedTime < 77)
         //{ caseF = 14; }
         //if (elapsedTime >= 77 && elapsedTime < 82)
         //{ caseF = 15; }
```

図 6　Search Panel データ保存自動化プログラム 2

2.2 VR を用いるための環境構築

本実験では視線追跡(eye tracking)を行うため、FOVE0 を用いた。FOVE0の大きな特徴は、現在スタンダードである画面全体にピントを合わせる VR HMD と異なり、赤外線アイトラッキングシステムを用いることで目の開閉状況、まばたき、片目つぶり、奥行き情報を含んだ視線まで把握できる点にある。

本実験では、各被験者にキャリブレーションを事前に行うことでまばたき、頭の角度(x , y , z)、左目と右目の座標(x , y , z)の誤差を減らしている。

また、フレームレートの数が解析においてより詳しいデータを得るために重要であるが、フォービエイテッドレンダリングにより注視点以外の解像度を落とす技術から Unity と同時起動であっても本実験では、FPS30での測定が可能になった。

また、実験環境として被験者は図8のような仮想現実の世界に入り、実験パネルの動画にしたがって実験を行う。実験パネルの中心座標は(0,0,0)をとり x ,y 座標において 90×90units のスケールを持っている。被験者、対象物の配置は測定の際なるべく頭を動かさなくてもいいようなスケールと距離に調整してある。Unity 上では、座標の定義がメートル単位で調整されているが頭の角度により常に縮尺は変化しているため本研究では units を単位として用いる。

解析の際に用いるデータとして、被験者の視線は実験パネルの外に出るとエラーとして出るようにプログラミングされている。よって被験者は実験パネルにのみ注目し実験を行う。VR 空間の照明に関しては No Shader でオブジェクトに均一に光が当たるように設定し配置している。

実験パネルには、動画を制御するスクリプト、視線データの出力を行うスクリプトの制御を一括で行うようにプログラミングしたため、初期状態では図9のようにグレーの Plane として表示されている。

なお、本実験は Unity 上の時間によって進められるため、動画に設定された時間が経過すると実験 Panel の画面が切り替わるようとなっている。スタートに関してはインターフェイスとしてキーボードにて操作が可能なため、被験者の準備が整い次第測定を開始する。後に動画の構成、タイムスケジュールについて述べる。アイトラッキングを用いた分析では、一般的に観測したデータから注視に関する情報を抽出する。眼球の角度、視点が1フレーム毎に一定の領域に留まっているとするときその眼球運動を注視とみなす。

本研究では、それぞれの目の視点の中間点を視点とし、1/30FPS 毎の視点を注視点と定義した。

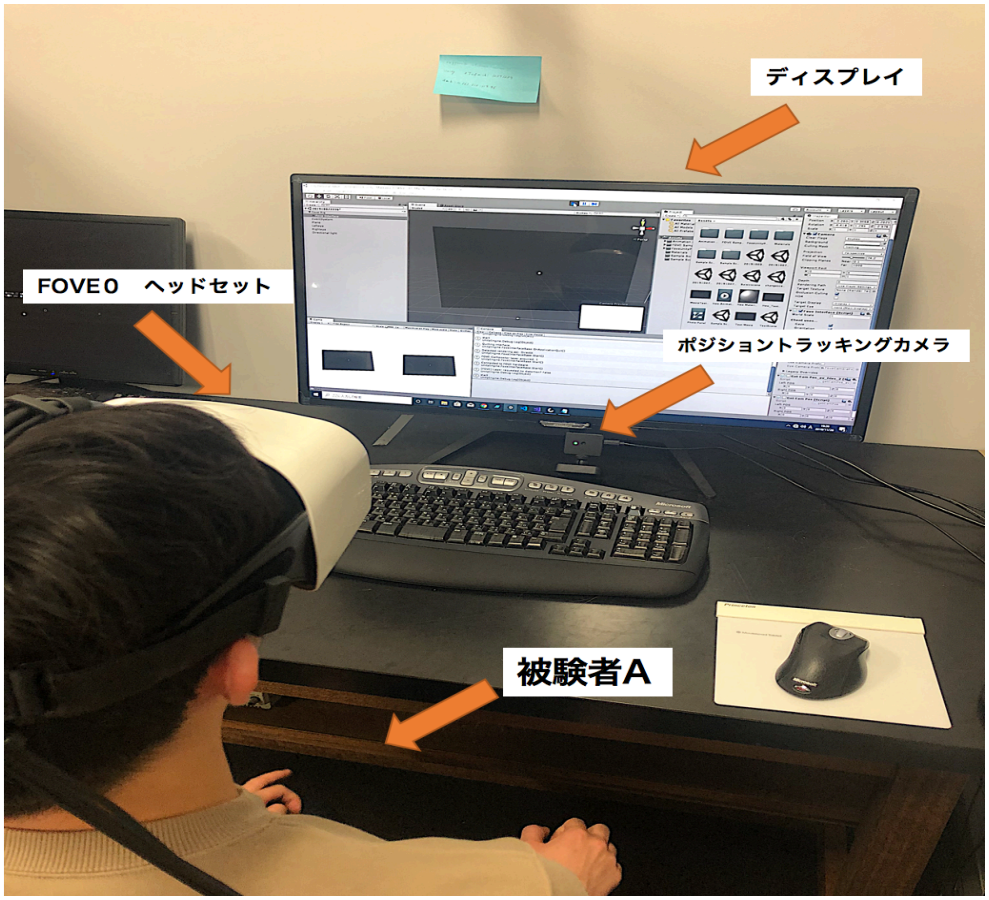


図 7　被験者の様子（被験者は図の様に実験を行う）

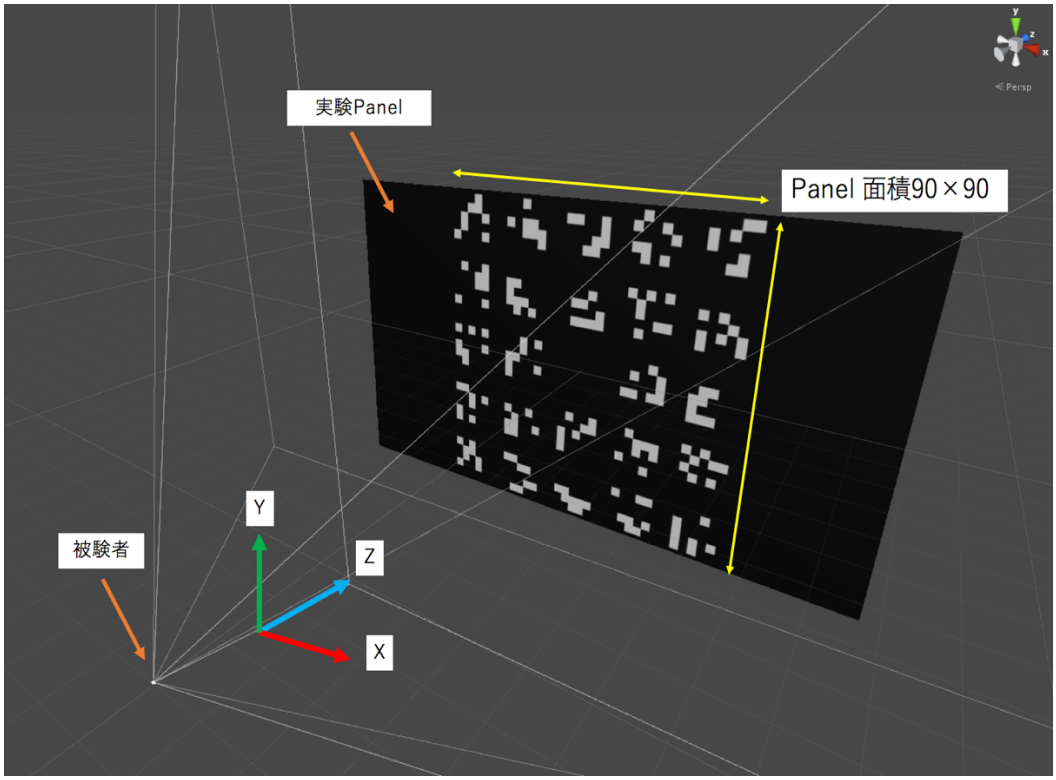


図 8　実験環境

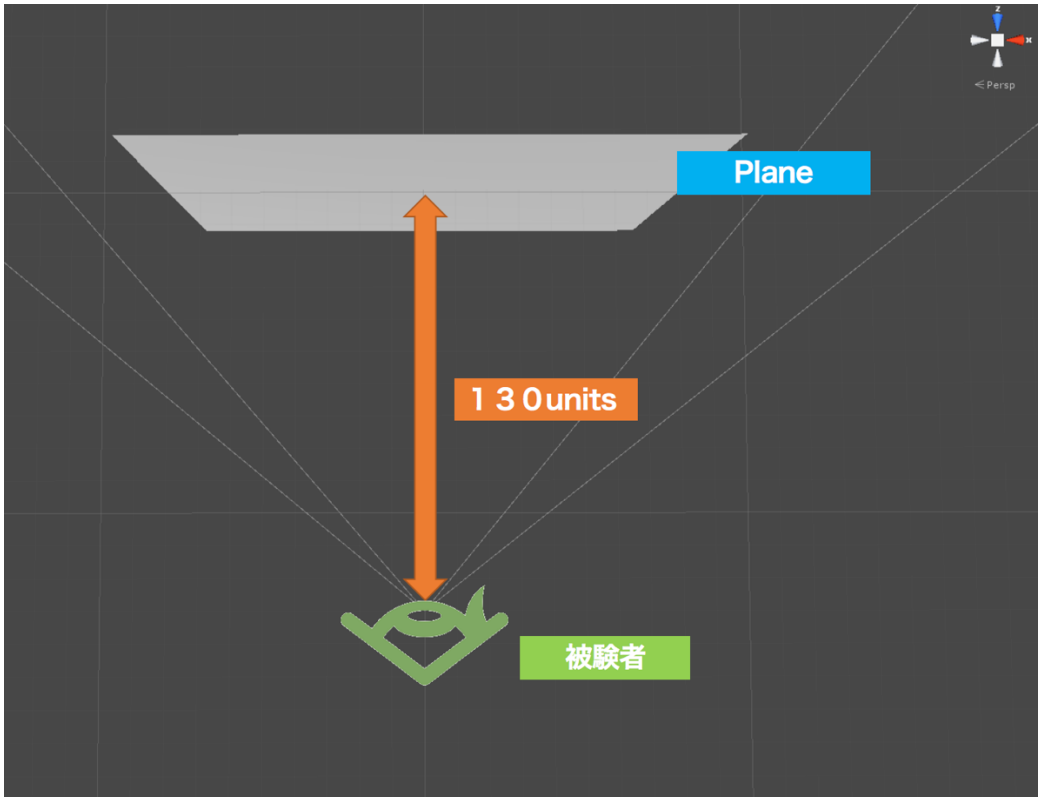


図 9　被験者頭上から見た実験環境

2.3 実験方法

2.3.1 視覚探索課題動画の作成

本実験では、視覚探索課題を動画として Python にて作成した。動画の構成は、Search Panel として、24 個のランダムに作成された定量的な図形が10秒間提示される。次に目標項目である Target Panel の図形が3秒間提示される。その後 Break Panel を2秒間提示するような 1set15秒の記憶セットを用いた。

本実験では 10set150 秒の視覚探索課題を行い、視線追跡機能付き VR にて右目と左目のそれぞれの座標、FPS30の条件で時間を計測した。なお、実験の際、被験者が体感する VR 空間では作成した動画は Panel として、図10のようなスケールを持ったスクリーンで表示される。Panel は 5×5 の黒と白のランダムな図形を生成し定量的な刺激項目として評価される。

動画は表3のように進行され、解析の際、Search Phase のデータを用いた。

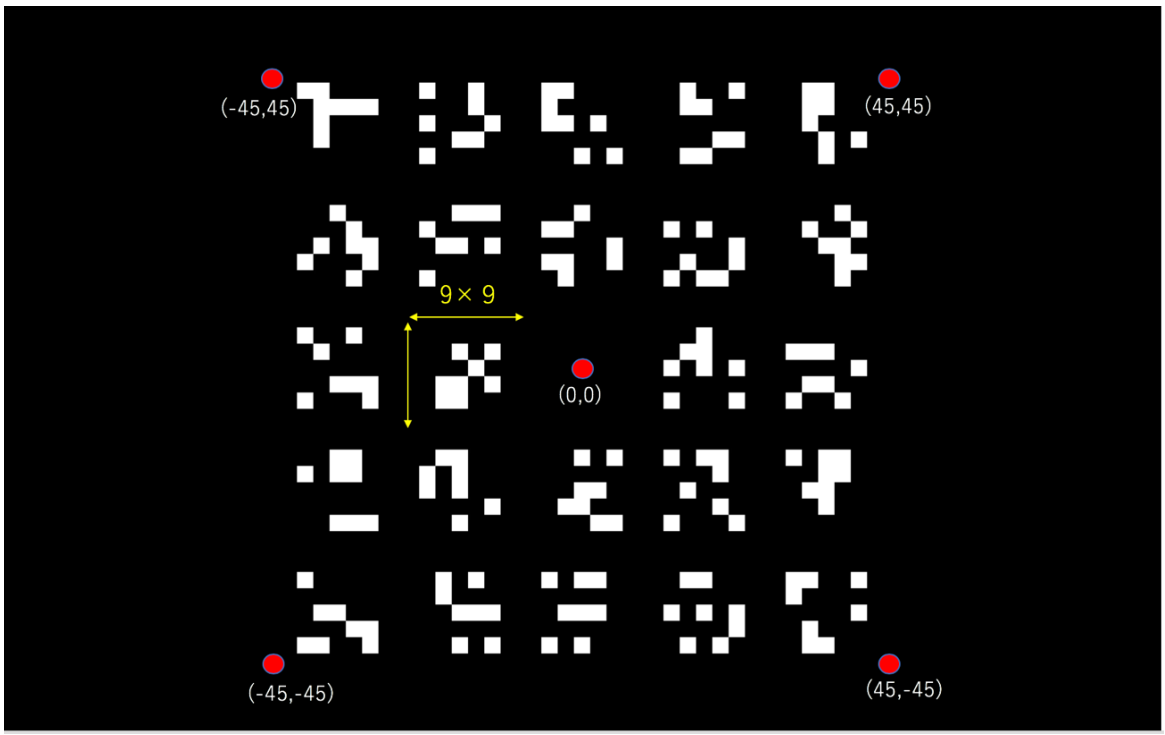


図 10 動画における Panel スクリーンのスケール

表 3 動画構成 Time Schedule

00:03~00:13	search	10s	01:18~01:28	search	10s
00:13~00:16	target	3s	01:28~01:31	target	3s
00:16~00:18	break	2s	01:31~01:33	break	2s
00:18~00:28	search	10s	01:33~01:43	search	10s
00:28~00:31	target	3s	01:43~01:46	target	3s
00:31~00:33	break	2s	01:46~01:48	break	2s
00:33~00:43	search	10s	01:48~01:58	search	10s
00:43~00:46	target	3s	01:58~02:01	target	3s
00:46~00:48	break	2s	02:01~02:03	break	2s
00:48~00:58	search	10s	02:03~02:13	search	10s
00:58~01:01	target	3s	02:13~02:16	target	3s
01:01~01:03	break	2s	02:16~02:18	break	2s
01:03~01:13	search	10s	02:18~02:28	search	10s
01:13~01:16	target	3s	02:28~02:31	target	3s
01:16~01:18	break	2s	02:31~02:33	break	2s

2.3.2　　視線情報、正答率を得るための測定方法

被験者は健康な男女１３人から実験を行なった。まず被験者は、実験説明書（図１１）を読み FOVE0 にてプロファイルを作成し、キャリブレーションを行った。その後、Unity 上にて動作確認テストを行い本実験に移った。

その際測定者は、キーボードの S ボタンにて動画の開始、自動保存ファイルプログラムを起動させる。実験中、被験者は Target Panel にて提示されるフェイズの図形の有無を Search Panel の記憶から答える。実験を継続し、10set 終了した地点で測定を終了した。自動保存されたファイルは 1set 毎の時間 10s で区切られ 1s 毎のデータ数は FPS に依存する。データの形は、Time, ave X , ave Y の要素を持った CSV ファイルとして出力される。

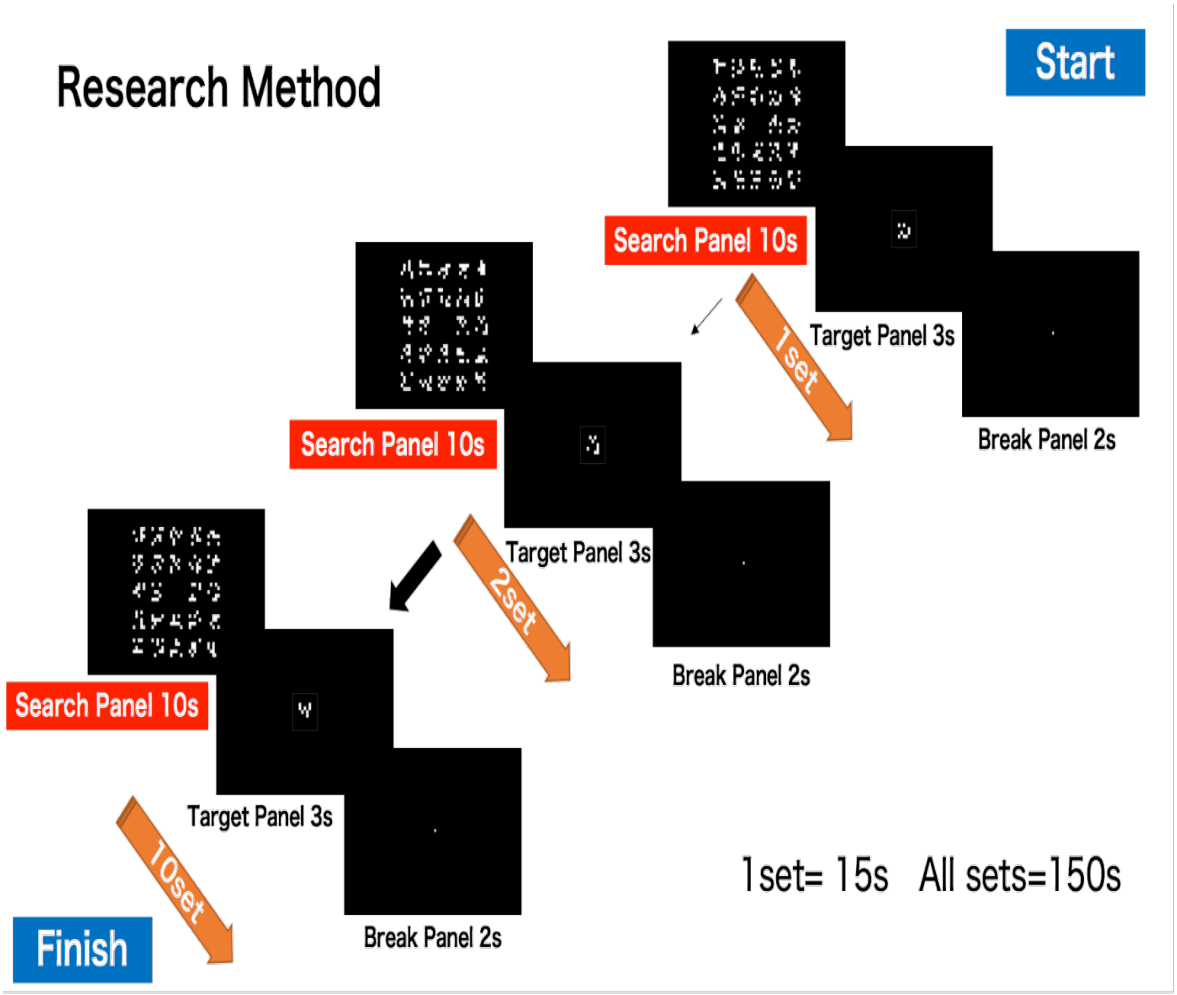


図 11　実験説明書図

第3章　　結果考察

3.1　　ヒストグラムと視線情報の結果考察

測定によって得られた視線情報から、Python プログラムを用いて解析を行なった。 測定に用いるデータは、被験者別に aveX , aveY を用いた。視線情報を x 座標=aveX, Y 座標=aveY にて被験者13人毎に図12、図13、図14のようにプロットした結果、被験者によって分布に偏りが見られる。これを統計学的に分析する手法としてヒストグラムの作成を 2D 解析 3D 解析と分けて行なった。なおヒストグラム作成の際に用いられる式は

$$n = \sum_{i=0}^{k} m_i \cdots\cdots \text{式 } 2$$

で計算される。n は全データ数、k は bins 数、m_i はヒストグラムである。本実験では、各要素が互いに素である区間である bins によって分けられ、実験に用いた Plate に対応させる際 bin=5 で作図を行う。今回の解析ではエントロピーをより正確に評価するため、bin=20 にて解析を行なった。2D ヒストグラムの解析結果の図は図15、図16、図17のようになる。

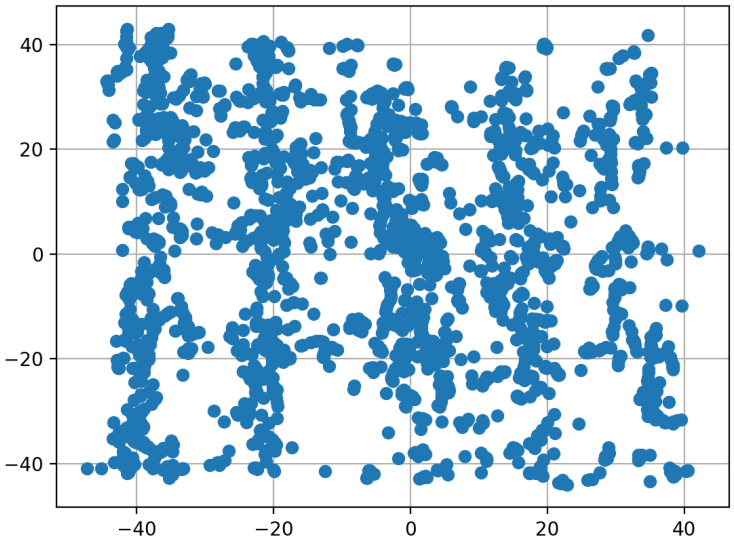


図 12　視線情報の散布図　1

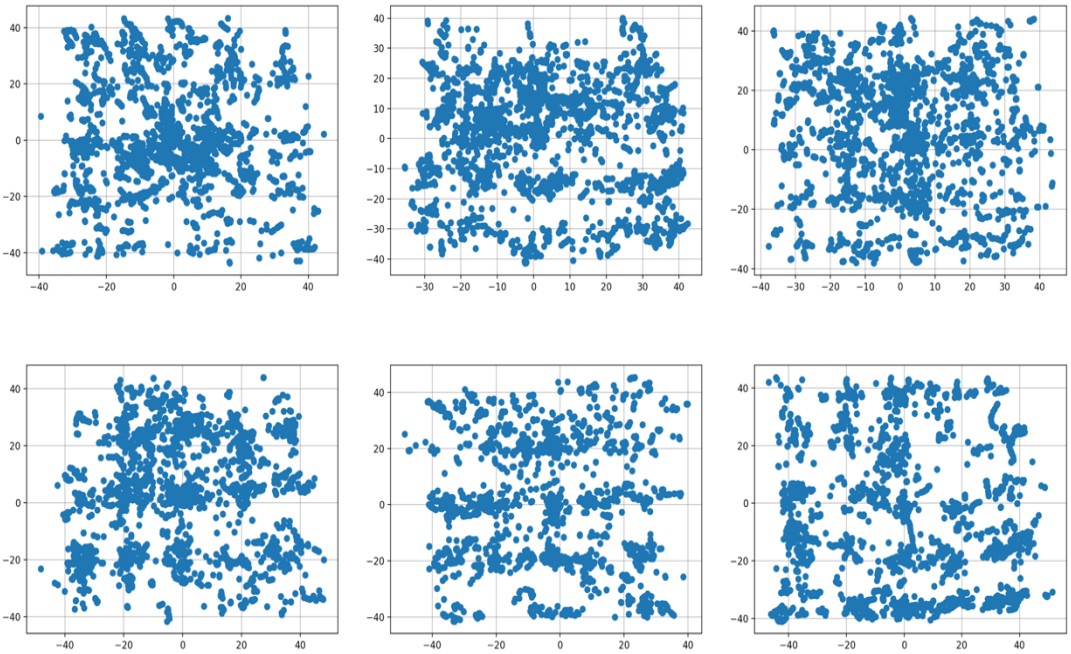


図 13 視線情報の散布図 2

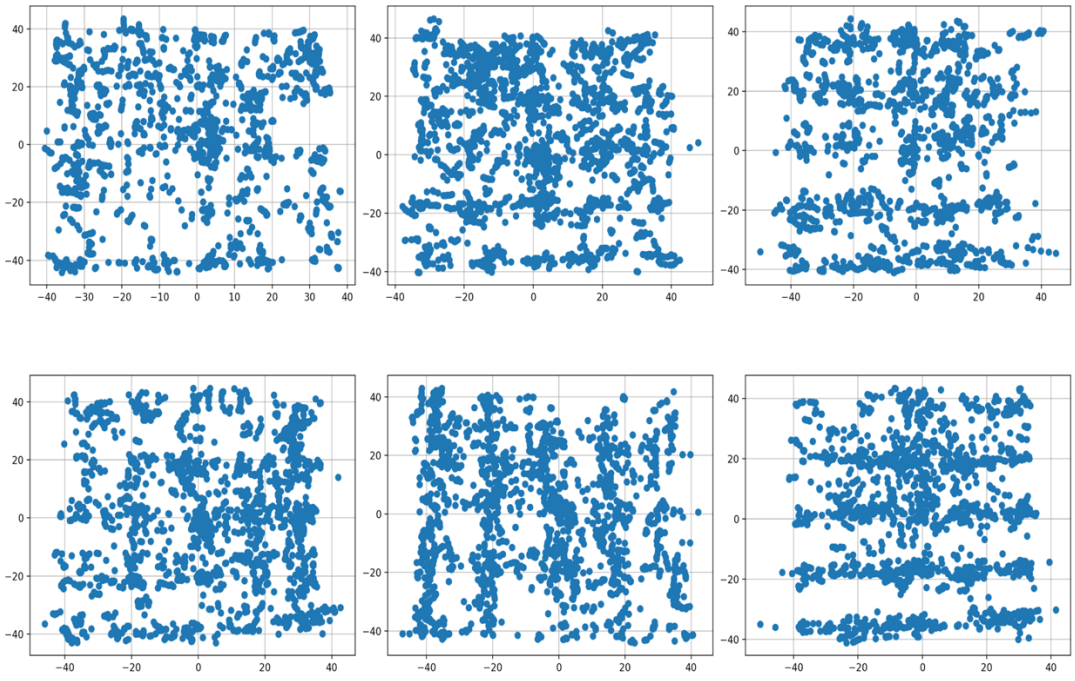


図 14 視線情報の散布図 3

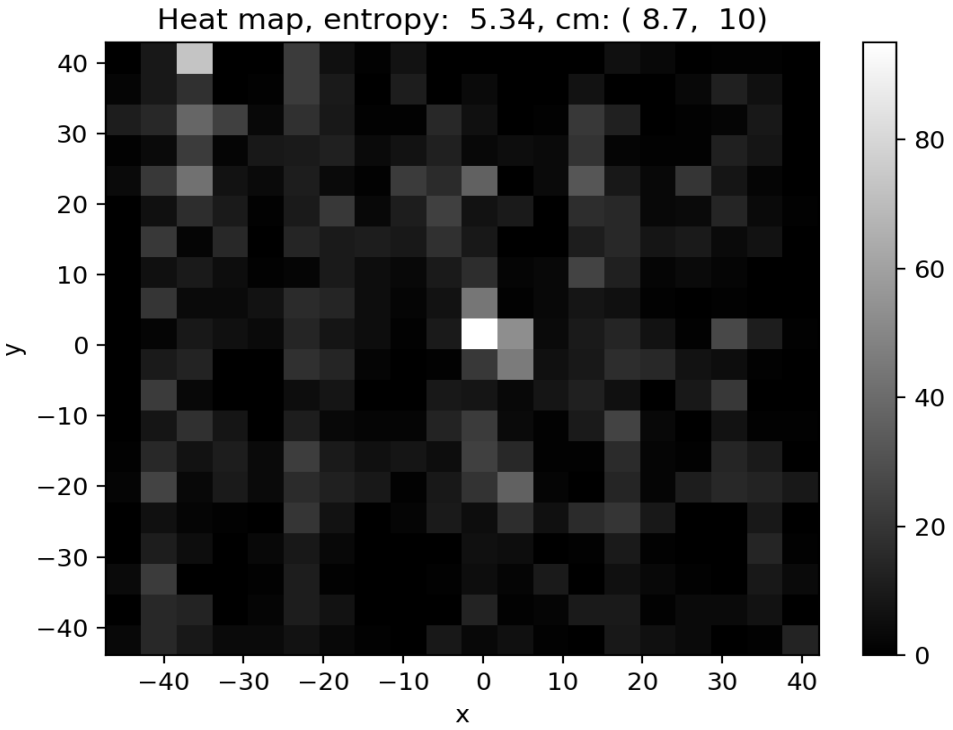


図 15 視線情報の2D ヒストグラム 1

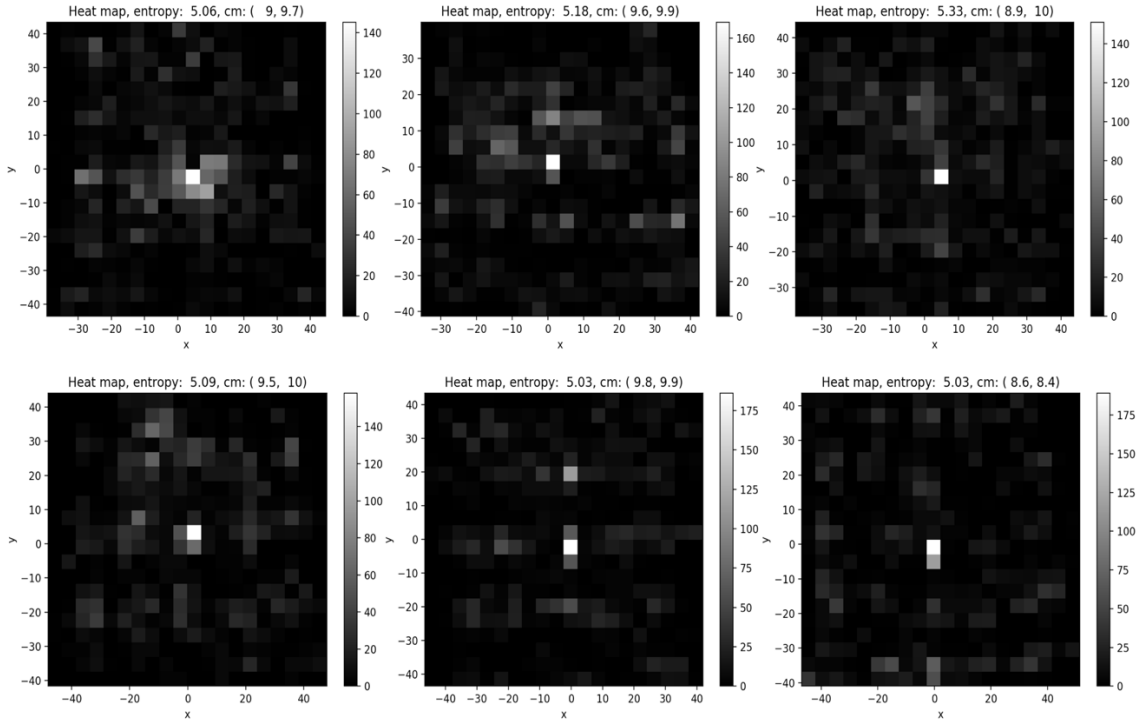


図 16 視線情報の2D ヒストグラム 2

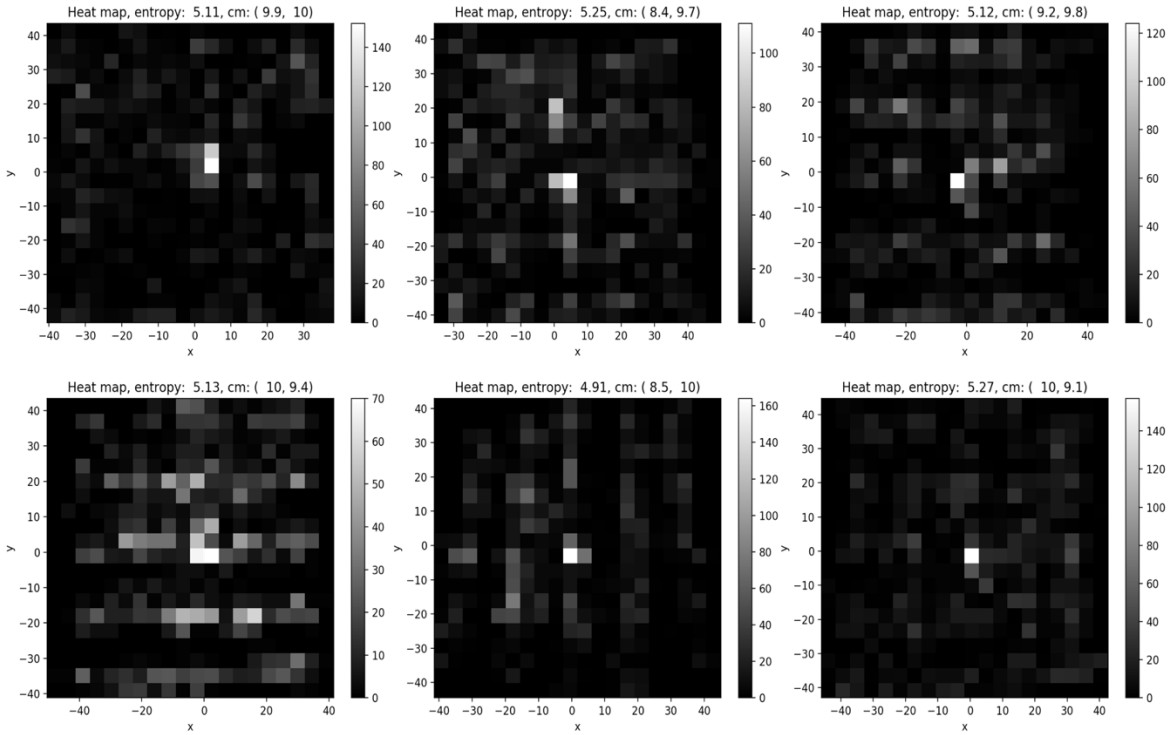


図 17 視線情報の2D ヒストグラム 3

ヒストグラムの結果から、視線領域の集まる分布に個人差が見られる結果となった。共通した傾向として、中心付近に最も高い分布が見られることは、Break Panel で中心点をおいたことによると考えられる。しかし、画面が Search Panel に切り替わる瞬間に視線が集まることは当たり前に考えられるが、測定全体を通してターゲットを設置していない中心に視線が集まる原因としては考えられない。このことは、2D ヒストグラムを被験者別に解析することで、中心以外で高い分布領域に形が見えてくる。本実験の目的である探索方略のモデル作成としてヒストグラムの形から先行研究で取り上げられた注意集中方略、注意拡散方略に加えた探索方略モデルを統計学的、心理学的に後の節で解析していく。

3.1.1　シャノンエントロピー

視線データのばらつきは被験者の探索方略に関係していると考え、この系におけるシャノンエントロピーを求めた。ここでは被験者における視覚探索中のデータを用いるため、統計的な意味を持つ平均シャノンエントロピー[3]について述べる。被験者毎に N 個からなる aveX, aveY=$\{x_1,x_2,x_3,\cdots,x_n\}(n\leq 0)$, $\{y_1,y_2,y_3,\cdots,y_n\}(n\leq 0)$とその行列に起因する確率$p=\{p_1,p_2,p_3,\ldots,p_n\}(\sum p_k=1,\ p_k\geq 0)$が(aveX or aveY, p)を完全事象系とするとき

$$S(p)=-\sum p_k log p_k \cdots\cdots \text{式 } 3$$

で与えられる。$S(p)$は、系のもつ不確かさ、乱雑さ、曖昧さなどを表す量であり、本実験における目線のばらつき度合いを表すシャノンエントロピーと呼ばれるものである。これを被験者毎に解析し、表4、図18に示した。

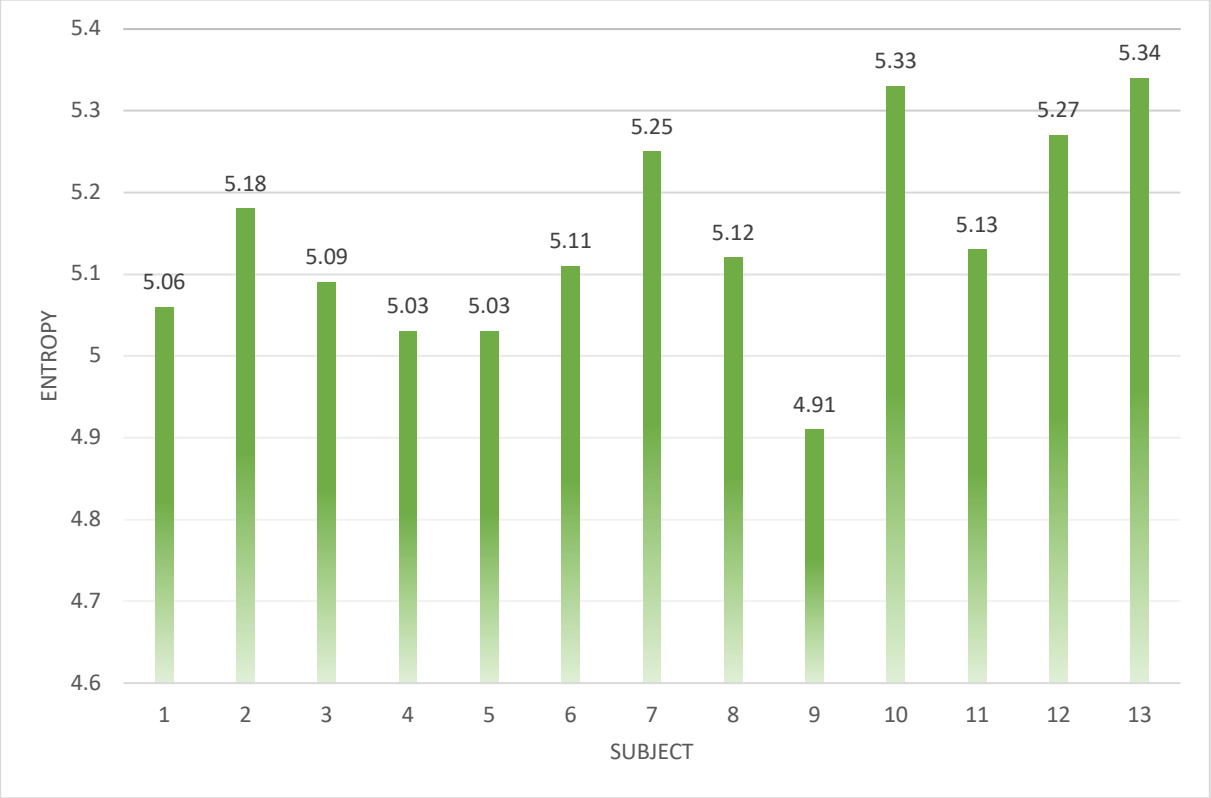


図 18　シャノンエントロピー比較

[3] https://whyitsso.net/math/statistics/information_entropy.html から引用

表 4 エントロピーと正答率の被験者13人の表

subject	entropy	correct answer rate
1	5.06	80%
2	5.18	40%
3	5.09	50%
4	5.03	30%
5	5.03	50%
6	5.11	50%
7	5.25	50%
8	5.12	30%
9	4.91	30%
10	5.33	20%
11	5.13	40%
12	5.27	50%
13	5.34	50%

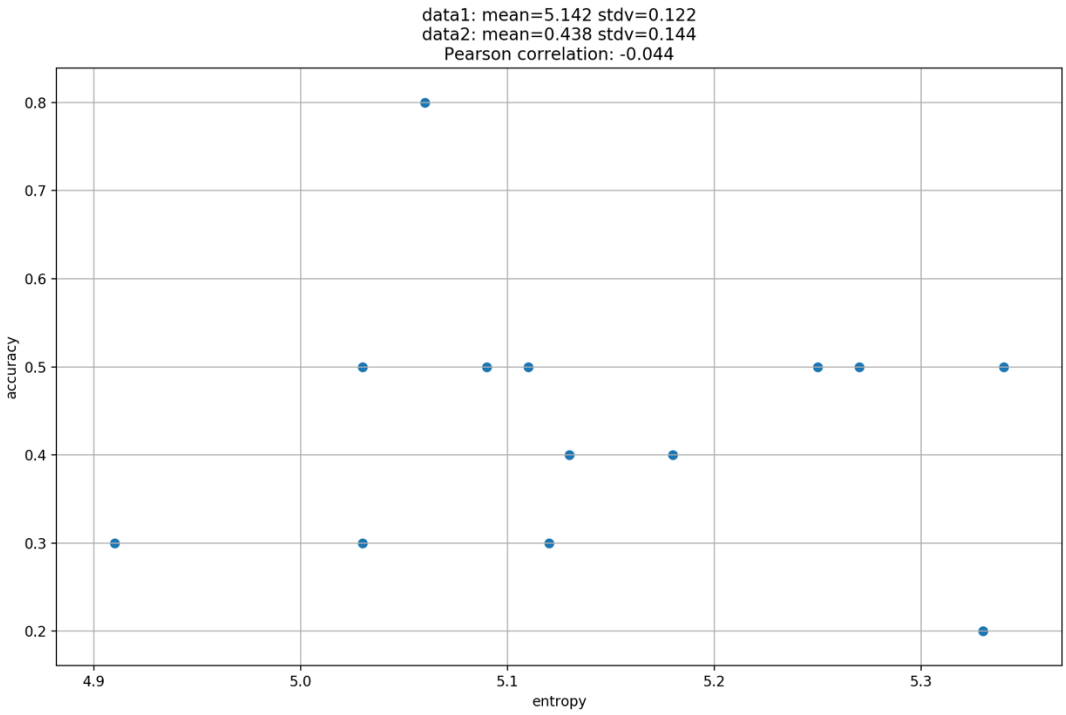


図 19 エントロピーと正答率の被験者13人の表の散布図

図18では、エントロピーに個人差が見られる。これは、視覚探索の際の視線の拡散度合を示すため、24個の刺激項目であるターゲットを覚える際の視野の広さに関係すると考える。表4から正答率とエントロピーの関連性を分析すると、エントロピーが最も高い被験者の値は 5.34 で、正答率は 50%である。逆にエントロピーの最も低い被験者の値は 4.91 で、正答率は 30%である。正答率の平均は44%でエントロピーの高い被験者は、24個のターゲットを満遍なく見ていると言えるが、他の被験者の結果から高いエントロピーを持っていても正答率に直接関連性があるとは言えない。また、図 19 から相関係数を求めると-0.04 と相関関係は見られなかったことから独立した結果であることが言える。

以上の結果から、視線の重心(Center of mass)を解析し、視線のばらつきの持つ偏りを分析する。

3.1.2 Center of mass

2D ヒストグラムの結果から正答率とエントロピーの関連性がみられなかった。このことから、別の視点として重心(Center of mass)の解析を行なった。重心の解析をすることで被験者別のエントロピーの高い領域の方向、言い換えると見やすい領域方向がわかる。なお解析は、Python プログラムによって行なった。本実験のプログラムでは、

from scipy.ndimage.measurements import center_of_mass[4] から、center_of_mass のライブラリーをインポートし、com=center_of_mass(H[0]) とすることで視線情報の重心を計算した。H(0)は 2D ヒストグラムの解析の際に aveX,aveY の行列を用いていたことから、2D ヒストグラム、散布図に対応したラベルの配列の重心を求めることができる。結果は表5のようになった。

Center of mass の解析結果から、視線情報の中心からの分布の偏りが明らかになった。重心の x 座標、y 座標の平均を計算すると(9.31 ， 9.47)のように右上の方向に偏りがあることがわかり、左下は見にくい、右上は見やすいという定量的な分析結果を述べることが可能である。この分析から、自動車のインターフェイス、スマートフォンなどのスクリーンの要素を配置する際に、右上に重要な項目を配置、左下に重要度が低い項目の配置をするようなことが、無意識下において人間の視線領域の情報処理、記憶選択が最も効率よく行われる配置であると言える。

統計学的に Center of mass の解析結果に、科学的根拠を示すため K-means 法を用いたクラスタリング解析を加えて本実験では解析を行なった。

[4] https://docs.scipy.org/doc/scipy/reference/generated/scipy.ndimage.center_of_mass.html を参照

表 5　被験者毎の重心(Center of mass)解析結果

subject	Center of mass
1	(9.9 , 7)
2	(9.6 , 9.9)
3	(9.5 , 10)
4	(9.8 , 9.9)
5	(8.6 , 8.4)
6	(9.9 , 10)
7	(8.4 , 9.7)
8	(9.2 , 9.8)
9	(8.5 , 10)
10	(8.9 , 10)
11	(10 , 9.4)
12	(10 , 9.1)
13	(8.7 , 10)

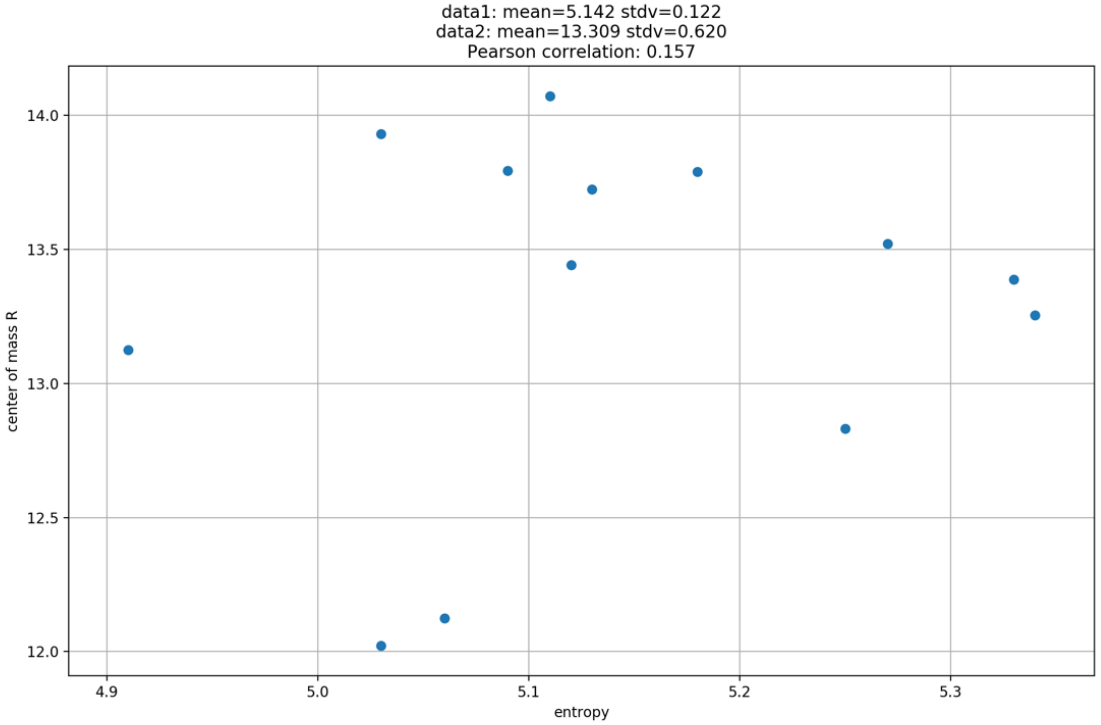


図 20　エントロピーと重心座標の R の散布図

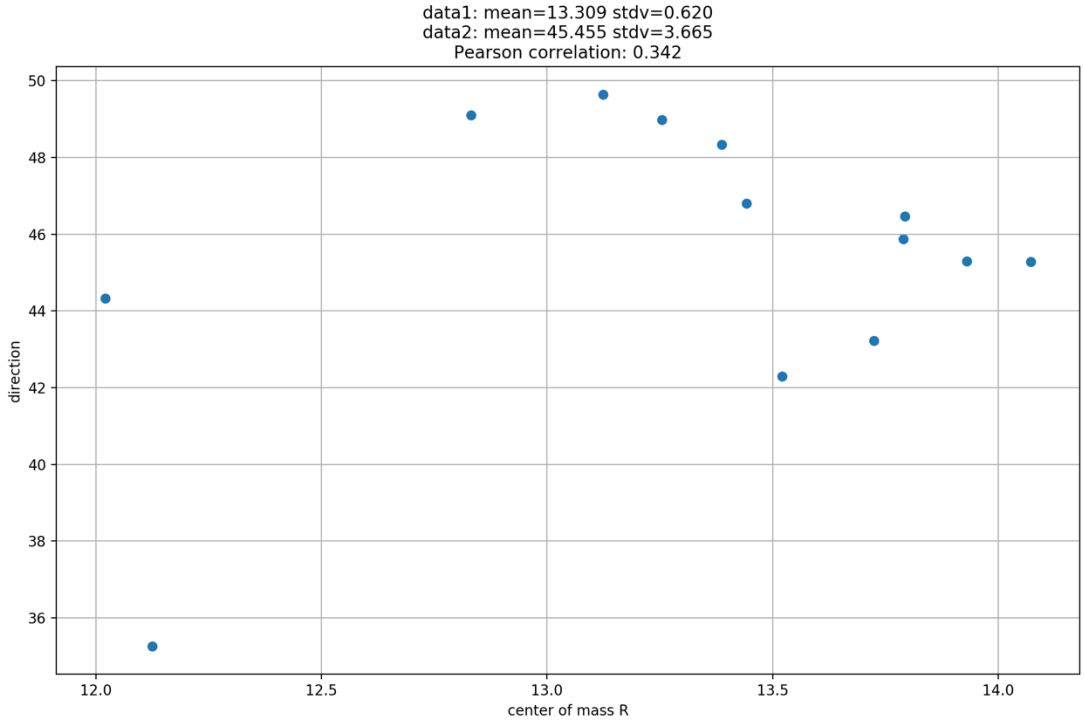


図 21　視線重心角度と重心座標 R の散布図

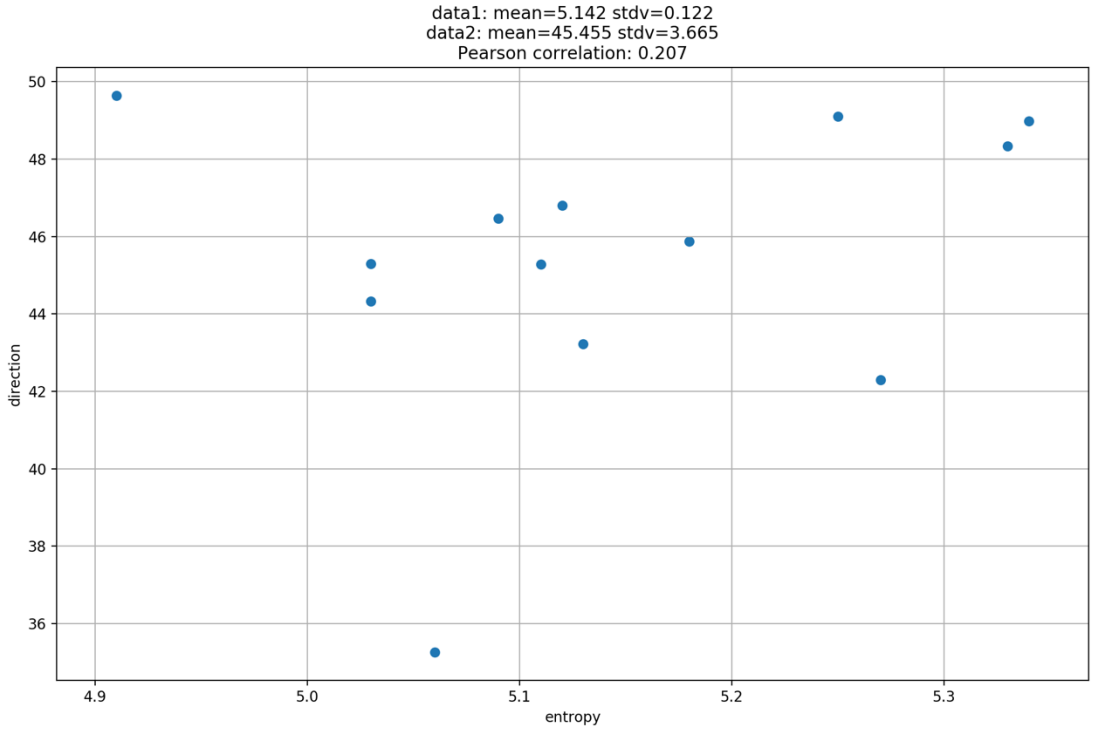


図 22　　視線重心角度とエントロピーの散布図

3.1.3　クラスター分析

クラスタリングの解析方法は、大きく分けて2種類ある。1つは最短距離法[5](hierarchical method)などの階層的手法、もう一種類は K-means 法[6]などの非階層的手法(non-hierarchical method)である。階層的手法の代表例を挙げると最短距離法(nearest neighbor method)がある。これは凝集型に分類され、N 個の対象のデータに対して1個の対象を選びクラスタを N 個生成する初期状態を作成する。初期状態から対象の2点間の距離 $d(x_1,x_2)$からクラスタ間の距離 $d(C_1,C_2)$を計算し、最も近いクラスタを逐次的に併合する操作を繰り返す。

この際、用いられる式は

$$\boldsymbol{d(C_1, C_2) = \min[x_1 \in C_1 , x_2 \in C_2]\, d(x_1, x_2)} \cdots\cdots \text{式 } 4$$

によって計算が行われている。本実験の解析では、プログラムのアルゴリズムの作成の難易度、分割数を自ら設定できる利便性から非階層的手法である K-means 法を用いた。K-means 法では、

$$\boldsymbol{\sum_{i=1}^{k} \sum_{x \in C_i} d((x, c_i))^2} \cdots\cdots \text{式 } 5$$

の式で、セントロイド c_i をクラスタの代表点として k 個のクラスタを分割する。最適解は、クラスタと代表点の計算を繰り返し行い求められる。

本実験では、k=4 とすることで視界を4分割し探索的なデータ分析を行なった。初期値は、(0,0)の中心点が視覚情報を多く含むため、対象数であるデータ数 N に対して評価関数を最小にするものを選ぶ。解析の結果、被験者13人

のクラスタリング解析のプロットは Center of mass の視線重心傾向に適応した分布を示した。よって図19のように被験者は一般的に視覚探索をし、記憶をする際に右上が高頻度、左下を低頻度で探す傾向があることがわかった。これは、被験者が日本人が多かったため、本などを右上から見る癖がついており、スタート時を含め視線傾向がたくなったことが考えられる。

---

[5] http://www.kamishima.net/jp/clustering/から引用

[6] http://tech.nitoyon.com/ja/blog/2009/04/09/kmeans-visualise/から引用

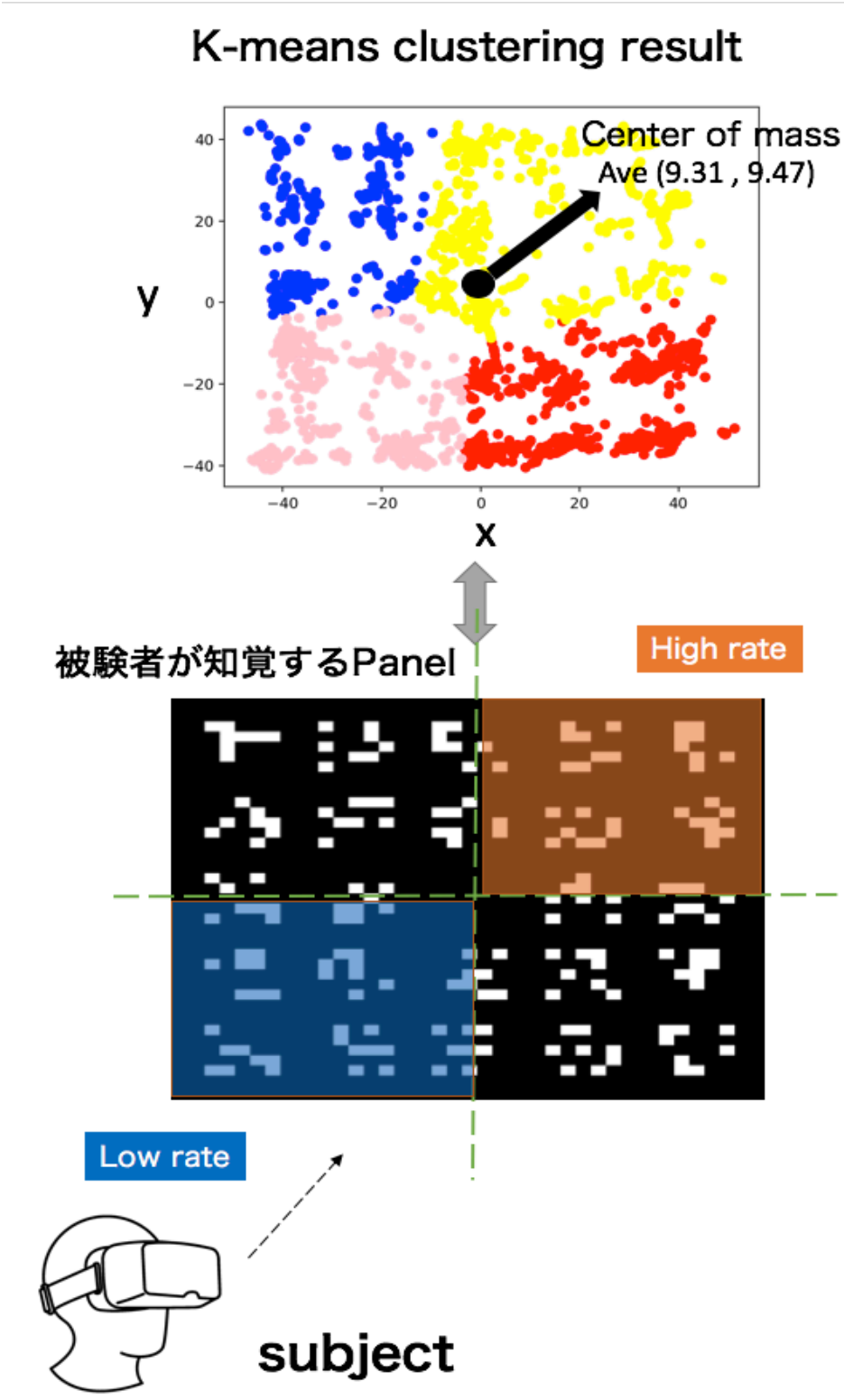


図 23 視線データの4分割クラスタリング結果と被験者が実際に体感する VR 空間での視覚探索傾向の図

3.2　　視線追跡の被験者別のモデルについて

視線情報 Time, x 座標, y 座標から、animation の作成を行い視線の運動を観察した。なお、その際に視線の速度はターゲットを移る時、ターゲットの形を慎重に覚える時に挙動が変化すると考えられる。本研究では、1秒間あたり 30 個のデータを取得できる利点から視線情報のベクトル、速度分析を行い視覚探索中の眼球運動を解析した。解析手法としては、Python を用いてプログラミングを用いてベクトル図、時間依存における眼球運動速度分布図、視覚探索中の x 座標、y 座標速度分布図を作成した。なおこの節では先行研究での注意集中方略、注意拡散方略のモデリングから解析結果を4つのモデルに定義し考察を行う。

なお本実験では、生物学的根拠を得るためマイクロサッカードの解析を行い、取得したデータの時間方向依存性を示し今後の知覚実験の研究への展望を示す。

3.2.1　ベクトル解析

本研究では取得したデータから、時間によって順番に出力されたデータを用いることで、要因間複数規則（方位に関する規則と、速度に関する規則）を解析した。解析のプロットの方法は、

X = (d[:-1,1])

Y = (d[:-1,2])

U= np.diff(d[:,1])

V= np.diff(d[:,2])

plt.quiver(X,Y,U,V,angles='xy',scale_units='xy',scale=1,color='red')

のプログラムから生成する。X,Y で CSV ファイルの aveX,aveY のデータを上から全て読み込み始点を決定した。また、np.diff()の関数によって1つ時間をずらし読みこむデータを U,V とし plt.quiver()からベクトルを自動生成した。

被験者13人を解析した結果から、大きな特徴を持つ4つのモデルに分類し定義する。Centralized type（中心集中型）では(0.0)付近でのデータ数が多く、ベクトル解析の結果から頻繁に中心付近を比較的遅いスピードで通ることがわかる。Diffusion type（拡散型）では、プロット結果では均等に画面全体を見ているように思われたが、3.1.3 の結果から右上高頻度探索を行う傾向があることがわかった。また、ほかのモデルと比較して次のプロットまで大きな変化量を持つ動きが多いことも特徴である。のちの項で述べるが、スピードを解析した結果として探索方略を持つ被験者より拡散して視覚探索をする被験者の方が、眼球運動が素早く行われていることがわかった。本実験では、被験者に探索方略の指示をしなかったのにも関わらず4パターンの探索方略に被験者13人が全て分けられたことから被験者の人数を増やすことによってより正確なモデリングが可能になると考える。なお、本研究では、視線情報の座標データを4モデルのプロット位置から分類した。

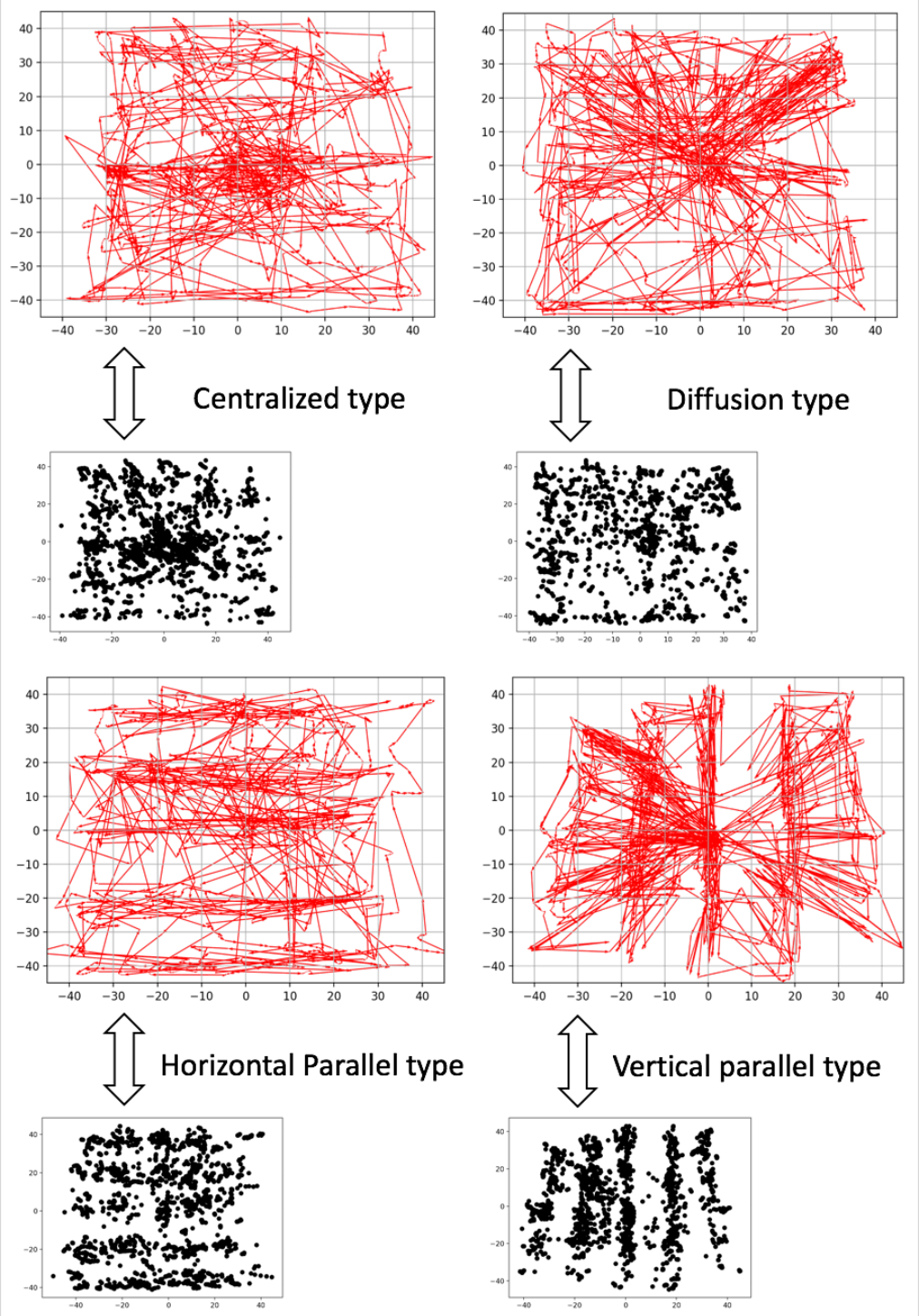


図 24 被験者4パターンのベクトル解析結果の図

3.2.1　マイクロサッカード運動

人間の眼球は、1点を見つめている間にも固視微動と呼ばれる微小運動を続けている。固視微動のうち、振幅と最大速度の関係が通常のサッカード[7]と類似している運動をマイクロサッカードと呼ぶ。本研究の実験結果から、マイクロサッカードはその発生頻度や方向は注意の影響を受けると言われているため解析を行った。

Engbert と Kliegl は Posner の手がかり課題において、マイクロサッカードの発生頻度は手がかり刺激提示直後に減少し、続いて刺激提示後 300ms から 400ms 後には増加するという時間依存性を持つことを示した。また、マイクロサッカード発生頻度が増加する期間には、手がかり刺激の方向に向かうマイクロサッカードが増加することを示した。これらの性質からマイクロサッカードは潜在注意の働きを可視化するための指標としての可能性を示唆されている。

本研究では、探索方略のパターンを4つに定義したことからマイクロサッカードの振る舞いがパターンによって違うと仮定し発生頻度と方向を分析した。

本研究では、マイクロサッカード運動の速度を 1m/sec の眼球運動とした。実験中には、被験者は刺激項目であるターゲット24個を絶え間なく覚えるために眼球運動を行なっている。本解析では、x 成分と y 成分の眼球運動の速度を 30FPS で計 100 秒間においてプロットをした。図21のように被験者データを拡大処理すると x,y 成分ともにマイクロサッケードを確認することができた。その結果から Centralized type, Diffusion type, Horizontal Parallel type, Vertical Parallel type の4つのモデルでマイクロサッカードの頻度を比較した。x 軸は x velocity(unity meter/sec),、y 軸は y velocity(unity meter/sec)である。また赤の点は x 青の点は y の時間依存の点を示している。解析結果から、マイクロサッケードが最も多く見られたモデルは Centralized type であった。これは、刺激項目を覚える際にほかのモデルの被験者と比べて狭い視野で対象を覚えていたことが示される。最もマイクロサッカードの出現頻度が少なかったのは Diffusion type であり、広い視野で視覚探索を行なっていたので出現頻度が少なかったと考えられる。本解析では、時間依存性は視覚探索の時間が長かったことから出現頻度の関連性は求められなかった。

[7] サッカード　眼球が小刻みに、高速で動く運動

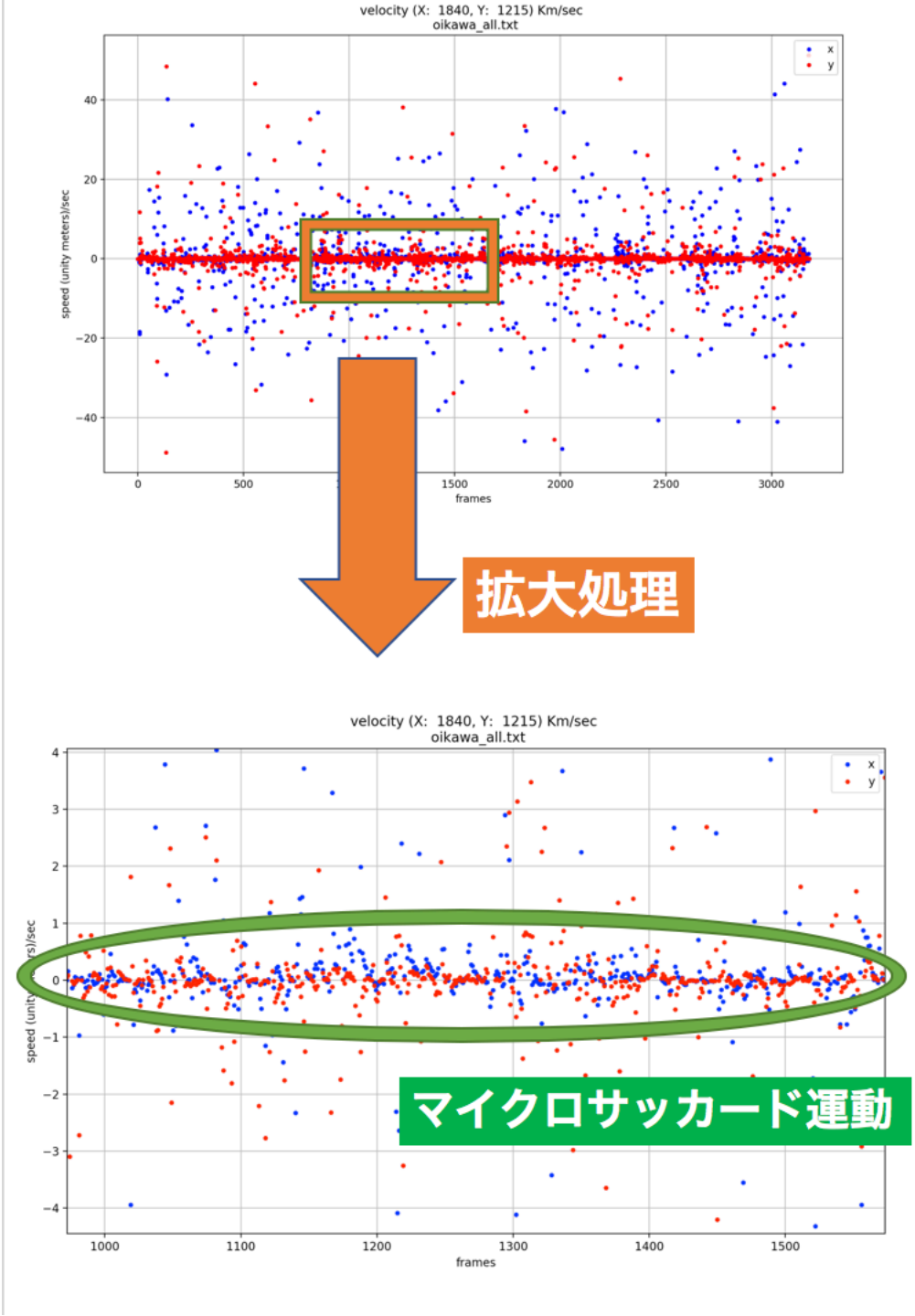


図 25 視覚探索中の被験者における x,y speed の 30FPS 時間依存グラフ

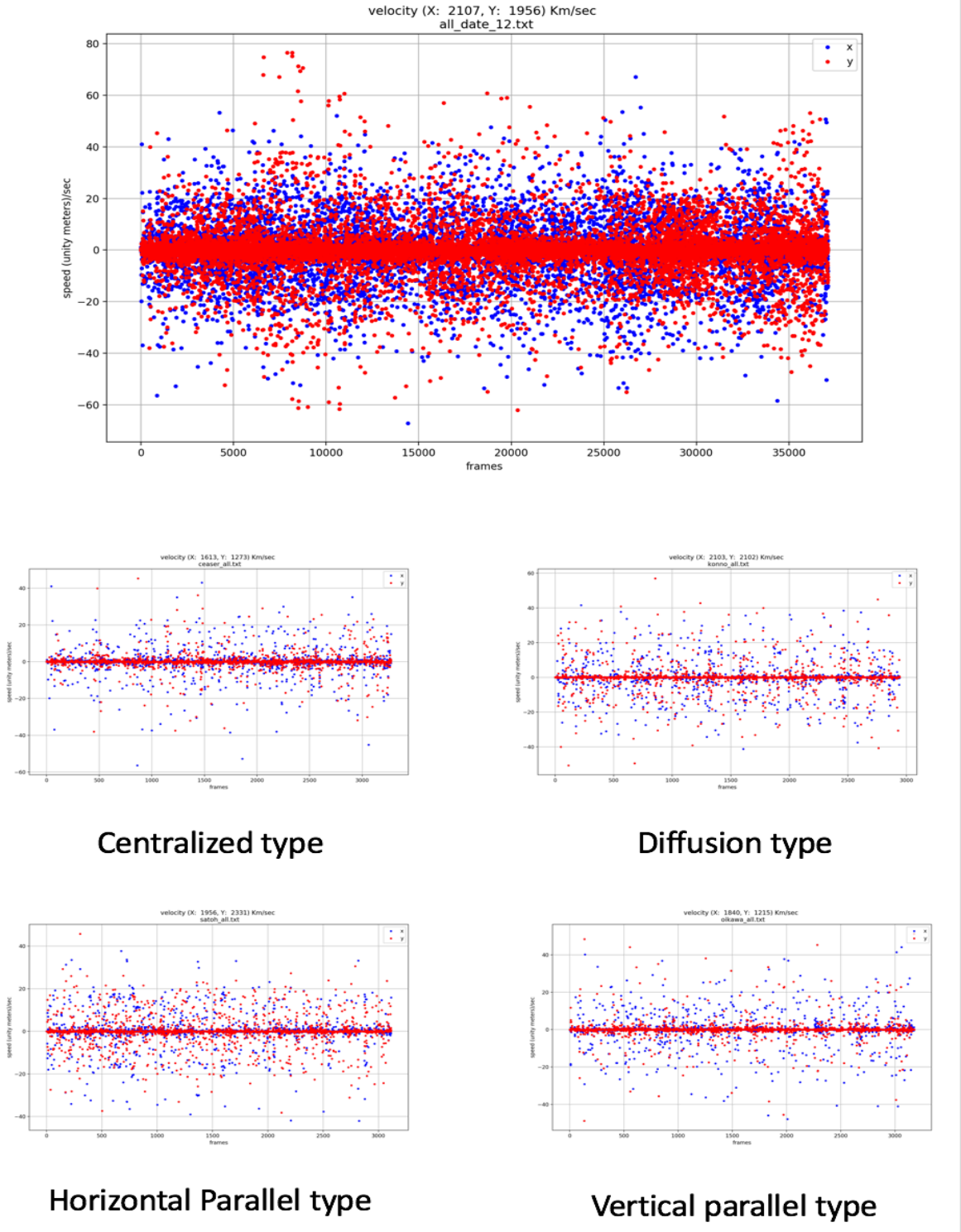


図 26 Eye Speed のモデルごとの解析結果図 上から順に All date,左 Centralized type , Horizontal Parallel type 右 Diffusion type, Vertical Parallel type

3.2.2　探索方略の方向速度依存性

3.2.1 では時間依存からマイクロサッカードを解析したが方向による解析をこの項では行う。ベクトル解析と時間依存における眼球運動速度分布図のプログラムから、時間ごとの視線データから差を取ることによって x 方向、y 方向の速度と向きを解析した。

解析結果は、図23のようになり、x 座標は x velocity、 y 座標は y velocity を示している。なお、マイクサッカードに対応する速度は原点を中心に分布している。この解析結果から、モデルの特徴ごとに速い眼球運動が分布している特徴がある。これは、複数のターゲットを探索するときに、縦に探すような Horizontal Parallel type では y 方向に大きな速度成分をもち、Vertical Parallel type では、x 方向に大きな速度成分を持つような分布を持つことをいう。特徴として見られる大きい速度成分は、モデルに応じて 1000m/sec 内に分布特徴をもち、 Centralized type では 500×500m/sec 領域に多くの分布を持つ。対象的に Diffusion type の被験者は、3.1.1 の結果からエントロピーが少ない傾向をもち、速度分布でも均等にあらゆる方面を見ていることがわかった。

本実験では、Diffusion type の被験者が少なかったが理由として周辺視[8]の能力があげられる。McConkie の研究では、被験者の視野を種々の大きさに制限した状態で、文を読ませる実験を行い視野の大きさと文章を読むスピードの関連性を調べた。結果、人間は文章の1文字を注視しているときに常にその周囲十数文字を同時に読んでいることが知られている。すなわち、本実験でも1つの刺激項目を覚える際に周辺の図形に関する情報を広い視野から脳に並列に中枢に送られており処理を行っている。よって Diffusion type と Centralized type の被験者の解析結果はこの脳による情報の処理と視野の広さに関連があると考えられる。以上の解析から改めて視線情報を 3D ヒストグラムで解析すると視線情報の分布の広さを確認することができた。

[8] https://www.jstage.jst.go.jp/article/jje1965/18/3/18_3_155/_pdf

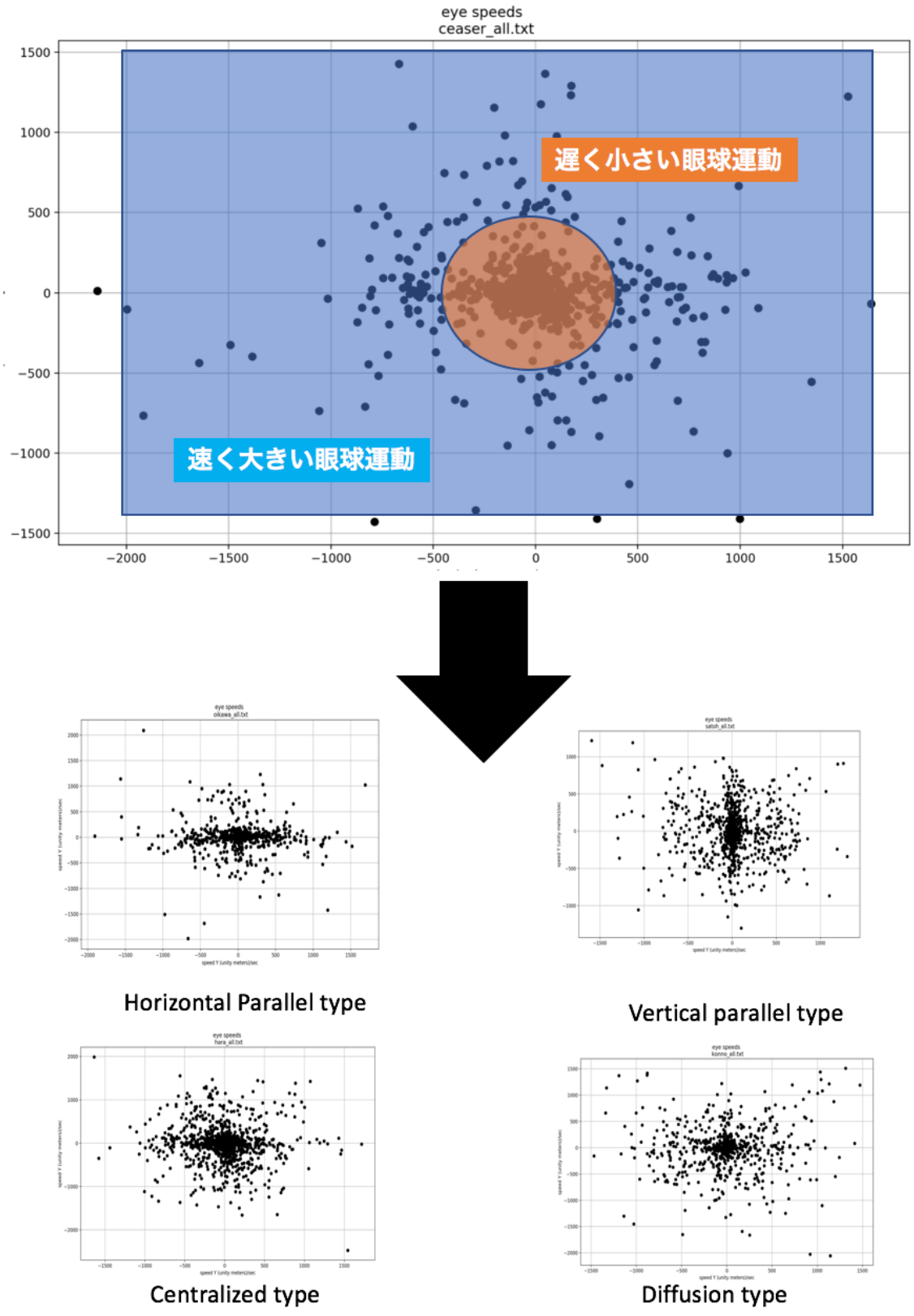


図 27 x,y 成分のの眼球運動速度分布

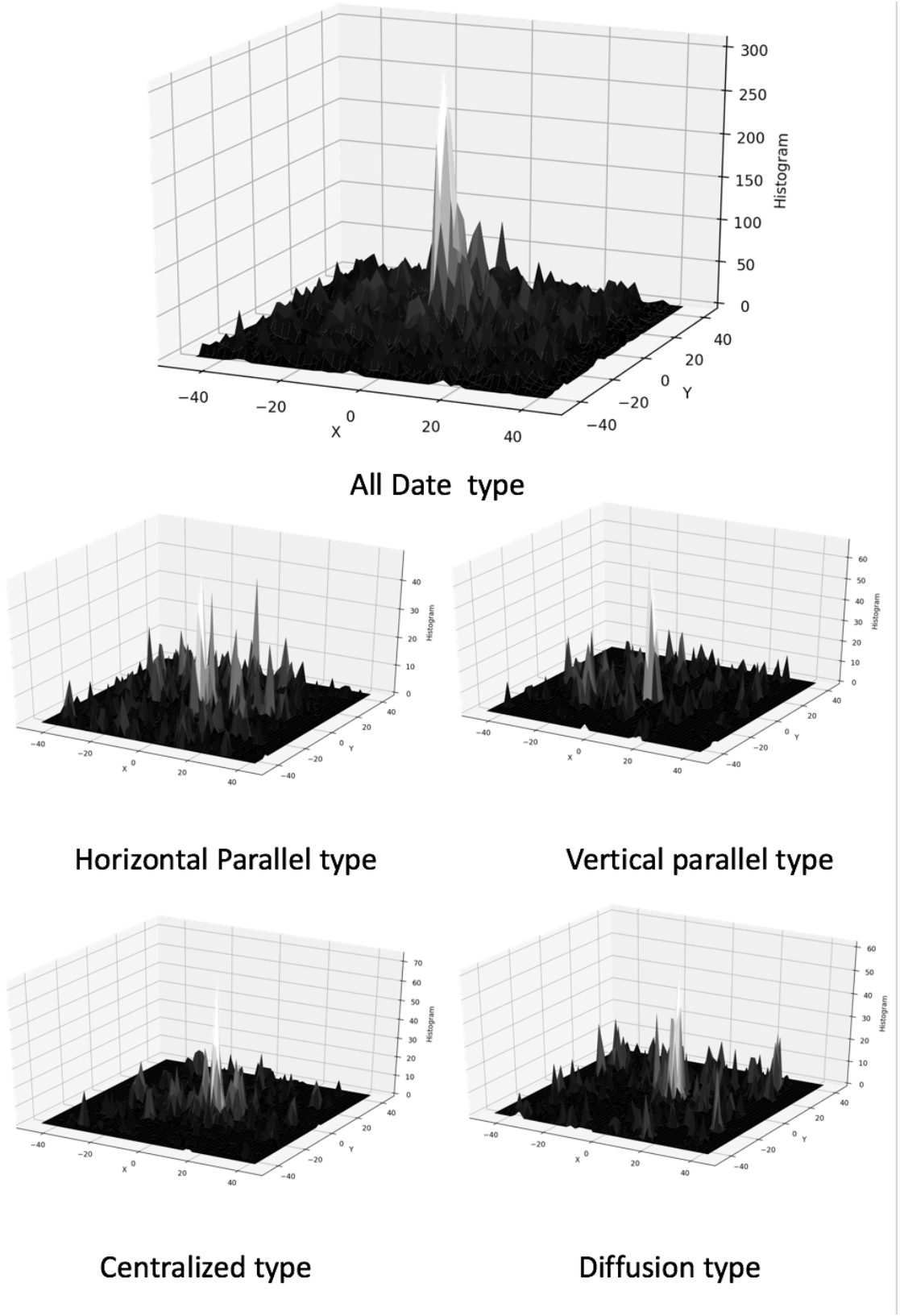


図 28 モデルごとの 3D ヒストグラム

第4章　　まとめ

本研究では、大きく分けて2つの結果を得ることができた。1つは、被験者全員に共通して見られる視線情報の特性としてエントロピー解析、座標データ分布解析、クラスタリング解析、Center of mass 解析から定量的に右上の領域を見やすい傾向があることの有用性を示した。2つ目に、個々のパーソナリティーから探索方略のモデルを作成し、多方面の分析から人間が潜在的にもつ意識を表すと言われるマイクロサッケードの個人差を示すことができた。

今後の展望として、本実験ではマイクロサッケードの時間依存性を発見できなかったことからより FPS をあげることができる実験環境を用いて被験者から多くのデータを取得する必要性があると考える。また、仮設としてエントロピーと正答率に関連性があると考えていたがベクトル解析の結果から、周辺視の能力と正答率に関連性があるという新しい仮説を立てることができたため、新たな視覚探索課題を用いて視線情報を取得する必要性があると考える。

また、本研究で測定の際に PC と FOVE0 の機械的な処理にばらつきが見られたので、全ての被験者のデータ数に十数個のばらつきがあった。これは実験方法として刺激項目を探す時間を長くすることによって解消されるのでより多くのデータを取得する環境づくりをしていきたい。

第5章　　謝辞

本研究を実施するにあたって、Micheletto ruggero 教授には研究を行う機会を与えていただき、また熱心な指導を賜りました。プログラミングについて知識と経験を惜しみなく教えてくださり、将来的に使える技術、考え方のきっかけを作っていただきました。ここに深く感謝をいたします。

また、ミケレット研究室の藤井祐輔さんにはお互いの研究が将来どのように生活に関わるか、実験から理論に渡って討論するなど貴重な時間を共に過ごしていただきました。ここに深く感謝いたします。同研究室の今野和樹さんには、本研究を進めて行く上で遅くまでプログラミングや実験に付き合っていただきました。心から感謝いたします。土屋さん、平井さん、小野さんには卒業論文を作成するにあたって、共に助け合い充実した日々を送ることができました。深く感謝いたします。

最後に、本研究を進めるに当たり、日々の生活を支えてくれた家族、友人、仲間、全ての方々に感謝の意を示し、皆様の今後の益々のご発展を心よりお祈り申し上げます。ここに謝辞といたします。

第6章　　参考文献

[1]大山正『新編感覚・知覚心理学ハンドブック』誠信書房　1994

[2]佐藤達哉『心理学総合事典』 海保博之監修　朝倉書店　2006

[3]APA〔編纂〕『APA 心理学大辞典』培風館　2013

[4]レファレンス協同データベース , レファレンス事例詳細, 埼玉県立久喜図書館提供, http://crd.ndl. go.jp/reference/modules/d3ndlcrdentry/index. php?page=ref_view&id=1000181979.(平 成 29 年 8 月 2 日取得)

[5] 視覚情報の優位性について https://crd.ndl.go.jp/reference/modules/d3ndlcrdentry/index.php?page=ref_view&id=1000181979

[7] Adam, N., & Collins, G. I. (1978) Late components of the visual evoked potential to search in short-term memory. Electroencephalography and Clinical Neurolhysiology, 44, 147-156.

[8] Atkinson, M. H., & Shiffrin, R. M. (1968) Human memory: A proposed system and its control processes. In K. W. Spence &J.T. Spence (Eds.), The psychology of learning and motivation: Advances in research and

theory (Vol. 2, pp. 89-195). Academic Press.

[9] Baddeley, A. (1986) Working memory. Oxford: Oxford Press.

[10] Baddeley, A. (1992) Working memory. Science, 255, 556-559.

[11] Baddeley, A., & Hitch, G,J. (1974) Workingmemory. In G. H. Bower (Ed.),The psychology of learning and motivation: Advances in research andtheory (Vol. 8, pp. 47-90). New York: Academic Press.

[12] Baddeley, A., Logie, R., Bressi, S., Delia Sala, S., & Spmnler, H. (1986)Senile dementia and working memory. Quarterly Journal of Experimental Psychology, 38A, 603-618.

[13] Broadbent, D. E. (1958) Perception and Communication. London: Pergamon Press.

[14] Brookhuis , K. A., Mulder, G., Mulder, L.J. M., Gloench , A. B. M., Van Dellen, H.J., Van der Meere,J.J., & Ellermann, H. H. (1981) Late positive components and stimulus evaluation time. Biological Psychology, 13, 107-123.

[15] Cave, K. R., & Wolfe,J. M. (1990) Modeling the role of parallel processing in visual search. Cognitive Psychology, 22, 225-271.

[16] Cerella, J. (1991) Age effects may be global, not local: Comments on Fisk and Rogers (1991).Journal of Experimental Psychology: Genaral, 120, 215-223.

[17] Cherry, E. C. (1953) Some experiments on the recognition of speech, with one and with two ears.Journal of Acoustic Society of America, 25, 975-

[17] Coles, M. G. H., Gratton, G., Kramer, A. F., & Miller, G. A. (1986) Principles of signal acquisition and analysis. In M. G. H. Coles, E. Donchin, & S. W. Porges (Eds.), Psychophysiology: Systems, processes, and applications (pp. 182-221). Amsterdam: Elsevier.

[18] Daneman, M., & Carpenter, P. A. (1980) Individual differences in working memory and reading.Journal of Verbal Learning and Verbal Behavior, 19, 450-466.

[19] Deutch, J. A., & Deutch, D. (1963) Attention: Some theoretical considerations. Psychological Review, 70, 80-90.

[20] Donchin, E., Karis, D., Bashore, T. R., Coles, M. G. H., & Gratton, G. (1986) Cognitive psychophysiology and human information processing. In M. G. H. Coles, E. Donchin, & S. W. Porges (Eds.), Psychophysiology: Systems, processes, and applications (pp. 244-267). Amsterdam: Elsevier.

[21] Duncan, J., & Humphreys, G. W. (1989) Visual search and stimulus similarity. Psychological Review, 96, 433-458.

[22] Duncan-Johnson, C. C. (1981) P300 latency: A new metric of information processing. Psycho ♪hysiology, 18, 207-215.

[23] Eason, R. G. (1981) Visual evoked potentials correlates of early neural filtering during selective attention. Bulletin ofPsychonomic Society, 18, 203-206.

[24] Eimer, M. (1993) Spatial cueing, sensory gating and selective response preparation: An ERP study on visuo-spatial orienting.

Electroencelholography and Clinical Neurophysiology, 88, 408-420.

[25] Eimer, M. (1994) An ERP study on visual spatial priming with peripheral onsets. Psycholhysiology, 31, 154-163.

[26] 松室 美紀, 三輪 和久 Miki Matsumuro, Kazuhisa Miwa 名古屋大学大学院 情報科学研究科

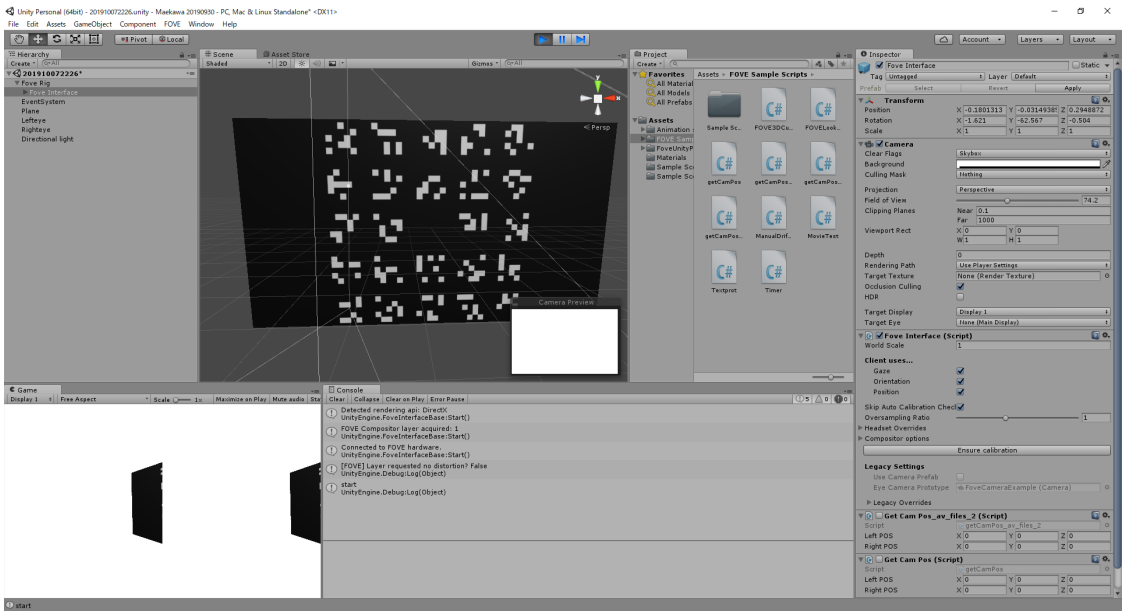
Graduate school of Information Science, Nagoya University

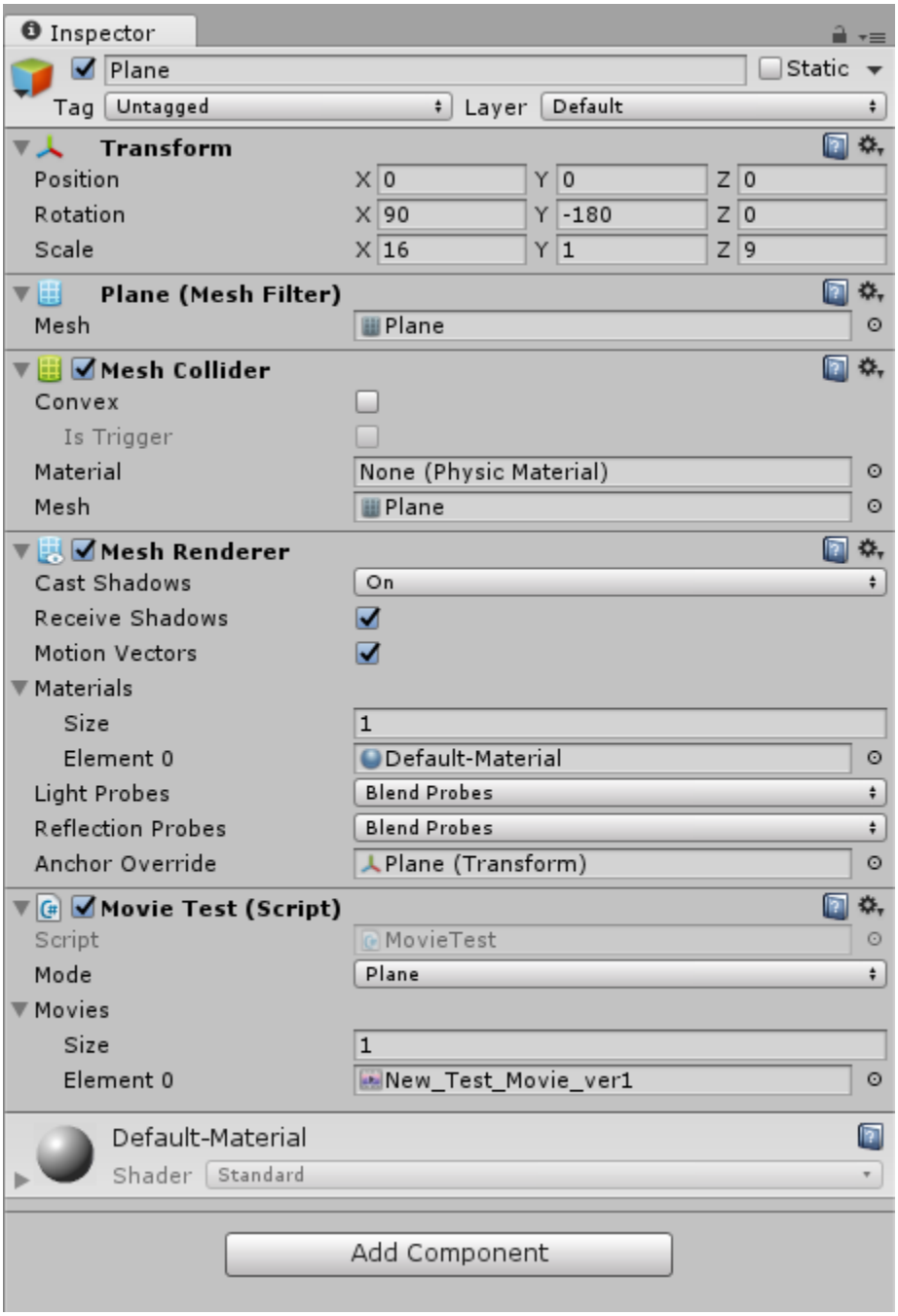
「眼球運動を用いた仮説生成における探索方略の検討」

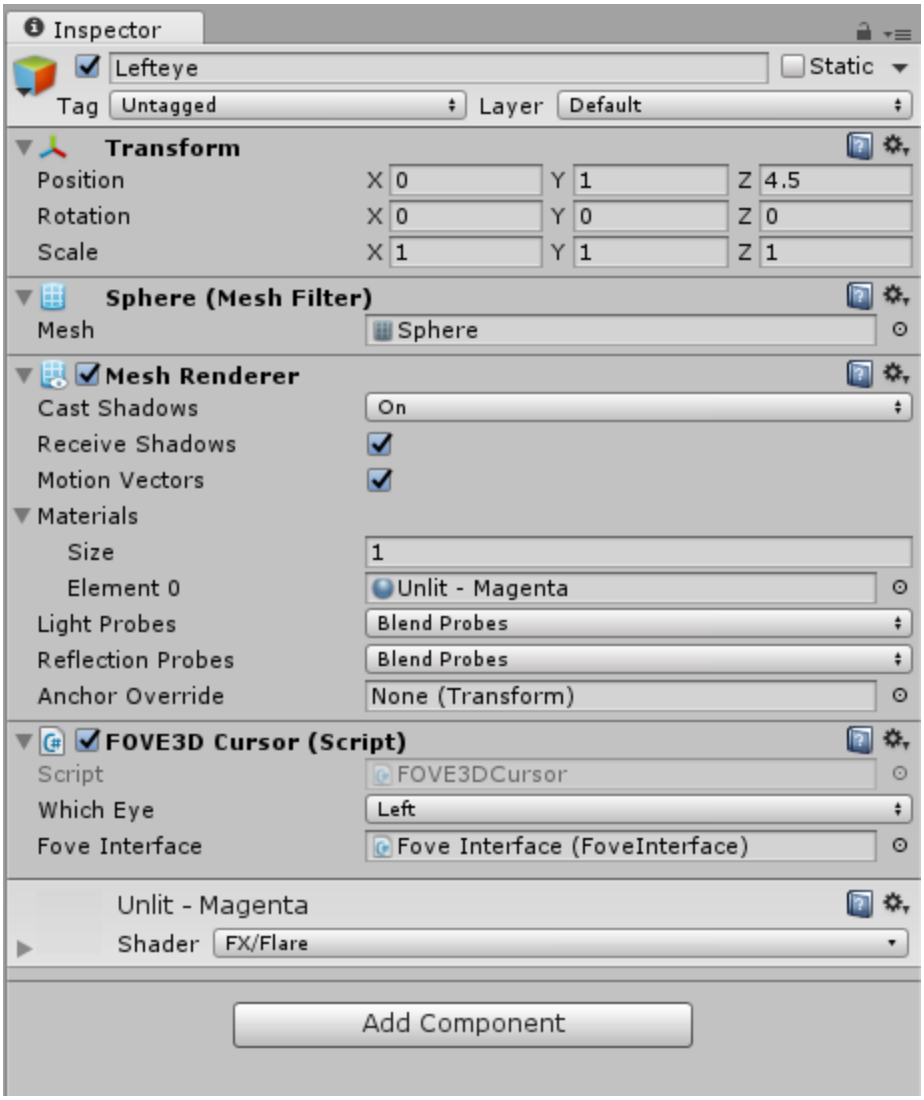
https://www.jcss.gr.jp/meetings/JCSS2011/proceedings/pdf/JCSS2011_P3-16.pdf

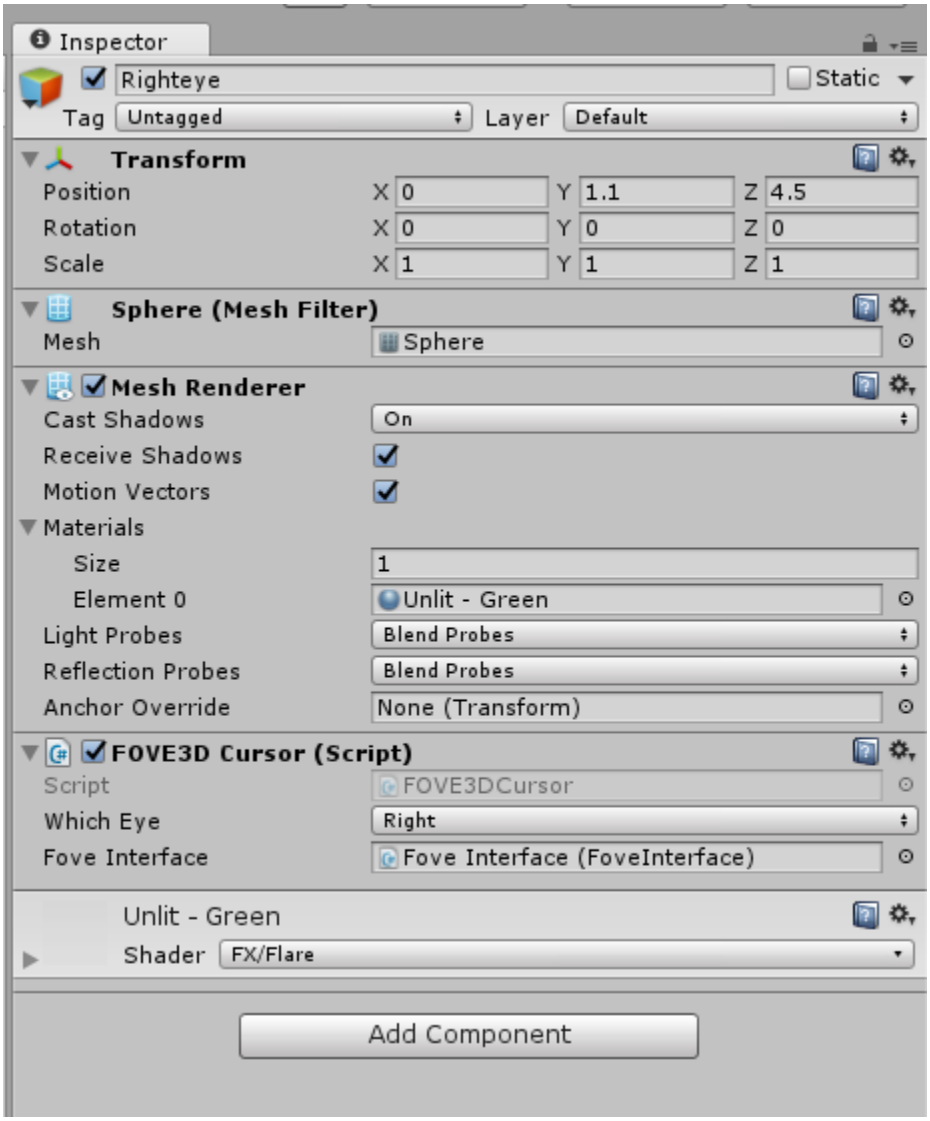
付録

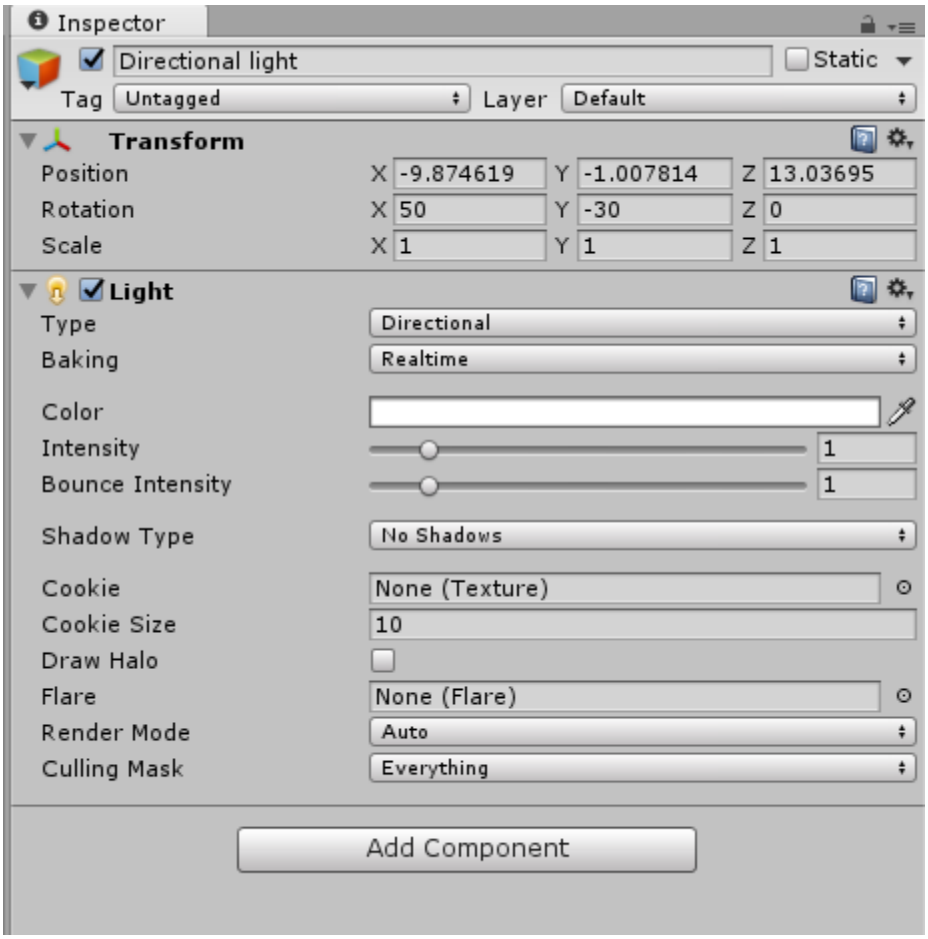
本研究で用いた実験環境、Python プログラムを後続の研究者のために付録としてまとめる。なお、Python プログラムはコピーアンドペーストしての有用性を考えテキストで処理をした。

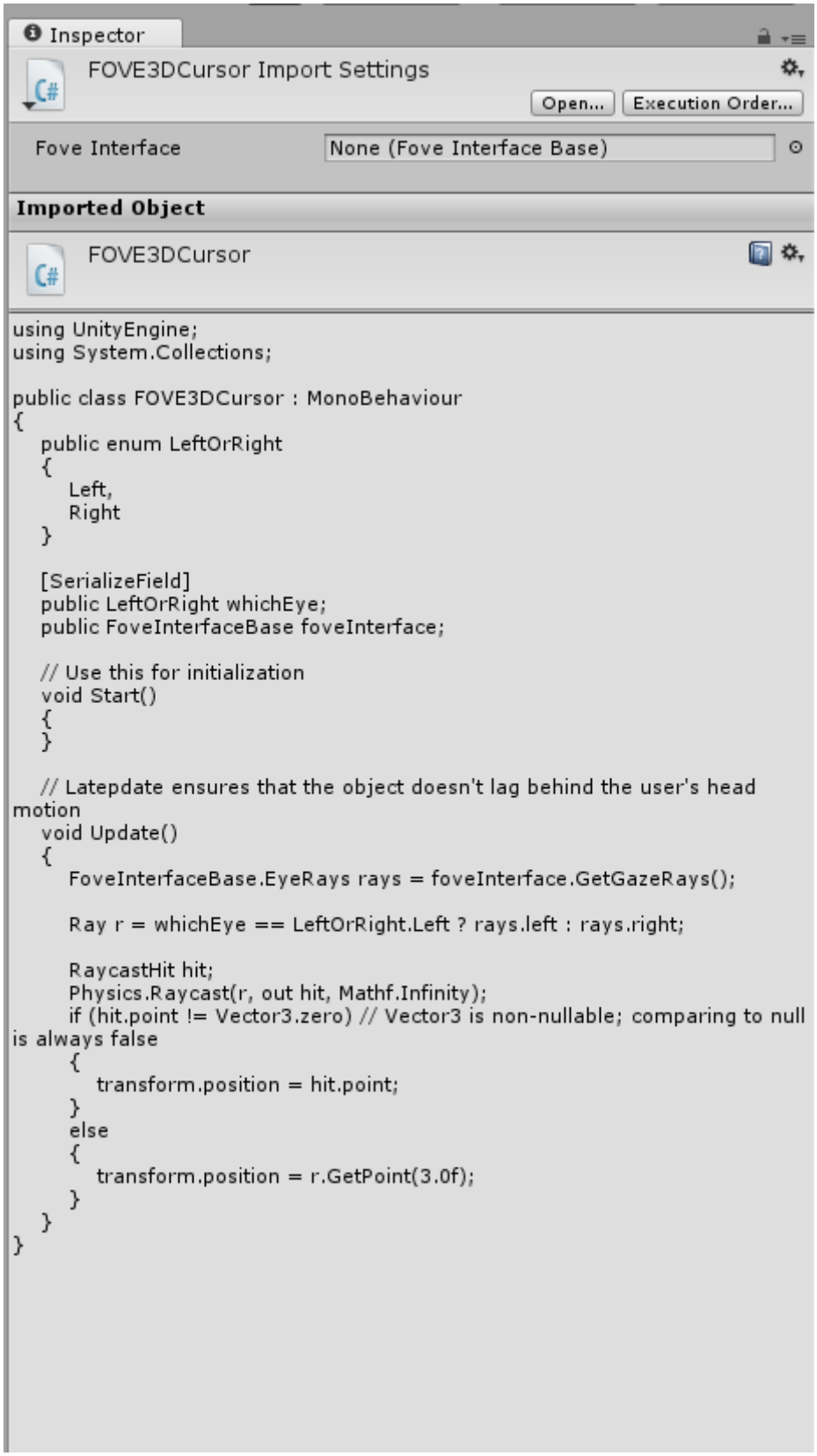



Inspector
Plane
Static
Tag Untagged
Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 90 Y -180 Z 0
Scale X 16 Y 1 Z 9
Plane (Mesh Filter)
Mesh Plane
Mesh Collider
Convex
Is Trigger
Material None (Physic Material)
Mesh Plane
Mesh Renderer
Cast Shadows On
Receive Shadows
Motion Vectors
Materials
Size 1
Element 0 Default-Material
Light Probes Blend Probes
Reflection Probes Blend Probes
Anchor Override Plane (Transform)
Movie Test (Script)
Script MovieTest
Mode Plane
Movies
Size 1
Element 0 New_Test_Movie_ver1
Default-Material
Shader Standard
Add Component


Inspector
Lefteye
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 1
Z 4.5
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Sphere (Mesh Filter)
Mesh
Sphere
Mesh Renderer
Cast Shadows
On
Receive Shadows
Motion Vectors
Materials
Size
1
Element 0
Unlit - Magenta
Light Probes
Blend Probes
Reflection Probes
Blend Probes
Anchor Override
None (Transform)
FOVE3D Cursor (Script)
Script
FOVE3DCursor
Which Eye
Left
Fove Interface
Fove Interface (FoveInterface)
Unlit - Magenta
Shader
FX/Flare
Add Component


Inspector
Righteye
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 1.1
Z 4.5
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Sphere (Mesh Filter)
Mesh
Sphere
Mesh Renderer
Cast Shadows
On
Receive Shadows
Motion Vectors
Materials
Size
1
Element 0
Unlit - Green
Light Probes
Blend Probes
Reflection Probes
Blend Probes
Anchor Override
None (Transform)
FOVE3D Cursor (Script)
Script
FOVE3DCursor
Which Eye
Right
Fove Interface
Fove Interface (FoveInterface)
Unlit - Green
Shader
FX/Flare
Add Component


Inspector
Directional light
Static
Tag
Untagged
Layer
Default
Transform
Position
X -9.874619
Y -1.007814
Z 13.03695
Rotation
X 50
Y -30
Z 0
Scale
X 1
Y 1
Z 1
Light
Type
Directional
Baking
Realtime
Color
Intensity
1
Bounce Intensity
1
Shadow Type
No Shadows
Cookie
None (Texture)
Cookie Size
10
Draw Halo
Flare
None (Flare)
Render Mode
Auto
Culling Mask
Everything
Add Component




```
using UnityEngine;
using System.Collections;

public class FOVE3DCursor : MonoBehaviour
{
    public enum LeftOrRight
    {
        Left,
        Right
    }

    [SerializeField]
    public LeftOrRight whichEye;
    public FoveInterfaceBase foveInterface;

    // Use this for initialization
    void Start()
    {
    }

    // Latepdate ensures that the object doesn't lag behind the user's head
motion
    void Update()
    {
        FoveInterfaceBase.EyeRays rays = foveInterface.GetGazeRays();

        Ray r = whichEye == LeftOrRight.Left ? rays.left : rays.right;

        RaycastHit hit;
        Physics.Raycast(r, out hit, Mathf.Infinity);
        if (hit.point != Vector3.zero) // Vector3 is non-nullable; comparing to null
is always false
        {
            transform.position = hit.point;
        }
        else
        {
            transform.position = r.GetPoint(3.0f);
        }
    }
}
```

Inspector

Imported Object

FOVELookSample

```
using UnityEngine;

public class FOVELookSample : MonoBehaviour {
	public Light attachedLight;
	public FoveInterfaceBase foveInterface;

	private Collider my_collider;
	private Material material;
	private bool light_attached = false;

	void Start() {
		my_collider = GetComponent<Collider>();

		if (attachedLight == null)
			attachedLight =
transform.GetComponentInChildren<Light>();

		if (attachedLight)
		{
			light_attached = true;
			attachedLight.enabled = false;
		}
		material =
gameObject.GetComponent<Renderer>().material;

		if (material == null)
			gameObject.SetActive(false);
	}

	void Update () {
		if (foveInterface.Gazecast(my_collider))
		{
			material.EnableKeyword("_EMISSION");

			if (light_attached)
			{

material.SetColor("_EmissionColor", attachedLight.color);
				attachedLight.enabled = true;

DynamicGI.SetEmissive(GetComponent<Renderer>(), attachedLight.color);
			}
		}
		else
		{
			gameObject.GetComponent<Renderer>
().material.color = Color.white;
			material.DisableKeyword("_EMISSION");

			if (light_attached)
			{
				attachedLight.enabled = false;

DynamicGI.SetEmissive(GetComponent<Renderer>(), Color.black);
			}
		}
	}
}
```

Imported Object

getCamPos

```
using System.Collections;
using System.IO;
using System.Collections.Generic;
using UnityEngine;

public class getCamPos : MonoBehaviour {

                        string FILE_NAME = "posCam_Test_test.txt";
                        //int time = 0;
                        int interval = 1;
  // public Vector3 objPOS; public Vector3 objROT;
   public Vector3 leftPOS;
   public Vector3 rightPOS;
   private float elapsedTime;
   private bool counter_flag = false;
   // private float tzero;
   private float avEyeX, avEyeY;

   // Use this for initialization
   void Start () {
                        Debug.Log ("Start Camera script");
                        string dateTime = System.DateTime.Now.ToString ();
                        StreamWriter sw = File.AppendText(FILE_NAME);
                        sw.WriteLine (" **** New Trial: "+dateTime+"
****");
                        sw.WriteLine ("SUBJECT NAME:");
                        sw.WriteLine ("SCRIPT TYPE:");
                        sw.Close ();

                                //heads = GameObject.FindWithTag
(["MainCamera"]);
                        }

                        // Update is called once per frame
            void Update () {

      var lEye = GameObject.Find("Lefteye");
      var rEye = GameObject.Find("Righteye");
      //StreamWriter sw1 = File.AppendText(FILE_NAME);
      //sw1.WriteLine(" **** UPDATE: ");
      //sw1.Close();

      if (Input.GetKey(KeyCode.S))
      {
         counter_flag = !counter_flag;
         //elapsedTime = 0;

      }

      if(counter_flag == true)

         //(time > interval)
                                {// "this" の意味はこのもの!

         elapsedTime += Time.deltaTime;

                                //          Vector3 mousePOS =
Input.mousePosition;
                                //objPOS = this.transform.position; // this
is the POSITION !!!
                                //objROT = this.transform.eulerAngles;
         leftPOS = lEye.transform.position; // this is the left eye position
         rightPOS = rEye.transform.position;
         //float x = HSV.x;
```

```
                                    //          Vector3 mousePOS =
Input.mousePosition;
                                    //objPOS = this.transform.position; // this
is the POSITION !!!
                                    //objROT = this.transform.eulerAngles;
        leftPOS = lEye.transform.position; // this is the left eye position
        rightPOS = rEye.transform.position;
        //float x = HSV.x;
        //float z = HSV.z;
        //float fx = transform.forward.x;
        //float fz = transform.forward.z;
        StreamWriter sw = File.AppendText(FILE_NAME);
        //sw.WriteLine ("Mouse pos: " + mousePOS);
        //sw.WriteLine(objPOS + " " + objROT + " " + leftPOS + " " +
rightPOS);
        sw.WriteLine(elapsedTime + " " +leftPOS + " " + rightPOS);
        //if (Mathf.Abs(leftPOS.z)>0.1 && Mathf.Abs(rightPOS.z)> 0.1)
        //    {
        //    avEyeX = (leftPOS.x + rightPOS.x) / 2;
        //    avEyeY = (leftPOS.y + rightPOS.y) / 2;
        //    sw.WriteLine(elapsedTime + ", " + avEyeX + ", " + avEyeY);
        //    }
        //sw.WriteLine(leftPOS.z + "," + rightPOS.z);
           //sw.WriteLine ("Cam rotation: " + objROT);
        //sw.WriteLine ("方向 " + transform.forward);
        sw.Close ();
                                    //Debug.Log ("write to file");
                                    //time = 0;
                                    }
                                    //else
                                    //time ++;
                    }
          }
```

Imported Object

getCamPos_av_files_2

```
using System.Collections;
using System.IO;
using System.Collections.Generic;
using UnityEngine;

public class getCamPos_av_files_2: MonoBehaviour
{

    string FILE_NAME = "posCam_file_oikawa";
    //int time = 0;
    int interval = 1;
    // public Vector3 objPOS; public Vector3 objROT;
    public Vector3 leftPOS;
    public Vector3 rightPOS;
    private float elapsedTime;
    private bool counter_flag = false;
    // private float tzero;
    private float avEyeX, avEyeY;
    private int caseF=-99;
    /*private StreamWriter sw0;
    private StreamWriter sw1;
    private StreamWriter sw2;
    private StreamWriter sw3;
    private StreamWriter sw4;
    private StreamWriter sw5;
    private StreamWriter sw6;
    private StreamWriter sw7;
    private StreamWriter sw8;
    private StreamWriter sw9;
    private StreamWriter sw10;
    private StreamWriter sw11;
    private StreamWriter sw12;
    private StreamWriter sw13;
    private StreamWriter sw14;
    private StreamWriter sw15;
    private StreamWriter sw16;
    private StreamWriter sw17;
    private StreamWriter sw18;
    private StreamWriter sw19;
    private StreamWriter sw20;
    */
    // Use this for initialization
    void Start()
    {
        Debug.Log("Start Camera script");
        string dateTime = System.DateTime.Now.ToString();
         StreamWriter sw0 = File.AppendText(FILE_NAME + "0.txt");
         /*StreamWriter sw1 = File.AppendText(FILE_NAME + "1.txt");
         StreamWriter sw2 = File.AppendText(FILE_NAME + "2.txt");
         StreamWriter sw3 = File.AppendText(FILE_NAME + "3.txt");
         StreamWriter sw4 = File.AppendText(FILE_NAME + "4.txt");
         StreamWriter sw5 = File.AppendText(FILE_NAME + "5.txt");
         StreamWriter sw6 = File.AppendText(FILE_NAME + "6.txt");
         StreamWriter sw7 = File.AppendText(FILE_NAME + "7.txt");
         StreamWriter sw8 = File.AppendText(FILE_NAME + "8.txt");
         StreamWriter sw9 = File.AppendText(FILE_NAME + "9.txt");
         StreamWriter sw10 = File.AppendText(FILE_NAME + "10.txt");
         StreamWriter sw11 = File.AppendText(FILE_NAME + "11.txt");
         StreamWriter sw12 = File.AppendText(FILE_NAME + "12.txt");
         StreamWriter sw13 = File.AppendText(FILE_NAME + "13.txt");
         StreamWriter sw14 = File.AppendText(FILE_NAME + "14.txt");
         StreamWriter sw15 = File.AppendText(FILE_NAME + "15.txt");
         StreamWriter sw16 = File.AppendText(FILE_NAME + "16.txt");
         StreamWriter sw17 = File.AppendText(FILE_NAME + "17.txt");
```

Inspector

```
        StreamWriter sw16 = File.AppendText(FILE_NAME + "16.txt");
        StreamWriter sw17 = File.AppendText(FILE_NAME + "17.txt");
        StreamWriter sw18 = File.AppendText(FILE_NAME + "18.txt");
        StreamWriter sw19 = File.AppendText(FILE_NAME + "19.txt");
        */
        sw0.WriteLine(" **** New Trial: " + dateTime + " ****");
        sw0.WriteLine("SUBJECT NAME:");
        sw0.WriteLine("SCRIPT TYPE:");
        sw0.Close ();

        //heads = GameObject.FindWithTag
("MainCamera");
    }

    // Update is called once per frame
    void Update()
    {
        caseF = -99;
        var lEye = GameObject.Find("Lefteye");
        var rEye = GameObject.Find("Righteye");
        //StreamWriter sw1 = File.AppendText(FILE_NAME);
        //sw1.WriteLine(" **** UPDATE: ");
        //sw1.Close();

        if (Input.GetKey(KeyCode.S))
        {
            counter_flag = !counter_flag;
            elapsedTime = 0;

        }

        if (counter_flag == true)

        //(time > interval)
        {// "this" の意味はこのもの!

            elapsedTime += Time.deltaTime;

            if (elapsedTime > 3 && elapsedTime < 13)
            { caseF = 0; }
            if (elapsedTime >= 18 && elapsedTime < 28)
            { caseF = 1; }
            if (elapsedTime >= 33 && elapsedTime < 43)
            { caseF = 2; }
            if (elapsedTime >= 48 && elapsedTime < 58)
            { caseF = 3; }
            if (elapsedTime >= 63 && elapsedTime < 73)
            { caseF = 4; }
            if (elapsedTime >= 78 && elapsedTime < 88)
            { caseF = 5; }
            if (elapsedTime >= 93 && elapsedTime < 103)
            { caseF = 6; }
            if (elapsedTime >= 108 && elapsedTime < 118)
            { caseF = 7; }
            if (elapsedTime >= 123 && elapsedTime < 133)
            { caseF = 8; }
            if (elapsedTime >= 138 && elapsedTime < 148)
            { caseF = 9; }
            //if (elapsedTime >= 54 && elapsedTime < 57)
            //{ caseF = 10; }
            //if (elapsedTime >= 57 && elapsedTime < 62)
            //{ caseF = 11; }
            //if (elapsedTime >= 64 && elapsedTime < 67)
            //{ caseF = 12; }
            //if (elapsedTime >= 67 && elapsedTime < 72)
            //{ caseF = 13; }
            //if (elapsedTime >= 74 && elapsedTime < 77)
            //{ caseF = 14; }
            //if (elapsedTime >= 77 && elapsedTime < 82)
            //{ caseF = 15; }
```

```
        if (elapsedTime >= 125 && elapsedTime < 135)
        { caseF = 8; }
        if (elapsedTime >= 138 && elapsedTime < 148)
        { caseF = 9; }
        //if (elapsedTime >= 54 && elapsedTime < 57)
        //{ caseF = 10; }
        //if (elapsedTime >= 57 && elapsedTime < 62)
        //{ caseF = 11; }
        //if (elapsedTime >= 64 && elapsedTime < 67)
        //{ caseF = 12; }
        //if (elapsedTime >= 67 && elapsedTime < 72)
        //{ caseF = 13; }
        //if (elapsedTime >= 74 && elapsedTime < 77)
        //{ caseF = 14; }
        //if (elapsedTime >= 77 && elapsedTime < 82)
        //{ caseF = 15; }
        //if (elapsedTime >= 84 && elapsedTime < 87)
        //{ caseF = 16; }
        //if (elapsedTime >= 87 && elapsedTime < 92)
        //{ caseF = 17; }
        //if (elapsedTime >= 94 && elapsedTime < 97)
        //{ caseF = 18; }
        //if (elapsedTime >= 97 && elapsedTime < 102)
        //{ caseF = 19; }
        ///

        // Vector3 mousePOS = Input.mousePosition;
        //objPOS = this.transform.position; // this is the POSITION !!!
        //objROT = this.transform.eulerAngles;
        leftPOS = lEye.transform.position; // this is the left eye position
        rightPOS = rEye.transform.position;
        //float x = HSV.x;
        //float z = HSV.z;
        //float fx = transform.forward.x;
        //float fz = transform.forward.z;
        //sw.WriteLine ("Mouse pos: " + mousePOS);
        //sw.WriteLine(objPOS + " " + objROT + " " + leftPOS + " " +
rightPOS);
        //sw.WriteLine(elapsedTime + " " +leftPOS + " " + rightPOS);
        if (Mathf.Abs(leftPOS.z) < 0.1 && Mathf.Abs(rightPOS.z) < 0.1)
        {
           avEyeX = (leftPOS.x + rightPOS.x) / 2;
           avEyeY = (leftPOS.y + rightPOS.y) / 2;

           if (caseF == 0)
           {
              StreamWriter sw0 = File.AppendText(FILE_NAME + "0.txt");
              sw0.WriteLine(elapsedTime + ", " + avEyeX + ", " + avEyeY);
              sw0.Close();
           } // case
           if (caseF == 1)
           {
              StreamWriter sw1 = File.AppendText(FILE_NAME + "1.txt");
              sw1.WriteLine(elapsedTime + ", " + avEyeX + ", " + avEyeY);
              sw1.Close();
           } // case
           if (caseF == 2)
           {
              StreamWriter sw2 = File.AppendText(FILE_NAME + "2.txt");
              sw2.WriteLine(elapsedTime + ", " + avEyeX + ", " + avEyeY);
              sw2.C...

<...etc...>
```

Imported Object

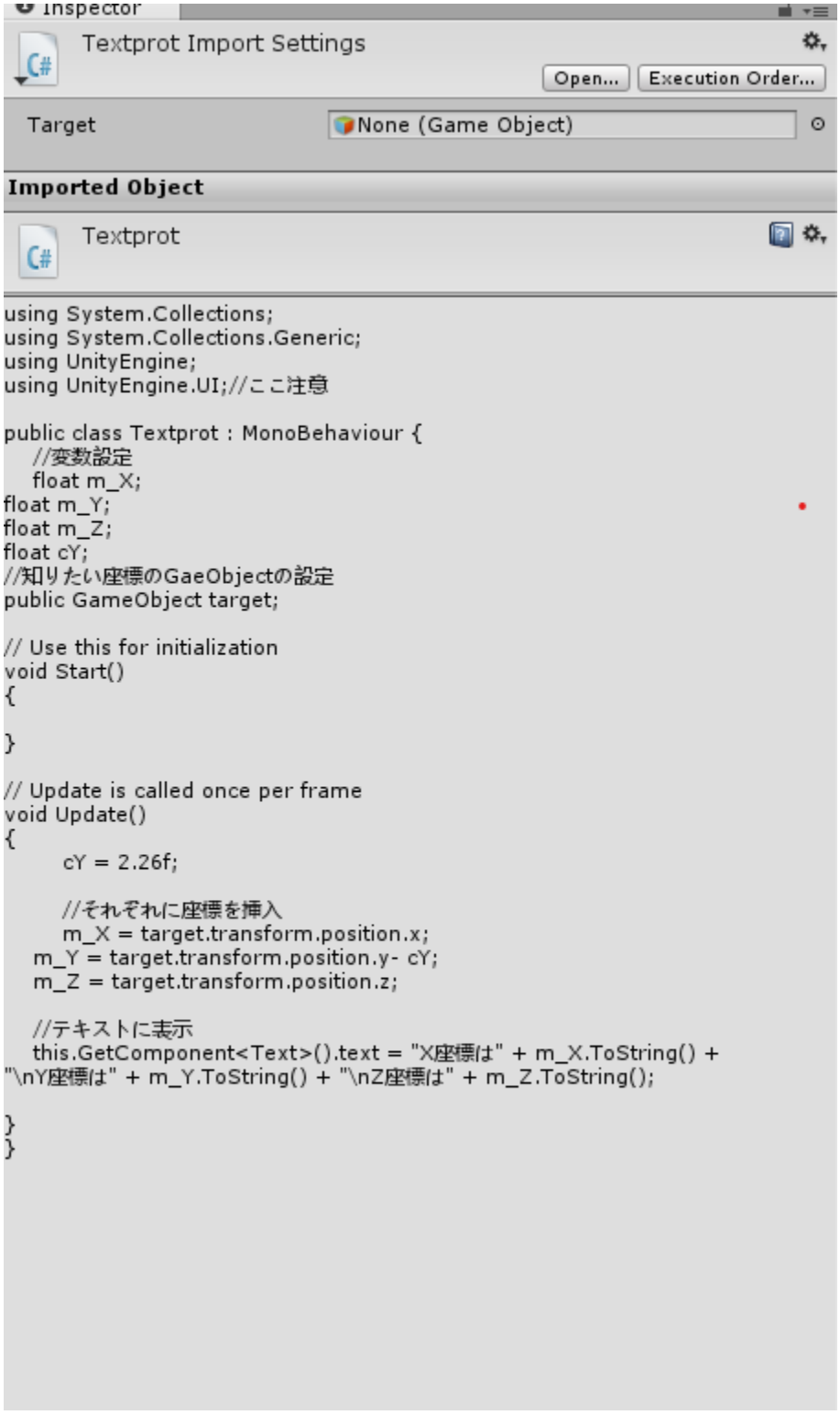
ManualDriftCorrection

```
using UnityEngine;
using System.Collections;

public class ManualDriftCorrection : MonoBehaviour
{
	// Use this for initialization
	void Start()
	{

	}

	// Update is called once per frame
	void Update()
	{
		if (Input.GetKeyDown(KeyCode.M))
		{
			this.GetComponent<Renderer>().enabled =
true;
		}

		if (this.GetComponent<Renderer>().enabled)
		{
			RaycastHit hit;
			FoveInterface foveInterface =
transform.parent.GetComponent<FoveInterface>();
			Ray ray = new
Ray(foveInterface.transform.position, foveInterface.transform.forward);
			if (Physics.Raycast(ray, out hit, 10.0f))
			{
				transform.position = hit.point;
			}
			else
			{
				transform.position =
foveInterface.transform.position + foveInterface.transform.forward * 1.5f;
			}
		}

		if (Input.GetKeyUp(KeyCode.M))
		{
			this.GetComponent<Renderer>().enabled =
false;
			Vector3 loc = transform.localPosition;
			Fove.Managed.SFVR_Vec3 driftLocation;
			driftLocation.x = loc.x;
			driftLocation.y = loc.y;
			driftLocation.z = loc.z;

FoveInterfaceBase.GetFVRHeadset().ManualDriftCorrection3D(driftLocation);
		}
	}
}
```

MovieTest

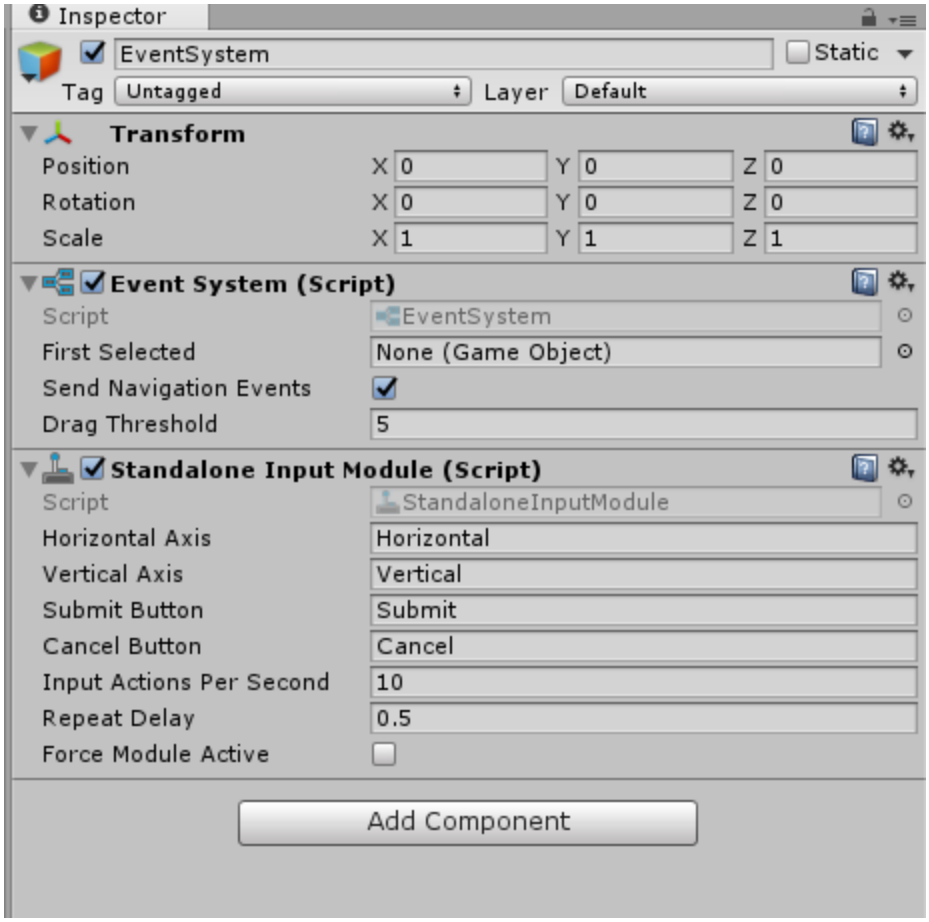
```
using UnityEngine;
using System.Collections;
using UnityEngine.UI;

public class MovieTest : MonoBehaviour
{

   enum Mode
   {
      Plane,
      RawImage
   };

   //  PlaneとRawImageで切り替える
   [SerializeField]
   private Mode mode;
   //  再生する動画
   [SerializeField]
   private MovieTexture[] movies;
   //  動画番号
   private int num;

   void Start()
   {
      //  最初の動画を設定
      num = 0;
      if (mode == Mode.Plane)
      {
         GetComponent<MeshRenderer>().material.mainTexture =
movies[num];
      }
      else
      {
         GetComponent<RawImage>().texture = movies[num];
      }
      //  動画をループ設定
      movies[num].loop = true;
   }

   void Update()
   {
      //  Sキーを押したら動画の再生とポーズを繰り返す
      if (Input.GetKeyDown("s"))
      {
         if (movies[num].isPlaying)
         {
            movies[num].Pause();
            Debug.Log("pause");
         }
         else
         {
            movies[num].Play();
            Debug.Log("start");
         }
         //  Qキーを押したら動画をストップ
      }
      else if (Input.GetKeyDown("q"))
      {
         movies[num].Stop();
         Debug.Log("stop");
      }
      else if (Input.GetKeyDown("n"))
      {
         //  動画を切り替える前に今再生している動画を止める
         movies[num].Stop();
```

```
    private int num;

    void Start()
    {
        //  最初の動画を設定
        num = 0;
        if (mode == Mode.Plane)
        {
            GetComponent<MeshRenderer>().material.mainTexture =
movies[num];
        }
        else
        {
            GetComponent<RawImage>().texture = movies[num];
        }
        //  動画をループ設定
        movies[num].loop = true;
    }

    void Update()
    {
        //  Sキーを押したら動画の再生とポーズを繰り返す
        if (Input.GetKeyDown("s"))
        {
            if (movies[num].isPlaying)
            {
                movies[num].Pause();
                Debug.Log("pause");
            }
            else
            {
                movies[num].Play();
                Debug.Log("start");
            }
            //  Qキーを押したら動画をストップ
        }
        else if (Input.GetKeyDown("q"))
        {
            movies[num].Stop();
            Debug.Log("stop");
        }
        else if (Input.GetKeyDown("n"))
        {
            //  動画を切り替える前に今再生している動画を止める
            movies[num].Stop();

            //  次の動画を指す
            num++;
            if (num >= movies.Length)
            {
                num = 0;
            }
            //  Textureを次のMovieTextureに変える
            if (mode == Mode.Plane)
            {
                GetComponent<MeshRenderer>().material.mainTexture =
movies[num];
            }
            else
            {
                GetComponent<RawImage>().texture = movies[num];
            }
            //  動画をループに設定
            movies[num].loop = true;
            Debug.Log("change movie");
        }
    }
}
```


Inspector
Textprot Import Settings
Open...
Execution Order...
Target
None (Game Object)
Imported Object
Textprot
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
using UnityEngine.UI;//ここ注意
public class Textprot : MonoBehaviour {
//変数設定
float m_X;
float m_Y;
float m_Z;
float cY;
//知りたい座標のGaeObjectの設定
public GameObject target;
// Use this for initialization
void Start()
{
}
// Update is called once per frame
void Update()
{
cY = 2.26f;
//それぞれに座標を挿入
m_X = target.transform.position.x;
m_Y = target.transform.position.y- cY;
m_Z = target.transform.position.z;
//テキストに表示
this.GetComponent<Text>().text = "X座標は" + m_X.ToString() +
"\nY座標は" + m_Y.ToString() + "\nZ座標は" + m_Z.ToString();
}
}

Timer

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;

public class Timer : MonoBehaviour
{

    // Use this for initialization
    void Start()
    {

    }

    // Update is called once per frame
    void Update()
    {
        Debug.Log("経過時間(秒)" + Time.time);
    }
}
```


Inspector
EventSystem
Static
Tag
Untagged
Layer
Default
Transform
Position
X 0
Y 0
Z 0
Rotation
X 0
Y 0
Z 0
Scale
X 1
Y 1
Z 1
Event System (Script)
Script
EventSystem
First Selected
None (Game Object)
Send Navigation Events
Drag Threshold
5
Standalone Input Module (Script)
Script
StandaloneInputModule
Horizontal Axis
Horizontal
Vertical Axis
Vertical
Submit Button
Submit
Cancel Button
Cancel
Input Actions Per Second
10
Repeat Delay
0.5
Force Module Active
Add Component

●3dHistogram_surf.py

```
from mpl_toolkits.mplot3d import Axes3D  # noqa: F401 unused import
import matplotlib.pyplot as plt
import numpy as np
d = np.loadtxt('konno_all.txt',delimiter=',')
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')
x=(d[:,1])
y=(d[:,2])
hist, xedges, yedges = np.histogram2d(x, y, bins=50, range=[[-45, 45], [-45, 45]] )
# Construct arrays for the anchor positions of the 16 bars.
xpos, ypos = np.meshgrid(xedges[:-1], yedges[:-1], indexing="ij")
zpos = 0
print(np.shape(hist))
print(np.shape(xpos),np.shape(ypos))
surf=ax.plot_surface(xpos,ypos,hist,rstride=1,cstride=1,cmap='gray',linewidth=1 )
ax.set_xlabel('X')
ax.set_ylabel('Y')
ax.set_zlabel('Histogram')
```

●Clustering_original.py

```
import csv
```

```
import random
import numpy as np
import matplotlib.pyplot as plt
def read_data(filename):
    data = []
    with open(filename, "r") as f:
        reader = csv.reader(f)
        for row in reader:
            data.append(row)
    data = [[float(data[i][j]) for j in range(0, len(row))] for i in range(0, len(data))]
    return data
def write_data(filename, cluster):
    with open(filename, "w") as f:
        writer = csv.writer(f)
        writer.writerows(cluster)
    print("File was written .")
def clustering(data, u, number, N):
    cluster = []
    for m in range(0, N):

        cluster.append([random.randrange(1, number+1),m+1,   data[m][1], data[m][2]])
    print("First cluster nuclear")
    for sample in cluster:
        print(sample)
```

```
    print("¥nStart Clustering")
    for count in range(0, 10000):
        u = center(u, number, N, cluster)
        cluster = belong(u, number, N, cluster)      cluster.sort()
    print("Finish clustering .")
    return cluster, u
def center(u, number, N, cluster):
    for n in range(0, number):
        x_center = 0.0
        y_center = 0.0
        sample_count = 0
        for m in range(0, N):
            if cluster[m][0] == n + 1:
                x_center = x_center + cluster[m][2]
                y_center = y_center + cluster[m][3]
                sample_count = sample_count + 1
        if sample_count != 0:
            x_center = x_center / sample_count
            y_center = y_center / sample_count
        u[n][0] = n + 1
        u[n][1] = x_center
        u[n][2] = y_center
    return u
def belong(u, number, N, cluster):
```

```
    for m in range(0, N):
        d_min = (cluster[m][2] - u[0][1])**2 + (cluster[m][3] - u[0][2])**2
        temp_number = 1
        for n in range(1, number):
            d = (cluster[m][2] - u[n][1])**2 + (cluster[m][3] - u[n][2])**2
            if d < d_min:
                d_min = d
                temp_number = n + 1
        cluster[m][0] = temp_number
    return cluster
def main():
    filename = "posCam_file_fujii_all.txt"
    data = read_data(filename)
    N = len(data)
    print('N:', N)
    number = 4
    u = [[1, 0.0, 0.0],
         [2, 0.0, 0.0],
         [3, 0.0, 0.0],
         [4, 0, 0]
     ]
    cluster, u = clustering(data, u, number, N)
    cA=np.array(cluster)
    print("shape",np.shape(cA))
```

```
    cl=np.array(["red","blue","yellow","pink"])
    group=cA[:,0]-1
    #print("group",group,"shape of group",np.shape(group),np.type(group))
    plt.scatter(cA[:,2],cA[:,3],c=cl[ group.astype(int) ])
    filename2 = "cluster.csv"
    write_data(filename2, cluster)
    print("Clustering")
    for sample in cluster:
        print(sample)
    print("Center")
    for sample in u:
        print(sample)
    print("population")
    g1=g2=g3=g4=0
    for data in cluster:
            if data[0]==1: g1=g1+1
            if data[0]==2: g2=g2+1
            if data[0]==3: g3=g3+1
            if data[0]==4: g4=g4+1
    print("g1=%d, g2=%d, g3=%d, g4=%d"%(g1,g2,g3,g4))
    plt.show()
if __name__ == "__main__":
    main()
```

●prot_heatmap_bin5_original_entr.py

```
import matplotlib.pyplot as plt
from numpy import *
from mpl_toolkits.mplot3d import Axes3D
import numpy as np
d = np.loadtxt('all_date_12.txt',delimiter=',')
yoko=(d[:,1])
tate=(d[:,2])
fig= plt.figure()
plt.plot(yoko,tate,"o")
plt.grid()
plt.show()
fig,ax=plt.subplots()
H = ax.hist2d(yoko,tate, bins=20,cmap="gray",vmin=0)
from scipy.stats import entropy
from scipy.ndimage.measurements import center_of_mass
print("shape H",shape(H[0]))
vis=H[0].flatten()
entr=entropy(vis)
com=center_of_mass(H[0])
print ("entropy",entropy(vis))
print ("center_of_mass",center_of_mass(H[0])  )
ax.set_title('Heat map, entropy: %5.3g, cm: (%4.2g,%4.2g)'%(entr,com[0],com[1]))
```

```
ax.set_xlabel('x')

ax.set_ylabel('y')

fig.colorbar(H[3],ax=ax)

plt.show()
```

speed_plot_lr.py

```
import matplotlib.pyplot as plt

import numpy as np

fname="all_date_12.txt"

d = np.loadtxt(fname,delimiter=',')

X=d[:,1]

Y=d[:,2]

T=d[:,0] # time

dX=(d[1:,1]-d[:-1,1])

dY=(d[1:,2]-d[:-1,2])

dT=(d[1:,0]-d[:-1,0])

print(X,dX)

xspeed = dX/dT # unity meters/sec

yspeed = dY/dT # unity meters/sec

plt.plot(dX,'b.',label='x')

plt.plot(dY,'r.',label='y')

plt.legend(loc='upper right')

avX=np.mean(np.abs(dX))

stdX=np.std(dX)

avY=np.mean(np.abs(dY))
```

```
stdY=np.std(dY)
print(avX,stdX)
print(avY,stdY)
plt.grid(True)
c=1000
plt.title("velocity (X: %5.4g, Y: %5.4g) Km/sec ¥n %s"%(avX*c,avY*c,fname))
plt.ylabel("speed (unity meters)/sec")
plt.xlabel("frames")
plt.show()
################ HIST
bins=500
plt.hist(dX,bins=bins,color="blue",alpha=0.5)
plt.grid(True)
plt.hist(dY,bins=bins,color="red",alpha=0.5)
plt.grid(True)
plt.show()
###### linear regression #########
from sklearn.linear_model import LinearRegression
xsp=xspeed.reshape(-1,1)
ysp=yspeed.reshape(-1,1)
model=LinearRegression().fit(xsp,ysp)
print("shape xspeed",np.shape(xsp))
print("shape yspeed",np.shape(ysp))
scr=model.score(xsp,ysp)
```

```
rc=model.coef_

ict=model.intercept_

plt.plot(xspeed,yspeed,'ko')

plt.title("eye speeds ¥n %s"%(fname))

plt.suptitle("rc: %5.3g int: %5.3g score: %5.3g"%(rc,ict,scr))

plt.ylabel("speed Y (unity meters)/sec")

plt.xlabel("speed X (unity meters)/sec")

plt.grid(True)

plt.show()

plt.show()
```

●vecter_master.py

```
import numpy as np

import matplotlib.pyplot as plt

plt.figure()

fig,ax= plt.subplots()

fname="tsukida_all.txt"

d = np.loadtxt(fname,delimiter=',')

X = (d[:-1,1])
```

```
Y = (d[:-1,2])

U= np.diff(d[:,1])

V= np.diff(d[:,2])

plt.quiver(X,Y,U,V,angles='xy',scale_units='xy',scale=1,color='red')

plt.xlim([-45,45])

plt.ylim([-45,45])

plt.grid()

plt.draw()

plt.show()
```