卒業論文

アイトラックを用いたタッチタイピングの視線追跡

関西学院大学理工学部

情報科学科 西谷研究室

27018575 井本皇己

2022年3月

概 要

本研究では, 西谷研究室が以前開発したタイピングを練習するソフトである shunkuntype と所有している pupil labs 製のアイトラックを使用する．これらを使い, タッチタイピングのデータの可視化及びアイトラックにおいての calibration について研究する．結果として,

- calibration の精度向上.
- calibration の精度向上の比較.
- タイピングデータから, 一日にした練習回数の積算と初日からの経過日数を自動的に抽出しそれをグラフ化.

を行った.

目 次

図 目 次

第1章　序論

物事の習い初めたばかりの初心者とある程度習得した非初心者とでは, 動作速度や見ている目線が異なる. そこで, 目の瞳孔を追跡するアイトラッキングの開発や研究が近年増加している. アイトラックを利用することで, 被験者がいつ, どこを, どのように見ているのかという３つの要素をデータ化し、被験者の視線を可視化することができる.

アイトラックの現状として, AI,Iot,VR とデジタル領域における各分野で技術進歩が加速しており, その影響が市場での急成長に現れている.

> 今年 1 月に発表された、リサーチステーション合同会社の調査によれば、アイトラッキング（視線計測）市場は今後急成長が見込まれ、その世界市場規模は 2020 年段階の推計で 5 億 6,000 万ドル、2025 年には 17 億 8,600 万ドルに達すると予測されています [1]。
>
> これまでデジタル領域における“間”を、アイトラッキング（視線計測）で埋めようとする動きが世界中、各分野で加速しており、その影響が市場の急成長にも表れていると考えています [1]。

ここでいう’間’というのは, ユーザーの動向の結果においてどこをよく見ていて, 興味があるのか, そして相関関係を導くことができるのかを正確に分析ができず, 何を改善すればいいのかが不明確だということである. なので, 視線の動きを正確に追跡することができれば, この’間’を埋めることができ, 新たな発見や改善ができるということで, アイトラックは急成長している.

そこで, 初心者と上級者のタッチタイピング中の視線の違いに注目することは, 初心者が自身のスキルの未熟さを理解する上で, 学習効果が非常に高いことが期待される. そして, アイトラッキングカメラ (Eye Camera) と主観視点カメラ (World Camera) を搭載した, メガネ型のウェアラブル端末を使いタイピング中の視線を追跡する. アイトラックでは pupil

labs 製の製品を使う. 様々なアイトラックが世にある中で, この製品は値段が抑えられており, かつ高性能で多種多様の設定が可能となる.

また, 西谷研究室が以前開発したタッチタイピングを行う際にタイピングソフト（shunkun-type）を使用し被験者からデータを取得する. このソフトは github 上に OSS で公開されている.

アイトラックを用いた視線追跡を行う設定は多種多様あり, 正確に追跡しようとすれば予期していない散らばった動きを記録してしまう場合が発生してしまう. この”目線の散らばった動き”になってしまう原因にあるのが前段階に行う calibration にあると判明した. 本研究は, この calibration の精度を高める手法及び, 瞳孔を追跡し目の動きと被験者からタイピング速度のデータを取得し, 初学者と上級者の目線の違いを理解することを目的とする.

第2章　手法

2.1　pupil coreについて

アイトラッカーにはpupil labs製のpupil coreという製品を使う.USB経由でLinux,Mac,WindowsのPCやタブレットと接続し、Pupil Capture,Pupil Playeといった無償のソフトウェア上でデータを記録することができる.Pupil capture は主にリアルタイムでの視線 (world view) と左右の瞳孔の動き (合計 3 つのカメラ) を合わせたデータとして表示し記録する．そして,pupil player は　pupil capture で記録したものをドラックアンドドラップすることで,本人からの目線ビデオを構築することができる.

2.1.1　操作手順

pupil core と pupil capture を繋いだ時の手順:
pupil capture を起動→setting を押す→ detective を押す (目をリアルタイムで見ることができる) → R ボタンで録画→→ calibration で視線確認→タイピングから視線データを収集する．接続した時のホーム画面 (world window) を図 2.1 に示し, 図に書いてある数字を以下に書く.

- 1.CPU や FPS の値が表示されている
- 2. 主に C, T, R のボタンがある. それぞれ Calibration, start and stop(検証の開始と停止), Recording である.
- 3. 設定画面
- 4. サイドメニューバー

図 2.1: ：world window[3].

2.2 shunkuntypeについて

shunkuntype はタッチタイピングの基礎であるホームポジションの練習から正しい指の定位置を練習することができる.

2.2.1 ダウンロード手順

shunkuntype:
主に terminal からインストールをする.

- sudo gem install shunkuntype

2.3 shunkuntype コマンド一覧

shunkuntype を使用する際, 下記のコマンド一覧にある shunkuntype -d, shunkuntype -c, shunkuntype -h を主に利用する. shunkuntype のコマンドを以下に示す.[図 2.2]

```
koki@imotokoorenoair ~> shunkuntype
Shunkuntype says 'Hello world'.
Usage: shunkuntype [options]
    -v, --version                   show program Version.
    -c, --check                     Check speed
    -d, --drill [VAL]               one minute Drill [VAL]
    -h, --history                   view training History
    -p, --plot                      Plot personal data
    -s, --submit                    Submit data to dmz0
        --reset                     reset training data
        --review [FILE]             Review training, TAGs=html or hiki
```

図 2.2: ：shunkuntype のコマンド画面.

2.3.1 shunkuntype 特徴と使い方

＜ shunkuntype -d について＞

- d の意味は drill(ドリル)
- shunkuntype -d X （X に数字を入れるとそのドリルができる）
- 1 分間で出てくる英単語を打つ.
- ドリルによって文字の長さや配置が異なる.
- 記号などの配置も覚えることができる.
- 間違えても文字を消さず, 進める.(ホームポジションをすぐ覚えやすくするため)
- cat ~/.shunkuntype/training_data.txt で履歴を見ることができる.

＜ shunkuntype -c ＞

- c の意味は check(確認)
- ドリルと違いこのコマンドは check なので 20 個の英単語を何秒で打ち終わるかを計測する.
- ただし,check するときの単語が少ない.
- cat ~/.shunkuntype/speed_data.txt で履歴を見ることができる.

＜ shunkuntype -h ＞

- h の意味は history(全体確認)
- shunkuntype -d でのドリルの一覧を確認することができる.
- 全部で 97 個の練習ドリルが用意されている.
- 練習されていないところは空白となる.

2.4 データの取得方法

データを取得するために, 関西学院大学人間福祉学部河鰭ゼミ (計 8 人) に協力をいただいた. 8 人の中には, 初心者とある程度タイピングができる者とで shunkuntype -c から速さのデータを取り, その比較を行うことにした. 条件を以下に示す.

- shunkuntype を初めて使う段階で shunkuntype -c(確認）データを取る.
- データ取得から 4 週間 shunkuntype -d で練習し, その後同様にデータを取得する.
- 練習前と練習後で速さを比較する.

2.5 Calibration

Calibration の意味は計測器と標準器との調整であるが, アイトラックでの Calibration はアイトラックマシンが計測する瞳孔の情報と実際の視線とのマッピングを取ることである.

- 瞳孔から視線座標へのマッピングを確立する.
- eye camera から目を緑色の枠内に入れる.
- 用途に合わせて calibration の種類を選択.

R ボタンで録画されたものは, 下記に記した 00X(001 から順番に動画が入る) に保存されている. 以下にこの操作によって生成されるファイルディレクトリにあるファイルとその内容をまとめている.

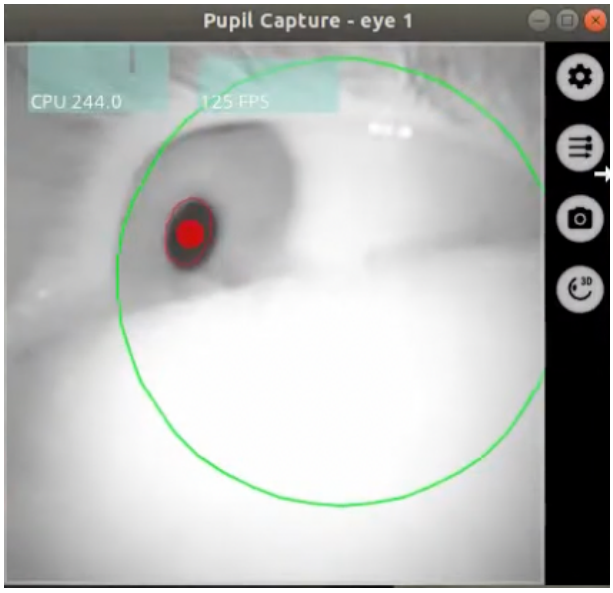

図 2.3: ：eye camera.

- 20XX_XXXX(自分で指定することができるフォルダ) その日の年月日

|- 00X(記録した順に 001 から番号が割り当てられるフォルダ)

|-- info.csv --- ファイルのメタデータが記録されたファイル

|-- user_info.csv --- 記録したユーザ情報を記録するためのファイル (自分で編集する)

|-- world.mp4 ---外側カメラの映像情報*

eye0.mp4 --- 内側カメラの映像情報*

|-- eye0.mp4

|-- world_timestamps.npy --- 外側カメラの映像のタイムスタンプ*

|-- ey0_timestamps.npy ---内側カメラの映像のタイムスタンプ*

|-- camera_calibration ---キャリブレーション情報が記録されたファイル

|-- pupil_data --- Python から見ることができる生データ

|-- exports(勝手に作られるフォルダ)

|--- pupil_gaze_positions_info.csv ---各パラメータの説明が記載されたファイル

|--- pupil_postions.csv --- 瞳孔の詳細情報が記録されたファイル

|--- gaze_positions.csv --- 視線の詳細情報が記録されたファイル

*は Pupil player を用いてしか見ることができない [2].

2.6 Calibrationの種類

2.6.1 Screen Marker Calibration 手順

図 2.4 に Screen Marker Calibration の様子を示した. このキャリブレーションは, デフォルトの手順であり, テレビや PC などの画面の時に行う. これは, 画面上の 5 つのマーカーを見ることでキャリブレーションを行っているため,2D が最も適応している.

1. Screen Marker Choreography を選択する
2. Monitor（複数のモニターの場合）を選択する.
3. c キーボードを押すか, ワールドウィンドウの左側にある青い円形のボタンをクリックして, キャリブレーションを開始する.
4. 画面上のマーカーを目で追ってください. キャリブレーション中は頭を動かさないようする.
5. キャリブレーションが完了すると, キャリブレーションウィンドウが閉じる [3].

2.6.2 Single Marker Calibration 手順

図 2.5 に Single Marker Calibration の様子を示した. このキャリブレーションは,2D や 3D といったほとんどの場合に適応することができる. これは, 円形キャリブレーションマーカーの中心を見ながらぐるぐると頭を動かすことで広範囲の注視角度をすばやくサンプリングすることができるからである.

1. Single Maker Choreography を選択する.
2. C キーボードを押すか, ワールドウィンドウの左側にある青い円形のボタンをクリックして, キャリブレーションを開始する.

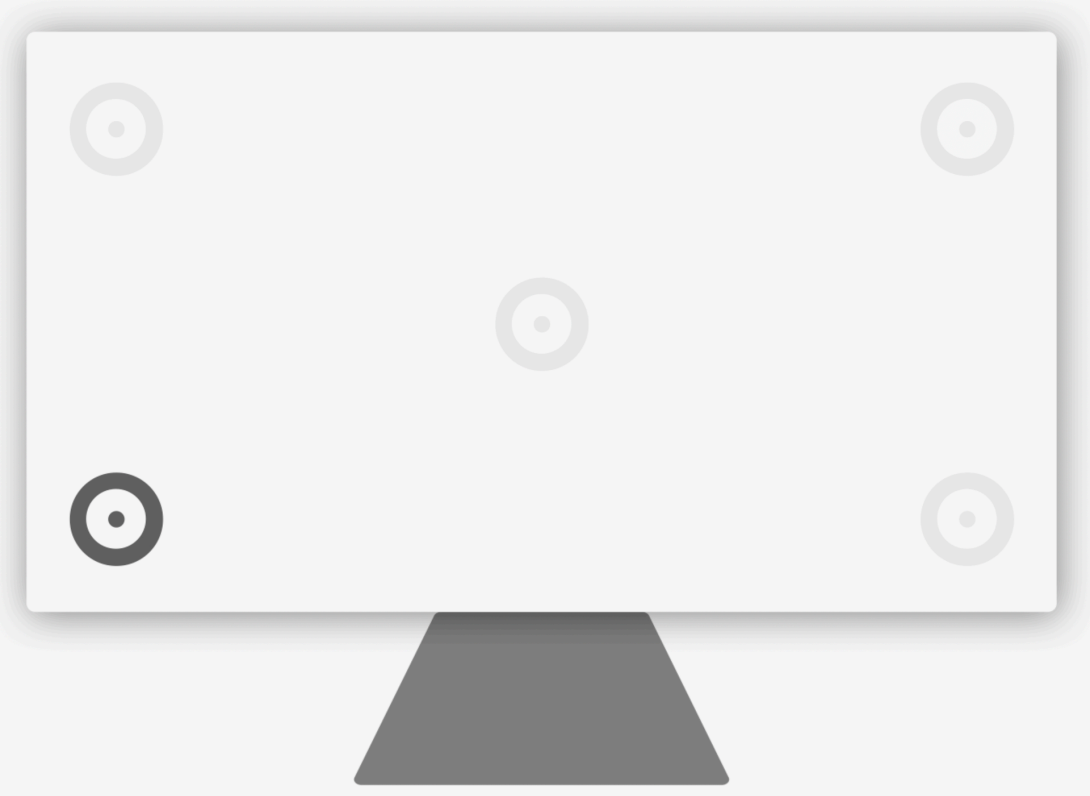

図 2.4: ：Screen Marker Calibration[3].

3. マーカーの中心を見る.
4. マーカーの中心を見ながらゆっくりと頭を動かします. 首を動かしながらすることは,FOV=Field of View（見える範囲）の広い領域をカバーする効率的な方法である.
5. C キーボードのボタンを押すか, 停止マーカーを表示して、キャリブレーションを停止する [3].

2.6.3 Natural Features Calibration 手順

図 2.6 に Natural Features Calibration の様子を示した. このキャリブレーションは, 特殊な場合にのみ使用する. 例えば, 本は一見 2D であるけれど, 読むときに奥行きが出ます. この奥行きによって見ている場所とアイトラックの視線にずれが生じるためこのようなキャリブレーションを使用する.

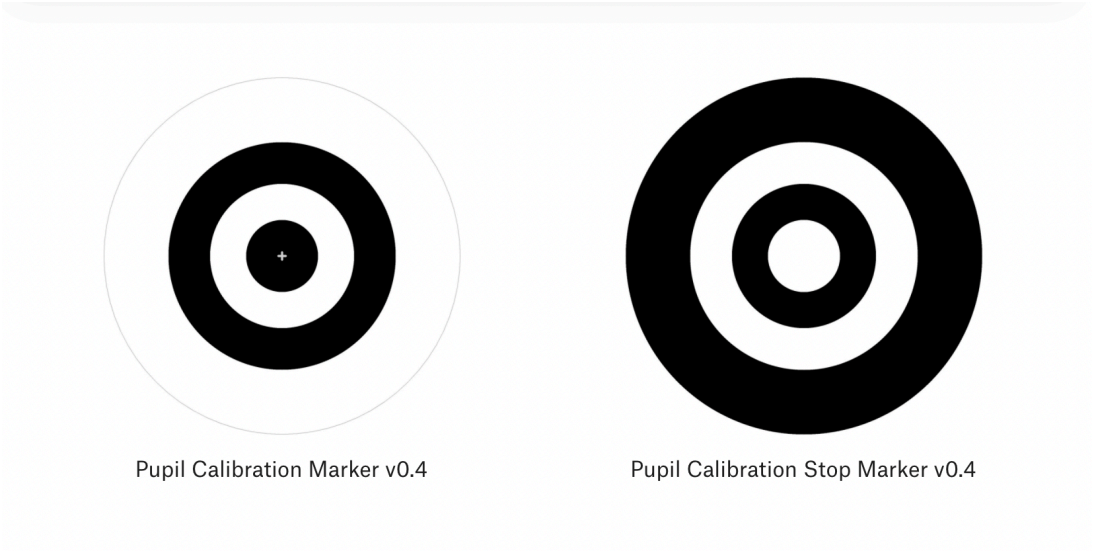

図 2.5: ：Single Marker Calibration[3].

1. Natural Features Calibration Choreography を選択する.
2. c キーボードを押すか, ワールドウィンドウの左側にある青い円形のボタンをクリックして, キャリブレーションを開始する.
3. 被験者（瞳孔ヘッドセットを装着している人）に視界内のポイントを見るよう指示する.
4. ワールドウィンドウでそのポイントをクリックする.
5. データがサンプリングされる.
6. 対象の視野をカバーするまで繰り返す（通常、約 9 箇所）
7. c キーボードを押すか, ワールドウィンドウの左側にある青い円形のボタンをクリックして, キャリブレーションを停止する [3].

2.7 データの視覚化

タイピングが早くなるには練習量とその期間が重要になってくる.

タイピング特性に関する従来の研究を述べる. タッチタイピング教育システムの開発と運用して得られた学習データ（速度, ミス率) の解析を行い, タイピン

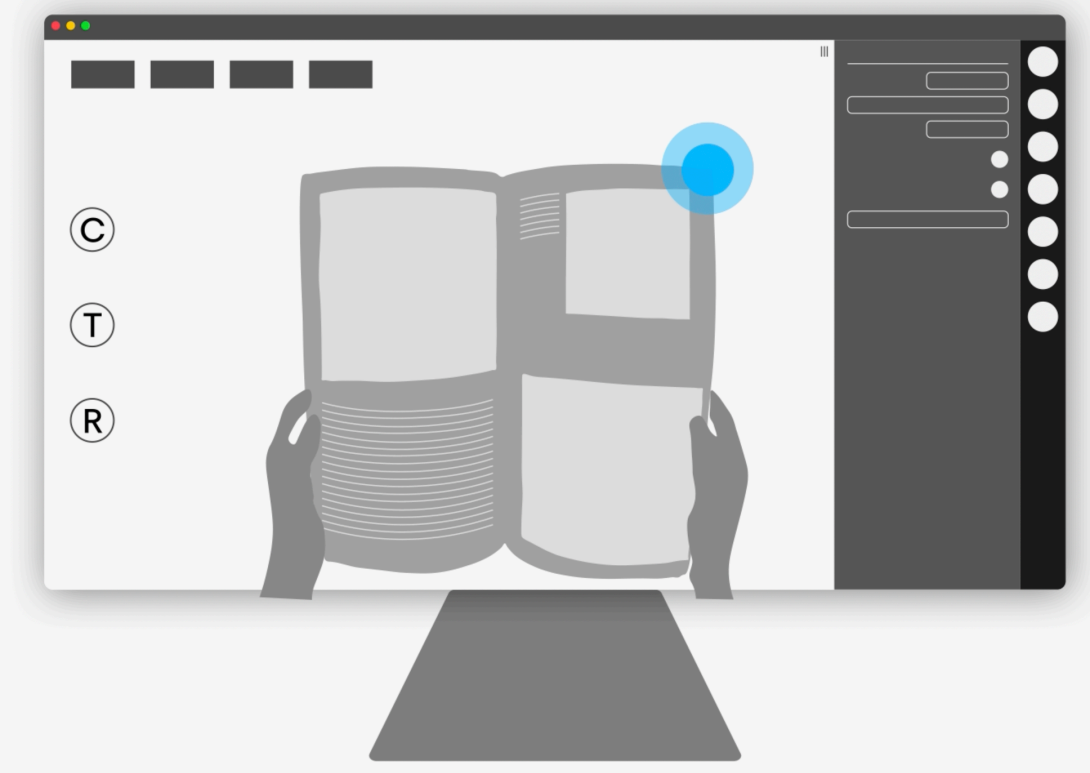

図 2.6: ：Natural Features Calibration[3].

グ教育の成長に焦点を当てている. その結果, タイプミス低下率と速度上昇率の二つには相関があり, また学習を日数を空けずに継続して行うことが重要であるということを述べている [4].

そこで, タイピングデータを, その日にした練習回数の積算と初日からの経過日数の自動的に抽出しそれをグラフ化させるプログラムを作成する.

- shunkuntype 終了後自動的に training_data.txt 内に保存される.
- 保存されたデータから必要なデータ (その日にした回数の積算と初日からの経過日数) を取り出す.
- ターミナルで　ruby ～.rb training.txt ＞ ～.csv　に変換.
- エクセルでグラフ作成

ソースコードは以下の通りである.

```
require 'date'
```

```
/** ファイルの読み込み */
file = ARGV[0] || 'training_data.txt'
lines = File.readlines(file)
/** 区分け */
init = Date.parse(lines[0].split(',')[0])
data = [[0,1]]
lines[1..-1].each do |line|
  date = Date.parse(line.split(',')[0])
  cur_data = data[-1]
  cur_date = (date-init).to_i
  if cur_data[0] == cur_date
    data[-1][1]+=1
  else
    data << [cur_date, 1]
  end
  # ["2021-04-08 15:03:33 +0900", "STEP-1.txt", "40", "60\n"]
  # [[4/8, 1], [4/9, 3], [4/13, 2]]
end
/** 日数を積算 */
sum = 0
new_data = []
data.each do |data|
  sum += data[1]
  puts "%4d, %4d" % [data[0],sum]
end
```

第3章　結果

3.1　calibration精度向上

calibrationが上手く行われていないと見ている目線が違ったり, 枠内から目線が外れるということがある. よって, 自身が実験したことで改善する設定や手順を以下に示す.
まず, 目と画面の距離を一定にする. 頭を固定し目だけを追うようにcalibrationの手順をする. これはscreen mekerのみ限定される. 理由はそれ以外のcalibration手順には,2Dと3Dに対応しているため距離を一定にしていなくても対応することができるからである. 逆にsingle maker calibrationでは, 頭をぐるぐると動かしながら目だけマーカーを見るようにする. 頭を動かすことで広範囲の領域にカバーすることができるからである.
natural features calibrationは複数の自分たちが決めたポイントを見ることでcalibrationできるシステムであるため, より正確な追跡には多くのポイントを見る必要がある.
そして, どのキャリブレーションにも当てはまることであるが, 目線がぶれる理由はcalibrationを終えた後に出る赤色の枠外にはみ出るか小さいため, トラッキングしたい目標を捉えきれていない. なので,eye cameraに映っているリアルタイムの動画は, 緑色の円内に収めるようにする. 枠が出てきない時は一度ソフトを落とすかアイトラックのカメラの装着をやり直すことを勧める. eye cameraは, リアルタイムで目の瞳孔を見ることができ, そのリアルタイムで見ることができるモードにも種類がある. 以下にそれを示す.

Algorithm Mode:2つの赤い円は, 最小および最大の瞳孔サイズ設定を表す. 緑の円は, 現在の見かけの瞳孔サイズを視覚化する. 緑の円（現在の瞳孔のサイズ）がすべての眼球運動の最小/最大範囲内になるように, 最小値と最大値を設定する.

Intensity Range：瞳孔と見なされるピクセルの最小の「暗さ」を定義する. 瞳孔検出の対象となるピクセルは, 青色で視覚化される。瞳孔の外側への漏れをできるだけ少なくしながら, 瞳孔が常に完全に覆われるように, 範囲を最小化

する [3].

また, 設定にも精度を高める手順があり以下にそれを示す.

- 設定の video source の Freme size を 30 から 60 に上げる.
- 次の calibration を実行する時は, 一度ソフトを落とす.

一度ソフトを落とす理由は, 前の calibration がまだ反映されている可能性がある. これは実際キャリブレーションの手順をしている際すでに終わった状況になったことがあったためである.

上手く calibration が出来ていると, 注視すれば黄色の枠が出ることや見ているところとアイトラックでの視線が合うようになる. しかしながら, リアルタイムの動画ではこの黄色の枠は出るが, 録画した動画では枠は消える. 最後にそれぞれの種類の calibration を実験する中で, 主観的であるが最も精度が良く上手くキャリブレーションできるのが single marker calibration であった. 他のキャリブレーションも上手く行く時があるが, 実際目線が散らばることが多々発生した.

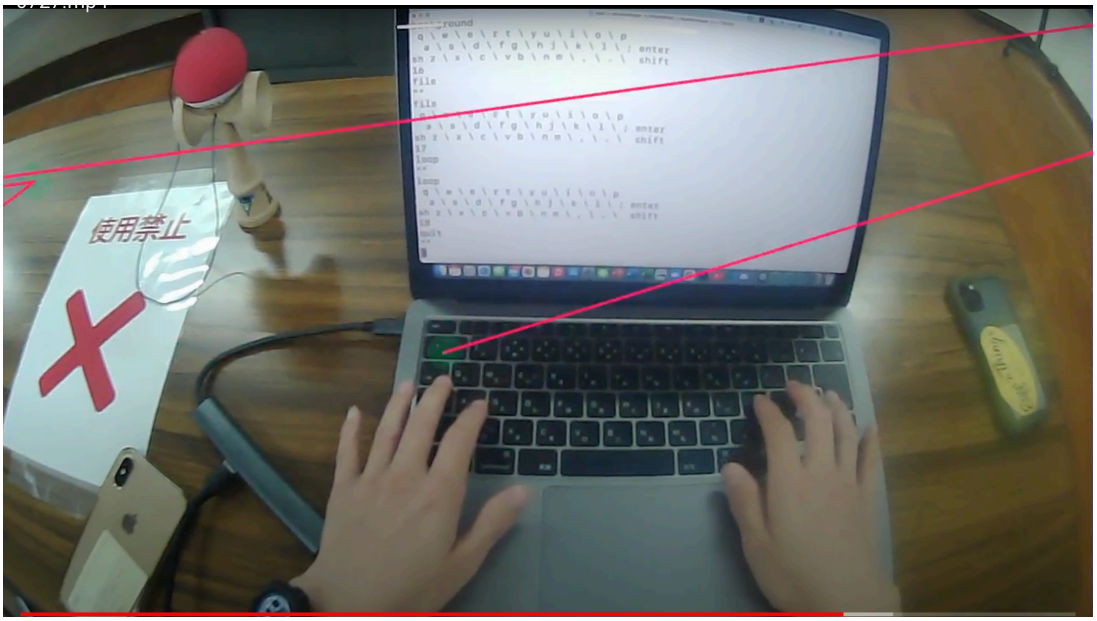

図 3.1: ： calibration 向上前.

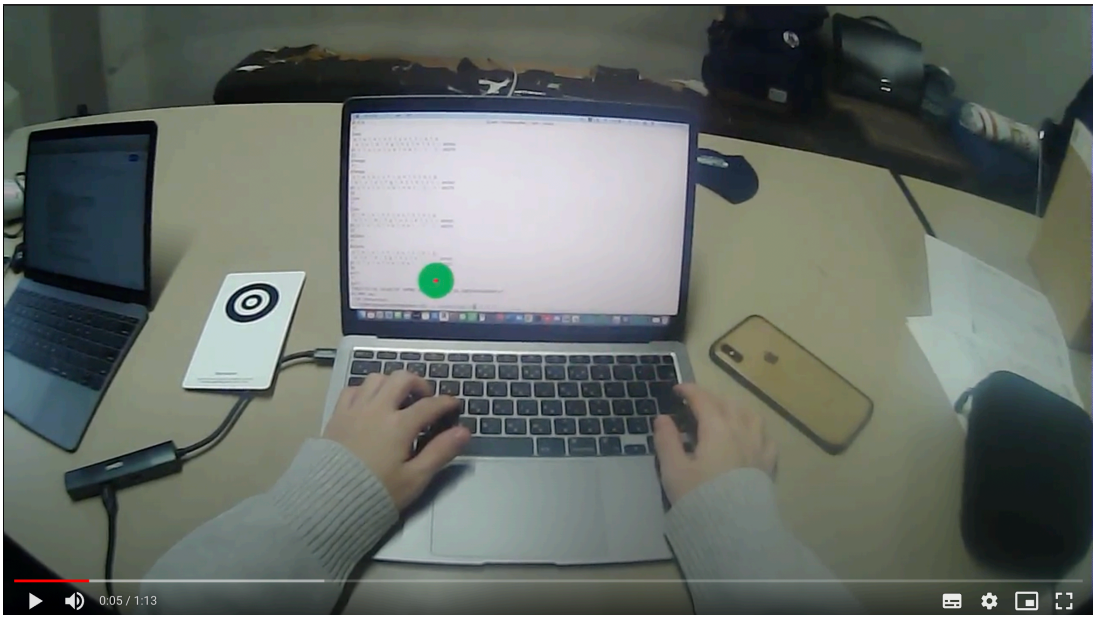

図 3.2: ： calibration 向上後.

3.1.1　calibration の精度の向上前 [図 3.1] と向上後 [図 3.2] の比較.

calibration の向上前と向上後の図から”目線の散らばった動き”ではなく, 視線がまとまっていることがわかる. 見ている目線とアイトラックの目線がしっかりと合い被験者がどこを見ているのかがわかるようになった. 以前までは calibration のやり方が上手くいかず視線が画面の外や見ていないはずの場所を見ることがあった. しかし, それがなくなるようになり注視すれば黄色の枠も現れるようになった。これは私見であるが, この枠の出現は向上前にはほとんど見られないものだった. けれども設定や手順を何度も踏んだことがこの calibration の精度を高くしたことに繋がった. 今回のキャリブレーションの精度を上げることにおいて,2D だけでなく 3D といった範囲までも向上することができた. アイトラックを用いてタイピングの視線追跡で２ D であるが,3D もできるようになったことでさらに汎用性が高くなったといえる.

3.2　練習量のグラフ化

グラフ化するために必要な条件として,shunkuntype を始める前の段階と 4 週間練習した後の状態とする. そこで, 練習量のデータを取るために関西学院大学人間福祉学部の河鰭ゼミ (計 8 人) に協力をいただいた. 8 人には shunkuntype -c(speed_data) を行いタイピ

ングの速さのデータを取った.

Ruby 言語のプログラムから被験者の練習量 (training_data) をグラフ化することができるようになった. これによって, 個人の練習量を可視化させるためにターミナルで実行し, グラフを作成することができる. また図 3.3 のグラフを見るとわかる通り, 縦軸に 1 日にした練習回数の積算で横軸には初日からの経過日数を表したグラフである. 横軸の 40〜105,120〜180 の期間は一定であり, 練習が不十分であることがわかる. 逆に 180〜240 の期間では練習を多くしたことがわかる. このようにどの時期ができていないかや集中して取り組んだ期間を瞬時にわかるようになる. そして同様に, 被験者のデータを speed_data を取ることで速さのグラフを出し, そこに相関は見られるかというグラフを作成しようと試みた. しかしながら, データの取り方として,PC を 1 台で 8 人分のデータ (speed_data) を取ってしまい, それぞれのグラフを作れない状態になってしまったことが反省点として挙げられる.

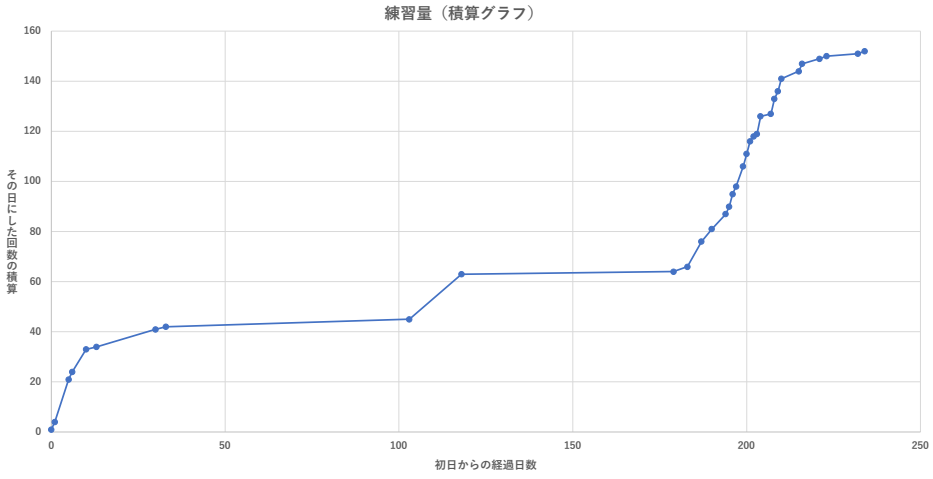

図 3.3: : 練習量グラフ.

第4章　考察

calibration 後に画面に表示される数字は calibration の正確の度数を示すものだという見解を持っている. しかし, その数字が高い時と低い時での差が顕著に現れていない.

- 数字は画面全体における calibration の範囲である.
- 精度であるが, 差が見られないだけである.

また,calibration の精度が上がったけれど, 目線が散らばった動きをするときが多少動画内であり, それが,calibration が上手くいっていないのか, 本当に無意識的にそういった目の運動がされているのかがわからない.

- 画面外に行く時や連続して散らばる運動が見られる時, アイトラックは上手く calibration できていない.
- 2D の場合では, カメラのレンズの歪みや焦点距離などによってずれが生じている.

こういった問題が今後の課題となる.

謝辞

本研究の遂行にあたって，終始多大なる有益な御指導，及び丁寧な助言をいただいた西谷滋人教授に深く感謝いたします．また，本研究を進めるにつれて，西谷研究室に所属する同輩たち，人間福祉学部河鰭ゼミ生の皆様, 並びに先輩方からの様々な知識の供給，ご協力をいただき，本研究を成就させることができました．この場を借りて心から深く御礼申し上げます．

参考文献

[1] BAE 編集部, ”AI ×アイトラッキングが生み出した、「視線推測」のポテンシャルとは？”, https://bae.dentsutec.co.jp/articles/hitokuse/.

[2] 牧野　考史, ”Pupil Labs のデータの見方”, https://qiita.com/makky0620/items/07dfe5414f5a38e322d1.

[3] Pupil labs 公式, ”Pupil labs”, https://pupil-labs.com/products/core/.

[4] 高岡　詠子, ”初心者のタイピング動作特性の解析”, file:///Users/koki/Downloads/typint_IPSJ-CE13120009%20(1).pdf.

[11pt]article [utf8]inputenc [T1]fontenc graphicx longtable wrapfig rotating [normalem]ulem amsmath amssymb capt-of hyperref 2022 年 6 月 23 日

目 次