誰でも使用できる視線入力装置のプログラミング研究

小池 麻悠子

【概要】
視線入力装置は、眼球の動きから注視している位置を検知し，各種データの入力を行う機器のことである。今のところ、肢体不自由者や AL 重度障がい者用の視線入力装置は流通しているが、高価であるため簡単に手に入らない。
そこで本研究では、先行研究の笹竹佑太さんが発明した「Python Web カメラで eye tracking（アイトラッキング・視線計測）をする」を使って、視線入力するプログラミングを目的に研究した。アイトラッキングとは、視点の場所や、眼球の動きを計測し、追跡する方法であるため、これを点に可視化させることでパソコンの画面上の文字と重ね合わせ、文字の入力ができるようにプログラミングを行った。結果、「あ、い、う、え、お」の視線入力に成功した。

A visual input line input device can detect the position that is being looked at based on eye movements and can input this data. Now, visual line input devices for physically disabled people and AL individual with developmental disabilities is in circulation, but it is not easily obtainable because it is so expensive. So, the purpose of this study is to create a program that visual line input device by using the “Eye tracking with a web camera” made by Sasatake Yuta on python. Eye tracking means to measure the eye’s movement and viewpoint, so I programed to put it on top of the letter on the screen of the PC by visualizing a point and to be able to input the letter. As a result, I succeeded in entering “A, I, U, E, O.”.

【実験器具】
・ Windows10（Corei5）
実験環境は屋内とし、照明として一般的な卓上スタンドライトを使用。

【実験方法】
眼球の動きをアイカメラが解析する角膜反射法では、LED を用いて眼球に赤外線を照射するが、今回の研究では、どこの家庭にでもある卓上ライトを使用した。また、実験を行うときはソフトコンタクトレンズを着用したままで行った。

【実験結果】
(1) 画面に文字を表示した。
Python に以下を入力した。
>>>import tkinter as tk
⇨「Tkinter」というファイルを開く。
Tkinter を tk と省略する。
>>>root = tk.Tk()
⇨メインウィンドウを作成する。
>>>root. geometry("1250x250")
⇨メインウィンドウを 1250×250 の大きさにする。
>>>canvas = tk.Canvas(root, bg = "white")
⇨メインウィンドウを白で塗りつぶす。
>>>canvas.pack(fill = tk. BOTH, expand = True)
⇨大きさの変動が可能なキャンバスを作成する。
>>>canvas. create_line (250, 0, 250, 250, fill = "Blue", width = 5)
⇨(250,0)から(250,250)までに太さ 5pt の青の線を引く。

>>>canvas. create_line (500, 0, 500, 250, fill = "Blue", width = 5)

⇨(500,0)から(500,250)までに太さ 5pt の青の線を引く。

>>>canvas. create_line (750, 0, 750, 250, fill = "Blue", width = 5)

⇨(750,0)から(750,250)までに太さ 5pt の青の線を引く。

>>>canvas. create_line(1000, 0, 1000, 250, fill = "Blue", width = 5)

⇨(1000,0)から(1000,250)までに太さ 5pt の青の線を引く。

>>>canvas.create_text(125, 150, text = "あ" ,fill = "Blue", font = ("",125)) = tk1

⇨(125,150)の位置に大きさ 125pt の「あ」を書く。それを tk1 とする。

>>>canvas.create_text(375, 150, text = "い" ,fill = "Blue", font = ("",125)) = tk2

⇨(375,150)の位置に大きさ 125pt の「い」を書く。それを tk2 とする。

>>>canvas.create_text(625, 150, text = "う" ,fill = "Blue", font = ("",125)) = kt3

⇨(625,150)の位置に大きさ 125pt の「う」を書く。それを tk3 とする。

>>>canvas.create_text(875, 150, text = "え" ,fill = "Blue", font = ("",125)) = tk4

⇨(875,150)の位置に大きさ 125pt の「え」を書く。それを tk4 とする。

>>>canvas.create_text(1125, 150, text = "お" ,fill = "Blue", font = ("",125)) = tk5

⇨(1125,150)の位置に大きさ 125pt の「お」を書く。それを tk5 とする。

図 2　画面上の文字

(2)　点を可視化した。

先行研究の "C:¥Users¥mayuko15¥Downloads¥eyetrack-main¥eyetrack-main¥Eye_Tracking_Program_by_dlib¥eye_tracking.py"に、以下を入力した。

※mayuko15 は個人情報を表すので、伏せています。自分のディレクトリ名に合わせてください。

>>>eye_img_gray = cv2.cvtColor(eye_img, cv2.COLOR_BGR2GRAY)

⇨先行研究の視線計測した画像をモノクロ化(＝グレースケール化)する。それを eye_img_gray とする。

>>>eye_img_negative = 255 - eye_img_gray

⇨モノクロ化した画像の白と黒を反転する。それを eye_img_negative とする。

>>>eye_img_negative.shape

⇨デスクトップ上に eye_img_negative を表示する。

>>>eye_mask = np.zeros_like(eye_img_negative)

⇨眼球の部分だけを取り出す。(＝マスク処理)

>>>eye_mask = cv2.fillConvexPoly(eye_mask, np.array(landmark_local), True, 1)

⇨マスク処理した画像の眼球に点を打つ。

>>>eye_img_negative_masked = np.where(eye_mask == 1, eye_img_negative, 0)

⇨点と点を線で結ぶ。

>>>height, width = eye_img.shape[:2]

⇨画像を大きく表示する。

>>>sum_x = np.sum(eye_img_negative_masked, axis=0)

⇨縦列ごとの和を求める。

>>>v = np.array([0.1, 0.2, 0.4, 0.2, 0.1])

⇨加重平均を求める。

>>>weighing_moving_ave_y =
np.convolve(sum_y, v, mode='same')
⇨移動平均を求める。
>>>fig = plt.figure(figsize=(6, 8))
⇨正方形を x 軸=6 個,y 軸=8 個積み重ねた大きさのグラフにグラフ化する。
>>>plt.subplots_adjust(wspace=0.5)
⇨グラフを 0.5 の幅に調整する。
>>>ax1 = fig.add_subplot(2, 1, 1)
⇨加重平均をグラフ化する。
>>>ax1.imshow(eye_img_negative_masked)
⇨eye_img_negative_masked を表示する。
>>>ax2 = fig.add_subplot(2, 1, 2)
⇨移動平均をグラフ化する。
>>>ax2.plot(sum_x)
⇨Sum_x に移動平均のグラフをプロットする。
>>>ax2.set_xlim([-0.5, 36.5])
⇨x 軸を-0.5 から 36.5 の範囲に制限する。
>>>sum_y =
np.sum(eye_img_negative_masked, axis=1)
⇨横列ごとの和を求める。
>>>v = np.array([0.1, 0.2, 0.4, 0.2, 0.1])
⇨加重平均を求める。
>>>weighing_moving_ave_y =
np.convolve(sum_y, v, mode='same')
⇨移動平均を求める。
>>>fig = plt.figure(figsize=(3, 12))
⇨正方形を x 軸=3 個,y 軸=12 個積み重ねた大きさのグラフにグラフ化する。
>>>plt.subplots_adjust(wspace=0.5)
⇨グラフを 0.5 の幅に調整する。
>>>ax1 = fig.add_subplot(2, 1, 1)
⇨加重平均をグラフ化する。
>>>ax1.imshow(cv2.rotate(eye_img_negative_masked,
cv2.ROTATE_90_COUNTERCLOCKWISE))
⇨eye_img_negative_masked,
cv2.ROTATE_90_COUNTERCLOCKWISE を 90°回転させた画像を表示する。
>>>ax2 = fig.add_subplot(2, 1, 2)
⇨移動平均をグラフ化する。
>>>ax2.plot(weighing_moving_ave_y)
⇨Weighing_moving_ave_y に移動平均のグラフをプロットする。
>>>ax2.set_xlim([-0.5, 20.5])
⇨x 軸を-0.5 から 20.5 の範囲に制限する。
>>>pupil_x =
np.argmax(weighing_moving_ave_x)
⇨weighing_moving_ave_x の中で一番値が大きいものが左から何番目かを求める。
>>>pupil_y =
np.argmax(weighing_moving_ave_y)
⇨weighing_moving_ave_y の中で一番値が大きいものが左から何番目かを求める。
>>>print(f'center：({pupil_x}, {pupil_y})')
⇨重心を求める。
>>>eye_img_copy = eye_img.copy()
⇨eye_img.copy()をおく。（＝eye_img_copy）
>>>cv2.line(eye_img_copy, (pupil_x - 2, pupil_y), (pupil_x + 2, pupil_y), (0, 255, 0))
⇨(pupil_x - 2, pupil_y), (pupil_x + 2, pupil_y)の重心位置に緑色で x 印を書く。
>>>cv2.line(eye_img_copy, (pupil_x, pupil_y - 2), (pupil_x, pupil_y + 2), (0, 255, 0))
⇨(pupil_x, pupil_y - 2), (pupil_x, pupil_y + 2)の重心位置に緑色で x 印を書く。
>>>tmp_img_RGB =
cv2.cvtColor(eye_img_copy, cv2.COLOR_BGR2RGB)
⇨×印を重ね合わせる。
>>>eye_img_gray = cv2.cvtColor(eye_img, cv2.COLOR_BGR2GRAY)
⇨重ね合わせた画像をカラーデータにする。
>>>plt.imshow(tmp_img_RGB)
⇨Tmp_img_RGB を表示する。

(3) 文字が入力されるようにした。
先行研究の
"C:¥Users¥[redacted]¥Downloads¥eyetrack-main¥eyetrack-main¥Eye_Tracking_Program_by_dlib¥eye_tracking.py"に以下を入力した。

```
>>>if tmp_img_RGB = tk1：
        print("あ")
    elif tmp_img_RGB = tk2：
        print("い")
    else:
        print("う")
```

⇨もし、tmp_img_RGB が tk1 にあったら、「あ」を入力する。それ以外で、tmp_img_RGB が tk2 にあったら、「い」を入力する。それ以外は、「う」を入力する。

```
>>>if tmp_img_RGB = tk5：
        print("お")
    elif tmp_img_RGB = tk4：
        print("え")
    else:
        print("う")
```

⇨もし、tmp_img_RGB が tk5 にあったら、「お」を入力する。それ以外で、tmp_img_RGB が tk4 にあったら、「え」を入力する。それ以外は、「う」を入力する。

以上より文字を入力することができた。

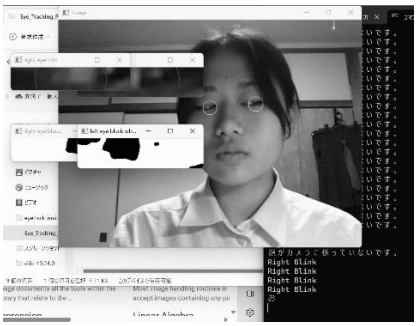

図 3　全体の画面

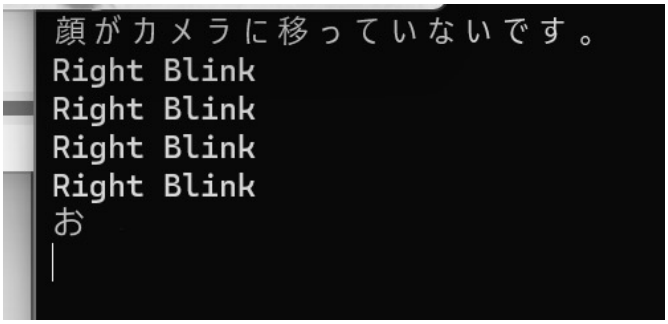

図 4　文字の入力

しかし、プログラミングが違うのかわからないが文字数を増やすと文字を打つことができなかった。

【結論】

実際に病院や特別支援学校で使われるような性能を持つ視線入力装置のプログラミングを作ることはできなかったが、基盤となるプログラミングは費用をかけずに作ることができた。しかし、パソコンの画面でもっと文字数を増やすと視線入力ができないことが分かったので、今後は、プログラミングについてもっと学習し、今後の研究で改善していこうと思う。

【参考文献】

佐竹佑太,Web カメラで eye tracking をする,qitta,2021-1-21
塩塚敬介,10 度肢体不自由教育における視線入力装置活用の現状と課題,教育情報研究,2020,Vol 35,No.2,p.3-14
小枝 正直,上田 悦子,中村 恭之, OpenCV による画像処理入門 改訂第 3 版 (KS 情報科学専門書), 講談社,2022-12-8
永田 雅人,豊沢 聡,実践 OpenCV 3 for C++画像映像情報処理, カットシステム,2017-8-1