

令和4年度卒業研究報告書

Webカメラによる視線トラッキングに関する研究

情報技術科 小松 篤志

指導教員 ソソラ

目次

第 1 章 はじめに.....	4
第 2 章 研究概要.....	5
2.1 概要.....	5
2.2 開発環境.....	5
第 3 章 眼の検出について.....	6
3.1 カメラの入力・画面表示方法.....	6
3.2 眼の検出.....	7
第 4 章 視線方向検出.....	9
4.1 全体構造について.....	9
4.2 左右の判定.....	9
4.2.1 初期案.....	9
4.2.2 正式アルゴリズム案.....	10
4.2.3 アルゴリズムの詳細.....	10
4.2.4 実験・検証.....	12
4.3 下向きの判定.....	17
4.3.1 初期案.....	17
4.3.2 正式アルゴリズム案.....	17
4.3.3 アルゴリズムの詳細.....	17
4.3.4 実験・検証.....	19
4.4 上向きの判定.....	22
4.4.1 初期案.....	22
4.4.2 正式アルゴリズム案.....	22

4.4.3	アルゴリズムの詳細	22
4.4.4	実験・検証	23
4.5	eye_direction()の引数	26
第 5 章	視線方向取得モジュールとしての実装	27
5.1	外部プログラムからの呼び出し方法	27
5.2	外部から呼ばれるメインとなるメソッド	28
5.3	値取得用メソッドの実装	29
5.4	ループ終了用メソッドの実装	30
第 6 章	アプリの作成	31
6.1	アプリの概要	31
6.2	アプリ仕様	32
6.3	ライブラリ「kivy」について	32
6.3.1	Kivy とは	32
6.3.2	選定理由	32
6.3.3	Kivy の基本	33
6.4	GUI 設計	35
6.4.1	レイアウト	35
6.4.2	メイン画面のネスト構造	36
6.4.3	操作説明画面のネスト構造	36
6.4.4	すべてのタグに使えるプロパティの説明	37
6.4.5	BoxLayout の縦横切り換え	38
6.4.6	Button のクリック処理	38
6.4.7	GUI の完成	38
6.5	視線方向取得モジュールの呼び出し	39

6.6 音を鳴らす仕組み.....	41
6.7 再生と停止の処理.....	42
6.8 BPM(テンポ)設定処理.....	43
6.9 音程モード切り替え処理.....	44
6.10 説明画面の表示方法.....	45
第 7 章 終わりに.....	46
参考文献.....	47

第1章 はじめに

近年、非接触の操作デバイスの需要が拡大しつつある。そこに、音声認識や体を使ったジェスチャーを画像認識するなどの多くの手法が使われている。一方、ある病気の場合は眼だけは動くが他の身体の部位は動かすことができないという事例があり、ジェスチャーだけでは操作できない。ここで、「眼の動きを使った表現」ができれば聴覚に障害を持った人でも使えると思った。

そこで、コミュニケーション手段の一つとして「眼の動き」に注目し、眼の動きをトラッキングして何かを操作できるツールを製作したいと考えた。

研究は以下の通りに進めた。

OpenCV や MediaPipe (FaceMesh) などの使い方を学習しながら、開発環境を設定した。

市販の WEB カメラを使用し、リアルタイムで眼の輪郭を含むキーポイントを取得し、EXCEL ファイルに書き出した。

キーポイントの座標データの解析による視線方向（右左上下）を判定するアルゴリズムを考えた。

取得した視線方向を活用し、音楽を演奏することができるデスクトップアプリケーションを作成した。



左を見ている様子



アプリメイン画面

第2章 研究概要

2.1 概要

本研究では, 特殊なハードウェアを必要とせずに, WEB カメラを使用してリアルタイムで虹彩, 瞳孔, および眼の輪郭を含むキーポイントにより, 眼の動きを追跡するアルゴリズムを開発する.

さらに眼の動きを利用したアプリケーションを作成する.

2.2 開発環境

本研究の開発環境を以下の表に示す. また, 開発環境のインストールおよびライブラリの説明は付録にて解説する.

OS	Windows 10
統合開発環境	Visual Studio Code
開発言語	Python 3.8.10
使用ライブラリ	OpenCV MediaPipe faceMesh Pandas Numpy Kivy Concurrent.ftures Winsound
カメラ	Integrated Camera(標準装備, 720p 30fps) C920n pro hd webcam(1080p, 30fps)

表 1 開発環境

第3章 眼の検出について

3.1 カメラの入力・画面表示方法

本研究のカメラ認識には「Opencv」を使用している。カメラの入力および画面出力の流れは下図に示す。

```
1  import cv2 as cv
2
3  cap = cv.VideoCapture(0)  #接続しているカメラの選択
4
5  while True:
6      ret, frame = cap.read() #1フレーム読み込み
7      if not ret:
8          break
9      frame = cv.flip(frame, 1) #表示される画像を鏡向きにする
10
11
12     cv.imshow('img', frame) #画面表示
13     key = cv.waitKey(1)     #キー入力待ち
14     if key == ord('q'):     #終了条件
15         break
```

図 1 カメラ入力から表示までのソースコード

VideoCapture()の引数で内蔵カメラ(標準カメラ)または後付けカメラの選択となる。対応引数は以下の表に示す。

引数に渡す値	認識する画像
0	内蔵カメラ (内蔵カメラがない場合)後付けカメラ
1	内蔵カメラ以外の後付けカメラ
動画のファイルパス	指定した動画

表 2 VideoCapture()の引数対応表

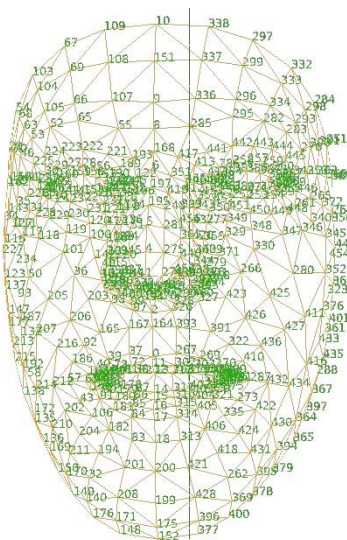
cap.read()は入力画像を1フレームしか読み込むことができないのでwhileの無限ループで読み込み、フレームを更新している。

waitKey(1)以降(図 1: 13 行目以降)などを使用し、無限ループを終了させる方法を記述しなければならない。

3.2 眼の検出

mediaPipe FaceMesh を利用し目の検出をする。顔に割り当てられている 468 点のポイントのうち以下の図に示す 40 点のポイントを利用する(以降これらのポイントをキーポイントと呼ぶ)。

参考文献【1】 参照



```
5 #目に割り当てられているキーポイントの値
6 LEFT_EYE = [362, 382, 381, 380, 374, 373, 390, 249, 263, 466, 388, 387, 386, 385, 384, 398]
7 RIGHT_EYE = [33, 7, 163, 144, 145, 153, 154, 155, 133, 173, 157, 158, 159, 160, 161, 246]
8 LEFT_IRIS = [474, 475, 476, 477]
9 RIGHT_IRIS = [469, 470, 471, 472]
```

図 2 眼に割り当てられたポイントの値

Facemesh を利用するために python の with 構文を使用する。With 構文はファイルのオープンからクローズまでを簡単にするものである。

Facemesh で画像を扱うためには cv2 で読み込んだ画像値(BGR)を標準値(RGB)に変換する必要がある。

虹彩の中心座標は図3の41,42行目で計算している。

```
14 mp_face_mesh = mp.solutions.face_mesh
15
16 with mp_face_mesh.FaceMesh(
17     max_num_faces=1,
18     refine_landmarks=True,
19     min_detection_confidence=0.5,
20     min_tracking_confidence=0.5
21 ) as face_mesh:
22     while True:
23         ret, frame = cap.read()
24         if not ret:
25             break
26         frame = cv.flip(frame, 1)
27         rgb_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB) #画像形式を変更(BGR → RGB)
28         img_h, img_w = frame.shape[:2] #画像の高さ, 幅を取得
29         results = face_mesh.process(rgb_frame) #facemeshに画像データを与える
30         if results.multi_face_landmarks:
31             #画像の中の眼の位置とキーポイントの対応付け
32             mesh_points = np.array([np.multiply([p.x, p.y], [img_w, img_h]).astype(int)
33                                     for p in results.multi_face_landmarks[0].landmark])
34             #目全体
35             #cv.polyline(frame, [mesh_points[LEFT_EYE]], True, (0, 255, 0), 1, cv.LINE_AA)
36             #cv.polyline(frame, [mesh_points[RIGHT_EYE]], True, (0, 255, 0), 1, cv.LINE_AA)
37             #瞳孔のみ(四角)
38             # cv.polyline(frame, [mesh_points[LEFT_IRIS]], True, (0, 255, 0), 1, cv.LINE_AA)
39             # cv.polyline(frame, [mesh_points[RIGHT_IRIS]], True, (0, 255, 0), 1, cv.LINE_AA)
40             #瞳孔のみ(○)表示
41             (l_cx, l_cy), l_radius = cv.minEnclosingCircle(mesh_points[LEFT_IRIS])
42             (r_cx, r_cy), r_radius = cv.minEnclosingCircle(mesh_points[RIGHT_IRIS])
43             center_left = np.array([l_cx, l_cy], dtype=np.int32)
44             center_right = np.array([r_cx, r_cy], dtype=np.int32)
45             cv.circle(frame, center_left, int(l_radius), (255,0,255), 1, cv.LINE_AA)
46             cv.circle(frame, center_right, int(r_radius), (255,0,255), 1, cv.LINE_AA)
47
48         cv.imshow('img', frame) #画面表示
49         key = cv.waitKey(1) #キー入力待ち
50         if key == ord('q'): #終了条件
51             break
```

図3 眼を認識する操作のソースコード

以上でカメラから眼の認識を行う処理の完成である。この時点で紫色の円が目の動きに合わせて動いている。

第4章 視線方向検出

4.1 全体構造について

本研究では上下左右の4方向の視線を判定することにした。判定結果を返すアルゴリズムが3つあり、左右の判定、上の判定、下の判定の3つを製作した。左右の判定は同じアルゴリズムで解決することができたが、上下の判定は目の大きさ、各キーポイントの移動距離など、判定に影響する部分の共有ができなかった。これらのことから上下の判定はアルゴリズムを別のものにした。構造のイメージ図を以下に示す。



図4 視線判定アルゴリズムの構造

4.2 左右の判定

4.2.1 初期案

まずは右を向いた時の判定を作成しようとした。その時に用いた方法が虹彩(黒目)の中心座標の変化を過去の時間との比較で判定するものだ。虹彩の中心座標を利用し、「50ms 前の x 座標より現在の x 座標が5以上増えていたら右を見ている。」というような判定を作成した。すると、右を見た一瞬は判定が出るが継続してその方向を見ていると、判定が出ないというようなことが起きてしまった。加え

て、この段階では 1 つのポイントしか使用しておらず、顔全体が移動したときも同様の判定が出てしまうことから初期構想は使用することができなかった。

4.2.2 正式アルゴリズム案

まず考えなければならないのが、顔が横にずれてしまっても目の判定に影響が及ばないようにすることである。これを実現するために目の動きを相対的な距離で判定できるように設計した。目頭と虹彩の中心との間の距離を利用する方法である。このようにすることで顔が横にずれてしまっても判定に影響する値に変化が生じにくくなった。さらにこの値を使用することにより初期案で問題になっていた継続的な判定の出力ができない問題を解決できる。また、左右それぞれの眼のデータを使うことでさらに精度の良い判定結果が得られる。

4.2.3 アルゴリズムの詳細

(1) キーポイント取得

眼のキーポイントのうち図 5 に示す①～⑤ (①—左目頭、②—右目頭、③—左虹彩の中心、④—右虹彩の中心、⑤—目頭と目頭の中心) のキーポイントの座標を取得する。

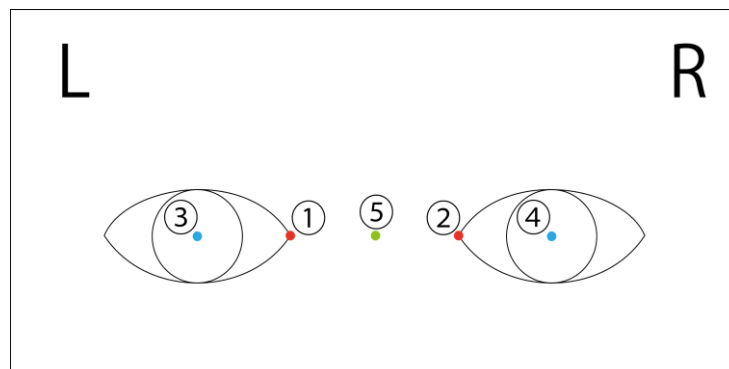


図 5 左右判定に必要なキーポイント

(2) 距離測定

キーポイント①～⑤の座標から異なるキーポイント同士の距離を求める。ここで、 i 番目のキーポイントの座標 $P_i(x_i, y_i)$ 、 j 番目のキーポイントの座標を $P_j(x_j, y_j)$ とすると i, j の距離 D_{i-j} を式 1 により求める。そして、図 5.2 (ア) には前へ向いている初期状態、(イ) には眼球を右に動かす状態の距離をそれぞれ色別で示した。

$$D_{i-j} = \sqrt{(X_i - X_j)^2 + (Y_i - Y_j)^2} \quad (1)$$

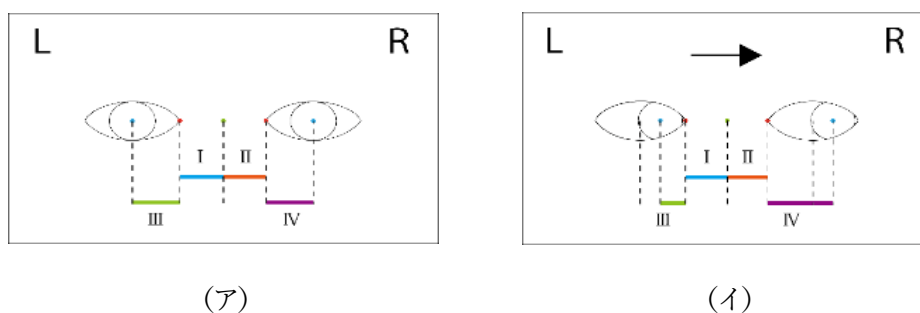


図 5.2 眼を右へ動かしたときの変化の様子

(3) 距離の変化の様子

頭を動かさない状態で、眼を右に動かす場合は D_{3-1} と D_{2-4} の値が変化される。

つまり、眼球を右に動かすと D_{3-1} (III) の値がある閾値より小さくなり、 D_{2-4} (IV) の値もある閾値より大きくなる。

4.2.4 実験・検証

(1) 実験内容

4.2.3 で提案したアルゴリズムのⅢとⅣの距離の変化をグラフ化して比較する.

(2) 実験条件

実験に用いたカメラは PC に標準で装備されている Integrated Camera(内蔵カメラ)を使用. 測定の流れは, ①カメラの前に座る. ②カメラから 40cm 離れる. ③指定の動きをする. これらの流れで測定する. この条件はすべてのアルゴリズムの実験で使用する.

(3) 検証した動き

左右の判定に検証した動き(視線移動プロセス)は以下の通りだ.



図 6 視線移動プロセス

(4) 実験結果

以下が実験によって得られたデータの一つである.

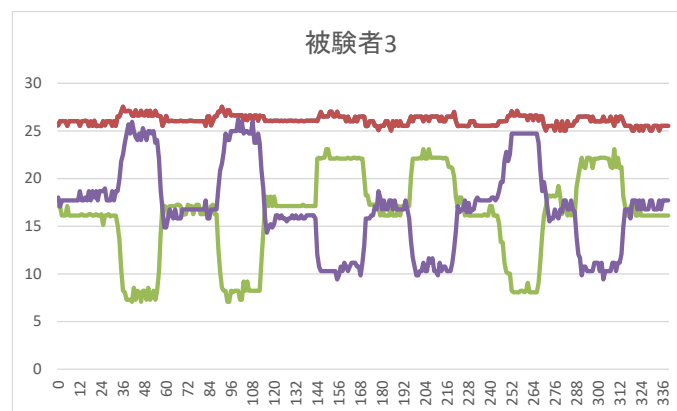


図 7 実験結果 1

横軸に時間、縦軸にそれぞれの間の距離を取っている。赤い線のデータは目頭間の値であり、カメラと顔との距離の指標として使用。

グラフと目の動きの対応を以下の図に示す。

正面を向いているときは互いの距離はおよそ同じなので直線状になる。

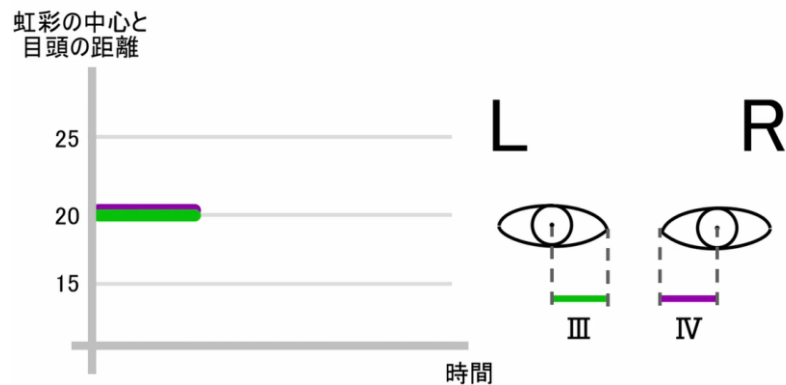


図 8 正面を向いているときのグラフと眼の対応

右を向くとそれぞれの値が変化する。

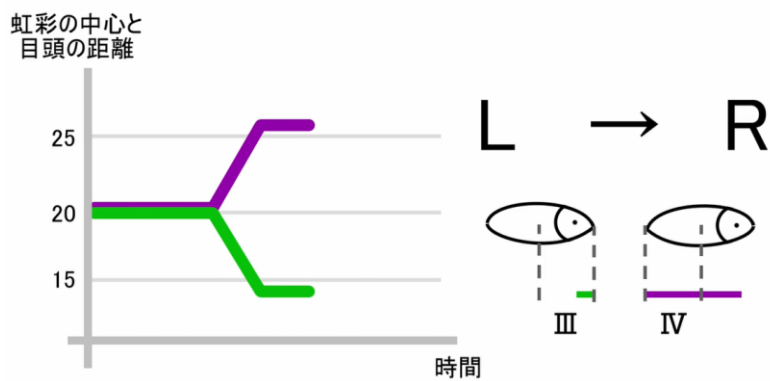


図 9 右を向いた時のグラフと眼の対応

右を見たときはIIIの値が少なくなり、グラフでも緑の線が低い値を取る。逆にIVの値は増加するのでグラフ上でも高い値を取る。

左を向いた時は図9と逆のようになり、その様子を図10とする。

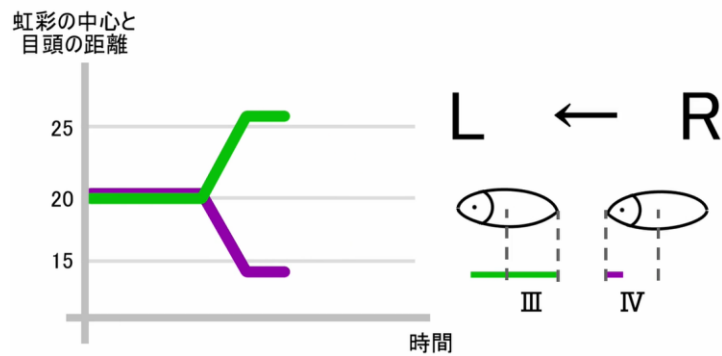


図 10 左を向いた時のグラフと眼の対応

(5) 閾値の設定

実験結果から同じ条件下であればある一定の値をそれぞれとることが分かった。そこで、左右判定に用いる閾値を図 11 の通り計測値の差で設定することにした。

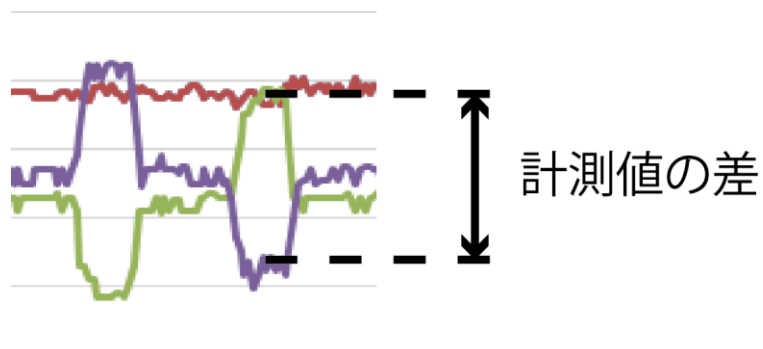


図 11 閾値に利用する値

実験データをもとに閾値を 10 と設定し、その値を超えたとき判定結果を返す。

(6) 閾値に利用する値の設定理由

初めはこのデータの左右それぞれ上の値を取ったとき、つまり正面を向いているときよりⅢが伸びるまたはⅣが伸びるときのように値が大きくなったときに判定を出そうとしたが、より目や斜視を考慮した結果この値が正確な左右を取れることが分かったので採用した。

(7) 実装(事前準備)

座標取得は指定された点の位置に画面サイズを掛けることで得られる。また, x 座標と y 座標を一つの配列にまとめることで一つのキーポイントとして扱うことができ, 管理しやすくなる。

座標取得メソッドを用意した。座標の取り方の例としてキーポイントの一つだけ抽出したソースコードを図12に示す。他のキーポイントを取得したソースコードはすべてのアルゴリズムをまとめて後ほど掲載する。

```
48  #目頭のキーポイント座標抽出
49  def get_keypoint(results, img_h, img_w):
50      left_inner_corner_eye_x = int(results.multi_face_landmarks[0].landmark[362].x * img_w)
51      left_inner_corner_eye_y = int(results.multi_face_landmarks[0].landmark[362].y * img_h)

      省略

68  |   eye_inner_L = np.array([left_inner_corner_eye_x, left_inner_corner_eye_y])
```

図 12 座標取得方法例

次に配列「data」を用意する。最終的な配列の中身は以下の図の通りになる。

```
["中心から左目頭", "中心から右目頭", "左目頭から虹彩の中心", "右目頭から虹彩の中心",  
 "上下右", "上下左", "左右判定", "目の面積右", "目の面積左", "右目下距離", "左目下距離"]
```

図 13 配列「data」の中身

次に視線の方向を取得するメソッド「eye_direction()」を作成する。引数は図14の通り。

eye_direction()は返値を持っており, 正面を見ている状態(どの判定も出ていない状態)は0を返す。

```
101  #4方向計算
102  def eye_direction(frame,
103                  data,
104                  center_right,
105                  center_left,
106                  lineToPoint_R,
107                  lineToPoint_L,
108                  lineToPoint_R_dist,
109                  lineToPoint_L_dist,
110                  eye_area,
111                  R_low_dist,
112                  L_low_dist):
```

図 14 eye_direction()の引数

(8) 実装(左右アルゴリズム)

最後に `eye_direction()` に左右の判定プログラムを実装し完成する。左右の返値はそれぞれ【右:5 左:-5】となっている。図 15 が判定プログラムである。

```
147     #右方向
148     if data[3] - data[2] > LR_THRESHOLD:
149         cv.putText(frame,
150                     text="RIGHT",
151                     org=(100, 400),
152                     fontFace=cv.FONT_HERSHEY_SIMPLEX,
153                     fontScale=1.0,
154                     color=(0,255,0),
155                     thickness=5)
156         return 5
157     #左方向
158     if data[3] - data[2] < -LR_THRESHOLD:
159         cv.putText(frame,
160                     text="LEFT",
161                     org=(100, 400),
162                     fontFace=cv.FONT_HERSHEY_SIMPLEX,
163                     fontScale=1.0,
164                     color=(0,255,0),
165                     thickness=5)
166         return -5
```

図 15 左右判定プログラム部分のソースコード

`cv.putText()` は opencv の機能で出力画面の中に文字列を表示させることができる。

実装が完了して判定結果を表示させるようにした結果を図 16, 17 に示す。



図 16 左を判定している様子



図 17 右を判定している様子

4.3 下向きの判定

4.3.1 初期案

左右の判定と同様に虹彩の中心と他のどこかとの距離変化を利用しようと考えた。しかし、眼の上下の動きを観察したところ、眼球のみが変化するのではなく、瞼なども同時に動いてしまうため左右の判定同様の仕様では解決できなかった。

4.3.2 正式アルゴリズム案

眼の動きを観察したところ、下を向いた時は白目を含む目の面積が大きく変化することが分かった。通常時(前を向いているとき)より値が下がったら下を見ている判定が返る。しかし、これだけでは問題がある。それは、人によって目の大きさが異なってしまう点である。そこで、判定条件をもう一つ加える。

さらに目の動きを観察したところ、目じりの動きは上下瞼に比べ大きな動きをすることはなかった。さらに左右の目と目の間の中心点(目全体の中心)もまた、固定点として使用することが可能であった。これら2点を結んだ直線を境界線(以降「下向き基準線」と呼ぶ)として、その線より虹彩の中心座標が低くなった時に判定を出すように考案した。

4.3.3 アルゴリズムの詳細

(1) キーポイント取得

眼のキーポイントのうち図 18 に示す (③—左虹彩の中心, ④—右虹彩の中心, ⑤—目頭と目頭の間, ⑩—左目じり, ⑪—右目じり) キーポイントの座標を取得する。

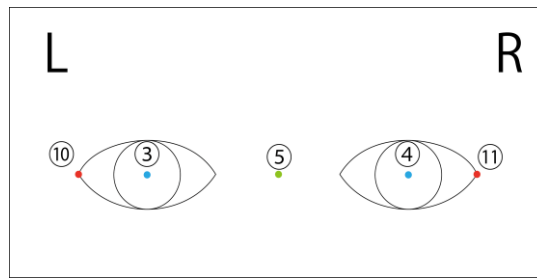


図 18 下判定に必要なキーポイント

(2) 距離, 面積の測定

2 点を通る直線から離れた位置にある点までの距離を求める. 基本的な計算はベクトルを使用したものである. 参考文献【3】を参照.

```

171 def pointToLine(line_a, line_b, point):
172     # a(x1, y1) = (2, 2)
173     x1 = line_a[0]
174     y1 = line_a[1]
175     # b(x2, y2) = (6, 4)
176     x2 = line_b[0]
177     y2 = line_b[1]
178     # c(x3, y3) = (4, 5)
179     x3 = point[0]
180     y3 = point[1]
181     u = np.array([x2 - x1, y2 - y1])
182     v = np.array([x3 - x1, y3 - y1])
183     L = abs(np.cross(u, v) / np.linalg.norm(u))

```

図 18 線と点の距離

加えて目の面積を取得する. 目の面積は Opencv に搭載されている `contourArea()` を使用する. キーポイントの配列を渡すことでその点によって囲われた部分の面積を取得できる.

```

eye_area_L = cv.contourArea(np.array(mesh_points[LEFT_EYE]))
eye_area_R = cv.contourArea(np.array(mesh_points[RIGHT_EYE]))

```

図 19 目の面積の取得

眼を動かしたときの変化の様子を以下の図に示す。

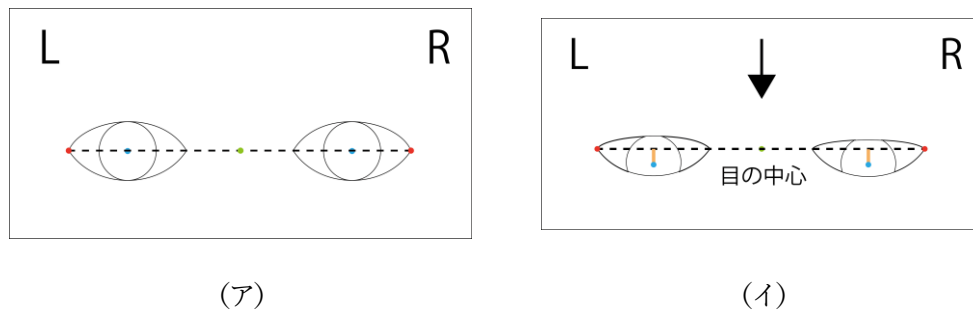


図 18.2 眼を右へ動かしたときの変化の様子

(3) 距離変化の様子

通常時(正面を見ているとき)に比べ下を見たときは下向き基準線より下に来ているのがわかる。さらに目の面積も小さくなっている。

4.3.4 実験・検証

(1) 実験内容

二つの実験を行う。一つは虹彩の中心位置の変化量の測定をすること。もう一つは目の面積の変化量を測定することである。

(2) 検証した動き

下の判定に検証した動き(視線移動プロセス)は以下の通りだ。



図 20 下の視線移動プロセス

(3) 実験結果(虹彩の中心と下向き基準線)

以下が虹彩の中心と下向き基準線の距離の変化によって得られたデータのの一つである。

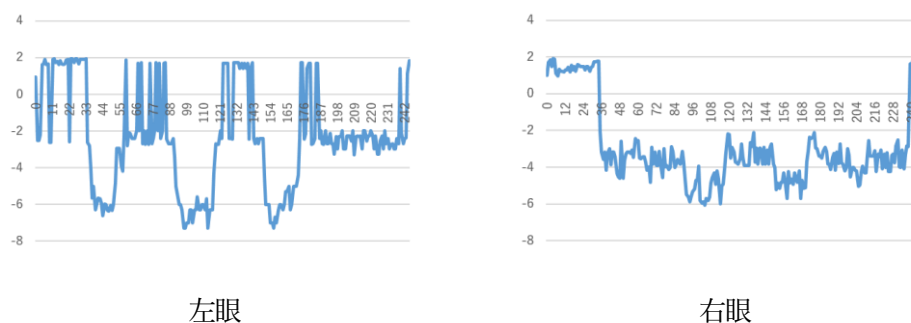


図 18.3 虹彩の中心と線の距離の変化

このグラフは横軸に時間, 縦軸に下向き基準線と虹彩の中心の距離を持っている。

下を向くと虹彩の中心点は下向き基準線より下に動くため値がマイナス値を取得する。

(4) 閾値の設定(虹彩の中心と下向き基準線)

今回は単純にグラフの谷部分(判定がわかりやすい部分)で閾値を設ける。絶対値へ変換し値は「3」と設定する。

(5) 実験結果(目の面積)

以下が目の面積を利用したデータである。

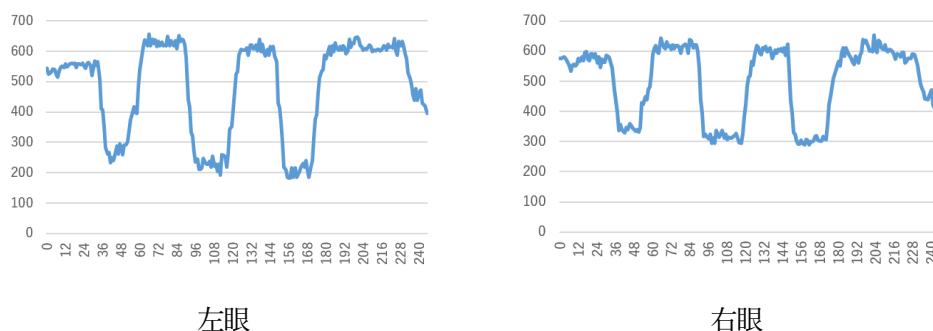


図 18.4 虹彩の中心と線の距離の変化

このグラフは横軸に時間, 縦軸に面積を持っている.

面積の 600 付近が通常時(正面を見ている)の状態, 値がそれよりも下がったときに下を見ていることになる.

(6) 閾値の設定(目の面積)

面積の閾値も単純に, ある値よりも数値が下がったら判定する方式にするため, 値を「270」と設定する. この値よりも小さくなったら下を見ている判定が返る.

(7) 実装

視線方向を取得するメソッド `eye_direction()` に下判定プログラムを実装する.

```
#下方向
if center_right[1] > lineToPoint_R[1] and lineToPoint_R_dist > 3 or center_left[1] > lineToPoint_L[1] and lineToPoint_L_dist > 3:
    if eye_area[0] < 270 or eye_area[1] < 270:
        cv.putText(frame,
                    text="DOWN",
                    org=(100, 400),
                    fontFace=cv.FONT_HERSHEY_SIMPLEX,
                    fontScale=1.0,
                    color=(0,255,0),
                    thickness=5)
        return -10
```

図 21 下判定のソースコード

実装が完了し, 判定結果を表示させるようにした結果を図 24 に示す.



図 22 下判定実装完了

4.4 上向きの判定

4.4.1 初期案

上判定のアルゴリズムも下判定と同様に虹彩の中心位置の変化と目の面積の変化を利用しようとした。しかし、虹彩の中心位置を絶対値で取得しているため同じ手法を用いるのは危険だった。加えて、目の面積は変化が非常に小さかった。以上のことから下判定と同じ手法を用いることができなかった。

4.4.2 正式アルゴリズム案

上を向いたときに利用できる見込みがある目の位置が一つだけ確認できた。それが虹彩の下部分と下瞼との間の距離である。基本的に通常時(正面を向いているとき)は虹彩の下部分は下瞼に隠れている。しかし、上を向いたときには白目が露出する。この変化を利用することにした。

4.4.3 アルゴリズムの詳細

(1) キーポイントの取得

眼のキーポイントのうち図 23 に示す⑥～⑨ (⑥—左虹彩の下, ⑦—右虹彩の下, ⑧—左下瞼の中心, ⑨—右下瞼の中心) キーポイントの座標を取得する。

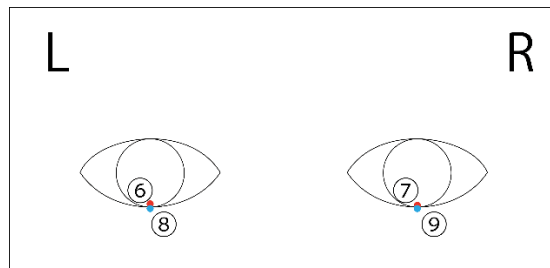


図 23 上判定に必要なキーポイント

(2) 距離の測定

線 6-8, 線 7-9 の y 座標の差で判定を製作する.

眼を動かしたときの変化の様子を以下の図に示す.

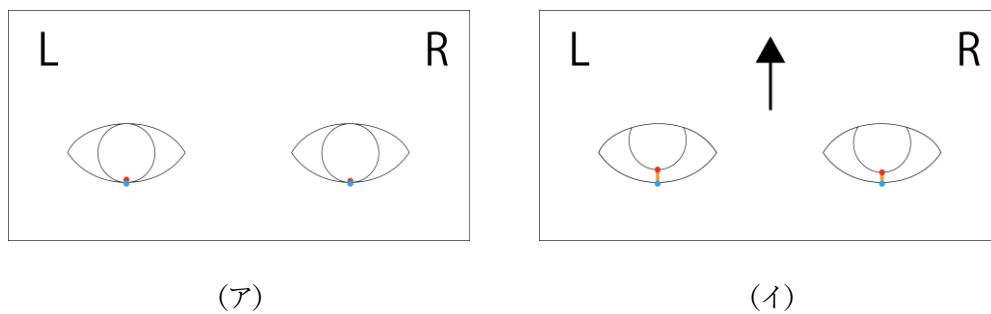


図 23.2 眼を上へ動かしたときの変化の様子

(3) 距離変化の様子

眼を上方向に動かすことで虹彩の下部分と下瞼の中心部分に距離の差が生まれる.

4.4.4 実験・検証

(1) 実験内容

通常値(正面を向いているとき)の値がどの値をとっているのか検証する. 変化後の値を測定する.

(2) 検証した動き

上の判定に検証した動き(視線移動プロセス)は以下の通りだ.



図 24 視線移動プロセス

(3) 実験結果 1

以下の結果が虹彩の下部分と下瞼の中心部分の距離の変化で得られたデータである。

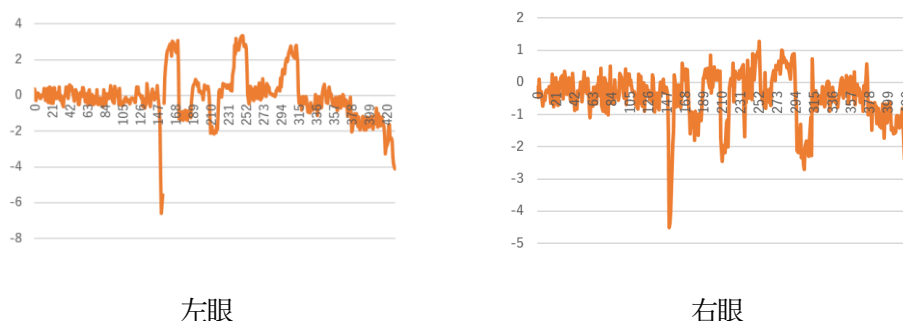


図 23.3 虹彩の中心と線の距離の変化

このグラフは横に時間軸、縦に計測した距離の値を所有している。

通常時(正面を向いているとき)の値は 0 を基準に ± 0.5 付近であることが分かった。

データからわかる通り、今までの計測値に比べると取得できる最大値が小さい。このままでは正面を見ているときの誤差値(波打っている部分)と判定値(山の部分が)わかりにくいので改善が必要である。

(4) 改善案

誤差の値を均すために指数移動平均法を用いる。移動平均法は時系列データを平滑化する手法で、さらに指数関数的に行うことで直近の値に近い形で平均をとることができるアルゴリズムである。この方法をプログラムに落とし込んだものを図に示す。

```
#指数移動平均
def ema(data):
    volume = 15
    ema = np.zeros(len(data))
    ema[:] = np.nan
    ema[volume - 1] = np.mean(data[:volume])

    for section in range(volume, len(data)):
        ema[section] = ema[section - 1] + (data[section] - ema[section - 1]) / (volume + 1) * 2

    return ema[-1]
```

図 25 指数移動平均法

(5) 実験結果(改善後)

以下が移動平均を用いて取得し直したものである。

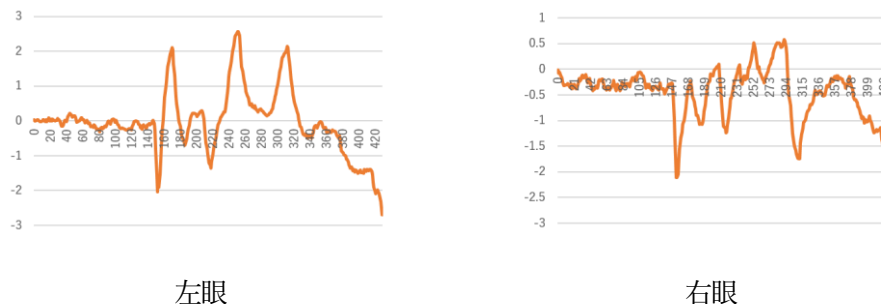


図 23.4 虹彩の中心と線の距離の変化

改善前に比べて判定結果がわかりやすくなった。

さらに、データを均すことで通常時で誤差が ± 0.5 を超えることはなくなった。

(6) 閾値の設定

通常時に誤差値が ± 0.5 を取らない。加えて、今回は上判定のみを作成することから、マイナスの値は考慮しなくてよい。以上のことから閾値を「0.5」と設定し、その値を上回ったら上判定が返るようにする。

(7) 実装

視線方向を取得するメソッド`eye_direction()`に下判定プログラムを実装する。

```
#上方向
if R_low_dist > 0.5 or L_low_dist > 0.5:
    cv.putText(frame,
               text="UP",
               org=(100, 400),
               fontFace=cv.FONT_HERSHEY_SIMPLEX,
               fontScale=1.0,
               color=(0, 255, 0),
               thickness=5)
    return 10
```

図 26 上判定のソースコード

実装が完了し、判定結果を表示させるようにした結果を図 24 に示す。



図 27 上を判定している様子

4.5 eye_direction()の引数

eye_direction()の引数の内容を表にまとめる。

変数, 配列名	内容	[0 番目の内容, 1 番目の内容]
frame	Opencv で読み取った画像情報	
data	図 13 を参照	
center_right	右虹彩の中心座標	[x 座標, y 座標]
center_left	左虹彩の中心座標	[x 座標, y 座標]
lineToPoint_R	目の中心と右目じりの線の中心座標	[x 座標, y 座標]
lineToPoint_L	目の中心と左目じりの線の中心座標	[x 座標, y 座標]
lineToPoint_R_dist	目の中心と右目じりの線から右虹彩の中心までの距離	
lineToPoint_L_dist	目の中心と左目じりの線から左虹彩の中心までの距離	
eye_area	目の面積	[左目の面積, 右目の面積]
R_low_dist	右虹彩下部分と下瞼の距離	
L_low_dist	左虹彩下部分と下瞼の距離	

表 3 eye_direction()の引数表

第5章 視線方向取得モジュールとしての実装

5.1 外部プログラムからの呼び出し方法

本研究で作成された視線方向取得プログラムは様々な用途で使われることを前提として作る必要があるため、メインクラス(メインファイル)とは別のクラス(ファイル)として扱う。

ここで設計上の問題点として、外部から呼ばれる機能であるにも関わらず、無限ループを使用している点が挙げられる。このループしているメソッド(視線判定メソッド)を外部から単純に呼び出そうとすると視線判定メソッドからメインプログラムに帰ってくることができない。そこで使用される技術として、「マルチスレッド」が必要となる。マルチスレッドとはアプリケーションのプロセス(タスク)を複数のスレッドに分けて並行処理する方式を指す。よってメインプログラムからはマルチスレッドを使用して視線判定メソッドを起動させなければならない。

メインのプログラムによっては視線判定メソッドから直接判定結果を受け取ることができない仕様で作成しなければならないことがある。そのために、値取得用のメソッドも追加で作成する。

すべてのプログラムを終了するとき外部からループを止める必要がある。ループを止めなければフリーズの原因になることから、ループ停止用プログラムも実装する。

視線方向取得プログラムを外部モジュール化したものを「iris.py」とする。

5.2 外部から呼ばれるメインとなるメソッド

先述している while でのループ処理が視線判定の主要機関である。これを外部から実行するためにメソッドとして扱う。名前を iris_cerector() として作成し、メインのループ部分をネストする。その様子を以下の図に示す。

```
216  #メインプログラムで起動されるモジュール
217  def iris_cerector():
218      #初期化
219      r_cx = 0
220      l_cx = 0
221      r_cy = 0
222      l_cy = 0
223      eye_area = 0
224      R_low_average = [0] * 15
225      L_low_average = [0] * 15
226
227      dataPOS = pd.DataFrame() #データ書き出し用準備
228
229      #メインループ
230      with mp_face_mesh.FaceMesh(
231          max_num_faces=1,
232          refine_landmarks=True,
233          min_detection_confidence=0.5,
234          min_tracking_confidence=0.5
235      ) as face_mesh:
236          while True:
237              ret, frame = cap.read()
238              if not ret:
239                  break
240              frame = cv.flip(frame, 1)
241              rgb_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
242              img_h, img_w = frame.shape[:2]
243              results = face_mesh.process(rgb_frame)
```

} 省略

図 28 外部から呼ばれるためのメソッド

5.3 値取得用メソッドの実装

値取得用のメソッド「get_direction()」を作成する。これはメインのプログラムから判定結果を取得できるメソッドである。

get_direction()を利用できるようにするために「set_direction()」を用意する。これは視線判定メソッドで得た値を加工して get_direction()で返せるようにするものである。これは視線判定メソッドで使われる。

以上の二つを実装したものを以下の図で示す。

```
194  #値加工用
195  def set_direction(dir):
196      global direction
197      global direction2
198      dir_val = {5: "e", -5: "c", 10: "d", -10: "f"}
199      dir_val2 = {5: "a", -5: "cc", 10: "g", -10: "b"}
200
201      if dir in dir_val:
202          direction = dir_val[dir]
203          direction2 = dir_val2[dir]
204
205  #メインのプログラムへの受け渡し
206  def get_direction(mode):
207      modes = {0: direction, 1: direction2}
208      if mode in modes:
209          return modes[mode]
```

図 29 メインのプログラムに値を渡す部分

後述する kivy を用いた開発で視線判定メソッドをマルチスレッドで呼び出すと、値の受け渡しを直接行うことができず、別のスレッドで値取得用のメソッドを呼び出さなければならない。本研究では少しメインのプログラムの仕様に寄せているため汎用性は低くなっているが、この部分を調整することで他のプログラムにも対応できるようになる。

5.4 ループ終了用メソッドの実装

現段階のループ終了条件はキーボードの「q」キーを押したときに終了することになっている。このままではメインのプログラムが終了しても視線判定だけ続いている状態となってしまう。メインプログラム終了時にフリーズしてしまう。それを防ぐためにメインプログラム側から終了コマンドを実行する必要がある。

ループ終了メソッド「set_finish()」を作成する。作成したものを以下に示す。

```
211  #ループの終了用
212  def set_finish(comand):
213      global finish_comand
214      finish_comand = comand
```

図 30 終了用コマンドのソースコード

変数 finish_comand は bool 型であり初期値は False である。メインプログラムから True が渡されることで以下に示すループ処理中の条件文で判定されループが終了する。

```
while True:
    ret, frame = cap.read()
    if not ret:
        break
    frame = cv.flip(frame, 1)
    rgb_frame = cv.cvtColor(frame, cv.COLOR_BGR2RGB)
    img_h, img_w = frame.shape[:2]
    results = face_mesh.process(rgb_frame)
    if results.multi_face_landmarks:
        省略
    if finish_comand:
        break
```

図 31 finish_comand の使われ方

第6章 アプリの作成

6.1 アプリの概要

本研究で作成されたアプリは視線の向く方向によって音程が変化する音楽演奏アプリである。アプリのみで操作の仕方などを知ることができるように操作説明画面をアプリ内に設けた。アプリの全体画面は図の通りである。



図 32 アプリのメイン画面



図 33 操作説明画面

6.2 アプリ仕様

- ・視線方向に合わせて音程が変わる
- ・画面右側が操作オブジェクトである
- ・画面右上の「BPM」欄でBPM(リズム)の変更が可能

各種ボタンの機能を以下の表に示す.

ボタンの名前	機能
mute	演奏の一時停止
play	演奏再開
up, down	音程の切替
set(BPM 欄)	BPM を確定する
?	操作説明
終了	アプリの終了

表 4 各種ボタンの機能

6.3 ライブラリ「kivy」について

6.3.1 Kivy とは

「kivy」は Python 上で GUI(グラフィカルユーザーインターフェース)を構成するためのオープンソースライブラリである. 対応 OS は Android, iOS, Linux, MacOSX, Windows 等の主要環境で動作させることができる.

6.3.2 選定理由

Python には GUI 開発ライブラリがいくつかある. 今回は研究題材のおまけアプリの作成ということもあり, 設計が簡単で短時間で習得できるかつ, 比較的インターネットに情報が多く掲載されていることを理由に kivy を選定することにした.

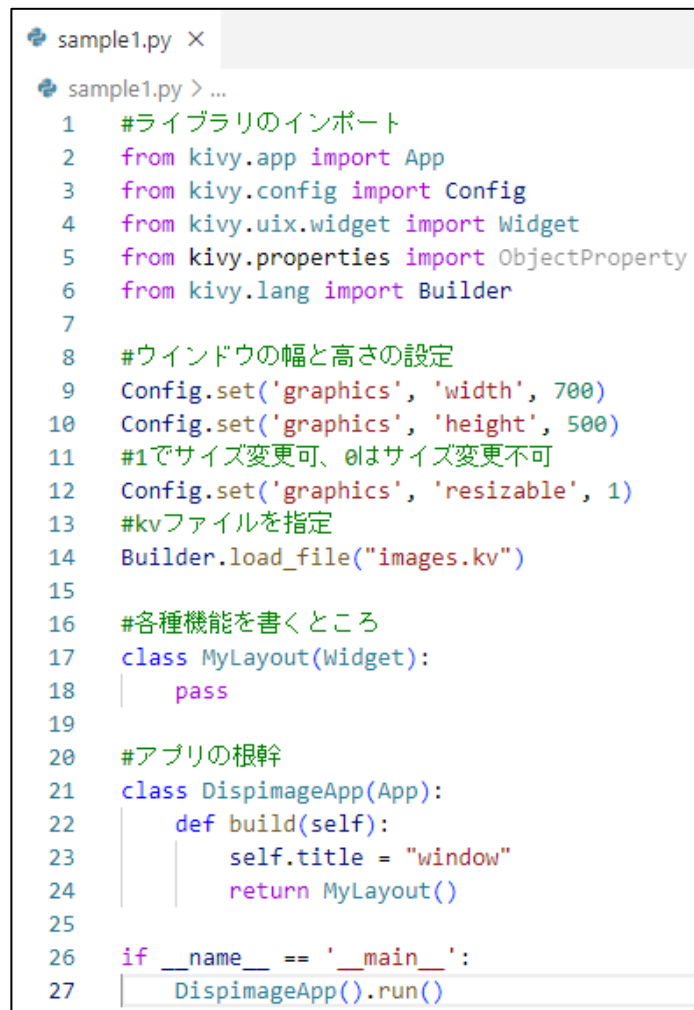
6.3.3 Kivy の基本

(1) ソースファイル

Kivy を扱う場合, メインのソースコード(プログラム名.py)一つでも実装できるが, 多くの場合 kivy 言語を扱うファイル(kivy プログラム名.kv)があると非常に作業がわかりやすくなる. 場合によってはファイル名に注意が必要な場合がある.

(2) メインファイルの説明

Kivy でアプリケーションを作るときの基盤となる部分を以下の図に示す.



```
sample1.py X
sample1.py > ...
1  #ライブラリのインポート
2  from kivy.app import App
3  from kivy.config import Config
4  from kivy.uix.widget import Widget
5  from kivy.properties import ObjectProperty
6  from kivy.lang import Builder
7
8  #ウィンドウの幅と高さの設定
9  Config.set('graphics', 'width', 700)
10 Config.set('graphics', 'height', 500)
11 #1でサイズ変更可、0はサイズ変更不可
12 Config.set('graphics', 'resizable', 1)
13 #kvファイルを指定
14 Builder.load_file("images.kv")
15
16 #各種機能を書くところ
17 class MyLayout(Widget):
18     pass
19
20 #アプリの根幹
21 class DispimageApp(App):
22     def build(self):
23         self.title = "window"
24         return MyLayout()
25
26 if __name__ == '__main__':
27     DispimageApp().run()
```

図 34 アプリメインファイル

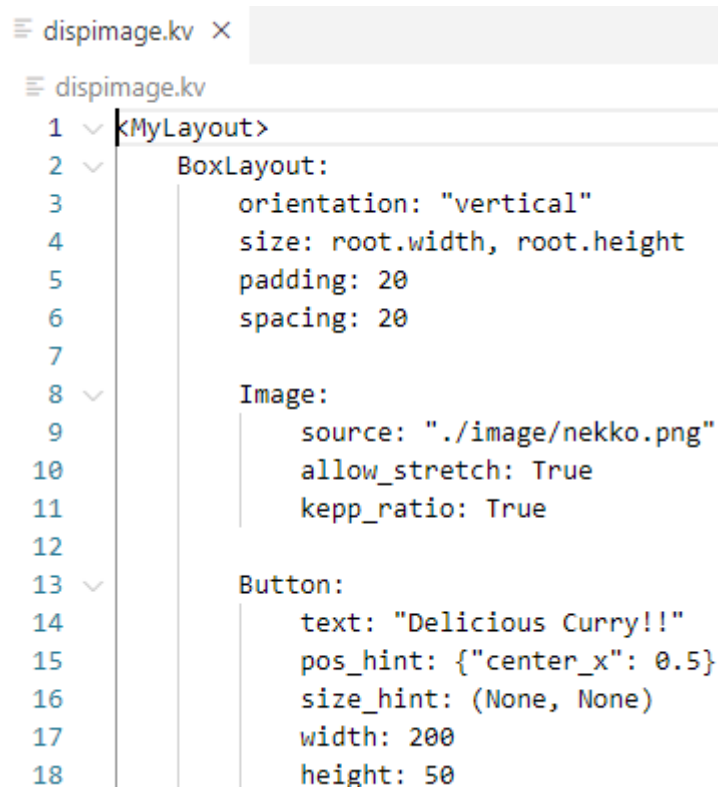
これをもとに構成されていく。ファイル名, 各種数値, class 名などは変更可能だが全体の形はおおむねこのままでなければならない。

アプリの根幹である「DispimageApp(App)」(図 33: 21 行目)は「○○App(App)」の形にしなければならない。

また, ○○.kv ファイルの読み込み方は「Builder.load_file(“ファイル名”)」(図 33: 14 行目)でできるが, .kv ファイルを○○App(App)の「○○」をすべて小文字にした○○.kv とすることで, 14 行目を記述しなくても読み込むことができる。ただしメインファイル(○○.py)と同じ階層である必要がある。

(3) .kv ファイルの説明

メインファイル(.py)に対応した記述をする必要がある。kv ファイルは HTML と CSS のような関係があり, 主に画面の装飾に使われるファイルである。以下に.kv ファイルの例を示す。



```
dispimage.kv ×
dispimage.kv
1 <MyLayout>
2     BoxLayout:
3         orientation: "vertical"
4         size: root.width, root.height
5         padding: 20
6         spacing: 20
7
8     Image:
9         source: "../image/nekko.png"
10        allow_stretch: True
11        keep_ratio: True
12
13    Button:
14        text: "Delicious Curry!!"
15        pos_hint: {"center_x": 0.5}
16        size_hint: (None, None)
17        width: 200
18        height: 50
```

図 35 .kv ファイルの例

図 33 の Widget を継承したクラス(図 33: 17 行目)に対応してコードを記述していく.

.kv ファイルは<クラス名>の下へネスト構造を作りレイアウトを製作していく.

<クラス名>で表記された元のクラスは root という扱いになり各種プロパティでは, root の変数やメソッドを呼び出すことが可能となる.

オブジェクトのタグ名の頭文字は大文字となり, プロパティはすべて小文字で表記される.

注意点として動画を読み込むために「VideoPlayer」というタグを使用するのだが, これを記述するだけでは動画は再生されない. 追加で「ffpyplayer」をインストールする必要がある.

6.4 GUI 設計

6.4.1 レイアウト

まずはウィジェット(ボタンなどのオブジェクト)の配置を考える. 視線の方向と音程を表す画像が一番注目される部分に配置する必要がある. このように考えたときに左上が最適と判断した.

次に操作パネルの配置を考える. 本アプリは基本的にマウス操作が必要になるため, 右手で直感的に操作できるように右半分に操作パネルを配置した.

終了ボタンはほとんどの windows 系アプリが右上で統一されていることから同じように右上に配置することにした.

操作説明用画面の設置も行った. アプリ内で操作方法などすべて完結できればユーザ的にも扱いやすいアプリになると考えた. 終了ボタンの横に配置することで目につきやすいと考えた.

6.4.2 メイン画面のネスト構造

アプリのレイアウトとソースコードのウィジェット(タグなど)を対応付けた図を以下に示す.

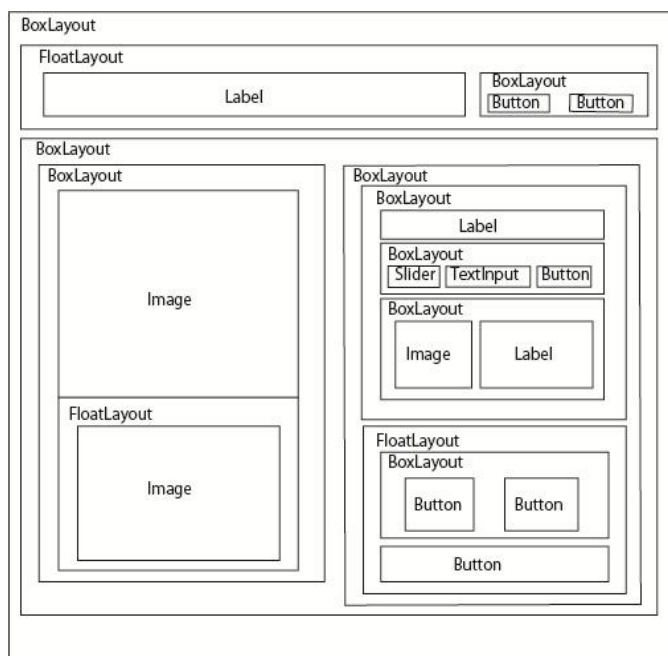


図 36 メイン画面のネスト構造

6.4.3 操作説明画面のネスト構造

操作説明画面のネスト構造を以下の図に示す.

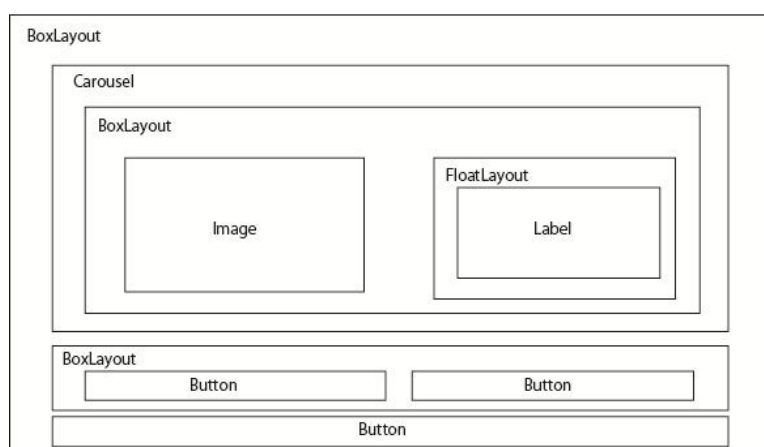


図 37 操作説明画面のネスト構造

6.4.4 すべてのタグに使えるプロパティの説明

(1) id

それぞれのウィジェットに id を付与することが可能

Id 指定で特定の動きをさせるために必要である.

(2) size

それぞれのウィジェットの領域を指定できる.

Size: x(数値), y(数値)

で利用できる.

(3) padding

css と同じ仕様で上下左右に空間を持たせることができる.

(4) color

ウィジェットの領域を塗りつぶす.

color: 数値/256, 数値/256, 数値/256, 1(透明度 0~1)

の表記で実装

(5) size_hint

ウィジェットの大きさを同階層のネストの割合で指定可能.

size_hin_x で x 軸方向の調整, size_hint_y で y 軸方向の調整をする.

size(30, 30)のように固定値で設定したい場合は size_hint(None, None)と指定しなければならない.

(6) pos_hint

主に Floatlayout の子要素で使用するようになる. 相対的な配置位置の調整が可能.

pos_hint: {"center_x": 数値, "center_y": 数値}が使いやすい.

6.4.5 BoxLayout の縦横切り換え

orientation プロパティで横に配置していくか, 縦に配置していくかを切り替えることができる. デフォルト(記述しない状態)では横に配置していく.

縦に配置する場合は, 「orientation: “vertical”」と記述する.

6.4.6 Button のクリック処理

基本的にはルートの関数を呼び出す. プロパティの記述は「on_press: root.メソッド()」の形式で行う.

6.4.7 GUI の完成

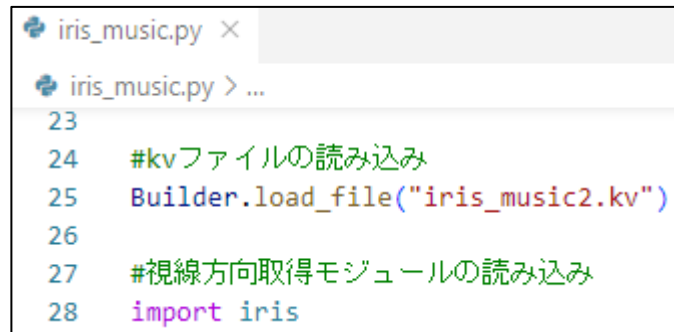
GUI(アプリ画面)の作成は以上で完了となる. 全体のソースコードは付録として後述する.

この後に, 視線トラッキングモジュールの呼び出しを行い各種ボタンに機能を割り振る.

ここまでで完成したファイルをそれぞれ「iris_music.py」と「iris_music2.kv」とする.

6.5 視線方向取得モジュールの呼び出し

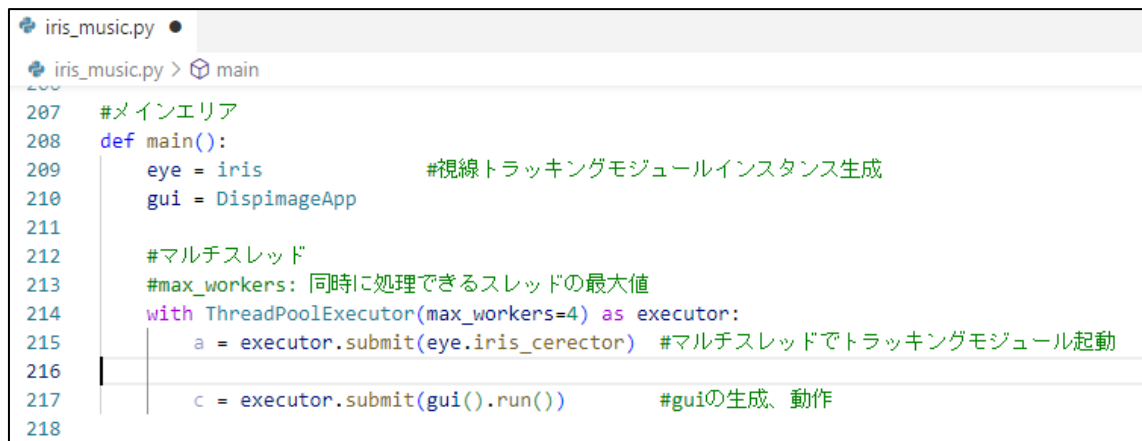
メインのプログラム(iris_music.py)に kivy ファイル(iris_music2.kv)と視線方向取得モジュール(iris).py を読み込む。



```
iris_music.py ×
iris_music.py > ...
23
24 #kvファイルの読み込み
25 Builder.load_file("iris_music2.kv")
26
27 #視線方向取得モジュールの読み込み
28 import iris
```

図 38 各種ファイルの読み込み

次に、インスタンスを生成し、ThreadPoolExecutor を利用しマルチスレッドで iris_cerector() を起動させる。



```
iris_music.py ●
iris_music.py > main
207 #メインエリア
208 def main():
209     eye = iris #視線トラッキングモジュールインスタンス生成
210     gui = DispimageApp
211
212     #マルチスレッド
213     #max_workers: 同時に処理できるスレッドの最大値
214     with ThreadPoolExecutor(max_workers=4) as executor:
215         a = executor.submit(eye.iris_cerector) #マルチスレッドでトラッキングモジュール起動
216
217         c = executor.submit(gui().run()) #guiの生成、動作
218
```

図 39 マルチスレッド処理

マルチスレッドで起動させる場合インスタンスを生成してから executor.submit() の引数に渡さない
と起動することができない。

Max_worker : executor.submit() を実行できる最大値を決定する。

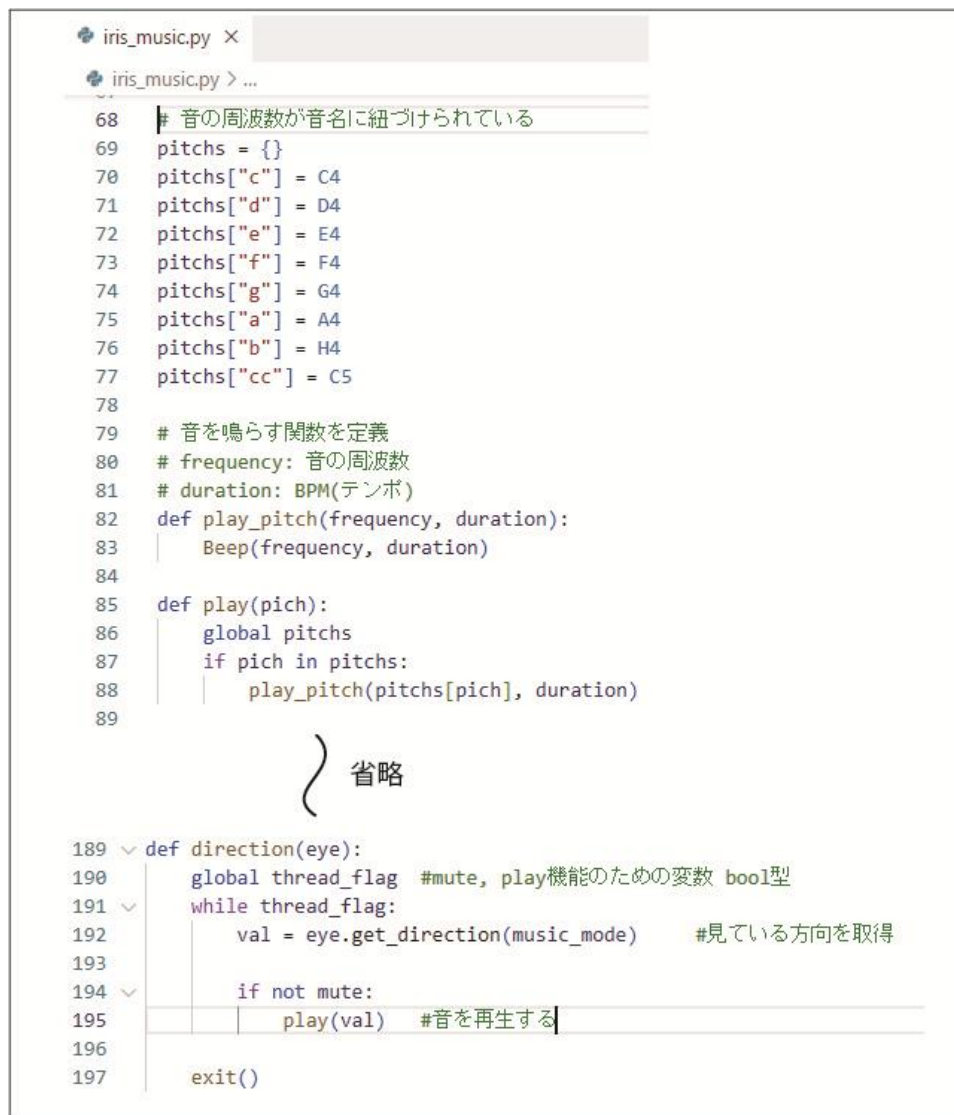
値取得用メソッドの「get_direction()」を別のスレッドで起動させる。今回は常に値を取り続ける必要があったため別メソッドでループ処理を行うように作成した。direction()に視線方向取得モジュールのインスタンスを渡すことでget_direction()が使えるようになる。

```
iris_music.py ×
iris_music.py > direction
189 def direction(eye):
190     global thread_flag #スレッドを終了するための変数 bool型
191     while thread_flag:
192         val = eye.get_direction(music_mode) #見ている方向を取得
193
194         if not mute:
195             play(val) #音を再生する
196
197     exit()
198
199 #メインエリア
200 def main():
201     eye = iris #視線トラッキングモジュールインスタンス生成
202     gui = DispimageApp
203
204     #マルチスレッド
205     #max_workers: 同時に処理できるスレッドの最大値
206     with ThreadPoolExecutor(max_workers=4) as executor:
207         a = executor.submit(eye.iris_cerector) #マルチスレッドでトラッキングモジュール起動
208         b = executor.submit(direction, eye)
209         c = executor.submit(gui().run()) #guiの生成、動作
```

図 40 get_direction()の呼び出し

6.6 音を鳴らす仕組み

各音程の周波数をそれぞれ辞書型配列で定義する. さらに, BPM(テンポ)を設定する. winsound というライブラリの beep() に定義した周波数と BPM を引数として渡すことで, BPM に合わせた音を鳴らすことができる.



```
iris_music.py x
iris_music.py > ...
68 # 音の周波数が音名に紐づけられている
69 pitches = {}
70 pitches["c"] = C4
71 pitches["d"] = D4
72 pitches["e"] = E4
73 pitches["f"] = F4
74 pitches["g"] = G4
75 pitches["a"] = A4
76 pitches["b"] = H4
77 pitches["cc"] = C5
78
79 # 音を鳴らす関数を定義
80 # frequency: 音の周波数
81 # duration: BPM(テンポ)
82 def play_pitch(frequency, duration):
83     Beep(frequency, duration)
84
85 def play(pich):
86     global pitches
87     if pich in pitches:
88         play_pitch(pitches[pich], duration)
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1188
1189
1190
1191
1192
1193
1194
1195
1196
1197
1198
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1388
1389
1390
1391
1392
1393
1394
1395
1396
1397
1398
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1410
1411
1412
1413
1414
1415
1416
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481
1482
1483
1484
1485
1486
1487
1488
1489
1490
1491
1492
1493
1494
1495
1496
1497
1498
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1510
1511
1512
1513
1514
1515
1516
1517
1518
1519
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529
1530
1531
1532
1533
1534
1535
1536
1537
1538
1539
1540
1541
1542
1543
1544
1545
1546
1547
1548
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582
1583
1584
1585
1586
1587
1588
1589
1590
1591
1592
1593
1594
1595
1596
1597
1598
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1610
1611
1612
1613
1614
1615
1616
1617
1618
1619
1620
1621
1622
1623
1624
1625
1626
1627
1628
1629
1630
1631
1632
1633
1634
1635
1636
1637
1638
1639
1640
1641
1642
1643
1644
1645
1646
1647
1648
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1668
1669
1670
1671
1672
1673
1674
1675
1676
1677
1678
1679
1680
1681
1682
1683
1684
1685
1686
1687
1688
1689
1690
1691
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718
1719
1720
1721
1722
1723
1724
1725
1726
1727
1728
1729
1730
1731
1732
1733
1734
1735
1736
1737
1738
1739
1740
1741
1742
1743
1744
1745
1746
1747
1748
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788
1789
1790
1791
1792
1793
1794
1795
1796
1797
1798
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1810
1811
1812
1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1868
1869
1870
1871
1872
1873
1874
1875
1876
1877
1878
1879
1880
1881
1882
1883
1884
1885
1886
1887
1888
1889
1890
1891
1892
1893
1894
1895
1896
1897
1898
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947
1948
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973
1974
1975
1976
1977
1978
1979
1980
1981
1982
1983
1984
1985
1986
1987
1988
1989
1990
1991
1992
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2048
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2068
2069
2070
2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093
2094
2095
2096
2097
2098
2099
2100
2101
2102
2103
2104
2105
2106
2107
2108
2109
2110
2111
2112
2113
2114
2115
2116
2117
2118
2119
2120
2121
2122
2123
2124
2125
2126
2127
2128
2129
2130
2131
2132
2133
2134
2135
2136
2137
2138
2139
2140
2141
2142
2143
2144
2145
2146
2147
2148
2149
2150
2151
2152
2153
2154
2155
2156
2157
2158
2159
2160
2161
2162
2163
2164
2165
2166
2167
2168
2169
2170
2171
2172
2173
2174
2175
2176
2177
2178
2179
2180
2181
2182
2183
2184
2185
2186
2187
2188
2189
2190
2191
2192
2193
2194
2195
2196
2197
2198
2199
2200
2201
2202
2203
2204
2205
2206
2207
2208
2209
2210
2211
2212
2213
2214
2215
2216
2217
2218
2219
2220
2221
2222
2223
2224
2225
2226
2227
2228
2229
2230
2231
2232
2233
2234
2235
2236
2237
2238
2239
2240
2241
2242
2243
2244
2245
2246
2247
2248
2249
2250
2251
2252
2253
2254
2255
2256
2257
2258
2259
2260
2261
2262
2263
2264
2265
2266
2267
2268
2269
2270
2271
2272
2273
2274
2275
2276
2277
2278
2279
2280
2281
2282
2283
2284
2285
2286
2287
2288
2289
2290
2291
2292
2293
2294
2295
2296
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378
2379
2380
2381
2382
2383
2384
2385
2386
2387
2388
2389
2390
2391
2392
2393
2394
2395
2396
2397
2398
2399
2400
2401
2402
2403
2404
2405
2406
2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441
2442
2443
2444
2445
2446
2447
2448
2449
2450
2451
2452
2453
2454
2455
2456
2457
2458
2459
2460
2461
2462
2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492
2493
2494
2495
2496
2497
2498
2499
2500
2501
2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518
2519
2520
2521
2522
2523
2524
2525
2526
2527
2528
2529
2530
2531
2532
2533
2534
2535
2536
2537
2538
2539
2540
2541
2542
2543
2544
2545
2546
2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581
2582
2583
2584
2585
2586
2587
2588
2589
2590
2591
2592
2593
2594
2595
2596
2597
2598
2599
2600
2601
2602
2603
2604
2605
2606
2607
2608
2609
2610
2611
2612
2613
2614
2615
2616
2617
2618
2619
2620
2621
2622
2623
2624
2625
2626
2627
2628
2629
2630
2631
2632
2633
2634
2635
2636
2637
2638
2639
2640
2641
2642
2643
2644
2645
2646
2647
2648
2649
2650
2651
2652
2653
2654
2655
2656
2657
2658
2659
26
```

6.7 再生と停止の処理

direction()の処理内にある play()メソッドを呼び出している部分の if 条件(図 41: 194 行目~195 行目)の bool 型変数「mute」の値を更新することで音の再生, 停止を制御できる仕組みをとっている.

条件を変更する処理は画面のボタンをクリックしたときに呼び出される.

ボタンを押したときの処理はクラス「MyLayout(Widget)」内に記述する. 二つの処理を以下に示す.

```
#muteボタンを押したとき(停止処理)
def mute_button(self):
    global mute
    mute = True

#playボタンを押したとき(再生処理)
def go_playing(self):
    global mute
    mute = False
```

図 42 再生, 停止処理のソースコード

6.8 BPM(テンポ)設定処理

基本的に変数「duration」の値を更新する。画面上の TextInput の値またはスライダーの値を取得して duration の値を更新する。

slider との値を連携する。Slider の使い方は参考文献【7】 slider の使い方を参照。

さらにtextInputからの入力にも対応するためTextInputの値を「self.ids.(kvで指定したID名).text」で取得する。ここで取得できる値は文字列なので整数型に変換する必要がある。

その後,BPM を以下の式で計算し duration に代入する。今回は最小値を 40, 最大値を 200 に設定した。

$$\frac{60 \text{ 秒}}{\text{指定したい BPM の数値}}$$

最後に TextInput 欄に残った文字列を空白リセットする。

これらの操作は「set」ボタンが押されたときに実行される。

以下に BPM 設定のソースコードを示す。

```
#setボタンを押したとき(BPMの更新)
def BPM_set(self):
    global duration    #BPMの値
    BPM = self.ids.BPM_input.text
    duration = int(60000 / int(temp_duration))
    if BPM.isdecimal():
        if int(BPM) >= 40 and int(BPM) <= 200:
            duration = int(60000 / int(BPM))

            self.ids.BPM_label.text = BPM    #画面表示を更新
            self.ids.BPM_slider.value = BPM  #スライダーの値をTextInputの値と同じにする
            self.ids.BPM_input.text = ""     #TextInputに残った値を空白リセット
            self.ids.BPM_input.text = ""     #数値が範囲外だった時に空白リセットさせる

#スライダーの値取得など
def BPM_slider(self, *args):
    global temp_duration #スライダーの値を格納
    temp_duration = int(args[1])
    self.ids.BPM_label.text = str(int(args[1]))
```

図 43 BPM 設定のソースコード

6.9 音程モード切り替え処理

視線方向取得モジュール(iris.py)の `get_direction()` に引数としてモード切り替え用の値を渡すことで音程モードの切り替えが可能。

iris_music.py の変数「music_mode」には0 または1 が代入される。0 が代入されたときは `get_direction` が「ド, レ, ミ, ファ」の音程を鳴らすための文字列を返す, 1 が代入されると「ソ, ラ, シ, ド」の音程を鳴らすための文字列を返す。この部分の iris.py のソースコードは図 29 である。

この処理はモード切り替えボタン(up, down)ボタンがクリックされたときに行われる。一つのボタンで処理を切り替えるため、表示の切り替えと機能の切り替えをする必要がある。変数「mode_text」にボタンに表示されている文字列を読み込み、その文字列が「up」「down」のどちらが表示されているかで判定する。この処理も同様にクラス `MyLayout(Widget)` 内に記述する。この部分のソースコードを以下に示す。

```
#music_modeの切り替え(「ソ, ラ, シ, ド」側にする)
def mode_high(self):
    global music_mode
    music_mode = 1

#musi_modeの切り替え(「ド, レ, ミ, ファ」側にする)
def mode_low(self):
    global music_mode
    music_mode = 0

#ボタンの機能とモードの切り替え
def mode_ch(self):
    mode_text = self.ids.mode_change.text
    if mode_text == "up":
        self.ids.my_image.source = "./image/new_gabc.png"
        self.ids.mode_change.text = "down"
        self.mode_high()
    else:
        self.ids.my_image.source = "./image/new_cdef.png"
        self.ids.mode_change.text = "up"
        self.mode_low()
```

図 44 モード切替のソースコード

6.10 説明画面の表示方法

説明画面は popup 要素で出現させる。MyLayout(Widget)のボタンを押した処理で別に定義したクラス「PopUpLayout(BoxLayout)」を機能させる popup 処理を行う。詳細の解説は、参考文献【8】
<https://senablog.com/python-kivy-popup/>を参照する。

ソースコードは以下の通り。

```
class PopUpLayout(BoxLayout):
    popup_close = ObjectProperty(None)

    def img_back(self):
        self.ids.carousel.load_previous()

    def img_next(self):
        self.ids.carousel.load_next()

class MyLayout(Widget):
    def popup_open(self):
        content = PopUpLayout(popup_close=self.popup_close)
        self.popup = Popup(title='操作説明', content=content,
                           size_hint=(0.7, 0.7), auto_dismiss=False)
        self.popup.open()

    def popup_close(self):
        self.popup.dismiss()
```

図 45 説明画面を表示するためのソースコード

第7章 終わりに

本研究は、コミュニケーション手段の一つとして「眼の動き」に注目し、眼の動きをトラッキングして操作できるツールに関する今後の研究の基盤になればよいと思う。

視線を取得する方法として角膜反射法(参考文献【9】)がある。しかし、この方法では眼に光を当てなければならず目への負担がどうしても避けられない。本研究では、外部からの人体への影響を極力減らして実行することが可能となった。

本研究を通して、眼の動きをトラッキングする専用のハードウェアが無くても眼の動きを検知することが可能であると確認できた。さらに、小さい子供から大人まで「眼の動き(上下左右)」を使って音楽を演奏することができるアプリを完成させた。

今後の課題として

左右判定の精度は十分に作成することができた。しかし、上下の判定はまだ改善の余地がある。特に上の判定は修正が必要である。

再初期段階で考えていた案は視線でPCのマウスのように画面上を操作できるようなツールだった。本研究では、市販のwebカメラから画像認識により目の動きを取得するため、目の細かい動きを検知するのが困難だった。そこで、妥協案として方向を取得するだけなら誤差と区別できる明らかなデータが取得できると考え実装した。

今後この研究がもし、引き継がれることがあるならば、上記に述べたように視線でPCのマウスのように画面上を操作できるつまり、「目の細かい動きを検知できる」機能を実装してほしい。

参考文献

【1】 Face Mesh の使い方

https://blog.deepblue-ts.co.jp/image-processing/how_to_use_face-mesh/

【2】 2 点間の距離

<https://www.higashisalary.com/entry/numpy-linalg-norm>

【3】 2 点を通る直線から離れた位置にある点までの距離を求める

<https://tokibito.hatenablog.com/entry/20121227/1356581559>

【4】 指数移動平均法

<https://bigdata-tools.com/moving-average/>

【5】 kivy の基本

<https://torimakujoukyou.com/category/python/kivy/>

【6】 kivy のレイアウト関連

<https://senablog.com/category/programming/python/kivy/>

【7】 slider の使い方

<https://torimakujoukyou.com/python-kivy-slider/>

【8】 kivy のポップアップ

<https://senablog.com/python-kivy-popup/>

【9】 角膜反射法

https://www.jstage.jst.go.jp/article/vision/3/2/3_81/_pdf