視線検出による非接触のPC制御システム

刈谷高校スーパーサイエンス部　情報班

研究の目標

手を使えない状態でもPC操作できるシステムの実現

Webカメラで視線を検出し、カーソル操作を行うプログラムを開発したが、視線検出精度が不完全だったので、精度の向上を目標とした。※2021年度前期研究…Ver.1　本研究…Ver.2

研究の方針

視線検出の精度不足は、機械学習の際のデータの偏りや不足に起因していると考えられるため、以下のことを行った。

・データ量を増加させるためのデータ収集用Webアプリ作成
・学習させる特徴点の座標追加
・データ収集時PC毎に位置調整用のキャリブレーションを行う

主な使用ライブラリ・ソフトウェア

・python――――機械学習等に使用したプログラミング言語
・spyder――――統合開発環境
・dlib ――――顔の特徴点検出に利用したライブラリ
・OpenCV――――画像処理のライブラリ
・scikit-learn――機械学習のライブラリ
・JavaScript――データ収集等に使用したプログラミング言語
・HTML――――上に同じ

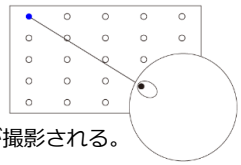
研究方法

①データの収集
自作のWebアプリで、定点を見ている顔の写真を撮影する
②特徴点の取り出し
虹彩などの顔の各所の座標を取り出して、データベース化する
③機械学習
特徴点から視線の座標を推定する式を線形回帰により作成する
④テスト・評価
推定した視線の先の座標と実際の座標との誤差を測る
⑤実用例
視線検出プログラムを使って実際にカーソルを動かす

ソースコード：
https://github.com/shotaro27/eye-tracker-webcam

①データの収集

(Ver.1)800枚→(Ver.2)**1300**枚撮影
画面に順番に表示させた25個の点を見ている顔を撮影する。
なお、顔の認識に成功した画像だけを使うので、学習やテストに使う画像の総数より多くの写真が撮影される。
今回は学習に777枚、テストに243枚使用



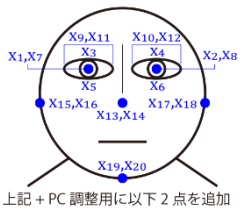
②特徴点の取り出し

顔の傾きや位置に合わせて変わる点の座標を取り出した。
虹彩の座標はそのままでは取り出せないため、
以下の処理を行った。


①特徴点をもとに目を切り抜く


②二値化処理する

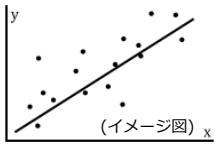

③虹彩の重心の座標を求める



上記＋PC 調整用に以下 2 点を追加
X21:目の寄り具合 X22:虹彩の y 座標

③機械学習

線形回帰とは
いくつかのデータの点の散らばりから
近似直線の式を計算して求める。



得られた式

x座標

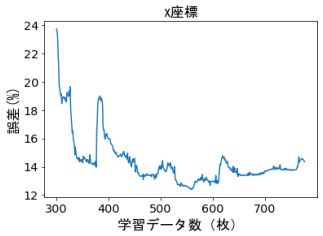
y_1=-167.272x_1-95.949x_2-0.565x_3+2.965x_4+0.074x_5-1.56x_6-1.308x_7 +3.949x_8-4.198x_9 -0.246x_{10}-0.106x_{11}+0.702x_{12}+3.028x_{13}-1.248x_{14} +0.124x_{15}-0.5x_{16}+1.814x_{17}-0.326x_{18} -0.81x_{19}+0.267x_{20}+74.721x_{21}-2.12x_{22}+377.591

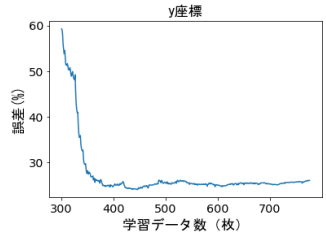
y座標

y_2=10.461x_1-37.132x_2+5.728x_3-1.153x_4-4.927x_5+2.998x_6+0.902x_7+0.373x_8+2.878x_9 +0.401x_{10}+1.071x_{11}+0.922x_{12}-1.856x_{13}-1.017x_{14}-0.559x_{15}-1.38x_{16}-1.868x_{17}-0.347x_{18} +0.187x_{19}+0.388x_{20}-139.244x_{21}+ 8.423x_{22}-427.597

④テスト・評価

テストデータを用いて、以下の値を求めた。
A:抽出した特徴点の座標を、学習で得た式に代入し、推定した座標
B:実際にテストデータを撮影したときに見ていた点の座標
これをもとに|A - B|の平均値を求め、学習モデルの誤差とした。
以下のグラフは、学習に用いるデータ数の増加に伴う誤差の変化を示す。

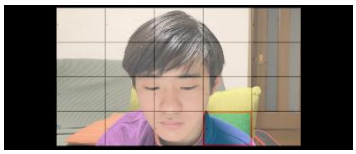




最終的な誤差　x座標：14.4%　y座標：26.1%　(Ver.2)
x座標 ： 28.9%　y座標 ：32.8%　(Ver.1)

⑤実用例

実用例① 視線によるカーソル移動 (資料動画0:00~0:32)



赤線で囲まれた5×4のマスにカーソルを一秒間入れると次のマスに移るプログラムをHTML・JavaScriptを用いて作成した。

実用例② 視線によるスクロール (資料動画0:32~1:00)

画面中央を見ているときはスクロールせず、画面上下を見たときだけスクロールするプログラムをPythonを用いて作成した。

現在では目の高さを一定に保っていなければきちんと認識できない。
事前にキャリブレーションを行う必要がある。

考察と今後の展望

Ver.1に比べて、Ver.2では視線によるカーソル移動がより正確になった。しかし、x・y座標共に学習するにつれ改善されなくなっていく。同じ内容のものを学習させているため、何枚か入れた時点で学習はほぼ完成され、それ以降は誤差の減少率が少なくなると考えられる。また同じ人を多く学習していることや、光の当たり具合によって虹彩の位置がしっかり認識できていないことも要因に挙げられる。
今後は特徴点と合わせて、画像から学習する方法も検証したい。