目線検出ソフトの作成

５年Ａ組　稲益　秀成
指導教員　米田　隆恒

１．要約

本研究では、目線の変化によってコンピュータを操作できるように、コンピュータを操作しているときの利用者の「黒目」の動きから目線を検出し、目線を使った入力装置となるソフトウェアを開発することを目的とする。

今回作成したソフトウェアは、Web カメラで利用者の顔を撮影し、その映像から目線を検出し、目線に合わせてマウスポインタを制御するものである。

キーワード　目、目線、目線検出、Web カメラ、OpenCV、Processing

２．研究の背景と目的

人間は他人の目を見ると、その目線と目線の先にあるものから、その人が何を見ているのかをおおまかに知ることができる。そこで私は、人間の目は視覚情報を得るだけでなく、自分がどこを見ているかを他人に知らせる役割も持っているのではないかと考えた。

今日幅広く普及しているコンピュータやスマートフォンなどが人間の目線を情報として取得できれば、これらの電子機器の使い方が広がるのではないだろうか。例えばコンピュータが利用者の見ているものを検出できれば、そこから利用者の興味のあるものを予測することができるかもしれない。

あるいは、マウスやタッチパネルの代わりに目線をポインティングデバイスとして使うことも考えられる。

これらの理由から、私はコンピュータを通じて利用者の目線を得る方法について考えていくことにした。

３．研究内容

まず、コンピュータが利用者の目線を取得する方法としては、Web カメラを用いた。Web カメラとは、Skype や FaceTime など、インターネットでテレビ電話を楽しむときに相手にリアルタイムで送信する、自分の顔を撮影するカメラのことである。最近のノートパソコンや一体型デスクトップパソコンには標準で搭載されていることが多く、外付けのものも安価で購入できる。(図１)

図１　Web カメラ

また、最近のスマートフォンや、タブレット端末などには同様な用途のカメラがほぼすべての機種に搭載されている。(図２)

図２　スマートフォン等のインカメラ

つまり、開発したソフトウェアをインストールするだけで、新たに専用の装置を追加しなくても、Web カメラさえあれば目線検出が行えるというメリットがある。

プログラミング言語は Processing を使用した。これは、ソースコードを多少変更するだけで、Windows OS、Mac OS X、Linux、さらには Android OS でもソフトウェアを実行することができるため、幅広いデバイスで目線検出が行えると考えたからである。

今回開発したソフトウェアのプログラムの流れは以下の通りである。

①Web カメラから映像を取得する。

②取得した映像に対して OpenCV ライブラリによる顔検出を行う。

③OpenCV の顔検出結果から、映像中の顔の中心の位置、顔の肌の色を計算する。

④顔の位置と肌の色のデータを用いて映像中の顔の範囲を検出する。

⑤顔の範囲内の黒目の位置を検出する。

⑥黒目の位置に合わせてマウスポインタを動かす。

また、以降のフレームでは前フレームの顔の位置と肌の色を用いて顔の追尾処理を行うので、毎回 OpenCV での顔検出（①～③) を行わず、④から⑥のみの処理を行う。

なお、OpenCV については次で説明する。

（１） OpenCV を用いた顔検出

映像から黒目の位置を検出するために、まず映像中の顔の範囲を検出するべきだと考えた。これは、最初に映像全体から黒目を探す方法を試みたところ、黒目と形状が似通っているものが背景に映り込んでしまうと誤認識してしまい、実用的ではないと判断したからだ。そこで現在、写真管理ソフトや SNS の Facebook、その他いろいろな場面で実用化されている顔検出技術を使った後、その顔の範囲から黒目を探した方が、背景の影響を受けにくいと考えた。

顔検出方法としては、インテル株式会社が発表したオープンソースライブラリーである OpenCV を利用したパターン認識を使う方法を選んだ。

図３　顔検出におけるパターン認識の例
（参考文献[５]より引用）

これは、鼻筋の方がその隣の目のある部分の方より手前にあり、光が当たりやすいため明るくなる、というような顔の一部分と別の部分との色の違いの特徴をデータベースに持っておき、（図 3 参照） このデータをパターンとして映像の左上から 1 ピクセルごとにずらしながら、今調べている範囲の映像の特徴が、データベース上の顔の特徴と一致するかを調べていく方法である。

この顔の特徴のデータのひとつひとつでの認識率は低いものだが、２つの部分の色の違いを調べていくため、シンプルで計算量が少なくて済む。それらをたくさん用意することで認識率を高めていくことができる。

しかし今回、製作するプログラムは、リアルタイムに目線検出を行える必要があるため、すべての処理が軽いものでなくてはならない。

OpenCV による顔検出はシンプルであるとは言え、CPU に Corei５を搭載している現在一般的なパソコンでも一枚の FullHD サイズの画像の処理に１秒近くかかるほど処理が重いものである。そこで、この方法の顔検出はプログラムを開始した初回フレームのみにし、以降のフレームでは顔の肌の色の部分を追尾するようにした。この方法により、高精度の顔検出結果をリアルタイムに得ることができると考えたからである。

（２） 初回フレーム以降の顔の範囲の検出

初回フレームでの OpenCV の顔検出結果から顔の中心位置、そして顔の範囲内の色より、肌の色を取得したあと、これらのデータを用いて顔の位置を追尾し、顔検出を続ける方法をとった。

また、追尾をする処理もできる限り軽くするために、前のフレームの顔の範囲の中心の点の位置からまず上下に色を調べ初回に取得した肌の色の色相と比較し、肌の色でなくなるまで上下に範囲をのばして調べ続ける。その後上下にのばした範囲から左右に肌の色が続く範囲を調べていくことで顔の範囲を調べる方法をとった。(図 4 参照) 次回フレームではこの処理で計算した新たな顔の範囲から中心を計算して、同様の処理を行う。

肌の色か否かを判断するのに色相を用いたのは、たとえ顔に陰ができても、色相は値の変化が小さいため、光源による影響を受けにくいと考えたからである。

（３） 黒目の位置の検出

(2)の方法で検出した顔の範囲から、黒目の位置と、黒目の中心点の位置を調べる必要がある。

黒目を検出する処理としては、黒目にみられる、「黒い」「２つ横に並んでいる」「顔の上半分にある」「隣に白目の白色がある」という４つの特徴を利用することにした。

特に、「隣に白目の白色がある」という判定によって、比較的形状が似ている眉毛や鼻の穴などとの誤認識を大幅に減らすことができた。

プログラムの処理としては、以下のような順で行った。

① 顔の上半分の範囲の中で、白色と黒色が隣り合っている部分を探す。

② 黒色と白色が隣り合う部分が４カ所以上ある横方向の列をさがす。
③ ②を満たす列が続いている範囲を目がある範囲とし、その中から黒い部分をさがす。
④ ③の結果から二つの黒目の範囲を探し、その黒目の位置を点で表すために黒い部分の重心を計算する。

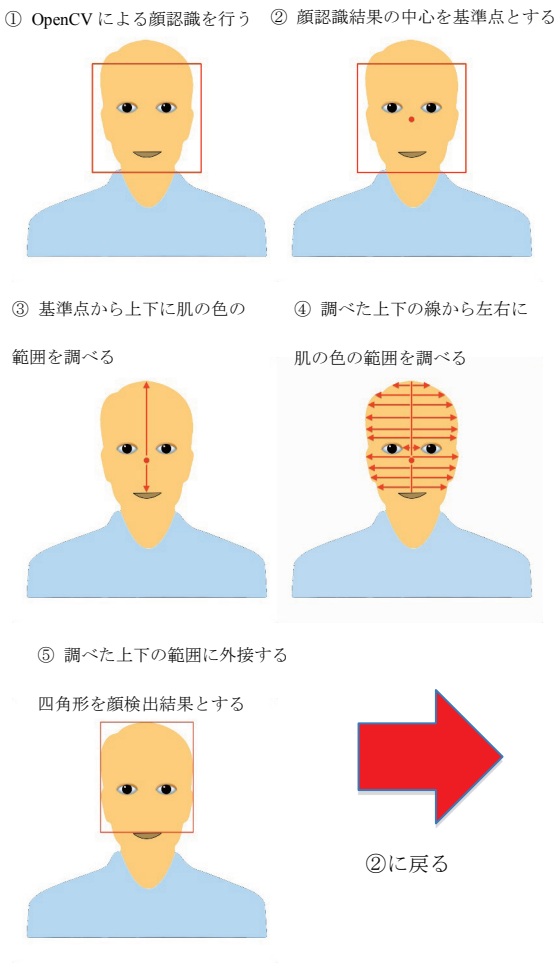

図４　顔認識の追尾の流れ

②において、４カ所としたのは、２つの黒目で白→黒と黒→白となる計４カ所あるはずだからである。

また、４カ所「以上」としたのは、図５のようにコンピュータのディスプレイの光で黒目の中央が白く反射してしまうときには４カ所より増えてしまうからである。

④において、黒目の点の計算に重心を用いたのは、図５のように黒目の中央がディスプレイの光で白く反射してしまっても、黒い部分の重心を計算することで、ある程度反射による誤差を減らすことができると考えたからである。

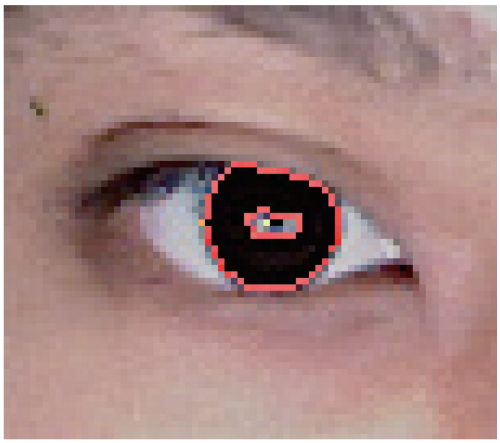

図５　黒目が反射している様子

（４） マウスポインタの制御

ここまでの処理によって、映像中の顔の範囲の中心、黒目の位置を点で取得できたため、この２つの位置関係からマウスポインタの制御を行った。

マウスポインタの制御を開始する前に、利用者に画面の上下左右の端をみてもらい、そのときの黒目と顔の位置関係を保存する。そして保存した位置関係と黒目の点と顔の点の位置関係を比較することで、マウスポインタを制御した。

4. 結果

今回、Apple 社製 iMac Mid2011(画面サイズ 21.5 インチ、 CPU corei5 2.7GHz、メモリ 4GB)のモデルで本ソフトウェアを

実行した。顔と画面との距離は約 60cm である。

実際に目を動かしてからプログラムがその動きを認識するまでの遅延は約 0.2 秒であった。

1920×1080 ピクセルのサイズの Web カメラ映像からは、21.5 インチディスプレイを横に約 22 等分、縦に約 11 等分した細かさで見ている場所を示すことができた。

しかし、まだ顔の位置の変化に対応していないため、利用者の顔の位置が変わってしまうと目線検出を続けることはできない。

また、顔の範囲の検出において、眼鏡を使用した場合を考慮できていないため、眼鏡を使用した場合は目線検出を行えない。

5. 考察

今回のような方法を使えば Web カメラの映像から利用者の目線を検出し、おおまかに画面のどのあたりをみているのかを取得することができると分かった。また、目を動かしてから反応までの遅延は約 0.2 秒と、実用的な速度で目線検出を行えた。

現在目線検出デバイスとして存在する、頭にとりつけ目に近い位置にカメラを設置して目線を検出するタイプの装置に精度は劣るが、本研究の方法では、デバイスに標準装備されたもの、あるいは安価に市販されているものを使用するため、追加費用をかけず手軽に、多くの人が目線検出を行えるというメリットがある。

しかしながら現在は顔の位置の変化に対応しておらず、実際に実用化するためには顔の位置の変化に対応する必要がある。人は無意識に目線の変化に応じて首が動いてしまうため、顔の位置を動かさないように意識していても、どうしても顔の位置が変わってしまうのである。

6. 課題と展望

今回の研究において、Web カメラで撮影した映像から目線を検出し、短い遅延でマウスポインタを制御することに成功した。しかしながら、現在、顔の位置の変化には対応しておらず、コンピュータを使用する上で目線の変化による首の動きの変化が意識していても発生してしまうため、正しく見ている場所にマウスポインタを移動させるのはとても難しい。そのため、まず顔の位置の変化にソフトウェアが対応する必要がある。そのためには顔と画面の位置関係を立体的に把握しなければならない。顔の向きは鼻頭の位置の変化で読み取ることができると考えている。顔と画面の距離は映像中の顔の大きさの変化でよみとることができると考えている。

そして、将来の展望としては、マウスポインタにリンクさせることでコンピュータの操作をするだけでなく、マウスやタッチ操作の補助としてこの目線検出システムを活用したいと考えている。具体的な例としては、文書をよんでいるときに、目線認識で利用者がどこを読んでいるかを知ることで、自動スクロール、自動ページ送りができるのではないかと考えている。

また、これが完成すると、体の不自由な人でも目線の動きだけでコンピュータを操作できるようになるだろう。例えば、画面に「あいうえお…」といったような文字の表を表示しておき、それを一定時間見ることでその文字を選択したことにすれば、文章入力はもちろんだが、その文字を音声で

出力することで、体が不自由な人でも会話することができるようになる。このような社会貢献につながると考えられる。

さらに、別の考え方として、利用者がインターネットブラウザを使っているときに、Web サイトのどこを見ているかがわかると、その利用者が引きつけられた内容がわかり、そこからその利用者の興味がある内容を知り、どのような内容が読んでもらえるかを今まで以上に調べることができるようになるので、うまく使えば、Web サイトの質の向上にも繋がると考えられる。

６．参考文献

[1] Processing Reference
「http://processing.org/reference/」
[2] OpenCV ¥ Library
「http://ubaa.net/shared/processing/opencv/」
[3] コンピュータビジョンのセカイ - 今そこにあるミライ
「http://news.mynavi.jp/series/computer_vision/008/index.html」
[4] RGB から HSV への変換と復元
「http://hooktail.org/computer/index.php?RGB%A4%AB%A4%E9HSV%A4%D8%A4%CE%CA%D1%B4%B9%A4%C8%C9%FC%B8%B5」
[5] OpenCV.jp
「http://opencv.jp/」

７．謝辞

今回の研究にあたり、サイエンス研究会物理班顧問の米田隆恒先生、藤野智美先生には、研究に関する助言や指導をしていただきました。また、同研究会のメンバーには研究のサポートをしていただきました。この場を借りてお礼申し上げます。