

G-01

組み込み Linux を使用した正面顔での視線方向認識の考察

Investigation on detecting the line of sight from frontal face photographs using Embedded Linux Device

難波田 ゆう子† 渡部 健二‡ 岡田 繁† 中野 秀男‡ 大西克実††
Yuko Naniwada Kenji Watanabe Shigeru Okada Hideo Nakano Katsumi Onishi

1. はじめに

肢体不自由者を対象とした視線入力装置は、カメラ(赤外線を含む)の高性能化、低価格化によっていくつかのメーカーで開発されるようになってきている¹⁾。しかし、安定した入力が行えるシステムは赤外線照射を利用したものが多く^{2) 3)}、高価であるとともに疾病の可能性が否定できず大きな普及には至っていない。

赤外線照射などを行わず、自然な環境下で視線の高性能な検出に関する研究がいくつか行われている。阿部ら⁴⁾は、重度肢体不自由患者の頭部移動が小さいことに着目し、室内照明のもとに、視線検出を行う方法を提案している。そのシステムでは、ユーザごとに上中下それぞれの位置の注視画像を記憶し、それらから光強度分布を求め基準データとしてユーザごとに辞書化することで、光強度の誤差を抑えている。

本研究は、このようなユーザごとの辞書化は行わず、一定の条件を満たした環境下であれば様々な人の顔に対応できる特徴を有する。また、正面に座っている人の2方向の視線を判別可能なシステムを非常に低価格な構成で提案する。低価格な構成を実現するために\$35で購入可能な名刺サイズ(85.60mm×53.98mm)のシングルボードコンピュータである Raspberry Pi と\$25で購入できる専用カメラモジュールを使用する。本システムを実現するためには他にSDカード、microUSB電源があれば可能であり、全体でも7,500円以下での構築が可能である。そして、カメラとの距離を0.5m~1.5mの違和感のない範囲で視線を認識できることを示す。

2. システム構成

本研究では、視線検出アルゴリズムは図1(a)に示されるような一般的なパソコン(クロック周波数1.8GHz)とモニター、モニター上部に備え付けられたウェブカメラにより開発される。開発されたアルゴリズムは、オープンソースハードウェアの Raspberry Pi(クロック周波数700MHz)上に専用のカメラモジュールを取り付けたシステム(図1(b))に移植される。専用カメラモジュールは\$25という安価さでありながらも、フルHDまで撮影することが可能である。また30fpsでの動画を撮影できる。性能としては現在市販されているウェブカメラでは当然のレベルであるが、フラットケーブルを使用したコンパクトなサイズであることもあり今後の発展を考え採用した。Raspberry PiではRaspbian(Debian LinuxのRaspberryPi用)のOSが使用される。

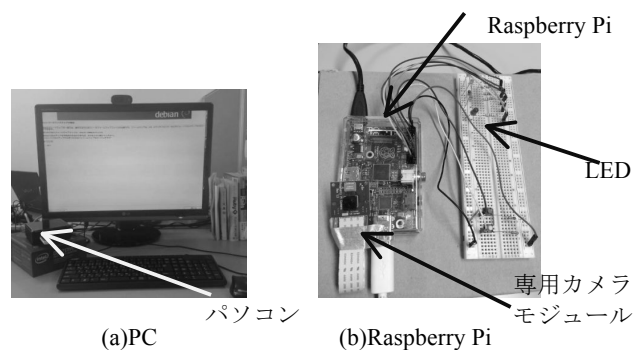


図1 システム構成

3. 視線方向判別手法

ここでは、人が正面および斜め方向に視線を向けたときの視線を Raspberry Pi と小型カメラによる小規模なシステムにより判別するための手法について説明する。

3.1 人間の上半身の撮影

まず、次節以降に示す画像処理手順において必要な顔の特徴の定性的性質およびパラメータ値を、多くの顔画像を基に獲得するため、3種類のカメラを使用して65名分の顔画像を収集する。ただし、すべての被験者に対して実験の目的および内容を事前に説明し、十分なインフォームドコンセントが得られた上で実験を実施している。

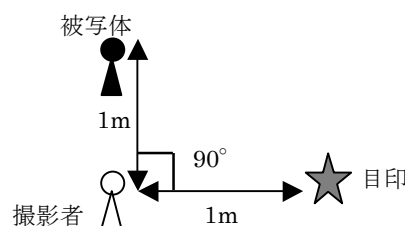


図2 評価用画像収集イメージ

人間が正面および斜め方向に視線を向けたときの上半身の撮影を、以下の方法で行う(図2)。

- (1) カメラの真正面1mの位置に座る。
- (2) カメラの1m右に目印を配置する。
- (3) 人間は顔をカメラに向け、カメラを見つめた状態で撮影。
- (4) 人間は顔をカメラに向けるが、視線は目印に移した状態で撮影。

† 電脳・工房, TAKUMICOBO

‡ 帝塚山学院大学, Tezukayama Gakuin University

†† 大阪市立大学, Osaka City University

3.2 画像処理による目頭と瞳孔の検出

Raspberry Pi と小型カメラによる小規模なシステムにより視線を認識するための画像処理手法として、顔画像から目頭と瞳孔を検出し、それらの位置関係から推定する方法を採用する。以下にその手順を説明する。

(1) 顔および目の検出

まず、上半身の画像から顔部分を検出する。顔検出には、Haar-like 検出器を使用した OpenCV に実装されているオブジェクト検出プログラムを利用する。図 3 の矩形で囲まれた部分が検出された顔領域である。以降の処理のために、顔領域を $400\text{pix} \times 400\text{pix}$ に正規化する。

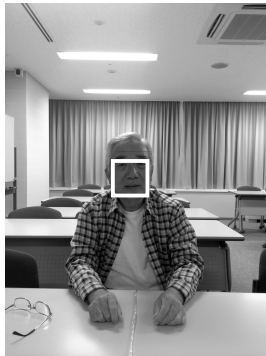


図 3 顔検出イメージ

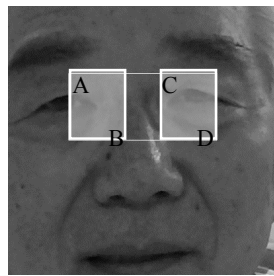
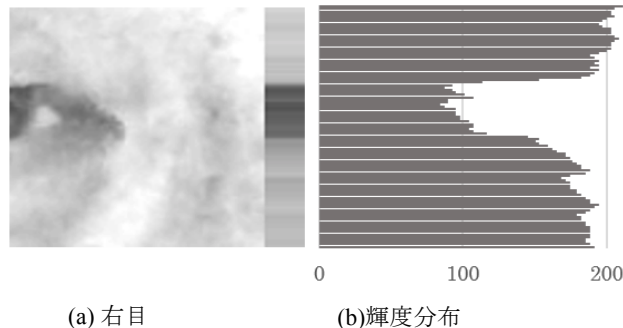


図 4 正規化された画像に対する目のエリア

次に、得られた顔画像から、目の上限下限を検出するためのエリアを決定する（図 4）。右目のエリアは正規化した画像の A(90,100)から B(180,199)の範囲とし、左目のエリアは C(220,100)から D(310,199)とする。これらの値は、人の目は顔の上半分にあること、また右目の幅、左目の幅、左右の目頭の距離がほぼ同じになることを収集した顔画像の分析により確認しており、それらに基づき決定している。

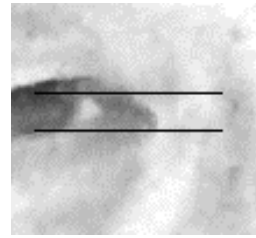
図 5 に、目の上限下限を決定するためのエリアの輝度から、横方向の輝度の最低値の輝度分布を表示した。図 5 の輝度分布において、下から上に1つつ候補を調べたときに130以下でしかも3倍した値が1つ上、2つ上、3つ上の合計値に30を足した値よりも大きい条件を満たしている場合にその候補を下限であると決定した。上限の決定は、次に再び130を超え、値を3倍したとき一つ下、二つ下、三つ下の合計値に30を足した値よりも大きい条件を満たしている候補を上限であるとした（図 6）。



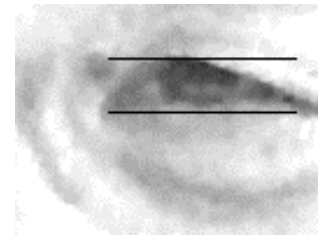
(a) 右目

(b) 輝度分布

図 5 目の輝度



(a) 右目

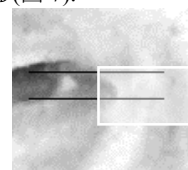


(b) 左目

図 6 目の上限下限

(2) 目頭候補検出

目頭の検出を行うために、まず目頭が存在する可能性が高い領域のエッジ検出を行う。縦方向は決定した目の下限の20[pix]下から上限まで、横方向の右目は130[pix]から180[pix]、左目は220[pix]から270[pix]までを二値化する。二値化に使用した閾値は、り収集した画像で必要なエッジが出ているか検証した結果40に決定された。目頭の検出には、エッジ画像の顔の中心の下方向から外側へ向かい、”>”や”<”のような2本の直線が角度をもってつながっている形をしたもので上側の直線が4[pix]以上、下側が4[pix]以上のものを探索すれば必ず目頭が含まれることを収集した顔画像の解析により確認しており、そのつなぎ目と考えられる部分（頂点の部分）を目頭候補として保持する（図 7）。



(a) 検索範囲



(b) 目頭検出用の二値化エッジ画像



(c) 目頭候補の検出結果

図 7 目頭候補の検出

(3) 瞳孔検出

目頭候補の中の最初に検出された目頭を使用し、瞳孔検出エリアを設定する（図 8）。具体的には、正規化された顔画像のグレースケール画像において、目の上限、下限、そして目頭の検出で最大10個まで検出した目頭候補の中で最初に検出された目頭を使用し、縦は目の上限と下限を、横は最初に検出された目頭の候補から左目は-100pixまでの幅、右目は+100pixまでの幅の中を瞳孔探索エリアとしている。続いて、瞳孔検出エリアを25近傍でメディアンフィルタを使用してぼかし（図 9）、2値化処理により黒目部分を検出する（図 10）。二値化の閾値の決定方法は、検証した結果、前述でぼかした画像の輝度をする。輝度の最低値が20以上であり、平均値との差が20以上の場合は最低値+20を閾値として使用し、そうでない場合は最低値に平均値との差を3で割った値を足した値を閾値とした。



(a) 右目



(b) 左目

図 8 瞳孔探索エリア



(a) 右目



(b) 左目

図 9 瞳孔探索エリアを 25 近傍でメディアンフィルタにかけた結果

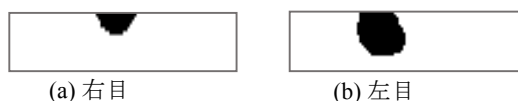


図 10 二値化の結果

(4) 目頭候補から目頭を決定

次に、目頭候補の検出で挙がっていた候補の中から目頭を確定させる。目頭の確定には、検出した左右の瞳孔を結ぶ直線の傾きを使用する。左右の目頭を結ぶ直線の傾きが最もその傾きと等しいペアを最終的に目頭として確定する。



図 11 瞳孔の傾きと目頭の確定

(5) 視線方向の推定

目頭と瞳孔の位置関係を利用して正面に視線が向いている時と 45 度先に視線が向いている時を推定する。収集した評価用のデータ全てに対して目頭と瞳孔の正解点を予め登録し、それらに基づいて正面に視線がある時と 45 度先に視線が向いている時の目頭と瞳孔との距離の平均値をカメラ毎に求めたものを表 2 に示す。目頭と瞳孔の検出が完了した後は、元データの座標へ戻して認識処理を行うので表 1 の値は元画像での距離となる。

表 1 カメラ別左右の目頭瞳孔間の距離差

カメラの種類	撮影画像 サイズ[pix]	左右の目頭瞳孔間の距離 差[pix]	
		正面視線 時	45 度視線時
iPhone5s	2448x3264	3.97	35.20
SONY DSC-W320	4320x3240	7.57	90.20
EOS Kiss X2	2256x1504	2.25	29.61

正面顔の正面に視線方向がある場合と 45 度先に視線方向がある場合とでは、目頭・瞳孔間の距離に大きな差が生じている。また、正面に視線がある場合には左右の目頭・瞳孔間の距離はほぼ等しくなり、左 45 度を向いている場合には左目の目頭・瞳孔間の距離は大きく、右目の目頭・瞳孔間の距離は小さくなることを確認した。これらをもとに、“検出した左右の目頭と瞳孔間の距離差[pix] ÷ 横幅[pix] × 100” で求めた値が 1 未満なら正面、それ以外は 45° として視線方向の判別を行なった。

また、3 種類のカメラを使用した理由は、複数の画像サイズで撮影した画像での瞳孔・目頭間の距離の差を求める必要があると考えたからである。

3.3 組み込み Linux 環境への移植

開発環境と Raspberry Pi 環境では同じ Debian Linux 系のディストリビューションで行なっているため、OpenCV のバージョンを揃えるだけで移植が可能となっている。

4. 結果と考察

4.1 収集した画像に対する認識結果

収集した画像を Raspberry Pi に組み込んだシステムを使用して視線を判別した結果を表 2 に示す。

表 2 Raspberry Pi の結果

		対象 数	検出率(%)		視線方 向の検 出率(%)	処理時 間(sec)
			顔	目頭・ 瞳孔		
iPhone5s	正面	11	63.6	100	100	7.73
	目印	11	54.5	100	100	7.6
EOS Kiss X2	正面	21	95.2	95	100	4.59
	目印	18	83.3	86.7	100	4.74
SONY DSC-W32	正面	24	91.7	86.3	100	15.35
	目印	25	88	81.8	100	15.08

表 3 Raspberry Pi2 の結果

		対象 数	検出率(%)		視線方 向の検 出率(%)	処理時 間(sec)
			顔	目頭・ 瞳孔		
iPhone5s	正面	11	63.6	100	100	3.02
	目印	11	54.5	100	100	3.03
EOS Kiss X2	正面	21	95.2	95	100	1.82
	目印	18	83.3	86.7	100	1.89
SONY DSC-W32	正面	24	91.7	86.3	100	6.13
	目印	25	88	81.8	100	6.00

表において、目頭・瞳孔の検出率は、顔を検出できたものの中でのパーセンテージを示し、視線方向の検出率は顔の検出が行え、且つ、目頭・瞳孔の検出が行えたものの中でのパーセンテージを示している。

リアルタイムで動作することを目指すには処理時間は大きくなったが、これは画像のサイズが大きいためであり、全ての検証において顔検出、目の検出、事前処理(ファイル読み込み等)の時間を別に計測したところ 6 7 %程度が顔検出にかかっていることが確認できた(図 12)。しかしながら顔の検出が行え、且つ、目頭・瞳孔の検出が行えたものの中で視線方向の判別率は 100%であり、提案したアルゴリズムで良好に判別できていることが確認できる。

Raspberry Pi と Raspberry Pi2 での結果を比較すると、同じ画像を入力した場合に検出結果は完全に一致したことを確認できた。Raspberry Pi2 での処理時間が Raspberry Pi に対して半分以下になっていることを確認できた。

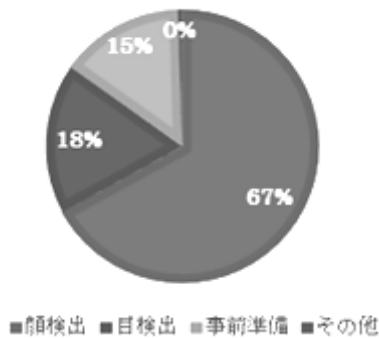


図 12 処理時間

4.2 Raspberry Pi 専用カメラモジュールを用いた実験と結果

5名の被験者を対象に、専用カメラモジュールを含む Raspberry Pi 上でのシステムを用いて画像獲得、解析を行った。カメラモジュールを使用した際の流れは、以下のようである。まず、画像サイズの指定をしてスナップショット撮影を行う。次に撮影した画像に対して視線の判別を行う。そして、LED へ判別結果の出力を行う。

表 4 視線認識成功回数

	フル HD						VGA					
	0.5m		1.0m		1.5m		0.5m		1.0m		1.5m	
	正面	目印	正面	目印	正面	目印	正面	目印	正面	目印	正面	目印
A	5	5	4	4	4	3	5	5	4	3	3	1
B	5	5	5	5	5	5	5	5	5	4	5	5
C	5	5	5	5	4	4	5	5	3	3	5	2
D	5	5	5	5	5	5	5	5	5	4	5	2
E	5	5	5	5	4	3	5	5	5	4	3	5

収集画像を使用した判別結果で処理時間が画像サイズに左右されることが確認出来たので撮影するスナップショットの画像サイズは、フル HD(1920[pix] × 1080[pix]) と VGA(640[pix] × 480[pix]) の 2 種類を使用し、各々、0.5m, 1.0m, 1.5m と距離を変えてそれぞれ 5 回ずつ正面視線と 45° 左右に視線を移動させた。5 名の被験者がそれぞれのサイズと距離、方向に対して何回視線方向を判別したかを記録した結果を表 4 に示す。値は 5 回中何回成功したかを意味している。ただし、Raspberry Pi 専用カメラモジュールを用いた場合でも顔検出された画像を 400[pix] × 400[pix] で正規化するため、3 種類のカメラの時と同じように判別を行っている。

実験結果ではフル HD の方が判別の成功率は高いが、使用感を被験者に確認すると、認識完了までの時間が長く、その待ち時間に対して違和感を抱くことがわかった。一方、VGA の方が認識率は低い、処理が高速に行われるため、フル HD で感じた違和感を抱く被験者はいなかった。

また、専用カメラモジュールを使用して 1m の距離で計測した際の処理時間を表 5 に示す。

表 5 Processing time using the Raspberry Pi Camera Module

	VGA(640×480)	フル HD(1920×1080)
1.0m	2.45sec	4.30sec

専用カメラモジュールを使用した場合でも時間がかかっているが、画像の撮影、顔検出、その後の視線判別については十分に動作していると考えられる。現在は、リアルタイム

感を出すには VGA 程度のサイズが限界かと考えられるが、ムーアの法則で考えると名刺サイズのシングルボードコンピュータも数年後には大きく進化し、さらに大きな画像をリアルタイムに処理できるようになると考えられる。

5. おわりに

本報告では、まず、Raspberry Pi に専用カメラモジュールを付加した安価なシステムにより視線の判別を実現するため、目頭と瞳孔の位置関係を用いた画像処理方法を提案した。そして、その方法を Raspberry Pi のシステムに組み込み、あらかじめ収集した画像に対して認識が正しく行われるかを確認した。さらに、Raspberry Pi に専用カメラモジュールを付加したシステムを用いて実際の人間の視線判別実験を行ったところ、良好な判別結果を得られることを確認した。また、本研究の最終段階で発売された Raspberry Pi の第二世代である Raspberry Pi2 では、第一世代の Raspberry Pi と比べて 2 倍以上の速度が出ていることが確認できた。この環境であればより早く、正確な認識が行えるようになるだろう。

本研究で実験したことは、視線における ON と OFF の 2 パターンの判断のためのものになるが、今後は視線入力システムのプラットフォームとして使用できるように判定の分解能を上げていく予定である。そして、視線入力システムの裏側で動作する視線認識のライブラリへ発展できるように更なる研究を続けていきたいと考えている。

6. 参考文献

- 1) Tobii Pro Japan, 視線制御 (入力) によるコミュニケーション, <http://www.tobii.com/ja-JP/eye-tracking-research/japan/about-tobii/history-at/eye-controlled-communication/>
- 2) 久野悦章, 八木透, 藤井一幸, 古賀一男, 内山嘉樹, EOG を用いた視線入力インタフェースの開発. 情報処理学会論文誌 1998 ; 39(5) : 1455-1462.
- 3) 伊藤和幸, 数藤康雄, 伊福部達, 重度肢体不自由者向けの視線入力式コミュニケーション装置. 電子情報通信学会論文誌 2000 ; J83-D-I(5) : 495-503.
- 4) 阿部清彦, 佐藤寛修, 大山実, 大井尚一, 視線による重度肢体不自由者向けコンピュータ操作支援システム. 映像情報メディア学会 2006 ; 60(12) : 1971-1979.