K-045

眼球運動により操作する重度肢体不自由者向けマウスポインタ
A Mouse Pointer Operated by Eye Movement for People with Severe Physical Disabilities

竹原　一行†
Kazuyuki Takehara

1．まえがき

現在のパソコンにおいて、マウスを用いたグラフィカルユーザインターフェース（GUI）は、直感的な操作が可能なため、主流のインターフェースになっている。そのため、通常の方法ではマウスを操作できない重度肢体不自由者がマウスポインタを操作する方法として、従来から、画面分割法や直交移動法などのスキャン法を基本とした方法が提案されている[1].

画面分割法では画面全体を分割し、分割した画面を順に自動スキャンする．ポインタを移動したい画面に来たとき、スイッチなどでその画面を選択する．その後、さらにその画面を分割してスキャンを始める．これを繰り返しポインタを操作する.

もう一つの直交移動法では、ポインタ自身が上下左右に自動的に移動するが、移動する前に、４方向を示すボタンを順にスキャンして、移動方向を選択する.

こうした方法を用いれば、重度肢体不自由者でもマウスポインタを操作することができる．しかし、これらの方法は GUI のような直感的なものではなく、コマンド入力に近い操作法のため、使いづらいという問題がある.

ポインタを操作する方法には、他に視線入力を用いる方法もある[2]．この方法は操作が直感的でわかりやすく、入力効率にも優れているという長所があるが、位置精度があまり良くないため、小さなターゲットにポインタを移動させることが困難である．また、視線入力装置は構成が複雑なため、コスト的な問題もある.

以上述べたように、従来の重度の肢体不自由者用ポインタ操作方法には問題があったため、ここでは眼球運動を利用した直感的な操作が可能で使いやすく、位置精度も良好なポインタの操作法について述べる.

2．ポインタの操作方法

2.1　概要

ここで述べるポインタ操作法は、位置精度に優れる直交移動法を基本にしている．また、直交移動法の欠点である使いにくいという点を改善するため、移動方向の選択方法は、直感的でわかりやすい眼球運動により行う方式を採用している．具体的には、まずビデオカメラで目の周辺の画像を取得し、画像処理で眼球運動を抽出、眼球運動の方向でポインタの移動方向を選択し、ポインタの移動を開始させている．移動方向は、上下左右の４方向から選択する．これを繰り返すことのより、希望する位置にポインタを移動させている.

2.2　基本構成

システムの外観を図 1 に示す．本システムは目の周辺の画像を 取得するための WEB カメラと称される安価なビデオカメラと画像処理を行うノートパソコンのみで構成されており、コスト的に有利である．表 1 に今回使用したビデオカメラの仕様を示す.

画像処理は米国 Intel 社で開発されたコンピュータビジョン向けライブラリである、OpenCV V.1.1 を用いて行っている ．また、ソフトを記述する言語は、Microsoft 社製の Visual C++ 2008 EE を用いている.

図１　システムの外観

表１　ビデオカメラの仕様

ビデオサイズ	320×240 ～　1280×720 画素
フレームレート	最大 30 fps
フォーカス	40cm ～ ∞　　（固定）
インターフェース	USB1.1 または 2.0

2.3　画像の取得方法

最初に、ノートパソコンのモニター上部に設置したカメラで、このシステムの使用者を撮影し、USB インターフェース経由でカラー画像 $Ap(a, b)$ を取得する．次にこの画像の中の顔の位置と大きさを計算し、その結果をもとに顔周辺をデジタル的に拡大した画像 $AZp(c, d)$を取得する．画像 $AZp(c, d)$から再度顔の位置と大きさを計算し、左右の目の周辺の画像 $AZBp(p, q)$を切り出す．その後この画像をグレー画像化し、画像 $Bp(p, q)$を得る．$Bp(p, q)$ のサイズは状況により多少異なるが 140×48 画素程度である．次に画像 $Bp(p, q)$を左右に二等分し、左側の目の周辺の画像 $BLp(i, j)$ と右側の目の周辺の画像 $BRp(i, j)$ を得る.

2.4　眼球運動の検出とマウスポインタの制御

取得した目の周辺の画像をもとに眼球運動を検出し、マウスポインタの移動方向を制御する方法として、従来から次のような方法がある[3]．まず、目の周辺の画像を処理して虹彩とその中央にある瞳孔（以下瞳と呼ぶ）の部分を抽出する．その瞳の位置が、目の内側に近い位置にあるか、あるいは外側や上側にあるかを判定して、マウスポインタの移動方向を選択する.

†マルヤス工業（株）　Maruyasu Industries Co., Ltd

従来の方法では、顔の位置がなんらかの原因で多少ずれた場合でも誤判定を生じないようにするには、瞳と瞳以外の部位との間の相対的な位置の変化を求めるなどの処理が必要になる[4][5]．ここではこうした顔のずれの対策として、次のような方法を用いている．例えばポインタの移動方向として右を選択したい場合、以下のように操作する．なお、ここでいう「右」とはカメラから使用者に向かって「右」であることを指す．

（1） 視線を適当な方向、例えば正面に向け、まばたきを行う．このまばたきを画像処理により検出し、検出した時点より 7 フレーム前の画像 $Bp1(p, q)$から図 2 に示すような画像 $BRp1(i, j)$ と画像 $BLp1(i, j)$を取得する．7 フレーム前の画像を用いるのは、まばたきの影響をなくすためである．

（2） 右に視線を向けてまばたきを行い、（1）と同様にして図 3 に示す画像 $BRp2(i, j)$ と画像 $BLp2(i, j)$を取得する．

画像 $BRp1(i, j)$ と画像 $BRp2(i, j)$ を比較すると、瞳の位置が右に移動していることがわかる．この瞳の右への移動を画像処理により抽出し、ポインタの移動方向として右を選択、移動を開始させる．以下、選択する方向を適宜変更しながら（1）と（2）を交互に繰り返し、ポインタを希望する位置まで移動させる．この方法では、基準となる画像 $BRp1(p, q)$ と $BLp1(i, j)$を判定の直前に取得するので、顔がゆっくりずれた時の影響を受けにくくなる．

2.5 眼球運動の検出のための画像処理

2.51 眼球運動の検出方法

眼球運動は、計算量の多いパターンマッチング処理に代えて、以下述べるような方法で抽出している．ここでは、右目の周辺の画像 $BRp1(i, j)$ と $BRp2(i, j)$ を例にとって説明する．まず、次式により差の画像 $SRs(i, j)$を計算する．

$$SRs(i, j) = BRp2(i, j) - BRp1(i, j) \tag{1}$$

次に、差の画像 $SRs(i, j)$を、正の領域 $SRsP(i, j)$ と負の領域 $SRsM(i, j)$ に分ける．

$$SRsP(i, j) = \begin{cases} SRs(i, j) & : SRs(i, j) > 0 \\ 0 & : SRs(i, j) \le 0 \end{cases} \tag{2}$$

$$SRsM(i, j) = \begin{cases} 0 & : SRs(i, j) > 0 \\ -SRs(i, j) & : SRs(i, j) \le 0 \end{cases} \tag{3}$$

$SRsP(i, j)$ の重心の座標を(rpx, rpy) とすると、

$$rpx = \frac{\sum_{i,j}[\, i \times SRsP(i, j)]}{\sum_{i,j} SRsP(i, j)} \tag{4}$$

$$rpy = \frac{\sum_{i,j}[\, j \times SRsP(i, j)]}{\sum_{i,j} SRsP(i, j)} \tag{5}$$

同様にして、$SRsM(i, j)$ の重心の座標 (rmx, rmy) を求め、最後に、右の目の眼球運動を表す、ベクトル $VR=(rmx - rpx, rmy - rpy)$ を求める．左の目の眼球運動を表すベクトル VL も、同様にして求める．また、処理を簡略化しているため、以上の処理は 15fps で行うことができた．

2.52 水平方向の眼球運動の検出

次に、この方法により、水平方向に視線を動かしたときの眼球運動を検出できることを、右方向の場合を例に説明する．図 2、3 は、先に述べたようにして取得した、視線を動かす前の基準画像と右方向に視線を向け、瞳が右に移動した時の画像である．図 4 には、図 2、3 の画像から計算した差の画像の正領域を、図 5 には負領域を示す．図 4、5 には、重心の計算結果を十字マークで示した．また、図 6 にはベクトル VR を図 2 の画像に重ねて示した．

図 4 の正領域の図には、瞳が右に移動したことにより、瞳から白目に変わった部分が、白い領域として表示され、また、図 5 の負領域の図には、白目から瞳に変わった部分が、白い領域として表示されている．そこで、正の白領域の重心を始点とし、負の白領域の重心を終点としたベクトル VR の方向に瞳が移動したと考え、ベクトル VR を右目の眼球運動の運動方向を表すベクトルとした．

図 2　右方向に視線を向ける前の画像 $BRp1(i, j)$

図 3　右方向に視線を向けたときの画像 $BRp2(i, j)$

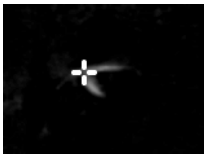

図 4　差の画像の正領域 $SRsP(i, j)$

図 5　差の画像の負領域 $SRsM(i, j)$

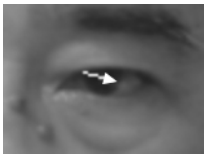

図 6　右方向に視線を向けたときのベクトル VR

従来の方法では、瞳であると認識した部位の重心の移動状態を検出して、移動方向を決めていたが、ここで述べている方法では、瞳の移動により変化した部位の正負の重心位置で移動方向を表すベクトルを決めている．そのため、瞳の移動量が小さくても、瞳の移動量が大きい場合に近い大きさのベクトルが得られる．したがって、瞳の移動量が小さくてもまぶたの移動などに起因する外乱の影響を受けにくく、瞳の移動方向を精度良く判定できる．その代わり、ベクトルの大きさは瞳の移動量の絶対値を正確には反映していないが、ここでは移動方向だけを検出できればよいので、この方法を用いている．

2.53 垂直方向の眼球運動の検出

次に、垂直方向へ眼球が移動したときの移動方向も検出できることを説明する．垂直方向への移動は水平方向への移動のときとは状況が多少異なるが、同じ方法で検出している．

ポインタを上方向に移動させようとして、視線を上に向けた場合を例にして説明する．図 7 は、基準となる画像を表し、図 8 は視線を上に向けたときの画像である．

図 7、8 に示す画像から計算した結果を図 9、10 に示す．図 9 は正領域 $SRsP(i, j)$ を、図 10 は負領域 $SRsM(i, j)$ を表す．図 9、10 を見ると、瞳が上に移動したことにより変化したと思われる部分もあるが、それ以外の要因で変化したと考えられる部分の方が多い．図 9 では視線が上を向いたことにより、下まぶたの縁の明るい部分が上がった部位と上まぶたが上がったことにより眉毛の位置が移動し、明るい部分に変化した部位が白く表示されている．また、図 10 では、上まぶたが上がったことにより、上まぶたのふちが上に上がった部分と、眉毛が上に移動した部分が白く表示されている．

正領域に示される部分は、下まぶたの縁と眉毛の下の部位であり、負領域に示されるのは、上まぶたの縁と眉毛の上の部位であるので、正の白領域の重心を始点とし、負の白領域の重心を終点としたベクトル VR は図 11 に示すように上向きのベクトルとなる．

図 7　上方向に視線を向ける前の画像 $BRp1(i, j)$

図 8　上方向に視線を向けた時の画像 $BRp2(i, j)$

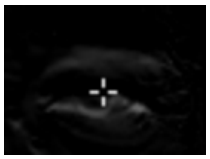

図 9　差の画像の正領域 $SRsP(i, j)$

図 10　差の画像の負領域 $SRsM(i, j)$

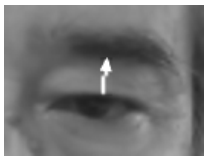

図 11　上方向に視線を向けたときのベクトル VR

2.6 マウスポインタの制御

2.61 マウスポインタの移動方向制御

次に、以上述べたようにして決定した、左右の目の眼球運動を表すベクトル VR と VL を用いて、マウスポインタの移動方向を制御する方法について説明する．

まず、各ベクトルの x 方向成分と y 方向成分を $VR =(rx, ry)$、$VL =(lx, ly)$ としたとき、これらの成分の内、絶対値が最大となる成分を求める．その結果、例えば、ベクトル VR の x 方向成分である rx の絶対値が最大で、$rx > 0$ なら、右方向を、$rx < 0$ なら左方向を移動方向として移動を開始する．もし、ry の絶対値が最大で、$ry > 0$ なら、上方向に、$ry < 0$ なら下方向への移動を開始する．

2.62 マウスポインタの移動距離の制御

ポインタの移動距離の制御は、次の２つの方法で、ポインタの移動を停止あるいは終了させることにより行っている．第 1 の方法では、ポインタが移動している方向とは異なる移動方向を選択することにより、移動を停止させる．例えば、右方向にポインタが移動しているときに、2.5 で述べた方法を用いて新たに上方向を選択すると、右方向への移動を停止し、上方向への移動を開始する．こうした移動方向の変更操作を繰り返すことにより、ポインタを希望する位置まで移動させることができる．

第２の方法では、第 1 の方法によりポインタを希望する位置まで移動させることができたとき、方向制御のときより閉眼時間の長いまばたき[6]を行って、ポインタの移動を終了させる．また、このとき同時にマウスクリックを実行している．

閉眼時間の長いまばたきは次のようにして検出している．ポインタの制御時に取得した画像のうち、7 フレーム前の両目の周辺の画像 $Bp1(p, q)$を基準として、最新の画像 $Bp0(p, q)$との間の差を計算し、その差の絶対値の合計 M を画像間の差の大きさと考え、その値を求める．

$$Ss(p,q) = Bp0(p,q) - Bp1(p,q) \tag{6}$$

$$M = \sum_{p,q} |Ss(p,q)| \tag{7}$$

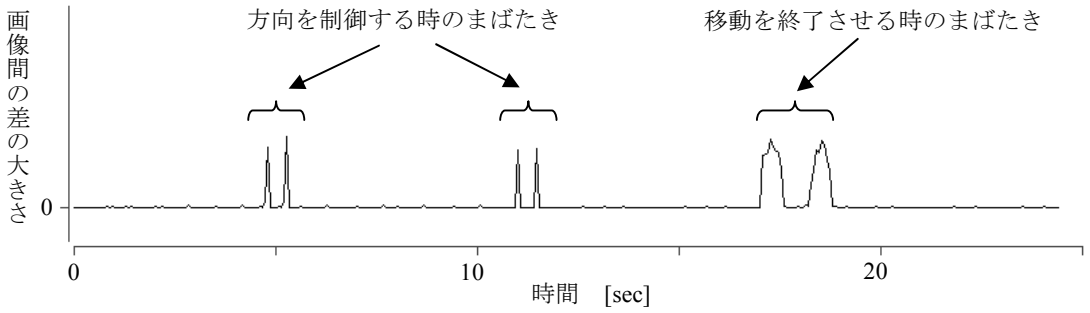

図 12　まばたきの判別

画像間の差の大きさの実測値を時系列データとして表したものを図 12 に示す．方向制御は、特に意識せずに自然な感じでまばたきを行っている．こうしたまばたきは 0.1 秒程度でまぶたの開閉を行っているため、画像を 15fps で取り込むと、まばたきによる画像間の差の変化は 1 ないし 2 フレームで終了する．そのため、図 12 にある、幅の狭いパルス状の信号として観測される．また、7 フレーム前の画像を基準として差を計算しているので、幅の狭いパルス状の信号は 1 回のまばたきに対して、２回現れる．

それに対して、通常よりまばたきによる閉眼時間を長くし、たとえば 1 秒程度とすると、閉眼時間は 7 フレームより長くなるため、幅が 7 フレームのパルスが 2 回現れることになる．この幅が 7 フレームのパルスを検出して、閉眼時間の長いまばたきがあったと判定し、ポインタの移動を終了させている．

3. 実験結果

実際にシステムを組んで、眼球運動を用いたマウスポインタの制御実験を行った．被験者は進行性かつ全身性の筋力低下がある重度の肢体不自由者である．その結果、まばたき検出はきちんとできていたので、ポインタの移動を開始させることはできた．しかし、思い通りの方向への移動させることが難しく、意図しない方向へ移動することの方が多かった．画像を記録してその原因を調べてみると、次のようなことが判明した．被験者は筋力低下のため、まぶたが下がり、いわゆる眼瞼下垂症状を呈していた．そのため、誤判定をしないように、眼球の移動をなるべく大きくしようと無理をすると、力が入ってまぶたが動き、このまぶたの動きにより、誤判定が生じていた．以上のようなことから、無理に眼球を大きく移動させずに、軽く視線を振る程度にして、再度試行してみた．2.52 で述べたように、本方式では、視線の移動量が少なくとも、十分に移動方向を検出できると考えたからである．その結果、希望する方向にポインタを移動させることができるようになった．また、軽く視線を振るだけで操作できるので、使用者の負担感も少なかった．

次に、ターゲットとして設定した、直径 20 画素の円までポインタを移動させる実験を行った．この円の直径は、インターネットのホームページの入力ボックスの高さと同程度に設定した． 実際にポインタを移動させてみたところ、多少の慣れは必要であるが、ターゲットの円までポインタを移動させることができた．

4. むすび

ここでは直交移動法を用いているため位置精度が良好であり、眼球運動で移動方向の選択を行っているため直感的な操作ができる、重度肢体不自由者向けのポインタ操作法について述べた．この操作法では、ノートパソコンのモニター上部に設置したカメラで目の周辺の画像を取得、画像処理して眼球運動の方向を判定し、その結果をもとにポインタの移動方向を選択している．このとき、判定の基準となる画像を、眼球運動の方向の判定の直前に取得しているので、顔のずれの影響を小さくできた．また、視線の移動量が少なくとも移動方向を効率よく検出できる処理方法を用いているため、使用者の負担感を軽減できた．実際にシステムを構築して実験を行い、ポインタを精度良くかつ直感的に操作できることを確認した．

参考文献

[1] 小山堅治, 石川真悟, 林豊彦, 中村康雄, 若林佑子, 遁所直樹, "マウスポインタ操作を代替するシングルスイッチスキャン法の操作性評価", 信学技報 WIT2007-37, (2007).

[2] 伊藤和幸, 伊福部達,"ビデオキャプチャ画像処理による視線検出及び意思伝達装置への応用", 信学論 (D-I), vol.J88-D-I, no.2, pp527-535, (2005).

[3] 比嘉広樹, 国吉真史, 堂上高司, 西原賢, "眼球運動を用いたヒューマンインターフェースの検討", 信学技報 MBE2006-67, pp5-8, Nov. 2006

[4] 伊藤和幸, "ビデオキャプチャによる眼球運動計測および環境制御への応用", ヒューマンインターフェース学会論文誌, vol.5, no.4, pp33-40, (2003).

[5] 長崎健, 秋田純一, 戸田真志, 川嶋稔夫, "頭部の移動を考慮した視線検出", 信学技報 WIT2006-41, (2006).

[6] 矢島大輔, 林豊彦, 渡辺哲也, 前田義信, 若林佑子, 渡辺諭, 阿部晃, 山口俊光, "反射型フォトセンサ列を用いた汎用シングルスイッチ VSN/1 による意識的なまばたきの検出法", 信学技報 WIT2009-50, pp11-16, (2009).