Vol.2019-HCI-185 No.23
Vol.2019-UBI-64 No.23
2019/12/11

Valve Icon: オーバーシュート後に生成される壁を用いたポインティング高速化手法の提案

木崎 駿也[1]　薄羽 大樹[1]　山田 開斗[1]　宮下 芳明[1]

概要：Graphical User Interface では，ボタンやウィンドウなどのターゲットにカーソルをあわせクリックすることで，そのターゲットを選択できる．この操作を行う際，カーソルを高速で動かすとターゲット上を通り過ぎてしまう「オーバーシュート」が発生する．そのため，ユーザはターゲットの大きさに対して適切な速度で操作する必要があり，また，オーバーシュートしてしまった場合には，再びターゲットにカーソルをあわせなければならない．本研究では，オーバーシュート後にターゲットの輪郭に壁を生成し，2 回目のオーバーシュートを防ぐことで操作時間を短縮する「Valve Icon」を提案する．Valve Icon では，ターゲット上にカーソルがあるとき，生成された壁にカーソルが衝突し停止するため，2 回目のオーバーシュートを防ぐことが可能である．そのため，1 回目のオーバーシュートの発生を恐れることなく，カーソルを高速に操作できる．従来手法との比較実験の結果，アイコン間の間隔が広い場合，提案手法はエラー率を増加させることなく操作時間を短縮できることがわかった．また，ターゲット幅が小さく間隔が狭い場合には，提案手法の操作時間は増加した．これらの実験結果をもとに，アプリケーション例を提案した．

1. はじめに

Graphical User Interface（GUI）では，カーソルを操作し，ターゲット（ユーザが選択したいアイコン）上でクリックを行うことで，ターゲットの選択が可能である．カーソルの操作時間は，カーソルを高速に操作することで減少できるが，速く操作をするとオーバーシュート（カーソルがターゲット上を通り過ぎること）が発生する．オーバーシュートが発生すると，再びターゲットにカーソルを合わせる必要があるため，オーバーシュートした分だけ操作時間が増加してしまう．そのため，オーバーシュートが発生しないよう速度を抑えたポインティングが要求される．

本稿では，オーバーシュート後にターゲットの輪郭に壁を生成し，壁にカーソルを衝突させてポインティングを行う「Valve Icon」を提案する（図 1a）．Valve Icon では，ターゲット上にカーソルがあるとき，カーソルが壁に衝突しターゲット上に停止するため，2 回目のオーバーシュートを防ぐことができる．そのため，高速でカーソルを操作し，ターゲットをオーバーシュートした場合でも（a1, a2），ユーザはカーソルの速度を落とすことなく，壁にカーソルを衝突させて，ポインティングが行える（a3）．これにより，カーソルの操作時間が減少されることが見込まれる．本稿では，従来手法と Valve Icon の比較実験を行い，

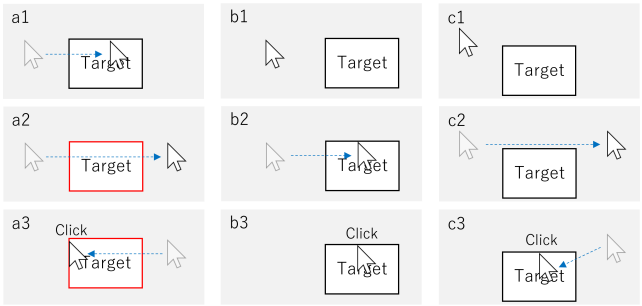

図 1　（a）カーソルがターゲットをオーバーシュートしたとき，ターゲットの輪郭に壁が生成される．（b）直接ターゲットをポインティングすることも可能である．（c）カーソルが一度もターゲット上に乗っていないため，壁は生成されない．

Valve Icon が有効に働く状況を調査する．また，実験結果を踏まえたアプリケーション例についても紹介する．

2. 関連研究

2.1 ターゲット幅を拡大する手法

Sticky Icons [1] では，カーソルがターゲット上にある場合，ゲイン（マウスの移動量に対するカーソルの移動量）を減少させる．つまり，カーソルの速度が遅くなるため，ユーザはアイコンの幅が大きくなったような操作感が得られる．Semantic Pointing [2] も同様の手法であり，カーソルとアイコン間の距離に応じてゲインを動的に変更する．Valve Icon では，カーソルをターゲットの壁に衝突させて

[1] 明治大学

© 2019 Information Processing Society of Japan

ポインティングを行える．壁に衝突することでカーソルは動かなくなり，壁より先にはカーソルを移動させることができないため，幅が無限大のターゲットをポインティングしているように感じられる．

Birdlime Icon [3] では，オーバーシュートが発生する前に，カーソルの移動方向へターゲット幅を拡大する．アイコンが隣接している場合，手前のアイコンの拡大された部分が奥のターゲットに重なってしまう可能性がある．一方で，同様の状況における Valve Icon では，手前のアイコンをオーバーシュートするとそのアイコンに壁が生成されるが，その壁がターゲットのポインティングを邪魔することはない（図 3）．

通常のカーソルでは，クリックイベントが発生する領域は 1 pixel のみであり，クリック範囲が拡大されたカーソルはエリアカーソルと呼ばれる [4]．エリアカーソルでは，カーソル自体がターゲットをオーバーシュートした場合にも，クリック範囲にターゲットが含まれていれば，ターゲットの選択が可能である．つまり，エリアカーソルは，ターゲットをクリックできる領域を拡大しているため，通常のカーソルを用いたポインティングでのターゲット幅の拡大と同様の効果が得られる [4]．Bubble Cursor [5] は，カーソルとアイコン間の距離に応じて，DynaSpot [6] は，カーソルの速度に応じて，クリック範囲が動的に変わる手法である．

2.2 ターゲットまでの距離を短縮する手法

カーソルとターゲット間の距離を短縮することで，高速にカーソルを移動することなく，操作時間を減少できる．Delphian Desktop [7] では，カーソルの速度からターゲットを推測し，カーソルを推測先のターゲットに移動させる．航空路マウス [8] では，マウスにジョイスティックが取りつけられており，ジョイスティックを倒した方向にあるターゲットにカーソルを移動させる．これらの手法では，カーソルはターゲット上に移動し停止するためオーバーシュートは起こらないが，アイコンが複数存在する場合には，誤ってターゲット以外のアイコンに移動してしまう可能性がある．

3. 提案手法: Valve Icon

Valve Icon を用いたポインティングでは，カーソルがターゲット上を通り，オーバーシュートしたときに，ターゲットの輪郭に壁が生成される．カーソルを再びターゲット方向に移動させると，ターゲットの壁に衝突し，停止する．もし，ユーザがオーバーシュートをしない場合，そのままクリックをしてターゲットを選択できる（図 1b）．カーソルがターゲット上を通らない場合，壁は生成されない（図 1c）．また，ターゲット以外のアイコンに壁を生成した場合に，壁がターゲットのポインティングの邪魔になることがある．その場合，壁の解除を行う必要がある．生成された壁は，カーソルの衝突後，カーソルを停止させてから（図 2a）もしくはカーソルを壁から離してから（図 2b）200 ms 後に解除される．カーソルを壁の方向に向かって移動させている間は，壁は解除されない（図 2c）．提案手法では，1 回目のオーバーシュート後に壁を生成し，2 回目のオーバーシュートを防ぐ．オーバーシュート前に壁を生成する場合，ターゲット以外のアイコンの壁にカーソルが衝突し，操作を妨げてしまう可能性があるためである．

ターゲットの選択後には，全ての壁が解除される．ターゲット選択後の 2 回目のポインティングでは，すべての壁が解除されているため，前回の壁に衝突することなくポインティングが行える（図 3）．

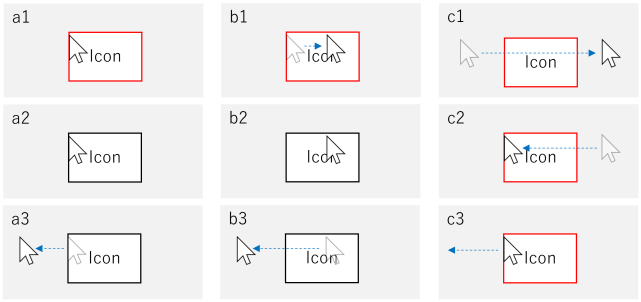

図 2 （a）カーソルを停止させてから 200 ms 後に壁が解除される．（b）カーソルを離してから 200 ms 後に壁が解除される．（c）カーソルを壁に向かって動かし続ける場合，壁は解除されない．

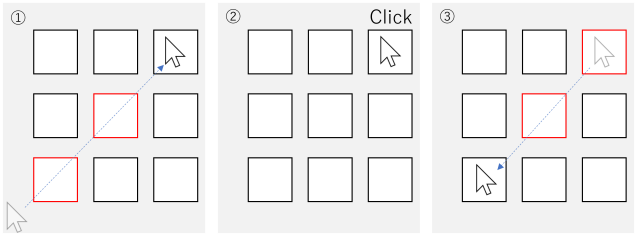

図 3 アイコンが複数ある場合における Valve Icon でのポインティング．

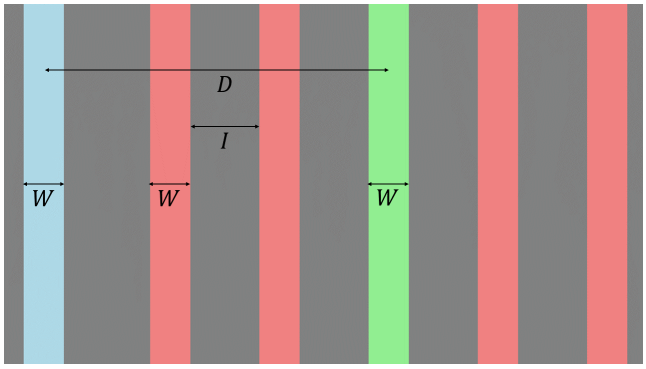

図 4 実験タスクの概要．青色：スタートエリア，赤色：非ターゲット，緑色：ターゲット．

© 2019 Information Processing Society of Japan

4. 評価実験

4.1 実験機材

PC は Alienware 17 R4（Intel Core i7-7700HQ, 2.8 GHz, 4 cores，8GB RAM，Windows 10）を使用した．解像度は 1920 × 1080 pixels（17.3 inches，381.89 × 214.81 mm）であった．入力デバイスは長さ 1.80 m のケーブルに繋がれる有線マウス（Logicool G-PPD-002WL）を使用した．カーソルの設定は Windows 10 のデフォルトであった．実験システムは Processing で実装されており，フルスクリーンで表示された．

4.2 実験参加者

12 名（9 名は男性，3 名は女性，平均年齢は 22.42 歳，標準偏差は 1.86 歳）が参加した．全ての参加者は右利きであり，右手でマウスを操作した．

4.3 タスク

従来手法と Valve Icon を用いて，1 次元のポインティングタスクを参加者に行わせた．PC には，青色のスタートエリア，赤色の非ターゲット，緑色のターゲットが表示された（図 4）．参加者が画面上の青色のスタートエリアをクリックすると試行の開始を示す効果音が鳴り，試行と計測が開始された．試行開始後，参加者はできる限り早く正確に緑色のターゲットを目指した．参加者がターゲット上でクリックを行った場合，その試行を成功とし，成功を示す音が鳴った．それ以外でクリックを行った場合，その試行を失敗（エラー）とし，失敗を示す音が鳴った．また，試行中のクラッチ（マウスを置き直す動作）を禁止した[*1]．参加者には，各試行の前にターゲットをどのようにポインティングするかを考えてからタスクを行うよう要求した．

4.4 実験デザインと手順

アイコンの幅（W）は，18，34，60，102 pixels（それぞれ，3.58，6.76，11.93，20.29 mm），距離（D）は，514，686 pixels（それぞれ，102.24，136.45 mm），アイコン間の間隔（I）は 0，18，34，102，∞ pixels（それぞれ，0，3.58，6.76，20.29，∞ mm）であった．$I = \infty$ の場合，非ターゲットは表示されなかった．比較手法（$Method$）は，従来手法（$Normal$），Valve Icon（$Valve$）の 2 種類であった．

非ターゲットが存在しない場合，Valve Icon では，カーソルを右端に移動させてからターゲット方向に移動することでポインティングが行える．また，非ターゲットがターゲットの両側に 1 個ずつ存在する場合も，カーソルを右端に移動させて，壁を解除するだけでポインティングが行える．そのため，ターゲットを手前と奥で囲むよう 2 個ずつ合計 4 個の非ターゲットを配置した．これにより，カーソルを右端に移動させてからターゲット方向に移動させる際，2 個の非ターゲットの壁を解除する必要があり，そのような操作をした場合には操作時間が増加すると考えられる．そのため，非ターゲットをオーバーシュートしないように気を付ける，または，オーバーシュートが発生しないように直接ターゲットを狙う必要が生まれると考えられる．

Valve Icon が有効に働く状況は，アイコン間の間隔も関係すると考えられる．アイコン間の間隔が狭いときには，ターゲットのみをオーバーシュートすることは難しく，非ターゲットもオーバーシュートしてしまうだろう．また，カーソルを壁から離すことで非ターゲットの壁を解除できるが，その際にもう 1 つの非ターゲットをオーバーシュートし壁が生成されてしまう可能性がある．一方で，間隔が広い場合には，ターゲットだけをオーバーシュートしてからターゲットの壁に衝突させることは容易であると考えられる．そのため，アイコン間の間隔を実験条件に加えた．

1 セットは，$4W \times 2D \times 5I = 40$ 試行であり，セット内の出現順序はランダムであった．$Method$ の順序はラテン方格法によって決められた．実験の開始前，参加者にはそれぞれのポインティング手法の説明を行い，それぞれの手法で 1 セットの練習と 15 セットの本番を行わせた．3 セットごとに 30 秒から 60 秒の休憩を行った．全試行は 14,440 回（$4W \times 2D \times 5I \times 2Method \times 15$ セット $\times 12$ 名）であり，1 名あたり 60 分を要した．

4.5 計測値

スタート領域をクリックしてからターゲットをクリックするまでの操作時間 MT，エラー率を記録した．

4.6 結果

14,227 試行のうち（173 試行を外れ値とした[*2]），エラーは 641 試行であった（4.51%）．以下では，繰り返しのある分散分析で分析を行い，多重比較には Bonferroni の法を用いた．従属変数は，$Method$，W，D，I であった．グラフ中のエラーバーは標準誤差を示し，***，**，* はそれぞれ，$p < 0.001$，$p < 0.01$，$p < 0.05$ を示す．

4.6.1 学習効果

ユーザは，従来手法に比べ Valve Icon の操作に慣れていない．そのため，従属変数を $Method$，セット番号（$SetNum$，1–15）として，学習効果を分析した．

$SetNum$ の主効果がみられたのは，MT のみであり（$F_{14,154} = 7.20, p < 0.001, \eta_p^2 = 0.40$），エラー率にはみら

[*1] クラッチは，フィッツの法則のモデル適合度を減少させることが知られている [9]．クラッチを許可した場合に，Valve Icon の適合度がもし低ければ，それがクラッチによる影響なのか Valve Icon による影響なのかが不明瞭になる．そのため，本稿ではクラッチを禁止した．

[*2] 平均 MT から $3SD$ 離れた試行を外れ値として扱った [10]．

© 2019 Information Processing Society of Japan

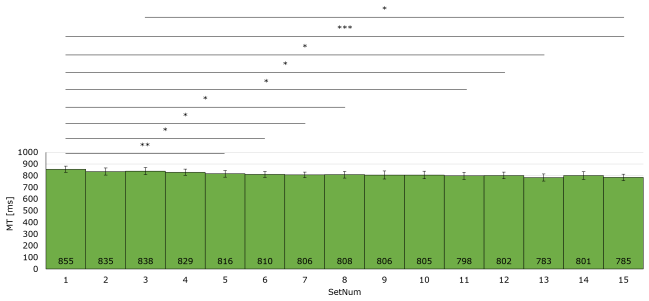

図 5 *SetNum* が *MT* に与える影響.

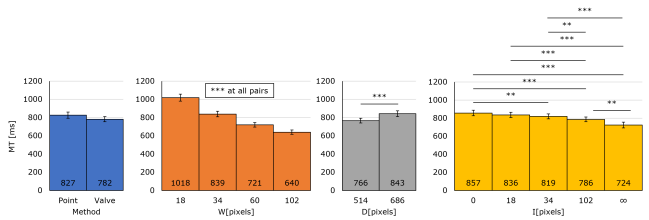

図 6 *MT* への影響.

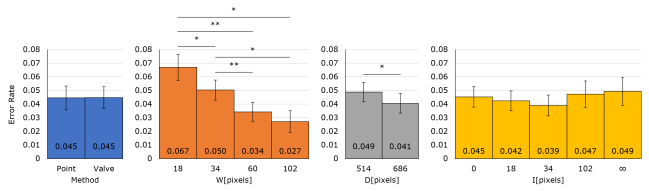

図 7 エラー率への影響.

れなかった（$F_{14,154} = 1.58, p = 0.089, \eta_p^2 = 0.126$）．また，*Method* の主効果は *MT* とエラー率のどちらにおいてもみられなかった（*MT* では，$F_{1,11} = 2.91, p = 0.12, \eta_p^2 = 0.21$; エラー率では，$F_{1,11} = 0.095, p = 0.763, \eta_p^2 = 0.0086$）．さらに，*Method* × *SetNum* の交互作用もどちらの値においてもみられなかったため（*MT* では，$F_{14,154} = 0.45, p = 0.957, \eta_p^2 = 0.039$; エラー率では，$F_{14,154} = 0.85, p = 0.61, \eta_p^2 = 0.072$），*SetNum* が *MT* に与える影響によって学習効果を分析する．

図 5 に示すように，$SetNum = 1$ は，複数の *SetNum* と比べて *MT* が遅く，また，$SetNum = 3$ は，$SetNum = 15$ と比べて遅かった．そのため，*MT* が有意に遅いセットを除いた 4 セット目以降を分析の対象とした．4 セット目以降の試行は 11,270 回（外れ値は 125 試行，エラー数は 514 回）であった．

4.6.2 操作時間 *MT*

分散分析の結果は，表 1 の通りであり，*W*，*D*，*I* には主効果がみられた．*Method*，*W*，*D*，*I* の多重比較の結果は，図 6 の通りである．

4.6.3 エラー率

分散分析の結果は，表 2 の通りであり，*W*，*D* には主効果がみられた．*Method*，*W*，*D*，*I* の多重比較の結果は，図 7 の通りである．

表 1 操作時間 *MT* の分散分析の結果

Factors	*DF*	*DFDen*	*F*	*p*	η_p^2
Method	1	11	3.07	0.107	0.218
W	3	33	307	< 0.001	0.965
D	1	11	125	< 0.001	0.919
I	4	44	46.6	< 0.001	0.809
Method × *W*	3	33	0.339	0.797	0.030
Method × *D*	1	11	37.7	< 0.001	0.774
Method × *I*	4	44	30.0	< 0.001	0.732
W × *D*	3	33	9.59	< 0.001	0.466
W × *I*	12	132	28.7	< 0.001	0.723
D × *I*	4	44	2.65	< 0.05	0.194
Method × *W* × *D*	3	33	7.17	< 0.001	0.395
Method × *W* × *I*	12	132	26.0	< 0.001	0.703
Method × *D* × *I*	4	44	7.33	< 0.001	0.400
W × *D* × *I*	12	132	2.91	< 0.01	0.209
Method × *W* × *D* × *I*	12	132	2.57	< 0.01	0.190

表 2 エラー率の分散分析の結果

Factors	*DF*	*DFDen*	*F*	*p*	η_p^2
Method	1	11	0.00	0.979	0.00
W	3	33	15.7	< 0.001	0.589
D	1	11	7.37	< 0.05	0.401
I	4	44	0.558	0.694	0.05
Method × *W*	3	33	0.924	0.440	0.08
Method × *D*	1	11	0.174	0.685	0.016
Method × *I*	4	44	7.64	< 0.001	0.410
W × *D*	3	33	1.92	0.145	0.149
W × *I*	12	132	0.658	0.789	0.056
D × *I*	4	44	2.65	< 0.05	0.194
Method × *W* × *D*	3	33	0.081	0.970	0.007
Method × *W* × *I*	12	132	1.98	< 0.05	0.152
Method × *D* × *I*	4	44	0.435	0.782	0.038
W × *D* × *I*	12	132	0.913	0.536	0.077
Method × *W* × *D* × *I*	12	132	0.920	0.529	0.077

4.6.4 モデル適合度

フィッツの法則 [11,12] によれば，*D* だけ離れた幅 *W* のターゲットを選択するまでの時間 *MT* は，2 つの回帰定数（*a* と *b*）を用いて，次のように表わせる．

$$MT = a + b \log_2 \left(\frac{D}{W} + 1 \right) \quad (1)$$

図 6 と図 7 で示されるように，*W* と *D* には，フィッツの法則における *W* と *D* と同様の効果がみられた．そのため，フィッツの法則をポインティングの操作時間推定モデルとし，適合度を検証した．*Method* 別にモデル適合度をみると，*Normal* では $R^2 = 0.964$ であり，*Valve* では $R^2 = 0.608$ であった（図 8）．また，*I* 間に差がみられたため（図 9），*I* 別にもモデル適合度を検証した．その結果，*Normal* ではすべての *I* において $R^2 > 0.90$ であったが，*Valve* では $I = 0$ と $I = \infty$ を除く *I* において $R^2 > 0.90$ であった（図 10）．

5. 考察

5.1 提案手法が有効な状況

ターゲットのみが存在する場合において（$I = \infty$），Valve Icon の操作時間は短かった（図 11）．また，ターゲットのみ

© 2019 Information Processing Society of Japan

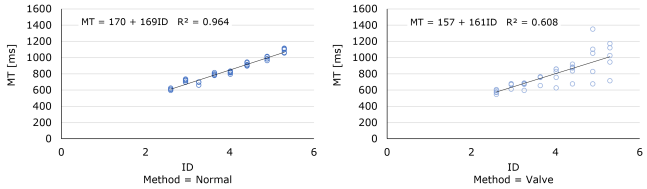

図 8　各 *Method* におけるモデル適合度.

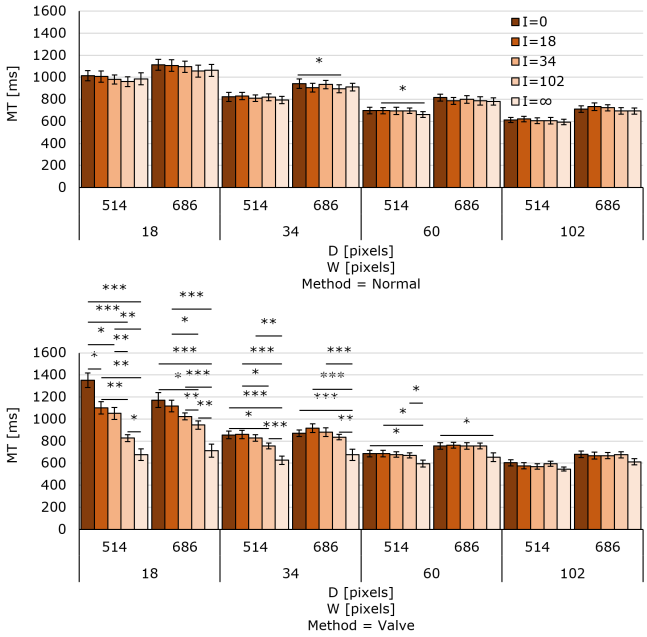

図 9　$Method \times D \times W \times I$ における MT への影響.

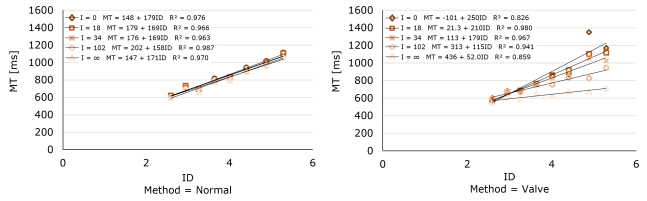

図 10　各 I における *Method* のモデル適合度.

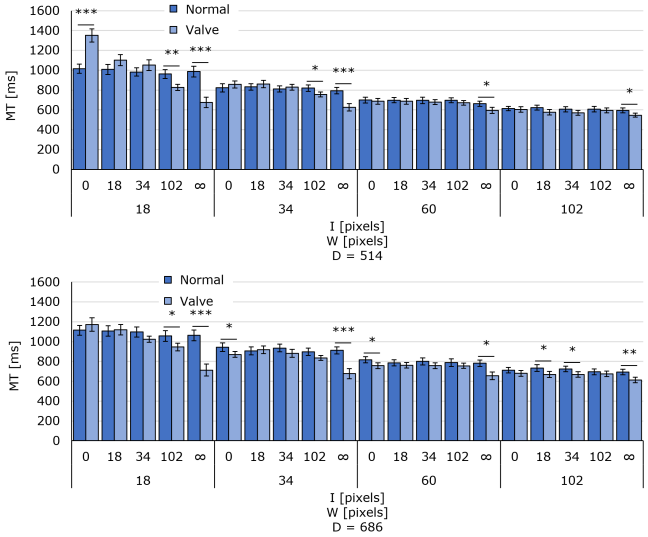

図 11　$Method \times D \times W \times I$ における MT への影響.

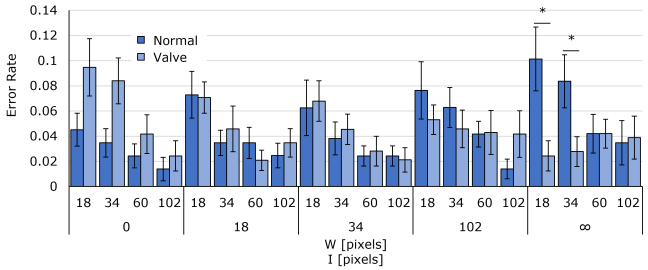

図 12　$Method \times W \times I$ におけるエラー率への影響.

が存在しターゲット幅が小さい場合に（$I = \infty$，$W \leq 34$），Valve Icon のエラー率は低かった（図 12）．つまり，Valve Icon では，ターゲットのみが存在する場合，非ターゲットに壁が生成されないため，高速に操作でき，操作時間が短縮される．また，壁で停止しターゲット上にカーソルがとどまれるためエラー率が低下していた．

ターゲット幅が小さく（$W = 18$），アイコン間の間隔が広い場合に（$I = 102$），Valve Icon は速かった（図 11）．これは，ターゲット幅が小さいためオーバーシュートが発生しやすく，従来手法の操作時間が増加したためであると考えられる．また，Valve Icon でのポインティングにおいて，間隔が広いときは，容易にカーソルを壁に衝突でき，操作時間が減少されたことも理由として考えられる．

$W \geq 60$ の場合，Valve Icon は従来手法と比べ操作時間が増加することはなかった．十分なターゲット幅がある場合，そもそもオーバーシュートが起こりにくいため，Valve Icon の操作時間は増加しないと考えられる．アイコン間の間隔が狭いときには，ターゲットをオーバーシュートしても，十分なターゲット幅があるため，非ターゲットをオーバーシュートする可能性は低い．そのため，Valve Icon では，ターゲットの壁にカーソルを衝突させることで操作時間を減少できたと考えられる．

5.2　従来手法が有効な状況

$W = 18$，$I \leq 18$ の場合，Valve Icon は従来手法と比べ操作時間が増加した．ターゲット幅が小さい場合にはオーバーシュートが発生しやすく，また，アイコン間の間隔が狭い場合，非ターゲットをオーバーシュートする可能性が高い．その場合には，Valve Icon では，非ターゲットに頻繁に壁が生成され，その壁の解除に時間がかかり，操作時間が増加した．また，Valve Icon で発生した壁を解除するには，壁に一度衝突させてからカーソルを停止させる，もしくは壁から少しカーソルを離す必要がある．壁からカーソルを離す際に，誤って非ターゲットをオーバーシュートし壁を生成してしまい，その壁の解除にも時間がかかってしまっていた．実験参加者によると，誤って生成された非ターゲットの壁の解除がストレスであったという意見が得られた．ターゲット幅が小さく，アイコン間の間隔が狭いときには，Valve Icon を使用すべきではないと考えられる．

© 2019 Information Processing Society of Japan

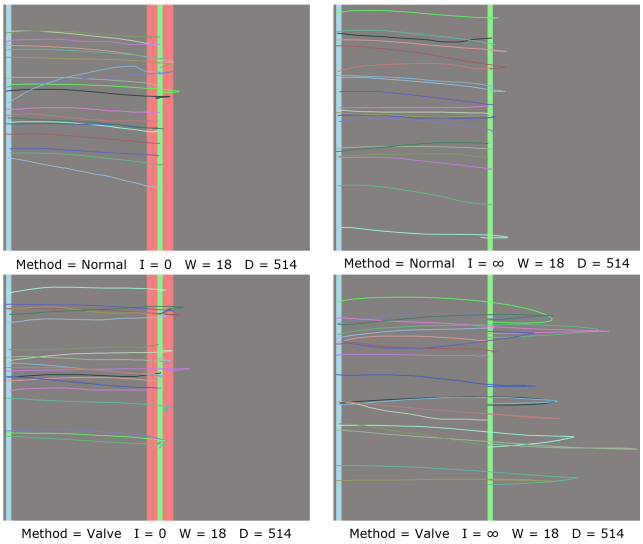

図 13 各条件におけるランダムに抽出された 20 試行分のカーソルの軌跡.

5.3 フィッツの法則の適合度

図 10 に示されるように，Valve Icon は，$I = 0$ と $I = \infty$ ではフィッツの法則に適合しなかった．図 13 は各条件におけるランダムに抽出された 20 試行のカーソルの軌跡を示しているが，Valve Icon では，$I = 0$ のときに非ターゲットをオーバーシュートしている軌跡がみられた．非ターゲットをオーバーシュートすると，非ターゲットに壁が発生し，その壁を解除する必要があるため，操作時間が従来手法に比べ増加する．そのため，$I = 0$ では，従来手法は適合したが，Valve Icon では適合しなかったのであろう．

$I = \infty$ のときの Valve Icon では，非ターゲットが存在しないため，高速でオーバーシュートすることでターゲットに壁を生成し，壁に向かって再び高速でカーソルを動かすことでポインティングを成功させられた（図 13）．そのため，従来手法に比べ，大幅に操作時間を減少でき（図 11），モデル適合度が減少したのだと考えられる．

6. アプリケーション例

実験結果の考察をもとに，Valve Icon が大きく操作時間を減少できるアプリケーション例を紹介する．

6.1 動画プレイヤーのシークバー

動画プレイヤーにはシークバーが存在する．シークバーの高さが大きいと動画のコンテンツ領域が減少されるため，既存の動画プレイヤーのシークバーは小さく設定されている．フィッツの法則によれば，ターゲット幅が小さいと操作時間が増加する．つまり，シークバーが小さければ，その分だけ操作時間が増加してしまうだろう．Valve Icon をシークバーに適用した場合，実験結果によれば，シークバーの大きさを維持したまま，操作時間を減少できるだろう（図 14）．また，シークバーのみに Valve Icon を適用し，

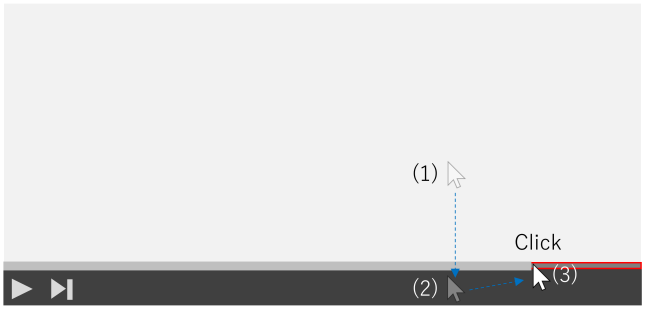

図 14 Valve Icon によるシークバーのポインティング．(1) カーソルがシークバーをオーバーシュートしたとき，シークバーの輪郭に壁が生成される．(2) シークバーの方向に向かってカーソルを高速で移動させる．(3) カーソルがシークバーの輪郭の壁に衝突し停止する．その後，ユーザはクリックによってシークバーを選択できる．

再生ボタンなどには適用しなければ，誤って非ターゲットに壁が生成されることを防げる．つまり，Valve Icon を使用することにより，コンテンツ領域を減らすことなく操作時間を減少できる．

6.2 拡張ディスプレイ時のメニューバー

macOS では，画面上部にメニューバーが存在する．通常では，画面上部に配置されているため，カーソルを上に勢いよく動かせば，スクリーンの端に衝突し，メニューバーに到達できる．そのため，メニューバーのポインティングにおいて，オーバーシュートは発生しない．しかし，拡張ディスプレイを上に配置している場合では，画面上部に衝突することなく，拡張ディスプレイにカーソルが移動してしまう．つまり，オーバーシュートが発生するため，ユーザはメニューバーの大きさにあわせて適切な速度でカーソルを操作する必要がある．Valve Icon をメニューバーに適用した場合，オーバーシュートが発生したとき，メニューバーに壁が生成される（図 15）．ユーザは故意に 2 回オーバーシュートしようとすればメニューバーに到達でき，その動作は通常のポインティングと比較すれば高速であるため，操作時間は減少できる．つまり，Valve Icon を用いれば，非拡張ディスプレイ時のメニューバーの利点を失わずに，少ない操作時間でポインティングを行えるだろう．

7. 制約と展望

7.1 2 次元のポインティング

本稿では，1 次元のポインティング実験のみを行った．一般的な GUI のターゲットは 2 次元であり，また，ターゲットの周囲には非ターゲットが存在する．1 次元のポインティングでは左右方向にのみ非ターゲットが存在し，実際の GUI 環境とは異なる．2 次元のポインティングでは，1 次元のポインティングとは異なり，非ターゲット上を通らないことも可能である．このとき，非ターゲットにはもちろん壁が生成されないため，壁の解除を行う必要がない．

© 2019 Information Processing Society of Japan

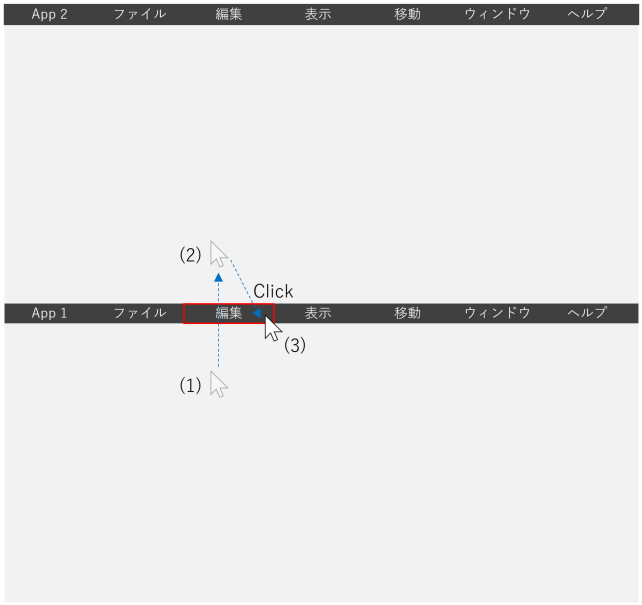

図 15　Valve Icon による拡張ディスプレイ時のメニューバーのポインティング．

また，非ターゲットをオーバーシュートし壁が発生した場合においても，非ターゲットを避けるように操作すれば，壁を解除せずにターゲットを選択できる．1 次元のポインティングと比較して，壁を解除する回数を減らせるため，Valve Icon がより有効に働く可能性がある．一方，ターゲットの壁に衝突させる際には，ターゲットを通るようにカーソルを動かす必要があるため，ターゲット幅が小さい場合には，1 次元のポインティングよりも困難になると考えられる．そのため，1 次元のポインティングでは，ターゲット幅が小さく，アイコン間の間隔が広い場合に，Valve Icon は有効であったが，2 次元のポインティングでは，そうならない可能性があると考えられる．

7.2　壁の解除方法

Valve Icon によって生成される壁は，本稿では，衝突後，カーソルを停止させてからもしくは壁からカーソルを離してから 200 ms 後に解除された．解除方法は他にも考えられるため，本実験の前に予備実験で他の方法について検証した．予備実験では，クラッチによって壁を解除する方法，一定時間後（200–1000 ms）に壁の解除を行う方法，本実験で用いた解除方法の 3 種類を試した．クラッチによって壁を解除する方法では，非ターゲットの壁を解除する際に，クラッチによって非ターゲットをオーバーシュートすることがあり，新たに非ターゲットに壁を生成してしまっていた．一定時間後に壁の解除を行う方法では，カーソルがターゲットの壁に衝突した後，ユーザの選択動作よりも壁が解除される方が速く，ターゲットを選択できずにオーバーシュートしてしまうことがあった．そのため，壁の発生時間を増加させ壁を解除したときに発生するオーバーシュートを防ごうとしたが，この方法では，壁の発生時間に応じて操作時間が増加してしまうため，Valve Icon がポインティングを高速化できなくなる．これらの予備実験を踏まえ，壁の解除方法を「カーソルの衝突後，カーソルを停止させてからもしくは壁からカーソルを離してから 200 ms 後」に設定した．カーソルが壁に衝突している間は壁が解除されないため，衝突中にオーバーシュートすることはない．また，非ターゲットに壁が生成された場合においても，200 ms で壁を解除できるため，操作時間の増加を抑えられる．実験結果をみてみると，Valve Icon の操作時間が従来手法よりも増加することはほとんどなく（図 11），本稿で用いた解除方法は有効であったと考えられる．

8.　結論

オーバーシュート後にターゲットの輪郭に壁を生成し，カーソルを壁に衝突させてポインティングを行う「Valve Icon」を提案した．比較実験の結果，アイコン間に十分な間隔がある状況で幅が小さいターゲットをポインティングする際に Valve Icon は有効であった．また，大きいターゲットをポインティングする際には，従来手法よりも操作時間が増加することはなかった．一方，小さいターゲットを十分な間隔がない状況でポインティングする場合に，提案手法の操作時間は増加した．これより，ターゲット幅が大きい，またはアイコン間に十分な間隔がある場合，Valve Icon を使うべきであるといえる．

参考文献

[1] Mandryk, R. L. and Gutwin, C.: Perceptibility and utility of sticky targets, *Proceedings of graphics interface 2008*, Canadian Information Processing Society, pp. 65–72 (2008).
[2] Blanch, R., Guiard, Y. and Beaudouin-Lafon, M.: Semantic pointing: improving target acquisition with control-display ratio adaptation, *Proceedings of the SIGCHI conference on Human factors in computing systems*, ACM, pp. 519–526 (2004).
[3] 築谷喬之，高嶋和毅，朝日元生，伊藤雄一，北村喜文，岸野文郎：Birdlime icon: 動的にターゲットを変形するポインティング支援手法，コンピュータ ソフトウェア，Vol. 28, No. 2, pp. 2_140–2_152 (2011).
[4] Kabbash, P. and Buxton, W. A. S.: The “Prince” Technique: Fitts’ Law and Selection Using Area Cursors, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’95, New York, NY, USA, ACM Press/Addison-Wesley Publishing Co., pp. 273–279, DOI: 10.1145/223904.223939 (1995).
[5] Grossman, T. and Balakrishnan, R.: The Bubble Cursor: Enhancing Target Acquisition by Dynamic Resizing of the Cursor’s Activation Area, *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI ’05, New York, NY, USA, ACM, pp. 281–290, DOI: 10.1145/1054972.1055012 (2005).
[6] Chapuis, O., Labrune, J.-B. and Pietriga, E.: DynaSpot: Speed-dependent Area Cursor, *Proceedings of the SIGCHI Conference on Human Factors in Com-*

© 2019 Information Processing Society of Japan

puting Systems, CHI '09, New York, NY, USA, ACM, pp. 1391–1400, DOI: 10.1145/1518701.1518911 (2009).

[7] Asano, T., Sharlin, E., Kitamura, Y., Takashima, K. and Kishino, F.: Predictive interaction using the delphian desktop, *Proceedings of the 18th annual ACM symposium on User interface software and technology*, ACM, pp. 133–141 (2005).

[8] 中辻智裕，山本景子，　倉本到，辻野嘉宏：航空路マウス：親指ジョイスティックでのターゲット方向指示によるポインティング高速化手法，インタラクション 2016 論文集，pp. 980–985 (2016).

[9] Casiez, G., Vogel, D., Pan, Q. and Chaillou, C.: RubberEdge: Reducing Clutching by Combining Position and Rate Control with Elastic Feedback, *Proceedings of the 20th Annual ACM Symposium on User Interface Software and Technology*, UIST '07, New York, NY, USA, ACM, pp. 129–138, DOI: 10.1145/1294211.1294234 (2007).

[10] Soukoreff, R. W. and MacKenzie, I. S.: Towards a standard for pointing device evaluation, perspectives on 27 years of Fitts' law research in HCI, *International Journal of Human-Computer Studies*, Vol. 61, No. 6, pp. 751 – 789, DOI: https://doi.org/10.1016/j.ijhcs.2004.09.001 (2004).

[11] Fitts, P. M.: The information capacity of the human motor system in controlling the amplitude of movement., *Journal of experimental psychology*, Vol. 47, No. 6, p. 381 (1954).

[12] MacKenzie, I. S.: A Note on the Information-Theoretic Basis for Fitts' Law, *Journal of Motor Behavior*, Vol. 21, No. 3, pp. 323–330, DOI: 10.1080/00222895.1989.10735486 (1989).

© 2019 Information Processing Society of Japan