ベイズ情報利得を用いたスクリーンキーボードカーソルの効率的移動手法

中嶋 誠 * 勝 世聡 * 五十嵐 健夫 *

概要. 本稿では, 方向キー操作によるスクリーンキーボードにおいて, ベイズ情報利得を用いた効率的なカーソル移動手法を提案する. スクリーンキーボードの改善としては, 盤面レイアウトを最適化する手法が提案されているが, ユーザの学習コストが問題となる. 本手法は, レイアウトは変更せず, カーソルの動きだけを変更する. 方向キーを押したときに, 単純に上下左右にカーソルが 1 つ移動するのでなく, 方向キーを押す回数が確率的に最も少なくなるように, 最適な位置へカーソルがジャンプする. ジャンプ先の計算には, ベイズ実験計画法をユーザインタラクションに応用した BIG(Bayesian Information Gain) アルゴリズムを用いる. さらに既存の BIG アルゴリズムに加えて, 新たに探索と活用の両方を考慮した目的関数を設定することで, 一度の方向キー押下で 1 マス以上カーソルが移動するシステムを実現した. 数値実験により, 提案手法を用いることで, 目的入力文字への到達に要するボタン押下数を 1 文字あたり平均 0.54 回削減できることが示された.

1 はじめに

コンピュータ黎明期から現在に至るまで, 文字の入力はヒューマンコンピュータインタラクション (HCI) における基本的なトピックの 1 つであり, 様々な手法が提案されてきた. 一般に, デスクトップコンピュータにおいては文字入力のデバイスとしてハードウェアキーボードを利用することが多い. 一方で近年普及の見られる小型の携帯端末等では, 筐体スペースの問題でハードウェアのキーボード機構を備えることが難しく, 代わりに画面に表示された文字をタップで選択するスクリーンキーボードがよく用いられている. またテレビや据置型ゲーム機のように画面と視聴者の距離があり, タッチパネルの利用が難しい場合は, リモコンやコントローラの方向キーによってスクリーンキーボード上のカーソルを移動し, 入力対象文字を選択する方法が一般的である.

本稿では, 方向キー操作によるスクリーンキーボードの入力高速化手法を提案する. 提案手法では, 方向キーを押したときに, 単純に上下左右にカーソルが 1 つ移動するのでなく, 方向キーを押す回数が確率的に最も少なくなるように, 最適な位置へカーソルがジャンプする. これにより, 目的の入力文字へ到達するまでに要する方向キー押下回数を削減できる. カーソルのジャンプ先の計算については, ベイズ実験計画法の観点から, ユーザの方向キー操作によってシステムから見た入力文字の曖昧さが減少すると考え, そのエントロピー減少をベイズ情報利得とし, 一度の入力で最も多くの情報利得を得られるカーソル位置を効率の良い移動先として選択する.

提案手法が対象とする方向キーによる文字入力は, 標準的なキーボード入力と比較してボタンが少ないため操作の自由度が低く, ユーザ側の習熟による速度向上の伸びしろが少ない. そのためアルゴリズムによる支援の意義があると考えられる. 特に, 赤外線リモコンと低スペック端末の組み合わせなどで, 方向キーを押してから画面上のカーソルが移動するまでに遅延がある場合などには, 押下回数削減の効果が大きいと考えらえる. また別の文脈として, 方向キー入力によるスクリーンキーボードは, 身体機能に制限がある人々の意思疎通手段として利用される場合がある. そういった利用シーンでは, 方向キーを 1 回クリックするにも多大な労力が必要であり, 本研究によって押下回数を削減することが大きな価値を持つといえる. 加えて, 研究の立ち位置という観点から, 本研究は文字入力という HCI 分野ではある意味研究し尽くされた分野であっても, 近年の確率的機械学習のコンセプトを取り入れることで, 新しいインタラクションの可能性があることを例示するものである.

2 関連研究

HCI 分野において, テキスト入力の支援手法が多数研究されてきた. 特にタッチ操作によるスクリーンキーボード入力の分野では, 画面上のキー配列を一般的な QWERTY 配列より効率的な配置に変更することで, 入力操作に要するコストを減らす手法が提案されてきた. メトロポリスキーボード [24] は, 英単語中の文字出現確率を考慮することで, 文字入力時の指移動量を最小化するキー配列手法である. Dunlop ら [5] は, 指移動量の最小化に加えて, 既存の QWERTY 配列との共通度を考慮することで, 入力コストが少なく, かつユーザの学習コストが少ないキー配列手法を提案した. また同様のキー配列最

* 東京大学, JST CREST

図 1. (a) 一般的なスクリーンキーボードのカーソル移動手法では目的文字 w への到達に 4 回の方向キー押下を要するが, (b) 提案手法では 3 回の押下で到達する.

適化手法を, スクリーンキーボード上をなぞって文字入力するジェスチャータイピングに応用する手法が提案されている [21, 3]. しかしこれらの手法は, ユーザが新しいキー配列に習熟するまでに学習コストがかかるという根本的な問題があり, 世間一般に普及するには至っていない. そのため, キー配列を維持したままで, 文字ごとのタップ判定範囲を変更することで入力エラーを減らす手法が提案されている [7].

我々の提案手法が対象とする, カーソル移動と決定操作によるスクリーンキーボード入力についても, 効率化手法が提案されてきた. Barrero ら [1] は, 方向キー操作を用いるデジタル TV 環境での文字入力エラーについて分析した. Bellman ら [2] は, 入力される確率の高い文字をグリッドの左上から順に配置し, 方向キー移動で文字を選択し, 一文字入力ごとにレイアウトを再配置する手法を提案した. また, 方向キーに限らず, 自由度の低い操作デバイスでの文字入力手法も研究されている. 松野ら [14] は, スクリーンキーボードの盤面を再帰的に分割・選択することで, 少ない操作ボタンでも効率的なテキスト入力手法を提案した. 田中ら [22] は, 4 ボタンそれぞれにいくつかの文字を割当て, ユーザが押したボタンの組み合わせから可能性のある単語を入力する手法を提案した.

限られた操作によるテキスト入力手法は, 身体機能に制限があるユーザを支援する技術としても研究されている [15]. Semantic keyboard[16] は, 身体の不自由なユーザらがスクリーンキーボードを 2D ポインタで操作する際に, 文字の登場確率に応じて Semantic pointing[4] の考え方を取り入れることでカーソル移動量を削減した. DUCK keyboard[18, 17] は, 視覚の不自由なユーザが読み上げ式スクリーンキーボードを利用するにあたって, 不正確な入力を許容して候補の単語を予測することで入力速度を向上した.

本稿の提案手法は, ユーザの入力に応じてシステムが持つ入力文字についての知識を更新するという観点で, Interactive Machine Learning (IML)[6] の一種と捉えることができる. インタラクティブな機械学習アプリケーションにおいて, 近年ではベイズ的な手法の利用が見られる. その中の一つのアプローチが, ベイズ最適化によってインタラクティブにデザイン空間の探索を行う研究である [5, 10, 9]. また別のアプローチとして, ユーザとソフトウェアのインタラクションをベイズ実験計画法 [11] の枠組みで捉えることで, システムがユーザに提示する情報を最適化する手法がある. Liu らは一連の研究 [12, 13] で, 情報理論の観点から, ユーザの操作が探索対象の曖昧さを減少させる点に着目し, その期待値を最大化する画面をユーザに提示する手法を提案した. 本稿の提案手法は, 同様の仕組みをスクリーンキーボード操作に応用したものである.

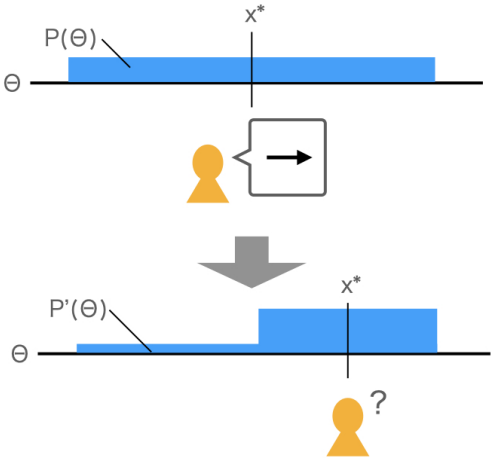

図 2. 二分探索タスクを例にすると, 提示されたビュー x^* に対するユーザの反応 y (この場合は →) によって, システムが持つ探索対象の確率分布 $P(\Theta)$ が $P'(\Theta)$ に更新され, エントロピーが減少する.

3 提案手法

我々の提案手法は, 方向キー操作で画面上のカーソルを移動するタイプのスクリーンキーボードにおいて, ユーザが方向キーを押した際のカーソル移動を効率化することで, 文字入力に要するキー押下数を削減する手法である. 通常のスクリーンキーボードでは, ユーザが方向キーを押下した方向に 1 マス分カーソルが移動し, 目的の文字に到達するまでその操作を繰り返す (図 1a). 一方提案手法では, ユーザが方向キーを押下した際のカーソル移動量が 1 マス分に固定ではなく, 入力文字候補を効率的に絞り込む位置へと適応的にカーソルが移動する (図 1b).

カーソル移動先の計算には, ベイズ実験計画法を利用した期待情報利得の最大化を考慮する. その要点は, ユーザの反応から最も多くの情報を引き出せる地点 x^* にカーソルを移動することである. ここでいう情報とは, 目的の文字についての確率分布の変動量である. 簡単のため一次元の例で説明すると, 単純な二分探索タスクにおいて, システムがユーザ

にある地点 x^* を見せて, 探索対象は現在位置の左右どちらにあるか尋ねるとする (図 2). 最初の時点でシステムは, 探索対象の位置については何の知識も持たないため, その確率分布 $P(\Theta)$ は一様分布である. ここでユーザが UI 操作で $\rightarrow$ と回答することで, システムは探索対象が現在位置より右にあることを知り, 確率分布を絞り込み $P'(\Theta)$ へと更新できる. ここで $P(\Theta)$ と $P'(\Theta)$ の間で減少したエントロピーを, 一度の操作で獲得した情報量とすると, その期待値が最大になる地点 x^* をユーザに提示する地点 (ビュー) として選択するのが, 探索という観点からは最も効率が良い戦略だと言える. これが提案手法の背後にあるアイデアである. 例のような単純な二分探索では, ビュー x^* をパラメータ空間 Θ の中点とすることで十分だが, 事前分布 $P(\Theta)$ が一様分布でない場合や, ユーザの行動 y に複数の選択肢があり, かつ確率的である場合に, 最適なビュー x^* をどのように選択するかは自明でない. そこで提案手法では, 獲得する情報量の期待値計算に基づくアルゴリズムによって x^* を決定する.

4 アルゴリズム

提案手法のアルゴリズムは, ベイズ実験計画法をヒューマンコンピュータインタラクション (HCI) に応用したものである. 以下に我々の利用するベイズ実験計画法の概要と, 提案手法への適用方法を述べる.

4.1 ベイズ実験計画法

元来のベイズ実験計画法は, 物理実験など複数回の実験を経て特定のパラメータを同定する場面で, トータル試行回数を少なくするために, 毎回の実験設定を適切に制御する手法である [11, 25]. Liu ら [12, 13] はベイズ実験計画法における実験 (experiment) と実験設定 (configuration) を, アプリケーションにおけるユーザの操作 (interaction) と GUI の状態 (view) に対応づけることで, HCI 分野への応用方法を示した. この形式でのベイズ実験計画法の HCI への応用を, 本稿では Liu ら [12, 13] に倣い, Bayesian Information Gain (BIG) アルゴリズムと呼ぶものとする.

BIG アルゴリズムは, インタラクティブシステム上で行われる探索的タスクにおいて, システムがユーザに提示する GUI の状態 (ビュー) を, そこから得られる情報利得の期待値が最大となるように最適化する手法である. ここでのビューは, 探索中のパラメータ空間における 1 点, または 1 範囲に相当する. それを見たユーザが, UI 上の操作 (アクション) によって, 現在地は目的地より上か下かといった判断結果を伝えることで, システムは探索対象の存在範囲を絞り込むことができる. つまり探索対象の確率分布についてエントロピーが減少する. このエントロピー減少を情報利得とし期待値を考えることで, システムがユーザに判断を仰ぐ価値のある点はどこかを求め, 最適なビューとして提示するというのが BIG アルゴリズムの要点である.

定量的に考えると, ユーザのアクションによって得られる情報利得は, そのアクション前後における探索対象の確率分布の差として KL ダイバージェンスにより計算できる. 以下で情報利得とその期待値計算についての定式化を述べる. 今, 探索対象がある離散パラメータ空間中で θ に位置する確率を, 確率変数 Θ を用いて $P(\Theta{=}\theta)$ とし, その確率分布を $P(\Theta)$ とする[1]. また, X をシステムが取り得る全てのビュー状態の集合, Y をユーザが取り得る全てのアクションの集合とすると, ユーザはシステムが提示したビュー $x \in X$ に基づき, アクション $y \in Y$ を行うことになる. これにより探索対象の確率分布 $P(\Theta)$ が, ビュー x とアクション y を与えられた場合の条件付き確率分布 $P(\Theta \mid x, y)$ へと更新される. すると, ユーザに提示する最適のビュー x^* は, 何らかの形でコストを定義するユーティリティ関数 $u(\cdot)$ と, 期待値 $\mathrm{E}[\cdot]$ を用いて,

$$x^* = \operatorname*{argmax}_{x \in X} \mathrm{E}[u(\theta, x, y)],$$

と定義される. 対象探索タスクにおいては, ユーティリティ関数を事前分布と事後分布の KL ダイバージェンス[2] とすることが一般的である [19]. その場合ユーティリティ関数は,

$$\begin{aligned} u(x, y) &\coloneqq -D_{\mathrm{KL}}(P(\Theta \mid x, y) \parallel P(\Theta))) \\ &\coloneqq \mathrm{IG}(x, y), \end{aligned}$$

と定義され, これを情報利得 (Information Gain: IG) と呼ぶ. この関数はユーザが選択したアクション y に依存するが, ユーザの選択を事前に知ることは一般的に不可能であるため, 取り得る全ての選択肢 $y \in Y$ についての期待値を計算することで, 期待情報利得 (Expected Information Gain: EIG) とする.

$$\begin{aligned} \mathrm{E}[u(x)] &\coloneqq -\mathop{\mathrm{E}}_{y \in Y}[D_{\mathrm{KL}}(P(\Theta \mid x, y) \parallel P(\Theta))] \\ &\coloneqq \mathrm{EIG}(x). \end{aligned}$$

これにより, ユーザに提示する最適なビュー x^* の選択を, 期待情報利得の最大化問題として定義できる.

$$x^* = \operatorname*{argmax}_{x \in X} \mathrm{EIG}(x). \tag{1}$$

[1] 本稿では表記を簡潔にするため, 以降 $P(\Theta = \theta)$ を $P(\theta)$ と表記する. 一言でいうと, $P(\Theta)$ はパラメータ空間 Θ 全体の確率分布であり, $P(\theta)$ は具体的な θ が与えられた時にそこが目標地点である確率である. また同様に, $P(X{=}x)$ を $P(x)$, $P(Y{=}y)$ を $P(y)$ と表記する.

[2] x が離散的であるときに, 確率分布 $f(x)$, $g(x)$ 間の KL ダイバージェンス $D_{\mathrm{KL}}(f(x) \parallel g(x)) = \sum_x f(x) \log_2 \frac{f(x)}{g(x)}$ である.

Liu ら [12, 13] の先行研究では式 (1) を用いているが, これはベイズ最適化問題における探索 (Exploration) と活用 (Exploitation) の, 探索だけを考慮した目的関数設計といえる. しかし我々のアプリケーションで検証したところ, 探索だけを目的関数とした場合, キーボード盤面の端で到達できないマスが生じるため, 活用もコストとして算入する必要があることがわかった. そのため我々は k を任意の定数として ($k = 0.3$), 位置 x の文字が探索対象である確率 $P(x)$ を活用コストとして加えた,

$$x^* = \underset{x \in X}{\operatorname{argmax}} \left(\mathrm{EIG}(x) + kP(x) \right), \tag{2}$$

を解くべき最適化問題とした.

最適ビューの選択にあたっては, 今回はビューの総数 $|X|$ が高々アルファベットの文字数程度のため, 総当たりで全てのビュー $x \in X$ について式 (2) のコスト項を計算し, 最大となるビューを提示する. また本稿で対象とするパラメータ空間 Θ は離散的であるため, $\mathrm{EIG}(x)$ に含まれる期待値計算は単純な足し合わせによって行うことができる.

$$\begin{aligned} \mathrm{EIG}(x) = &\sum_{y \in Y, \theta \in \Theta} P(y, \theta \mid x) \log_2 P(\theta \mid x, y) \\ &- \sum_{\theta \in \Theta} P(\theta) \log_2 P(\theta). \end{aligned} \tag{3}$$

式 (3) に含まれる事後確率 $P(\theta \mid x, y)$ は陽的に求めることができないため, ベイズの定理

$$P(\theta \mid x, y) = \frac{P(y \mid \theta, x) P(\theta)}{P(y \mid x)},$$

を用いて式変形すると,

$$\begin{aligned} \mathrm{EIG}(x) = &\sum_{y \in Y, \theta \in \Theta} P(\theta) P(y \mid \theta, x) \log_2 P(y \mid \theta, x) \\ &- \sum_{y \in Y} P(y \mid x) \log_2 P(y \mid x), \end{aligned} \tag{4}$$

となり, ここで

$$P(y \mid x) = \sum_{\theta \in \Theta} P(y \mid \theta, x) P(\theta),$$

である. $P(\theta)$ は事前確率であり, $P(y \mid \theta, x)$ を後述するモデル化したユーザのアクション確率として与えることで, 式 (4) の全ての項を数値的に計算可能となる.

実際のインタラクションにおいては, ユーザが探索対象に到達するまでに, システムによる最適ビュー x_i^* の提示と, ユーザのアクション y_i, それに応じた確率分布 $P_i(\Theta)$ の更新を繰り返し行うことになる. 確率分布の更新はベイズの定理を用いて, 全ての $\theta \in \Theta$ について,

$$P_{i+1}(\theta) = P_i(\theta \mid x_i, y_i),$$

を計算する. それによって得られた確率分布 $P_{i+1}(\Theta)$ を新しい事前分布として, 次に表示するビュー x_{i+1}^* を計算する.

表 1. モデル化したユーザのアクション確率

x と θ の位置関係	各ボタンを押す確率 (%)				
	←	→	↑	↓	OK
同じ	1.25	1.25	1.25	1.25	95
右	1.25	95	1.25	1.25	1.25
右上	2	47	47	2	2
上	1.25	1.25	95	1.25	1.25
左上	47	2	47	2	2
左	95	1.25	1.25	1.25	1.25
左下	47	2	2	47	2
下	1.25	1.25	1.25	95	1.25
右下	2	47	2	47	2

4.2 BIG キーボード

前項で説明した BIG アルゴリズムの枠組みにより, スクリーンキーボードカーソルを効率的に移動する手法を述べる. 任意のアプリケーションに BIG アプリケーションを適用するには, そのパラメータ空間 Θ, 初期の確率分布 $P(\Theta)$, 隠れ変数である探索対象の位置 θ, モデル化されたアクション確率 $P(y \mid \theta, x)$ を定義する必要がある. 提案手法のスクリーンキーボードにおいては, それぞれ以下に対応する.

- パラメータ空間 Θ: 入力可能なアルファベット文字全体 (a-z, カンマ, ピリオド)
- 事前分布 $P(\Theta)$: 一様分布もしくは文字の登場確率に応じた分布
- 隠れ変数 θ: ユーザが入力の目的とする文字
- ビュー x: カーソルの位置
- アクション y: ユーザの押したボタン (上下左右, 決定)

また, モデル化したアクション確率 $P(y \mid \theta, x)$ のテーブルによって, カーソル現在位置が x で, 目的文字が θ の時にユーザが各ボタンを押す確率を定義しておく (表 1). ユーザは画面上で x から見た θ の位置によって, 一定確率で方向キー (←→↑↓), または決定キー (OK) を押すが, 目的文字がカーソル現在位置の (i) 斜め方向にある, (ii) 垂直または水平方向にある, (iii) 同じ位置にある, 場合で行動の割合が異なる. 我々は経験的に, ユーザは約 95%の確率で正しいボタンを押すものとしてモデル化を行った. ここでの正しいボタンとは上のパターン分けそれぞれの場合において (i) 対象の方向ボタン 2 つのうちいずれか, (ii) 対象の方向ボタン 1 つ, (iii) 決定ボタン, である.

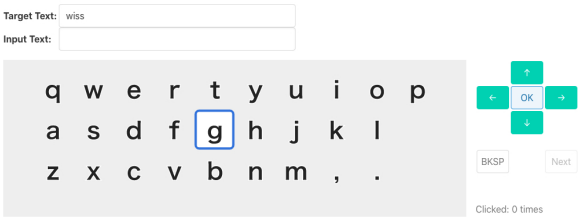

図 3. 提案手法を実装したスクリーンキーボード.

5 結果

我々は提案手法のスクリーンキーボードを実装し, 基本的な挙動を確認した (図 3). 一例としてキーボードで文字 w を入力する場合の挙動を示す. まずカーソルの初期位置は g にあるものとする. 標準のスクリーンキーボードでは, g から ← を 3 回, ↑ を 1 回押して, 合計 4 回の押下で w に到達する. 提案手法の BIG キーボードでは, まず g から ← を押して d に移動, ← を押して s に移動, 最後に ↑ を押して合計 3 回の押下で w に到達する.

同様に, 初期位置から開始して, ユーザがミスなく操作して各文字へ到達するのに要する方向キーの押下回数を計算により求めると, 標準スクリーンキーボードでは 1 文字あたり平均 3.0 回, BIG キーボードでは 1 文字あたり平均 2.46 回であった. よって提案手法では, 押下回数を 1 文字あたり平均 0.54 回削減できることが示された.

我々の提案手法では, 最適ビューの選択に探索と活用両方のコストを考慮する改良を行ったが, これによる既存手法との挙動の違いについても述べる. 特に問題となるのは目的入力文字が p の場合である. 既存手法 (式 1) では, 初期位置から ↑→→ の入力で o に到達し, さらに → を押してもカーソルはそれ以上右に移動せず, 結果として文字 p を入力することができない. 提案手法 (式 2) では, 初期位置から ↑→→ の入力で o に到達し, さらに → を押すことで, 問題なくカーソルが p に移動する. 既存手法が上手く動作しないのは, 探索の観点からすると, p の地点に行ってユーザの判断を仰いだとしても, それ以上候補文字の絞り込みに貢献しないため, 移動する価値がないとしてカーソルが o に留まる結果になるためである. 提案手法では, 探索への貢献だけでなく, 探索によって目的文字である確率が高いとされた場所に移動する行為 (活用) にも価値を与えることで, 意図した挙動となった.

6 むすび

本稿では, ベイズ実験計画法をユーザインタラクションに応用した BIG アルゴリズムによって, スクリーンキーボードにおけるカーソル移動を効率化する手法を提案した. そのために必要なパラメータを定義し, 作成したユーザアクションの確率テーブルを利用して, 必要な計算が実際に可能であることをプロトタイプにより示した. また理論面で, 既存研究で用いられた BIG アルゴリズムの目的関数に, 探索だけでなく活用のコストを算入することで, キーボード盤面の端における挙動が正しくなることを確認した. 計算による検証では, 目的文字に到達するために必要な方向キー押下数が, 1 文字あたり平均 0.54 回削減できることが示された.

今後の課題としては, ユーザスタディによって, 単純な押下回数削減だけでなく, ユーザエクスペリエンスが改善されているか検証の必要がある. 提案手法は標準的なキー配列と操作方法であるため学習コストは低く, その点については問題が少ないと思われる. 一方で, 初期位置付近にある f や h といった文字の入力時に, カーソルの挙動が非直感的になる場合があり, それによって引き起こされるユーザの誤操作がパフォーマンスに悪影響を及ぼす可能性がある[3]. またユーザアクションの確率モデルについて, 本稿では経験的にそれらしい値を設定して利用したが, 十分な根拠のある数値とは言い難い. これについては実際のユーザ行動について統計データを集めることで, より正確なモデル化が可能になるものと思われる.

最後に将来的な改善可能性について述べる. まず自然な拡張として, ひらがな五十音のスクリーンキーボードのように, より大きな盤面に対して BIG アルゴリズムを適用することで, 削減される押下回数のインパクトが大きくなると予想される. またユーザインタフェースとして, DriftBoard[20] と組み合わせて小さなディスプレイでの利用が考えられる. アルゴリズム面では, 文章の入力を想定した場合, 現在入力されている文字に応じた, 次の文字の登場確率を事前分布 $P(\Theta)$ として導入することで, パフォーマンス向上が想定される. 既存の確率的アプローチを, ヒューリスティックなパラメータ調整に頼らず自然な形でシステムへ導入できることは BIG アルゴリズムの利点である. 加えて, 現在の実装では毎回カーソルを同じ初期位置に戻しているが, 直前の文字や, これまで入力した文章を元に, より良い初期位置へ自動的に移動することで, 方向ボタンの押下回数をさらに削減できると考えられる.

BIG アルゴリズム自体に関する問題として, 現在のアルゴリズムは情報理論の観点から最適な挙動であっても, ヒューマンインタフェースとしては最適な挙動ではない可能性がある. そのため, 操作を行うのが機械ではなく人間であることを考慮した目的関数設計の余地がある. 具体的には, ユーザにとっては目的地を通り過ぎてから戻るようなジグザグの軌跡をとるカーソル挙動は非直感的で, 慣れるまで

[3] f の入力は ←→ の 2 押下, h の入力は →← の 2 押下を要する.

予想がつきにくい. 結果として, 一般的なカーソル移動手法ではスタートからゴールまでに必要な操作を最初に頭に思い浮かべて一括実行できるのに対し, 提案手法では毎回の移動ごとに現在位置を確認し適切なアクションを判断する認知的コストが細切れに発生する. このような人間の認知コストを考慮に入れた数値的最適化とユーザインタフェースのデザインは, 既存の心理学的認知モデル等と合わせることで [8, 23], 今後 HCI 分野における発展可能性が残されていることが考えられる.

謝辞

本研究は JST CREST JPMJCR17A1 の支援を受けたものである.

参考文献

[1] A. Barrero, D. Melendi, X. G. Paneda, R. Garcia, and S. Cabrero. An Empirical Investigation Into Typing Errors in Interactive Digital Television Applications. *Int. Journal of HCI*, 31(3):210–225, 2015.

[2] T. Bellman and I. S. MacKenzie. A Probabilistic Character Layout Strategy for Mobile Text Entry. In *Graphics Interface*, 1998.

[3] X. Bi and S. Zhai. IJQwerty: What Difference Does One Key Change Make? Gesture Typing Keyboard Optimization Bounded by One Key Position Change from Qwerty. CHI '16. ACM, 2016.

[4] R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic Pointing: Improving Target Acquisition with Control-Display Ratio Adaptation. CHI '04, p. 519–526. ACM, 2004.

[5] M. Dunlop and J. Levine. Multidimensional Pareto Optimization of Touchscreen Keyboards for Speed, Familiarity and Improved Spell Checking. CHI '12, p. 2669–2678. ACM, 2012.

[6] J. A. Fails and D. R. Olsen Jr. Interactive machine learning. IUI '03, pp. 39–45. ACM, 2003.

[7] A. Gunawardana, T. Paek, and C. Meek. Usability Guided Key-Target Resizing for Soft Keyboards. IUI '10, p. 111–118. ACM, 2010.

[8] B. E. John. Using Predictive Human Performance Models to Inspire and Support UI Design Recommendations. CHI '11, p. 983–986. ACM, 2011.

[9] Y. Koyama, I. Sato, and M. Goto. Sequential Gallery for Interactive Visual Design Optimization. *ACM Trans. Graph.*, 39(4), 2020.

[10] Y. Koyama, I. Sato, D. Sakamoto, and T. Igarashi. Sequential Line Search for Efficient Visual Design Optimization by Crowds. *ACM Trans. Graph.*, 36(4), 2017.

[11] D. V. Lindley. On a Measure of the Information Provided by an Experiment. *The Annals of Mathematical Statistics*, 27(4):986–1005, 1956.

[12] W. Liu, Rafael Lucas D'Oliveira, M. Beaudouin-Lafon, O. Rioul. BIGnav: Bayesian Information Gain for Guiding Multiscale Navigation. CHI '17, p. 5869–5880. ACM, 2017.

[13] W. Liu, O. Rioul, J. McGrenere, W. E. Mackay, and M. Beaudouin-Lafon. BIGFile: Bayesian Information Gain for Fast File Retrieval. CHI '18. ACM, 2018.

[14] S. Matsuno, S. Chida, N. Itakura, T. Mizuno, and K. Mito. A method of character input for the user interface with a low degree of freedom. *Artificial Life and Robotics*, 24(2):250–256, 2019.

[15] O. Polacek, A. J. Sporka, and P. Slavik. Text input for motor-impaired people. *Universal Access in the Information Society*, 16(1):51–72, 2017.

[16] M. Raynal, I. S. MacKenzie, and B. Merlin. Semantic Keyboard: Fast Movements between Keys of a Soft Keyboard. Computers Helping People with Special Needs, pp. 195–202. Springer, 2014.

[17] M. Raynal and P. Roussille. Use of DUCK Keyboard for a Daily Use. ICCHP '18, pp. 399–406. Springer, 2018.

[18] P. Roussille, M. Raynal, and C. Jouffrais. DUCK: A DeDUCtive Soft Keyboard for Visually Impaired Users. IHM '15. ACM, 2015.

[19] K. J. Ryan. Estimating Expected Information Gains for Experimental Designs With Application to the Random Fatigue-Limit Model. *Journal of Computational and Graphical Statistics*, 12(3):585–603, 2003.

[20] T. Shibata, D. Afergan, D. Kong, B. F. Yuksel, I. S. MacKenzie, and R. J. Jacob. DriftBoard: A Panning-Based Text Entry Technique for Ultra-Small Touchscreens. UIST '16, p. 575–582. ACM, 2016.

[21] B. A. Smith, X. Bi, and S. Zhai. Optimizing Touchscreen Keyboards for Gesture Typing. CHI '15, p. 3365–3374. ACM, 2015.

[22] K. Tanaka-Ishii, Y. Inutsuka, and M. Takeichi. Entering Text with a Four-Button Device. COLING '02, p. 1–7. Association for Computational Linguistics, 2002.

[23] D. P. Wallach, S. Fackert, and V. Albach. Predictive Prototyping for Real-World Applications: A Model-Based Evaluation Approach Based on the ACT-R Cognitive Architecture. DIS '19, p. 1495–1502. ACM, 2019.

[24] S. Zhai, M. Hunter, and B. A. Smith. The Metropolis Keyboard - an Exploration of Quantitative Techniques for Virtual Keyboard Design. UIST '00, p. 119–128. ACM, 2000.

[25] 松井 孝太, 金森 研太, 豊浦 和明, 竹内 一郎. ベイズ最適化の基礎と材料工学への応用. まてりあ, 58(1):12–16, 2019.