Vol.2009-HCI-134 No.4
2009/7/16

タイピングの特性を用いた文字入力中のコマンド入力方式

片山拓也[†1]　村尾和哉[†1]
寺田努[†2]　西尾章治郎[†1]

現在，PC 用のキーボードは主に文字入力に用いられ，ファンクションキーやショートカットキーなどを活用することで数多くの機能を実現している．しかし，これらの入力によって高速な操作が可能となる一方，キーと機能の組み合わせを覚えなければならないという問題点がある．そこで本研究では，従来の文字入力を行う「タイピング」に加えて，キーボードをなぞる「ストローク」およびキーボードをある形状で押す「スタンプ」の 3 種類のコマンドの入力をシームレスに行う方式を提案する．提案手法はキーボードの入力特性から「タイピング」,「ストローク」,「スタンプ」を自動的に判定をするため，特別な装置を用いることなく文字入力中のシームレスなコマンド入力を実現する．提案手法を用いることで，利用者が覚えやすいストロークやスタンプと欲しい機能の組み合わせを定義できるなどより直観的な操作が可能となる．

A Command Input Method while Texttyping using Typing Characteristics

Takuya Katayama,[†1] Kazuya Murao,[†1]
Tsutomu Terada[†2] and Shojiro Nishio[†1]

Keyboard is mainly used for text input, and also it has functional keys and shortcut keys to realize lots of functions. Though these input methods enable us to input faster, users have to learn relationships between keys and functions. In this paper, we propose three kinds of input methods: *stroke* that traces a shape on a keyboard, *stamp* that presses a shape on a keyboard, and usual *texttyping*. Our system recognizes texttyping, stroke, or stamp just from characteristics of typing trajectory automatically, to enable us us seamlessly to input command while texttyping. By using our system, users can define relationships between stroke or stamp and function as they like, and intuitive input can be achieved.

1. はじめに

現在，キーボードとマウスはその操作性の高さからコンピュータの入力デバイスとして不可欠なものとなっている．特に，文字入力に用いられるキーボードは，ある程度習熟した利用者が手軽に特定の機能を実行するために，しばしばショートカットキーと呼ばれる，利用頻度が高い機能を特定のキーの組合せ入力で実行する入力方式が用いられる．しかし,「それぞれの機能に対応するキーを覚えておかなければならない」,「ツールによって使用するコマンドが異なる場合がある」などショートカットキーはコンピュータの習熟度が低い利用者は使いづらい．

これまでにキーボードにおける入力を効率化するための様々な研究が行われ，多種多様な文字入力デバイスが提案されているが，現在のキーボードの形状は基本的に万国共通であり，広く普及しているため，新たな形状のデバイスを設計することは利用者への敷居が高い．

そこで本研究では，通常の文字入力を行う「タイピング」に加えて，キーボードの上をなぞる「ストローク」，およびキーボードの範囲を押す「スタンプ」の 2 種類のコマンド入力によって直観的に特殊機能の入力を行う方式を提案し，特別な装置を用いることなく，既存のキーボードのみを用いて，文字入力中にシームレスにコマンド入力が行える方式を構築する．提案方式は，キー入力の特性から現在の入力が,「タイピング」,「ストローク」,「スタンプ」のいずれかを自動的に判別する機能をもつため，利用者は一般の入力とコマンド入力をシームレスに行える．本研究ではシステムのプロトタイプを実装し，利用者評価によって識別精度を調査した．さらに，コマンドと機能を関連付けるインタフェースを構築した．

以下，2 章で関連研究を紹介し，3 章で構築したコマンド入力方式について説明し，4 章で構築したプロトタイプシステムを紹介し，5 章で提案システムの評価について述べる．そして，6 章で考察を述べ，最後に 7 章で本研究のまとめを行う．

2. 関連研究

これまでにキーボードに文字入力以外の様々な機能を付加する研究が数多く行われている．ポインティング機能をキーボードに追加する研究として，Pointing Keyboard[1)] があ

†1 大阪大学大学院情報科学研究科
Graduate School of Information Science and Technology, Osaka University
†2 神戸大学大学院工学研究科
Graduate School of Engineering, Kobe University

Vol.2009-HCI-134 No.4
2009/7/16

る．これはキーボード上に 2 次元座標検出のための赤外線センサを重ねた構造になっており，キーボードを「押す」動作と「なでる」動作の違いを検出し，キーボード面上でキー入力とポインティング操作の両操作を行うことができるものである．しかし，このシステムの実現には，新たにキーボードに赤外線センサを取り付ける必要があり，既存のキーボードには適用できないという問題点がある．FingerWorks 社の TouchStream[2] も同様に専用のキーボードの盤面を設計したものである．TouchStream は盤面を叩けばキーボードとして利用できるが，2 本指でなぞるとポインティング，3 本指で叩けばダブルクリックなど数十種類のジェスチャ入力が可能なキーボードである．これらに対して提案システムでは，既存のキーボードから得られる情報のみを用いた柔軟なコマンド入力を実現する．

また，ThumbSence[3] は，ノート PC 用のポインティングデバイスとして広く普及しているタッチパッドの左右のマウスボタンの操作時にキーボードのホームポジションから手が離れるという欠点を解決するため、キーボードのホームポジションに手を置いたままでもマウスボタンを操作することを可能にする技術である．具体的には，タッチパッドに指が触れている間は，キーボードにマウスのボタンの機能や，更にはその他のショートカットコマンドを割り当てるというものである．これにより，キーボードのホームポジションに手を置いたまま様々な操作が可能になるが，どのキーにどのようなコマンドが割り当てられているのかを記憶するのは，煩雑である．これは，ショートカットキーにも言えることである．ある程度 PC の操作に慣れている人はショートカットキーを使いこなすが，PC の操作が未熟な利用者にとっては習熟に時間を要する．提案システムでは，単純な操作でコマンド入力できるだけでなく，利用者がその場で新しくコマンドを作成し，機能を割り当てることを可能にしているため，PC の操作が未熟な利用者でも，容易に使いこなせるコマンド入力方式であると言える．また PC の操作に慣れている人でもショートカットキーには割り当てられていない新たな機能を割り当てることが可能である．

ExpressiveTyping[4] は，ノートパソコンが落下時にハードディスクを緊急停止させるために用いる内蔵型加速度センサの値を用いて打鍵圧を取得し，打鍵圧に応じたフォントサイズの変更を行う技術である．この技術のように，利用者が直観的に操作できるシステムが提案されており，提案システムもその取り組みの一環といえる．

3. コマンド入力方式の設計

3.1 システム構成

提案システムの動作フローを**図 1** に示す．なお，ここでタイピングはに単語や文章を入力している状態，ストロークは丸や四角，三角といった図形をキーボードをなぞって一筆書きで描いた状態，スタンプはグーやパーなどの形を手で作ってキーボードを押した状態を表す．ストロークやスタンプを用いることで，ウィンドウのサイズをキーボードの指のスライド幅で調節したり，キーボードを叩くことでウィンドウを閉じるといった直観的な操作が可能になる．

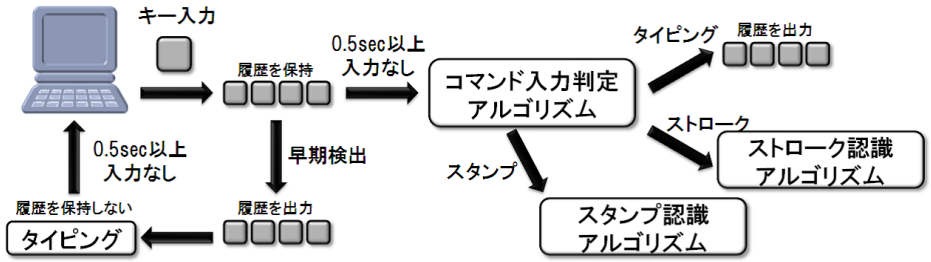


図 1 システムの動作フロー
Fig. 1 Flow of the system operation

提案システムでは，ストロークやスタンプが入力された際にキーの入力を無効化してコマンドに置き換える必要があるため，キーイベントが発生したらその履歴をシステムで保持する．そして，キーイベントが一定時間 (初期値は 0.5 秒に設定) 発生しなければ，その履歴がタイピングかコマンドかを判別し，処理を行う．その際，その履歴がタイピングだと判断されれば保持したキー入力の履歴を OS に流し，コマンドだと判断されれば，そのコマンドに応じたイベントを発生させる．一方，キーイベント発生時に文字を出力せずにただ履歴を蓄積すると，キー入力を行ってもそれが画面に反映されないという利用者にとっての違和感を生じさせるため，タイピングに関しては可能な限り早期に検出する必要がある．

3.2 コマンド入力の判定方法

提案システムでは，新たに特別な装置を用いることなく，キー入力の履歴のみからコマンド入力を識別する．予備実験として，タイピング，ストローク，スタンプの 3 種類の入力を 30 回ずつ行った場合のキー入力履歴を取り，

- 時間範囲 t_{span} : 最初のキーダウンから最後のキーダウンまでにかかった時間
- 押下時間 t_{push} : 各キーの平均押下時間
- 移動距離 c_{dist} : 次のキーまでの平均移動距離
- 座標分散 c_{dev} : キー座標の分散

Vol.2009-HCI-134 No.4
2009/7/16

図 2 キーと座標の対応
Fig. 2 Relationship between keys and coordinates

- 打鍵数 n：入力されたキーの数

を調べたところ**表 1** のようになった．ここで，**図 2** に示すように，キー座標 (x,y) は (0,0)〜(75,25) の範囲であらかじめマッピングしており，c_{dist}，c_{dev} は，入力文字の座標列 $S = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ が与えられた時にそれぞれ以下の式で求められる．

$$c_{dist} = \frac{1}{(n-1)} \sum_{k=1}^{n-1} \sqrt{(x_{k+1} - x_k)^2 + (y_{k+1} - y_k)^2}$$

$$c_{dev} = \frac{1}{2n} \sum_{k=1}^{n} (x_k - \bar{x})^2 + (y_k - \bar{y})^2 \qquad (\bar{x} = \frac{1}{n} \sum_{k=1}^{n} x_k, \quad \bar{y} = \frac{1}{n} \sum_{k=1}^{n} y_k)$$

表 1 より，以下の特徴が抽出された．

- t_{span}：スタンプの値の平均が極端に小さいことが分かる．これは，スタンプ入力の際は同時に入力する全てのキーを打鍵しているためである．
- t_{push}：大きな特徴は見られなかった．
- c_{dist}：ストロークとスタンプの値の平均が小さく，ストロークの標準偏差が極端に小さいことが分かる．この理由として，ストローク入力の際は指をキーボード上で滑らせて入力するため常に隣接するキーに移動している，スタンプ入力の際は，片手で入力しているため，離れたキーが同時に押されることがない，ということが挙げられる．
- c_{dev}：スタンプの値の平均が小さいことが分かる．これは，スタンプの移動距離が小さい理由と同様である．
- n：スタンプの値の平均が少ないことが分かる．これは，一般のキーボードでは内部の電気回路的な制約により，押下したキー全ては入力できないということが原因として挙げられる．

表 1 予備実験結果
Table 1 Result of pilot study

項目	入力方式	平均	標準偏差	最短	最長
t_{span}(msec)	タイピング	1044.5	674.6	0.0	2515.6
	ストローク	700.5	176.9	468.75	1109.4
	スタンプ	65.7	88.5	0.0	250.0
t_{push}(msec)	タイピング	137.0	35.1	108.4	243.4
	ストローク	109.7	14.5	85.4	153.4
	スタンプ	273.4	153.6	31.3	878.9
c_{dist}	タイピング	17.3	6.1	0	25.5
	ストローク	6.0	0.3	5.5	7.1
	スタンプ	6.8	4.6	0.0	16.8
c_{dev}	タイピング	93.3	42.5	0.0	158.8
	ストローク	30.7	10.0	15.0	60.7
	スタンプ	11.9	14.3	0.0	92.4
n	タイピング	12.0	6.3	1	20
	ストローク	12.5	2.7	8	17
	スタンプ	3.1	1.8	1	8

得られた特徴を考慮して，コマンド入力を識別するアルゴリズムを以下のように設計した．

if($n > 20$ or $n = 1$) then typing
else if($t_{span} < 250$ and $c_{dist} < 10$) then stroke
else if($c_{dev} < 100$) then stamp
else typing

しかし，この設計ではキーストローク間隔が 0.5 秒以上開くまでキー入力の履歴を記録して，コマンド入力の判定後に文字列を画面上に表示する，この入力と出力のタイムラグを解消するタイピング早期検出を以下のように設計した．前述の特徴からコマンド入力の c_{dist} と n が小さいこと，また今回提案するコマンド入力では同じキーが連続で打鍵される可能性がないことに着目して，現在，キー入力の履歴の座標列 $S' = \{(x'_1, y'_1), (x'_2, y'_2), \ldots, (x'_n, y'_n)\}$ がある状態，新たにキー (x'_{n+1}, y'_{n+1}) が押された時に，以下のいずれかの条件を満たす場合は，その履歴をタイピングであると判定して，現在のキー入力の履歴をシステムで保持せず，すぐに画面に表示するという処理を加えた．

- $\sqrt{(x_{n+1} - x_n)^2 + (y_{n+1} - y_n)^2} > 20$
- $n \geq 20$
- $x'_n = x'_{n+1}$ and $y'_n = y'_{n+1}$

Vol.2009-HCI-134 No.4
2009/7/16

さらに，その後もタイピングが続けられると想定し，タイピングの早期検出が発生してから，キーイベントが一定時間 (初期値は 0.5 秒に設定) 発生しなくなるまで，その間に発生したキー入力はタイピングであるとみなす．

3.3 ストロークの認識

システムでストローク入力と判断されたキー入力の履歴は，DP マッチングを用いた文字列比較を改良した手法によって識別する．DP マッチングを用いた文字列比較では，文字列の不一致の原因として「挿入・削除」と「置き換え」の 2 種類を定義し，それぞれに適当なコスト $Cost_{del-ins}$，$Cost_{replace}$ を定めて文字列同士の乖離度をスコア化する．具体的には “かたやま” と “いかりやま” の乖離度は，“かたやま” の先頭に ‘い’ を挿入して (空白文字を挿入して ‘い’ に置き換えて)，‘た’ を ‘り’ に置き換えれば “いかりやま” になることから，挿入コストと置き換えコストの和 ($Cost_{del-ins} + 2Cost_{replace}$) で表わされる．この文字列比較の手法をストロークの軌道の比較に用いる．**図 3** で示すように，一般的なキーボードのアルファベットキーは周囲のキーと 6 方向で接しているため，文字列の軌跡を図 3 のように，右上に移動した場合は ‘A’ に，同様に右に移動した場合，右下に移動した場合，左下に移動した場合，左に移動した場合，左上に移動した場合はそれぞれ ‘B’，‘C’，‘D’，‘E’，‘F’ に置き換えた．例えばキーボードを上を円の形になぞって「‘B’‘G’‘Y’‘U’‘J’‘N’‘B’」の順にキーが押された場合，その軌跡は “FABCDE” という文字列に変換される．また，挿入・削除のコストを 1 に設定し，置き換えのコストを軌跡のずれの角度の大きさにより ‘B’ と ‘C’ や ‘F’ と ‘A’ など隣合う場合は 2，‘B’ と ‘D’ や ‘F’ と ‘B’ などの場合は 4，‘B’ と ‘E’ や ‘F’ と ‘C’ など正反対の場合には 6 となるように設定した．そして，システムにはあらかじめ全てのストローク入力と正解のストロークラベルの組を登録しておき，未知のストローク入力に対して登録しているストローク入力との距離を求め，最も近いストローク入力のラベルを認識結果とし，それに応じたコマンドを出力する．以下にストローク入力同士の距離を求める際の詳細な計算手順を具体例を用いて説明する．

「‘B’‘G’‘Y’‘U’‘J’‘N’‘B’」と「‘Z’‘A’‘W’‘3’‘E’‘D’‘X’‘Z’」の 2 種類のストロークの距離を計算する．まず 2 つのストロークの軌跡を文字列に変換する (**図 4**(a))．そして，それらの置き換えコストを求める (図 4(b))．その後，テーブルの左上から右下に向かって置き換えコストと挿入・削除コストの和を計算する (図 4(c,d))．ここで，挿入・削除はテーブルを右，あるいは下に動くことを意味し，左上からの移動の場合はそのコストがかからない．そして，最終的に一番右下で計算されたコスト，この例の場合は 3 が 2 つのストロークの距離である．

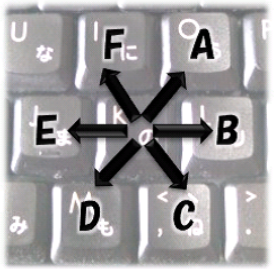
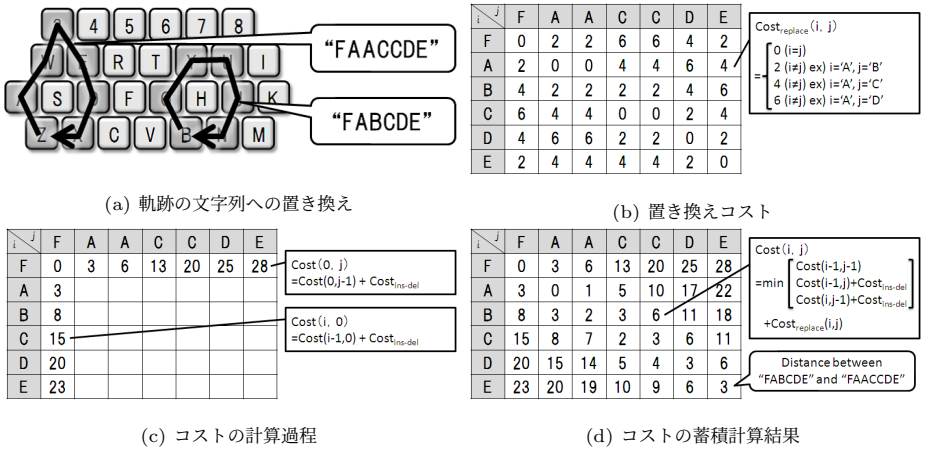


図 3 軌跡の置き換え
Fig. 3 Substituting trajectory for character string



(a) 軌跡の文字列への置き換え
(b) 置き換えコスト
(c) コストの計算過程
(d) コストの蓄積計算結果

図 4 ストロークの距離の導出過程
Fig. 4 Process of calculating the distance between two strokes

提案システムでは，利用者が学習用のストロークとそれに対応する処理の組を自由にシステムに登録できるようにした．

3.4 スタンプの認識

システムでスタンプ入力と判断された履歴は，キー座標の集合から特徴量を抽出して識別する．抽出する特徴量は，履歴のキーの数と，それらの X 座標，Y 座標それぞれの平均，分散，幅の 7 つである．あらかじめ特徴量と正解のスタンプの組をシステムに学習させてお

Vol.2009-HCI-134 No.4
2009/7/16

(a) グー　(b) パー　(c) チョップ

図 5　スタンプ入力の例
Fig. 5　Example of “stamp” input

き，未知のスタンプを認識する際は，その未知のスタンプの文字集合の特徴量と学習した特徴量のユークリッド距離を計算する．その後，k 近傍法 (k-nearest neighbor algorithm) 用いて推定された正解ラベルのスタンプが入力されたと判断し，その入力に応じたコマンドを出力する．k 近傍法は，入力データの近傍 k 個にある学習データ群の多数決によって決定するものである．

提案システムでは,「グー」(**図 5**(a)),「パー」(図 5(b)),「チョップ」(図 5(c)) という 3 種類のスタンプ入力を想定し，6 人の被験者からそれぞれの入力につき 30 個，計 540 個のデータを学習用に収集した．

4. 実　　装

提案システムを応用したプロトタイプシステムとして，コマンド入力と機能を関連付けるインタフェース (**図 6**) を実装した．このシステムでは，ユーザがその場で定義したコマンドを，あらかじめ作成された数十種類の機能の中から好きなものに関連付けて使用する．前述したように，ショートカットは「それぞれの機能に対応するキーを覚えておかなければいけない」,「ツールによってコマンドが異なる場合がある」などの理由からコンピュータの習熟度が低い利用者にとっては敷居の高いものになっている．それに対して提案システムでは，それぞれの機能に対するコマンドをその場で定義できるため，コンピュータの習熟度が低い利用者が，覚えられるだけのコマンドを登録して用いることができる．また，複数の動作を組み合わせた複雑な操作も，ひとつのコマンド入力で定義して実行することが可能であるため，コンピュータの操作に精通した利用者でも提案システムは有効である．

図 6　コマンド入力と機能を関連付けるインタフェース
Fig. 6　Screenshot of interface associating commands and functions

表 2　タイピングの評価
Table 2　Evaluation of typing

被験者	入力文字数	ストローク	スタンプ	識別アルゴリズム使用回数	タイムラグ発生文字数
被験者 1	214	1	0	6	46
被験者 2	242	0	1	6	45
被験者 3	140	0	3	8	48
被験者 4	253	1	12	18	92
被験者 5	202	0	2	7	32
被験者 6	198	0	0	12	48
合計	1249	2	18	57	311

5. 評　　価

提案システムの評価として，20 代の男性，女性の被験者 6 人から採取したデータを用いて入力の識別精度の解析を行った．

5.1 タイピング

被験者には自由に文章をタイピングしてもらった．各被験者の入力文字数とコマンド入力と誤認識した回数，前述したタイピングの早期検出の条件に当てはまらずにコマンド入力の識別アルゴリズムによってタイピングと判断された回数，入力してからのタイムラグが発生した文字数を調べたところ，結果は**表 2** のようになった．

表 2 より，平均して 100 タイピングする度に約 1.6 回の割合でシステムがタイピングをコマンド入力と誤認識することが分かる．そこで，システムが誤認識した履歴を解析したと

Vol.2009-HCI-134 No.4
2009/7/16

ころ，コマンド入力と判断された際のタイピングの履歴は

“asa”，“no”，“mi”，“kyou”，“ra”，“kyou”，“ki”，“nomo”，“no”，“kou”，“da”，“momo”，“dar”，“dara”，“te”，“iku”，“de”，“[BackSpace][¥]”，“rea”，“kono”

であり，これら 20 個の文字列のうち，それのみで単語となり得るのは半数の 10 個である．それ以外の誤った判断の原因は被験者のタイピングの癖が影響していると考えられる．また，前述したタイピングの早期検出の条件に当てはまらなかった回数は 57 回あるが，そのうち 50 回が一つのキーしかなかった場合であり，タイピングの早期検出は正常に動作したことが分かる．それ以外のタイムラグが発生した事例についても，そのほとんどが 2，3 キーが押される間にタイピングの早期検出が発生し，タイムラグは 0.3 秒以内に抑えられており，利用者に不快感を与えることはなかった．

タイピングをコマンド入力と誤認識することを防ぐための手段として，コマンド入力の識別アルゴリズムにかけるまでの待ち時間を長くする，という対策が考えられる．それにより，現在の待ち時間では発生しなかったタイピングの早期検出が発生する可能性が高まるが，一方で，1 つのキーしか入力されなかった場合のタイムラグが大きくなるというトレードオフが存在する．

5.2 ストローク

被験者にはストローク入力として,「丸」,「四角」,「三角」と各々が自由に定義したストロークコマンドの 4 種類を 3 回ずつシステムに学習させた後，それぞれのコマンドについて 20 回ずつ入力してもらってその精度を評価した．結果を**表 3** に示す．

表 3 より，全体の認識精度は 89.4%となり，高い精度で入力コマンドを区別することができた．また，ストロークの他のコマンドと判断を誤ったものの中には，爪をキーボードにひっかけたことによる入力ミスも含まれている．このことから，利用者がストロークの入力動作に慣れることで，そのような入力ミスがなくなり，認識精度はさらに向上すると考えられる．

5.3 スタンプ

被験者にはスタンプ入力として，前述した「グー」,「パー」,「チョップ」の 3 種類のスタンプ入力をそれぞれ 20 回ずつ入力してもらい，その認識精度を評価した．その結果を**表 4** に示す．

表 4 より全体の認識精度が 40.6%と低い値になった．さらに，その不正解の履歴の多くがタイピングと誤認識されていた．その理由として以下の 2 点があげられる．

- 十分な長さの履歴が取得できない

表 3 ストローク入力の評価
Table 3 Evaluation of “stroke” input

被験者	入力コマンド	正解数	不正解数 (ストローク)	不正解数 (タイピング or スタンプ)	認識精度
被験者 1	丸	14	6	0	70%
	四角	17	3	0	85%
	三角	17	2	1	85%
	波形「〜」	19	0	1	95%
被験者 2	丸	20	0	0	100%
	四角	14	5	1	70%
	三角	19	1	0	95%
	ガンマ「$gamma$」	18	2	0	90%
被験者 3	丸	20	0	0	100%
	四角	18	2	0	90%
	三角	19	1	0	95%
	無限大「∞」	20	0	0	100%
被験者 4	丸	17	1	2	85%
	四角	18	2	0	90%
	三角	20	0	0	100%
	チェック「$\surd$」	15	4	1	75%
被験者 5	丸	16	4	0	80%
	四角	15	3	2	75%
	三角	19	0	1	95%
	直線「—」	20	0	0	100%
被験者 6	丸	20	0	0	100%
	四角	18	2	0	90%
	三角	20	0	0	100%
	ひし形「$\diamond$」	16	3	1	80%
全体	——	429	49	10	89.4%

「グー」の入力時に，それをタイピングと誤認識された場合の履歴を調べたところ，その約 70% が 1 キーしか取得できていなかった．提案システムでは 1 キーの履歴は全てタイピングとして扱われる．

- タイピングの早期検出の誤動作

「パー」と「チョップ」の入力時の誤認識の約 95%,「グー」の入力時の誤認識の約 25%がこれにあたる．これより，タイピングと判断する値の設定ミスが考えられ，もう一度閾値を見直す必要がある．それ以外にも，キーボードの内部の電気回路的な制約から，ある特定のキーの組が押された時に実際には押されていないキーが押されたというイベントが発生するが，このこともタイピングの早期検出の誤動作を招いた要因の一つだと考

表 4 スタンプ入力の評価
Table 4 Evaluation of "stamp" input

被験者	入力コマンド	正解数	不正解数 (スタンプ)	不正解数 (タイピング or ストローク	認識精度
被験者 1	グー	6	5	9	30%
	パー	6	1	13	30%
	チョップ	6	1	13	30%
被験者 2	グー	13	1	6	65%
	パー	5	6	9	25%
	チョップ	15	0	5	75%
被験者 3	グー	12	5	3	60%
	パー	6	5	9	30%
	チョップ	13	3	4	65%
被験者 4	グー	5	10	5	25%
	パー	4	2	14	20%
	チョップ	4	2	14	20%
被験者 5	グー	7	8	5	35%
	パー	7	0	13	35%
	チョップ	4	3	13	20%
被験者 6	グー	14	2	4	70%
	パー	8	4	8	40%
	チョップ	11	2	7	55%
全体	——	146	60	154	40.6%

えられる．
これらの問題を解決するためには，キーイベントの履歴以外の要素を使うことが有効になる可能性がある．具体的には，打鍵圧の値などが考えられる．ExpressiveTyping[4] では，ノートパソコンが落下時にハードディスクを緊急停止させるために用いる内蔵型加速度センサの値を用いて打鍵圧を取得することに成功している．本研究においても，加速度センサを用いることでスタンプの衝撃を検知でき，それによりスタンプの検出精度が向上すると考えられる．

6. 考　　察

本章では，提案システムの実用例，応用例を以下に示す．

6.1 他アプリケーションとの連携

提案システムは設定した任意のウィンドウメッセージを送信する機能を備えているため，ウィンドウメッセージの受け皿を作るだけで，提案するコマンド入力を組み込んだアプリケーションを作成可能である．具体的には，スタンプ入力を用いたもぐらたたきや，ストローク入力の軌跡を刀の太刀筋と見立てたゲームなど，直観的な操作を伴うプログラム，あるいは，パワーポイントなど資料作成の際にストロークで描いた図形を挿入するマクロなどに簡単に組み込むことができる．

6.2 認証への応用

認証の手法として，現在一般に普及しているのは暗証番号やパスワードである．それ以外にも，指紋センサなど生体情報を用いて認証する試みもみられるが，新たにハードウェアが必要であることや，プライバシーの問題により広く普及するには至っていない．文献 5), 6) では，あるキーを打ってから次のキーを打つまでの打鍵時間情報や打鍵圧には個人差があり，暗証番号やパスワードに加えてこれらの情報を利用することで，より高度なセキュリティをもつ認証が可能になる．そして，この認証方法にはプライバシーに関わる生体情報ではなく，後天的な情報を用いること，万が一他人に知られたとしても，筆跡と同じように簡単に真似ることが困難であることなどの利点がある．それを応用して，ストローク入力の認識を活用して，キーボード上をなぞり，あたかもサインをしているような感覚で利用する認証に用いることもできるだろう．

7. おわりに

本研究では，従来の文字入力である「タイピング」に加えて「ストローク」と「スタンプ」という 2 種類のコマンド入力を提案し，特別な装置を用いることなく，文字入力中にシームレスにコマンド入力が行える方式を構築した．提案方式は，キー入力のイベントの履歴を保存し，キーボードの入力特性を利用して，一般の文字入力としてのキー入力か 2 種類のコマンド入力のどちらであるかを判定する．本研究で構築したシステムの 6 人の被験者による評価から，ストロークは非常に高い精度で識別できることが分かり，スタンプについても認識精度を上げるための要因がいくつか見つかった．

今後の課題として，認識精度向上のための改良や，提案システムを活用したアプリケーションの実装などを行う予定である．

参 考 文 献

1) 塚田 有人, 星野 剛史: Pointing Keyboard: キー入力／ポインティングが可能な入力デバイス, 第 10 回インタラクティブシステムとソフトウェアに関するワークショップ (WISS 2002) (2002).

Vol.2009-HCI-134 No.4
2009/7/16

2) FingerWorks, Inc: TouchStream Stealth, http://www.fingerworks.com.
3) J. Rekimoto: ThumbSense: Automatic Mode Sensing for Touchpad-based Interactions, CHI 2003 extended abstracts on Human factors in computing systems, pp. 852-853 (2003).
4) K. Iwasaki, T. Miyaki, and J. Rekimoto: Expressive Typing: A New Way to Sense Typing Pressure and Its Applications, ACM CHI2009 work in progress, pp.4369-4374 (2009).
5) 北川高雄, 宮川裕之, 古谷野英一: 本人識別のための打鍵圧および打鍵間隔時間の有用性に関する研究, 日本経営工学会誌, Vol.39, No.6 p. 415 (1989).
6) R. Joyce, G. Gupta: Identity Authentication Based on Keystroke Latencies, Communications of the ACM, Vol.33, Issue 2 pp. 168-176 (1989).