KooSHO - 空中での手書きジェスチャーに基づく日本語入力環境

萩原　正人　　益子　宗
楽天株式会社楽天技術研究所
{masato.hagiwara, so.masuko}@mail.rakuten.com

1　はじめに

スマートフォン, タブレット, ヘッドマウント型コンピュータなどの新たな携帯機器の出現に伴い, 直感的かつシームレスなインターフェースであるいわゆる NUI (Natural User Interface) の重要性が高まっている. 特に, 直感的で自然なテキスト入力は快適なユーザー体験を実現するために欠かせず, 従来のキーボードおよびマウスに替わる様々な入力法が提案されてきた.

今日, 携帯端末において広く利用されている入力法のひとつに音声認識があり, 米アップル社の *Siri* をはじめとするいわゆるコンシェルジュサービスの UI として広く使われている. しかしながら, 周囲の雑音に弱く, 周囲に入力内容が漏れてしまうという問題もあり, 実用にあたってはプライバシーの課題が残る. 仮想キーボード[1]を使うこともできるが, 操作が難しく, 高速に入力した場合に誤りを起こしやすいという欠点がある.

本稿では, 人間にとって直感的な方法である手書き入力に注目する. 現実空間への描画を可能とするペン型入力装置 [10] や, 磁石を用いた装着型手書き入力装置 [2] など, 数多くの手書き入力デバイスが提案されている. しかし, これらのシステムでは利用者が手袋などを装着する必要があり, 特にモバイル環境においては負担となり得る. そこで, 特殊な装置の装着を必要としない手法, 例えば手書きジェスチャー動画に基づきアルファベットを認識するシステム [8] や, 可視光と赤外線照射による画像に基づく手書き文字入力システム [12] が提案されている. これらを含む従来研究のほとんどが単一の文字の入力しか扱っていないが, 単語, フレーズ, そして文の入力における総合的なユーザー体験も, 特に日本語のようなかな漢字変換を伴う言語においては実用上重要である.

そこで本稿では, 空中での手書きジェスチャーに基づく日本語入力環境 KooSHO を提案する. KooSHO は, 検索エンジンのクエリ入力に焦点を当てており, 文字認識, かな漢字変換, 検索結果の表示を統合したユーザー体験を実現する. 筆者の知る限りでは, 手書きジェスチャーに基づき, 単一の文字入力以上の日本語入力環境を実現したのは本システムが初めてである.

図 1 に, KooSHO を用いて日本語テキストを入力する過程を示す. まず, (a) ユーザーはアルファベットの形を空中に描く. この際, 手の位置は Microsoft Kinect[2]を用いて認識する. KooSHO システムは手の軌跡から文字を認識し, かな漢字変換を経て, その結果をユーザーの肩を中心とする円形状に表示する (同図 b). 最後に, 選ばれた単語・フレーズをクエリとして検索された結果 (例えばウェブ文書, 商品など) を, 同様に円形状に表示する (同図 c) . ユーザーは, 検索結果に触れることで文書を選ぶことも, 入力を続け, さらに長いフレーズを作ることもできる.

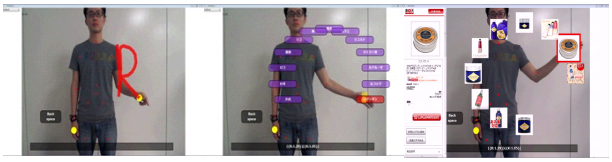

図 1: KooSHO のテキスト入力過程 — (a) 文字の入力 (b) かな漢字変換結果の表示 (c) 検索結果の表示

KooSHO システムは, シームレスかつ頑健な日本語入力を実現するために以下のような特徴を備えている:

自由文字形 Graffiti 2 [4] やその変種 [8] など, 手書き認識のための独自の文字形を使う必要がなく, キーボードや各種入力デバイスに不慣れなユーザーであっても訓練無しで自然・直感的な入力が実現できる.

頑健な認識・変換 空中での手書き文字認識では, 各文字および字画の開始・終了点を取得できないため, 一部の文字対, 例えば “K” と “R” などの識別が困難である. これに対して KooSHO では, システムが用いられる特定の分野において訓練された識別的かな漢字変換モデルを採用することにより, 当該分野内において妥当な候補のみを優先的に表示する.

サジェスチョンおよび子音入力による高速な入力 ジェスチャーに基づく入力では, キーボード等と比べ入力に長い時間がかかる. KooSHO システムでは, 第一に, ユーザーが入力しそうな単語・フレーズをサジェスチョンとして提案することにより, 入力速度の改善を図る. 例えば, ユーザーが “H” と入力した後, 「本体」「本皮」などが提案される. また, 第二に, ローマ字の母音を省略して子音のみを用いたかな漢字変換も可能とする [6]. 例えば, 「福袋」を入力するために, “HKBKR” と入力することができる. その後, KooSHO ではビタビサーチを入力子音列に対して直接適用し, かなを経ずにかな漢字混じり文へと変換する.

本稿では, 本システムの性能を評価するために文字認識とかな漢字変換の精度を測定した. また, 音声認識ソフトウェア *Siri* , 仮想キーボードと本システムを比較することにより総合的なユーザー体験を評価した.

[1] http://www.youtube.com/watch?v=h9htRy0-sUw
[2] http://www.microsoft.com/en-us/kinectforwindows/

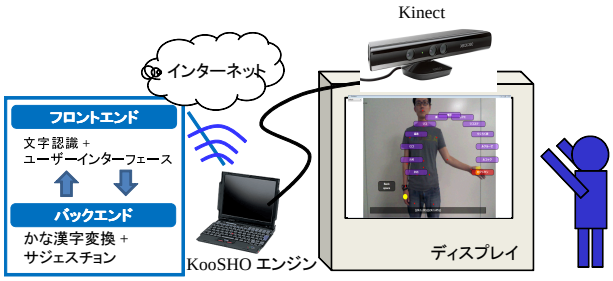

図 2: KooSHO システムの構成

B 7,7,7,7,7,7,7,6,4,4,4,4,2,1,1,1,1,1,1,0,1,1,1,1,1,2,2,2,2,5,5,5,5,8,8,7,7,
7,7,6,6,6,6,6,4,4,4,4,5,5,5,5,8,8,8,7,7,7,7,6,6,6,6,6,6,6,6,3,3,3,3,4,4,4,4,
4,4,4,4,4,4,4,4,4,4,4,4

H 4,7,7,7,7,7,7,7,8,8,7,7,7,7,7,6,7,4,4,4,4,1,2,2,2,1,1,1,1,1,1,1,1,1,4,4,4,5,
5,5,5,5,5,5,5,5,5,8,5,5,4,4,4,4,1,0,0,0,0,1,0,1,1,1,0,0,1,4,4,4,4,4,7,8,8,8,7,
7,7,7,7,7,7,6,6,7,6,4

K 7,7,7,7,7,7,7,6,6,6,7,6,7,7,7,4,4,4,4,2,2,2,2,2,2,2,2,2,2,2,2,2,2,4,4,4,4,6,
6,6,6,6,6,6,3,7,4,4,7,7,8,8,8,8,8,8,8,5,8,8,8,8,8,8

R 7,7,7,7,7,6,7,6,6,4,4,4,4,2,1,1,2,2,1,1,1,1,1,0,1,1,1,4,1,2,5,5,5,5,8,8,8,7,7,
7,7,6,6,6,6,6,6,3,3,3,3,3,4,4,4,4,4,5,8,8,8,8,8,8,8

↖:0 ↑:1 ↗:2 ←:3 ●:4 →:5 ↙:6 ↓:7 ↘:8

図 3: 素性表現したテンプレートの例

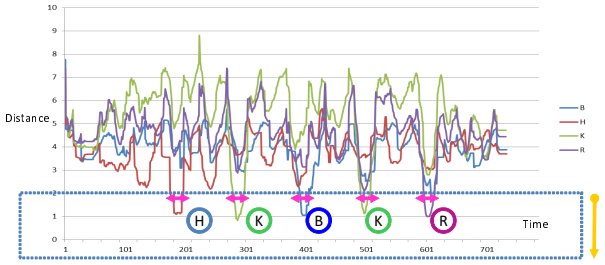

図 4: 入力 “HKBKR” におけるテンプレートとの距離の変化

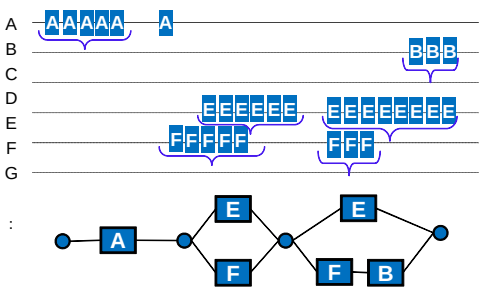

図 5: 文字のまとめ上げとその結果のラティスの例

2 文字認識

図 2に，KooSHO システムの構成を示す．ユーザーが空中に描いた軌跡は Kinect によって認識され，KooSHO システムへと送られる．本節では，文字認識および UI を担うフロントエンド部について述べる．

Kinect センサーは可視光カメラと赤外線レーザー深度センサを搭載しており，SDK に含まれる標準機能を使い，人体の骨格認識が容易に実現可能である．KooSHO システムでは，手の座標を二次元正規化平面に投影することにより x-y 座標を取得し，10 フレーム幅のウインドウ内における座標の中央値を計算し，より滑らかな手の軌跡が得られるようにした[3]．

文字認識は，ユーザーの描いた軌跡と，文字のテンプレートを連続的にマッチングさせることによって実現する．テンプレートは，あらかじめ教師データとして与えられた各アルファベットの軌跡であり，図 3 のような形で表される．入力された軌跡とテンプレートは，移動方向を 8 方向＋停留の 9 種類に離散化して符号化される．前フレームとの座標差異が 3 ピクセルより大きければ移動，それ以外は停留とみなす．この符号化方式は，文字のスケーリングや，描写速度の多少な変化に対して頑健であるが，筆順の変化に対しては頑健ではない．なお，図 2 のように正面からの撮影に加え，例えばヘッドマウント型装置からの撮影も，テンプレートを同じ条件で作成しさえすれば対応可能である．

符号化した軌跡は，各文字のテンプレートと DP マッチングにより照合する．この処理は，文字の置き換えのみを許可した編集距離の計算とほぼ同様であるが，置き換え処理のコストは似た方向間が小さくなるように設定した．具体的には，同一の方向はコスト 0，隣接する方向（例えば上と右上など）はコスト 1，反対の方向（右上と左下など）はコスト 3，それ以外はコスト 2 と定義した．停留と全ての方向に対するコストは 1 に設定した．

毎フレームごとに DP マッチングを実行すると，そのフレームで終わる軌跡と各テンプレートに対する距離を得ることができる (図 4)．距離がある閾値（2.0 に設定）を下回った時，その文字が入力されたとみなす．

なお，このマッチング方法では，距離がある閾値を下回っている間はずっとその文字が認識され続ける．また，例えば文字 “E” の中の “F” のように，ある文字が他の文字の字画を含んでいる場合，一時的に両方の文字が検出される場合がある．この問題に対しては，3 フレーム以上連続して検出された文字のみをまとめ上げることで対処する．また，同時に 2 種類以上の文字が検出された場合，それらは認識候補としてすべてバックエンドに送られ，かな漢字変換の対象となる．この文字認識結果は，文字の連接と選択からなるラティス構造として表現される（図 5）．各文字には，テンプレートと軌跡の距離の逆数として計算されるスコアが割り当てられ，ラティス内のある経路のスコアは，その経路上に含まれる各文字のスコアの積として計算される．この経路のスコアを以下では文字認識スコアと呼ぶ．

3 かな漢字変換

本節では，入力された子音をかな漢字混じり文へと変換するかな漢字変換手法（バックエンド）について述べる．前述の通り，バックエンドへと送られるのは子音列[4]からなるラティス構造であるため，省略された母音を推測し，かな漢字混じり文を復元する必要がある．

単純には，入力された子音列を生成するあらゆる

[3]なお，現在のところ Kinect センサーは大型であり身体に装着するのは現実的ではないが（ただし，[3] のような例もある），例えば Project Glass https://plus.google.com/+projectglass/posts のような眼鏡型デバイスに KooSHO システムを組み込むことは十分現実的である．

[4]なお，ユーザーは（誤って，もしくは故意で）母音を省略せずに入力することもできる．

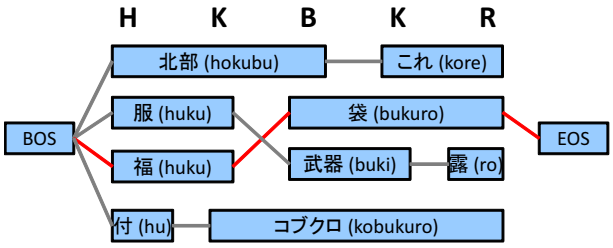

図 6: 子音に基づくビタビサーチの例

ユニグラム素性		バイグラム素性	
表層形	y_i	品詞バイグラム	c_{i-1}, c_i
表層形および品詞	y_i, c_i	表層形バイグラム	y_{i-1}, y_i
品詞	c_i		
表層形および読み	y_i, x_i		

表 1: かな漢字変換に用いた素性テンプレート

x_i, y_i, c_i は位置 i における表層形，読み，品詞を表す．

可能なカナ列に展開する方法が考えられる．例えば，入力された子音列が “HKBKR” であれば，「ハカバカラ」「ハカバカリ」... というように可能な母音を補完して展開する．しかし，各子音に対して後続する母音は通常 5 種類あり，この例では入力は少なくとも $5^5 = 3125$ 種類のカナ列に展開される[5]．文字認識結果の子音のラティス構造と組み合わせると，可能なカナ列の数に関して組み合わせ爆発が容易に起こるため，この展開手法は現実的ではない．

実際，このようにして展開された大量のカナ列のほとんどが日本語として意味をなさないものであり，かな漢字変換の順位付けの際に淘汰されてしまう．そこで，全ての可能なカナ列を列挙するのではなく，入力の子音列を直接復号化し，かな漢字混じり文を得る手法を考える．そのために，変換モデルとして識別的かな漢字変換モデル [7] に変更を加えたものを用いる．子音に対して適用されるという点以外は，日本語の形態素解析 [11] に用いられる通常のビタビサーチと同様である[6]．ビタビサーチでは，以下のように解析が進む：1) 入力の各文字位置から始まる辞書見出し語を列挙する 2) 見つかった見出し語に対応する節点を持つラティス構造を構築する．現在注目している節点から，直前の隣接する節点のうち，スコアを最大とするようなものを辺で繋ぐ．ここでのスコアは，各節点および辺に対して計算される素性 (表 1) に基づき計算される．最後に，3) 構築したラティスを文の最後から逆に辿ることにより，スコアが最大となる経路を取得できる．図 6 に，構築されたラティス構造およびスコアが最大となる経路の例を示す．なお，代わりに各節点に対して上位 N 個の後方節点を持たせることにより，上位 N 個の解を得ることができる．

[5]実際には，例えば「シ」は “si” とも “shi” とも綴られるので，“SH” という入力に対しては「シ」に加え，サ行とハ行の組み合わせ（例：「ソフ」）の両方の可能性があり，これが組み合わせ爆発をより一層深刻にしている．

[6]本システムでは，特別な未知語モデルは使用せず，単一の文字からなる見出し語を全て用意することにより，どのような入力に対しても解が出力されるようにした．

KooSHO で用いたかな漢字変換の違いは，子音列から直接見出し語を検索できるようにすることである．辞書構造をメモリにロードする際に見出し語を可能な子音の列に展開することによりこれを実現できる．例えば，「福袋」という見出し語であれば，“hkbkr,” “hukbkr,” “hkubkr,” “hukubkr,” “hkbukr,”... という子音（＋母音）の列が索引構造（通常はトライによって実現）に格納される．カナ 1 文字に対して生成される候補は母音あり／なしの高々 2 種類であるため，この処理によって組み合わせ爆発が起きることは少なく，ほとんどの場合，候補は 1,000 個以下に収まる．「シ」を “si” と “shi” に展開するなどの表記ゆれの処理もこの段階で行われる．カナをローマ字に変換するには，Microsoft IME のローマ字入力表[7] を用いた．なお，「ca→ カ」や「qu→ ク」などの，他のローマ字で代用可能でありかつ使用頻度の低いローマ字は，展開の効率のため取り除いた．

4 候補サジェスチョン

候補サジェスチョン機能は，ユーザーの現在の部分的な入力から，入力されそうな単語・フレーズを予測し候補を提示するものである．本システムでは，楽天市場[8] の 2011 年における頻度上位 2000 検索キーワードを用いた．各キーワードに読みを付与し，前節で述べたのと同様の手法で可能な子音列へと変換し，サジェスチョン用トライ構造へと格納する．実行時には，入力を用いてこのトライを接頭辞検索することにより候補を列挙する．候補は検索頻度によりソートされ，かな漢字変換の結果と共に提示される．かな漢字変換結果と候補サジェスチョンの最終的なスコアは，3 節で述べた文字認識スコアによって重み付けされ，順位付けされる．

5 評価実験

3 節で述べた素性の重みの学習には，平均化パーセプトロン [1]（繰り返し数 3）を用いた．品詞としては，UniDic [9]の上位 3 階層（例：名詞 -固有名詞 -人名）を用いた．訓練コーパスとしては，以下の 3 つを組み合わせて用いた 1) 現代日本語書き言葉均衡コーパス (BCCWJ) の CORE サブコーパス．60,374 文，1,286,899 形態素からなる [9]．2) EC（電子商取引）コーパス - 楽天市場から抽出した商品名および商品説明文 1,230 項目．上記 BCCWJ と同じ基準により形態素分割した．118,355 形態素からなる．3) EC クエリログ - 4 節で述べた頻度上位 2,000 検索キーワードを，2 と同様に形態素分割したもの．辞書には UniDic を用いた．

文字認識 まず，文字認識モデルの精度を評価した．A から Z までのそれぞれの文字について，2 人の被験者が 3 回ずつ入力を試み，文字が正しく認識される割合を計算した．結果として，文字の認識精度は文字によって大きく異なることが分かった．アルファベット中に類似した字型の無い文字，例えば A, H, S, Z などについては，精度は 100% であった一方，D, I, P などについては精度は 50% 以下に低下した．これらはそれぞれ

[7]http://support.microsoft.com/kb/883232/ja
[8]http://www.rakuten.co.jp/
[9]http://www.tokuteicorpus.jp/dist/

P, J, R などとして誤認識されることが多かった. 全体での文字認識精度は 76% であった.

かな漢字変換 次に, 子音列を入力とし, かな漢字混じり文を出力とするかな漢字変換の精度を評価した. 評価には, ACC (平均精度), MFS (平均 F 値), MRR (平均順序逆数) の 3 つを用いた. それぞれの正式な定義は文献 [5] に譲るが, 直感的には, ACC は候補の最上位が正解と一致する割合, MFS は, 候補の中で最も正解と近いものが, どのぐらい正解と一致しているかを文字単位で評価したもの, MRR は正解が平均して候補の n 位に見つかる時に $1/n$ となるような指標である.

テストセットとして, 楽天市場の商品名およびクエリログからランダムに選択した単語・フレーズ 100 個を用いた. このテストセットは訓練コーパスとは互いに素である. その結果, ACC = 0.24, MRF = 0.50, MRR = 0.30 という結果であった. この評価では, 部分ごとの入力を許可していないため, 実際よりは厳しい評価となっている. 例えば, 「フィットネスシューズ」は最上位に現れなかったが, 「フィットネス」「シューズ」と分割して入力することにより変換できる. また「まつげ」と「まつ毛」などの表記ゆれが原因で正解とならなかった場合があるが, このような多少のゆれは実用では問題無い場合がある.

総合評価 最後に, *Siri*, KinEmote [10]を用いて制御した仮想キーボード（Tablet PC 入力パネル）, そして KooSHO を, 精度, 入力速度, ユーザー体験の観点から総合的に評価する.

まず, 上記 100 個の単語・フレーズからなるテストセットを用いて, *Siri* を評価した. その結果, 認識精度は 85% であり, 成功した場合には3～4秒以内に認識された. ただし, 14% のクエリについては何回か試行した後も認識させることができなかった. このようなクエリの一つは, 「オーガランド」のような新語・未知語であり, この認識はシステムの言語モデル, 語彙サイズに依存する. もう一つは, 「放送」「包装」と「ミョウバン」「明晩」などの同音異義語であり, これは文脈を与えるかもしくは上位 N 解の選択を許さなければ解決が難しい. これは, KooSHO のような視覚的なフィードバックが有効に働く場合である.

次に, Kinect により制御した仮想キーボードを評価した. Kinect を用いてキーボードを制御する場合, 目的のキーに手を置くのに細かい動作を強いられるため, 非常に難しく, 実用的な使用はほぼ不可能であることが分かった.

最後に, KooSHO システムを使ってクエリを完成させるのに要する時間を測定した. 時間はクエリによって大きく異なり, “C” などの認識率の低い文字を含む単語, 例えば「チーズ」については入力時間が長くかかった. 平均で入力に要した時間は 35 秒弱であった.

6 おわりに

本稿では, 空中での手書きジェスチャーに基づく日本語入力環境 KooSHO を提案した. KooSHO は, 手書きジェスチャーに基づき, 単一の文字入力を越える日本語入力を実現した最初のシステムであり, 自由文字形, 頑健な認識・変換, サジェスチョンおよび子音入力による高速入力が可能であるという特徴を持つ. 評価実験の結果, クエリ入力という文脈において, KooSHO は音声認識や仮想キーボードと比べてより実用的である可能性を示した.

なお, KooSHO システムの文字認識は体の回転や筆順の変化に対して頑健ではない. 前者に対しては, 体の面に並行な平面を考え, 軌跡をその平面上に投影することにより解決できる. また, 後者に対しては, より多くのユーザーから, 多様な字型や筆順を持つテンプレートを収集することにより対応可能である.

KooSHO は日本語入力環境であるため, 近代的な入力メソッド (IME) の備える機能, 例えば, ユーザー入力からの学習機能やスペルミスを含む入力の変換などがそのまま応用できる可能性がある. 特に, 入力・変換誤りや文節区切り等の訂正をどのように NUI 環境において実現するかは今後の課題である.

[10]http://www.kinemote.net/

参考文献

[1] Michael Collins. Discriminative training methods for hidden markov models: theory and experiments with perceptron algorithms. In *Proc. of ACL*, pp. 1–8, 2002.

[2] Xinying Han, Hiroaki Seki, Yoshitsugu kamiya, and Masatoshi Hikizu. Wearable handwriting input device using magnetic field. In *Proc. of SICE*, pp. 365–368, 2007.

[3] Shota Kaneko and Jiro Tanaka. Building collaborative work environment using augmented reality (in japanese). In *Proc. of the 74th Annual Convention IPS Japan*, 2012.

[4] Thomas Költringer and Thomas Grechenig. Comparing the immediate usability of graffiti 2 and virtual keyboard. In *Proc. of CHI 2004*, pp. 1175–1178, 2004.

[5] Haizhou Li, A Kumaran, Vladimir Pervouchine, and Min Zhang. Report of news 2009 machine transliteration shared task. In *Proc. of NEWS*, pp. 1–18, 2009.

[6] Kumiko Tanaka-Ishii, Yusuke Inutsuka, and Masato Takeichi. Japanese text input system with digits. In *Proc. of HLT*, pp. 1–8, 2001.

[7] Hiroyuki Tokunaga, Daisuke Okanohara, and Shinsuke Mori. Discriminative method for japanese kana-kanji input method. In *Proc. of WTIM*, 2011.

[8] Ho-Sub Yoon, Jung Soh, Byung-Woo Min, and Hyun Seung Yang. Recognition of alphabetical hand gestures using hidden markov model. *IEICE Trans. Fundamentals*, Vol. E82-A, No. 7, pp. 1358–1366, 1999.

[9] 前川喜久雄. Kotonoha『現代日本語書き言葉均衡コーパス』の開発. 日本語の研究, Vol. 4, No. 1, pp. 82–95, 2008.

[10] 山本吉伸, 椎尾一郎. 空気ペン一空間への描画による情報共有一. 情報処理学会第 59 回全国大会講演論文集, pp. 3ZA–3, 1999.

[11] 工藤拓, 山本薫, 松本裕治. Conditional random fields を用いた日本語形態素解析. 情報処理学会研究報告, NL161 巻, 2004.

[12] 園田智也, 村岡洋一. 空中での手書き文字入力システム. 電子情報通信学会論文誌. D-II, Vol. J86-D-II, pp. 1015–1025, 2003.