Vol.2010-HCI-137 No.11
2010/3/19

多項目パイメニューにおける入力特性を考慮した領域割当てアルゴリズム

宮本雅勝[†1]　寺田努[†1]　塚本昌彦[†1]

近年，省スペースな環境での入力手法としてペンとパイメニューを組み合わせた方式が提案されているが，文字入力などの多項目選択に適していないものが多い．そこで筆者らの研究グループではこれまで，多数項目を選択できる入力手法としてアナログジョイスティックと階層型パイメニューを組み合わせたインタフェースを提案してきた．本研究では，このインタフェースにおいてより直観的で効率的な入力を目指し，選択項目の使用頻度と，パイメニュー内でのアナログジョイスティックの入力の特性に基づいた領域割当てアルゴリズムを提案する．提案するアルゴリズムは，評価実験により有効であることを確認した．

An Area Allocation Algorithm for Hierarchical Pie Menu

Masakatsu Miyamoto,[†1] Tsutomu Terada[†1]
and Masahiko Tsukamoto[†1]

Recently, there are several input methods that combine the pen-based input with the pie menu as input interfaces in the environments where working space is limited. We have proposed a new space-saving input interface that combines an analog joystick and a hierarchical pie menu because conventional interfaces have problems in the case where there are many items in the menu. In this research, we propose new area allocation algorithm for input items based on the frequency and input characteristic. Proposed method allocate an appropriate area to an item in the pie menu. The evaluation results confirmed that an intuitive input interface is achieved with high-speed and high accuracy.

†1 神戸大学
Kobe University

1. はじめに

近年，コンピュータの小型化，高性能化により，コンビニでの商品管理や，医療現場でのカルテ入力など様々な環境でコンピュータが使用されている．

これらの環境に適した入力手法として，ペンでタッチした部分の周囲だけで操作できるパイメニューを用いたインタフェースに関して多くの研究が行われている[1),2)]．パイメニューとは，図 1 に示すように選択項目を円周上に配置したメニューであり，選択項目が全て中心から等距離にあるため，ペンでタッチした位置から近い距離だけで操作を行える．直観的でわかりやすいメニュー配置をしており，少数のメニュー項目を選択する場合に素早い選択が行えるインタフェースである．

一方，この手法を応用して，文字入力などの多項目選択を行うインタフェースの研究も行われている[3),7),8)]．しかし，従来のペンとパイメニューを組み合わせた多項目入力インタフェースは，選択項目を増やした場合，入力が遅くなったり精度が悪くなるといった問題が生じるため，文字入力などの多数の選択項目への対応に適していなかった．

そこで筆者らの研究グループではこれまで，多数項目を選択できる入力手法としてアナログジョイスティックと階層型パイメニューを組み合わせたインタフェースを提案してきた．本研究では，このインタフェースにおいてより直観的で効率的な入力を目指し，選択項目の使用頻度と，パイメニュー内でのアナログジョイスティックの入力の特性に基づいた領域割当てアルゴリズムを提案する．提案するアルゴリズムを用いることで，選択項目の使用頻度に応じてパイメニュー内の適切な位置に適切な領域を割り当てることができ，メニュー項目が増加した場合でも高速かつ高精度な入力が実現できる．アナログジョイスティックの入力特性とは，アナログジョイスティックの傾けやすい方向と，入力しやすい位置や領域パターンを数値化したものである．提案するアルゴリズムは，評価実験により有効であることを確認した．

以下，2 章で関連研究について述べ，3 章ではインタフェースの設計について述べる．4 章では，パイメニューの入力特性を評価し，5 章で領域割当てアルゴリズムについて述べる．6 章で評価と考察を行い，最後に 7 章で本論文のまとめを行う．

2. 関連研究

ペンとパイメニューを組み合わせた文字入力インタフェースに関して多くの研究が行われている．

Vol.2010-HCI-137 No.11
2010/3/19

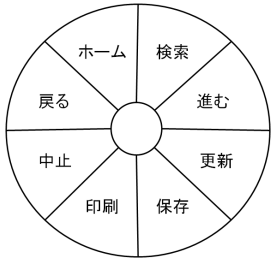

図 1 パイメニューの例
Fig. 1 An example of a pie menu

Venolia らによる T-Cube は 1 ストロークで 1 文字を入力する方式で，パイメニューのある項目をペンでタッチするとタッチした箇所に新たなパイメニューが表示され，入力したい文字の方向へペンを滑らせることで項目を選択する[3]．また，T-Cube を平仮名に対応させたインタフェースとしてかな T-Cube がある[4]．パイメニューは 8 分割から 12 分割の間に急激にエラー率が上昇するとされているため T-Cube で表示されるパイメニューは全て 8 分割の構造をしている[5]．しかしこの方式では，2 段階の操作を必要とするため入力速度が遅くなり，習得にも時間がかかる．

また，1 ストロークで複数文字を入力する手法として FlowMenu がある[6]．FlowMenu はパイメニューの各領域をペンで横切り，元の位置に戻るようにストロークを描くことで複数の文字を入力する．ペンでのタップ操作を必要としないため，タップミスの問題を解決している．Perlin らによる Quikwriting は，中心領域からペンで領域間を移動し，再び中心に戻ってくることで文字を入力するインタフェースである[7]．しかし，1 つの文字を選択する際のストロークが複雑であり，操作の習得に時間がかかるという問題点がある．Mankoff らによる Cirrin は，円周を 26 分割して，各マスにアルファベットを配置し，ペンが通過した部分のアルファベットを入力できる[8]．だが，1 つ 1 つの文字が表示されている領域が狭く，隣の領域を通過してしまうなどの選択ミスが起こる可能性がある．

3. アナログジョイスティックとパイメニューを用いたインタフェース

これまでに述べたように，パイメニューは選択項目が環状に配置されたメニューで，少数の選択項目の中から意図した項目を素早く選択できる．また，項目が円の中心を基準として配置されていることから，操作するデバイスの移動量が少なくてすみ，十分な場所を確保できない環境でも操作が行える．パイメニューのそれぞれのメニュー項目は，サイズを十分大きくとるべきであるが[9]，多数の選択項目が存在する場合，十分な領域をそれぞれ項目に割り当てることができない．そのため，パイメニューを応用して多数の選択肢に対応させる場合，入力速度が遅くなってしまう問題や，ペンでのストロークを覚える必要が生じ，操作の習得に時間がかかっていた．

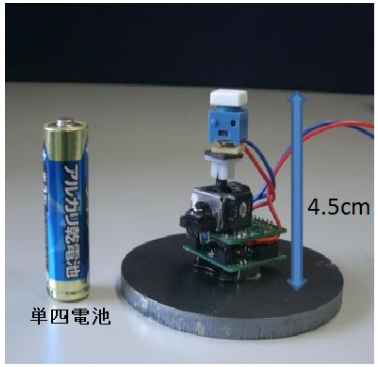

図 2 ハードウェア
Fig. 2 A hardware of analog joystick.

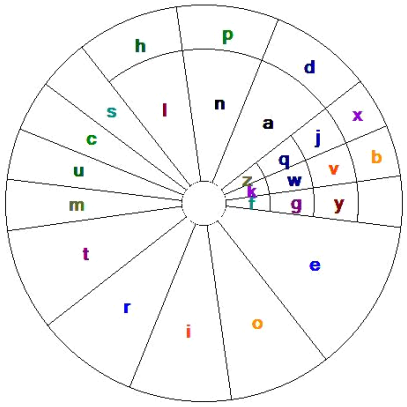

図 3 領域割当ての例
Fig. 3 An example of area Allocations.

そこで，筆者らの研究グループではこれまで，多数の選択項目に対応し，スペースが制限された環境で高速かつ高精度な入力を実現するインタフェースとして，可変領域をもつ階層化されたパイメニューインタフェースを提案してきた (図 2，3)[10]．提案したインタフェースの特徴を以下に示す．

- 階層化されたメニュー項目
- アナログジョイスティックの採用
- 項目の使用頻度に基づく領域割当てアルゴリズム

提案する階層型パイメニューは，角度を 24 分割，4 つの円を重ねあわせた 96 個の領域持つとする．円を重ねて同一方向に 4 つの領域がある部分を行，角度で 24 分割し，同一円周上に 24 個の領域がある部分を列と定義し，96 個の領域は 4 行 24 列の領域で形成される．

Vol.2010-HCI-137 No.11
2010/3/19

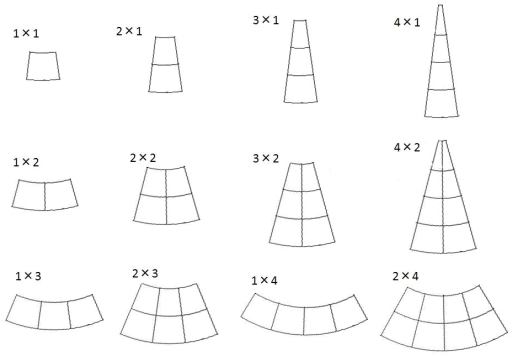

図 4 項目に与える領域のパターン
Fig. 4 Possible number of areas for each item.

これにより，同一方向に数種類の選択領域を与えることができ，多くの選択肢の中から 1 段階の操作で意図した項目を選択できる．また，図 2 に示すアナログジョイスティックを図 3 に示すパイメニューと組み合わせることで，アナログジョイスティックの傾ける方向と量を調節し，ジョイスティックの特性を利用した単純な入力が可能であり，省スペースな入力インタフェースが実現できる．さらに，選択項目数が増えた場合に全ての項目に対して均等な領域を与えると 1 つ 1 つの領域が小さくなるため，領域サイズを可変にし，項目に適切に割り当てられるようにしている．従来手法では，選択項目の使用頻度に応じた領域割当てを行っていたが，アナログジョイスティックには入力しやすい方向や領域パターンがあることがわかっており[11]，この特性を考慮した領域割当てを行うことでより高速に精度よく入力が行えると考えられる．

4. 入力特性評価

パイメニュー内のアナログジョイスティックの入力特性を調べるために，各領域パターンにおいて，配置される位置に応じて入力のしやすさがどう変化するかという特性を調べた．

アナログジョイスティックには，傾けやすい方向と傾けにくい方向があり，パイメニュー上に配置されている項目の位置や大きさ，領域の形によって，選択の難易度に差が生じると考えられる．そこで，領域の種類と位置における選択の容易さ (入力特性) を評価するために，特性評価実験を行った．実験に用いるパイメニューは，前章で述べたものと同様で，円

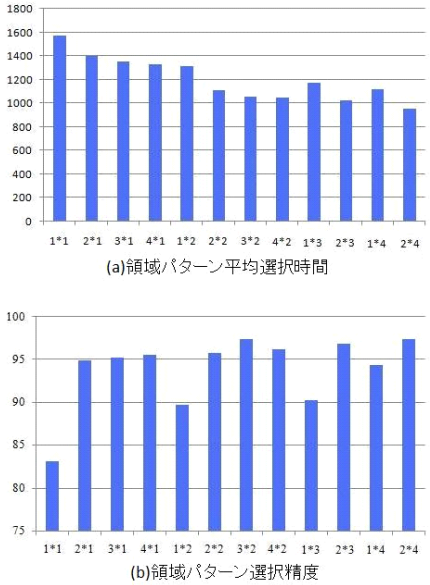

図 5 領域パターン別の平均選択時間と精度
Fig. 5 Input accuracy for each area.

を 4 段，角度の分割数を 24 に設定しており，96 個の領域をもたせている．

図 4 のように 1×1，1×2，1×3，1×4，2×1，2×2，2×3，2×4，3×1，3×2，4×1，4×2 の 12 種類の領域パターンを用意した．例えば，2×2 領域とは，2 行×3 行 から形成される領域である．実験は，この 12 種類の各領域パターンをランダムにパイメニュー内の全ての位置に表示させ，被験者にアナログジョイスティックを用いて表示された領域を選択させることで行った．

領域パターンはそれぞれ，パイメニュー上に表示されうる全ての位置で表示させた．例えば，図 4 の 2×2 は，中心からの距離別に 3 パターンの配置方法があり，角度別に 12 パターンあるため 36 通りのパターンがある．実験は，4 人の被験者に 2 回ずつ行わせ，全ての位置に表示される全ての領域パターンを選択し終えた時点で終了とした．この実験の結果から，各領域パターン別に，中心からの距離やパイメニュー内の角度に応じた入力速度と精度について分析する．

Vol.2010-HCI-137 No.11
2010/3/19

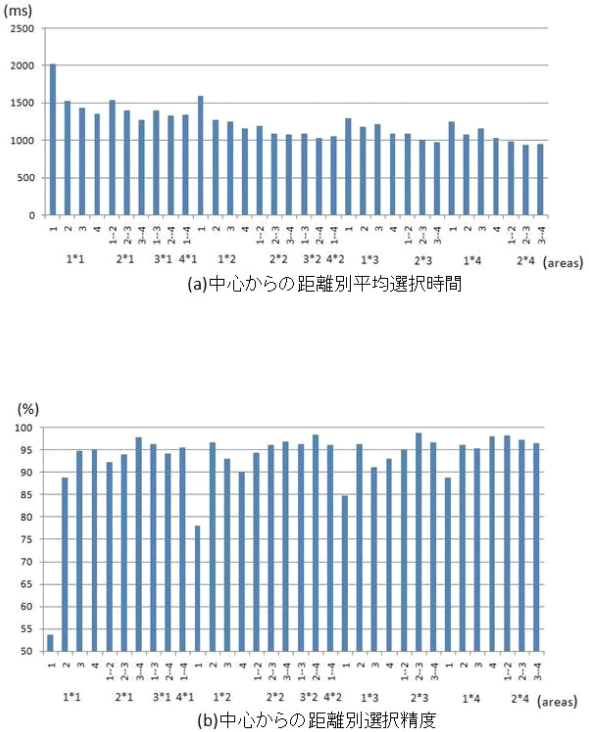

図 6 中心からの距離別の平均選択時間と精度

図 5 は，各領域パターン別の平均選択時間および精度の結果，図 6 は，各領域パターンごとに，配置した円周位置 (中心からの遠さ) 別に記録した速度と精度の結果であり，図 7 は，各領域パターンごとに，配置した角度別に記録した選択時間の結果である．図 6 の横軸の 1-2 とは，1 行目と 2 行目の 2 行分の領域のことであり，図 7 の横軸の角度の値は，その角度をスタート地点として時計回りに領域を配置したことを示している．ただし，角度は右方向を 0° とし，そこから時計回りに 1 行あたり 15° ずつ増えていくものとする．

図 5 より，領域が大きくなるにつれて入力速度の平均が速くなることがわかる．また，精度に関しては，1 行しかない領域である 1×1，1×2，1×3 が明らかに他の領域パターンよりも低い．

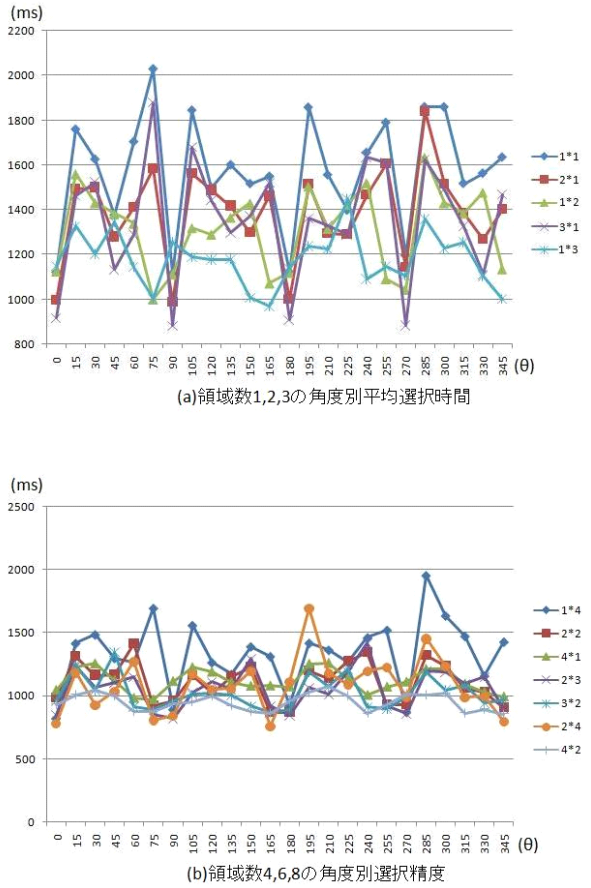

図 7 中心からの距離別の平均選択時間と精度
Fig. 7 Input speed for each angle.

次に，円周位置別での結果を考察する．図 6 より，1 行しかない領域は，1 番内側の円周に配置されている場合に明らかに速度が遅く精度が低い．特に精度に関しては圧倒的に低く，1×1，1×2，1×3 領域の 1 番内側に配置したもの以外は，ほぼ 90 %以上の精度が得られた．また，領域は，中心に近い位置よりも，外側に配置された方が全体的に速度が速く

なっている．

さらに，角度別での結果を考察する．図 7 より，角度による選択時間の差は非常に大きいことがわかる．特に，0°，90°，180°，270° の方向の領域は，他の領域に比べて選択速度が速い．これは，ジョイスティックを傾ける方向として，垂直・水平方向が簡単であるためだと考えられる．

以上の考察より，入力特性に基づいて適切な領域割当てを行うためには，予め領域パターンを決めるのではなく，その角度や中心からの距離に応じて，選択しやすい領域を割り当てる必要があると考えられる．

5. 領域割当てアルゴリズム

前節の考察を基に，領域割当てを行う．各領域パターンごとに，配置する位置に応じて入力速度や精度が変わるため，それぞれの場合に入力しやすさを表す指数が必要となる．そこで，各領域パターンが配置される全ての位置に対して領域スコアを与える．1 文字あたりの入力時間 (入力速度) を t，入力精度を a，領域数を n とし，領域スコア A は以下の式で定義する．

$$A = a \div t \times n \tag{1}$$

領域スコア A は，入力にかかる時間が少なく，精度が高いほど大きな値となる．また，領域が大きくなるほど入力速度，精度が増すと考えられるため，領域数 n を掛けることで，領域が大きいほどスコアが高くなるように設定した．つまり，入力が行いやすい領域ほどスコアの値が高くなる．*12* 種類の領域パターンが，パイメニュー上の全ての位置における場合それぞれに与えられるため，合計で *816* 個の値がある．

領域スコアの値を用いて，パイメニュー内での入力特性を考慮した領域割当てを行う．領域割当てを行う前の初期状態を，各領域が 1×1 で，*96* 個の領域がある状態とする．その中から最も領域スコアの悪い領域を選択し，隣の領域と結合させることで，全体の領域スコアを改善する．このとき選択された領域には，隣合う領域が最大で *4* つ存在する．隣合う全ての領域とそれぞれ結合した場合で比較し，その中で最も領域スコアが改善される組み合わせを新たに領域として決定する．この操作を領域数が項目数と等しくなるまで繰り返すことで，パイメニュー内の領域スコアが改善される．但し，結合可能な領域は，結合後の領域パターンが入力特性評価を行った *12* 種類の中に当てはまるものとし，領域結合のパターンとしては，図 *8* に示すように，*2* つの領域を *1* つに結合，*3* つの領域を *1* つに結合，*4* つの領域を *1* つに結合の *3* 種類を用いる．

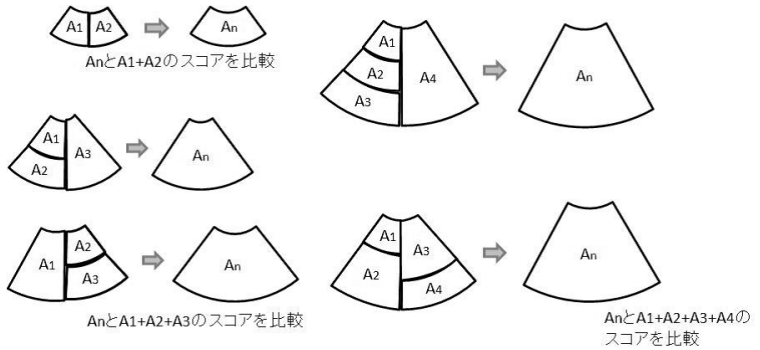

図 8 領域結合の例
Fig. 8 An example of area uniting

領域割当ての手順を以下に示す．

(1) 領域スコアの値を昇順にソートし，最もスコアが低い領域 A_1 を選択する．

(2) 図 *8* のように，選択する領域 A_1 が，隣合う領域 A_2 と結合可能な場合 *2* つの領域を結合する．結合後の領域を A_n とし，$A = (A_n$の領域スコア$) - (A_1 + A_2$の領域スコア$)$ を計算する．

(3) A_2 と結合出来ない場合，図 *8* のように，A_1 と A_2 の両方に隣合う領域 A_3 を含めると結合可能な場合，*3* つの領域を結合し，$A = (A_n$の領域スコア$) - (A_1 + A_2 + A_3$の領域スコア$)$ を計算する．

(4) A_3 も結合出来ない場合，図 *8* のように，A_2 と A_3 の両方に隣合う領域 A_4 を含めると結合可能な場合，*4* つの領域を結合し，$A = (A_n$の領域スコア$) - (A_1 + A_2 + A_3 + A_4$の領域スコア$)$ を計算する．

(5) A_4 も結合出来ない場合，A_1 結合可能な領域はないものとする．

(6) A_1 に隣合う全方向の領域に対して，*(2)*〜*(5)* の処理を行い，A の値が最大となる場合の A_n を新しい領域として決定する．

(7) *(1)* に戻る．領域数=項目数になった時点で終了．

(8) スコアが高い領域順に，使用頻度の高い項目を各領域に割り当てる．

以上の手順により，入力特性に基づいた領域割当てが行える．この時点での領域割当ての例を図 *9* に示す．

使用頻度と入力特性両方に基づく領域割当てを行うためには，使用頻度の比率と領域スコ

Vol.2010-HCI-137 No.11
2010/3/19

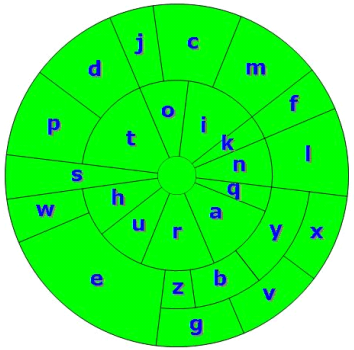

図 9 入力特性に基づく領域割当て
Fig. 9 Input characteristic based area allocation.

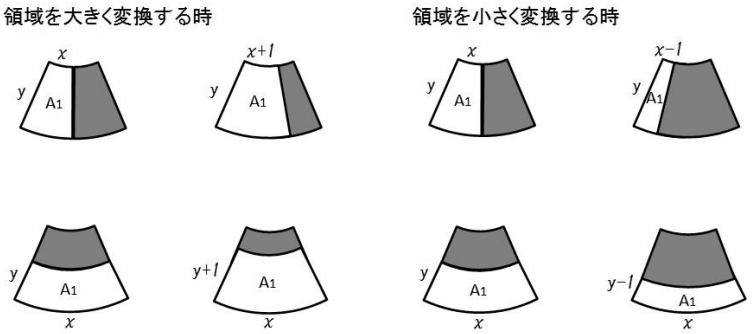

図 10 領域変換のパターン
Fig. 10 Pattern of area conversion

アの比率を近づける必要がある．そこで，領域の変換を行うことで上記を実現する．

領域割当てを行う前の初期状態を図 *9* に示す入力特性に基づいた領域割当てが完了した状態とする．このとき割り当てられている項目の使用頻度と，その領域のスコアの割合の差が大きい領域を選択し，両者の差が全体的に小さくなるように変換する．例えば，図 *9* の状態では，使用頻度が高い方から *5* 番目の値は *7.69* であるのに対して領域スコアの割合は *5.39* であり，両者の差は *2.3* と大きい．一方，使用頻度の *10* 番目の値 *4.61* に対して，*10* 番目の領域スコア割合は *4.28* であり，両者の差は *0.33* しかない．このように，差が大きい領域と小さい領域が存在するが，この状態での使用頻度の比率と領域スコアの比率の差を合計すると *27.83* であった．ここで，最も両者の差が大きい領域を変換すると，変換した領域と変換に関わった領域のスコア割合が変化する．そのとき，再び各項目の使用頻度と領域スコア割合の差の合計を計算し，*27.83* を下回れば，変換後の領域をそのまま割り当てる．この操作を繰り返すことで，使用頻度と領域スコアの割合の差を小さくし，使用頻度と入力特性に基づく領域割当てを行う．

選択した領域の変換方法を図 *10* に示す．領域を大きくするときは，x 方向に *1* 列増やすか，y 方向に *1* 行増やし，小さくするときは，x 方向を *1* 列減らすか，y 方向を *1* 行減らす．その処理を行った時，隣合う領域が評価に用いた *12* 種類のパターンに含まれる場合，変換可能とする．領域を変換後，再度領域スコアの割合と，使用頻度を比較し，全体の差が最小となる場合の領域を新たに領域として決定する．この操作を繰り返すことで，項目の使用頻度と，頻度に応じて項目が割当てられる領域のスコア割合を近づけ，適切な領域割当てを実現する．

領域割当ての手順を以下に示す．

(1) 使用頻度と領域スコアの値を降順にソートする．i 番目の項目の使用頻度を f_i，領域スコアを A_i とする．領域スコア割合 $P_{1i} = A_i \div \sum_{i=1}^{N} A_i = 1$ を計算する．ただし，割り当てる項目の総数を N とする．

(2) 使用頻度と領域スコア割合の差の絶対値 $Abs_{1i} = |f_i - P_{1i}|$ を計算し，降順にソートする．

(3) Abs_{1i} が最も大きい領域 A_1 を選択し，変換可能な場合領域変換を行う．変換可能でなければ，Abs_{1i} が次に大きい領域を選択する．

(4) 領域変換後の領域スコア割合 $P_{2i} = A_i \div \sum_{i=1}^{N} A_i = 1$ を計算し，変換後の両者の差の絶対値 $Abs_{2i} = |f_i - P_{2i}|$ を計算する．

(5) $\sum_{i=1}^{N} Abs_{2i} - \sum_{i=1}^{N} Abs_{1i} < 0$ かつ，最も小さい時の変換領域を新たに領域として決定する．

(6) 選択領域 A_1 が変換可能な全ての変換パターンで $\sum_{i=1}^{N} Abs_{2i} - \sum_{i=1}^{N} Abs_{1i} > 0$ の時，*(3)* に戻り，Abs_{1i} が次に大きい領域を選択する．

(7) *(1)* に戻る．全ての領域において，$\sum_{i=1}^{N} Abs_{2i} - \sum_{i=1}^{N} Abs_{1i} > 0$ となるとき，終了．

(8) スコアが高い領域順に，使用頻度の高い項目を各領域に割り当てる．

以上の手順で，領域スコアの割合と，使用頻度の比が近づくことになり，使用頻度と入力特性に基づいた領域割当てが行える．図 *11* に領域割当ての例を示す．

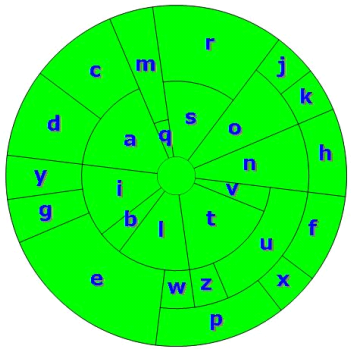

図 11 使用頻度と入力特性に基づく領域割当て
Fig. 11 A frequency and input characteristic based area allocation.

本来，全ての領域において，項目の使用頻度の比率と領域スコア割合の比率がほぼ等しくなることが理想的だが，本アルゴリズムは，両者の差が大きい領域の変換を繰り返すことで両者の差の合計を減らして比率をなるべく近づけるという手法であり，全ての領域で両者がほぼ等しくなるわけではないが，本アルゴリズムにより，*1* 項目当たりの差を *1.07* から *0.46* まで減らすことができた．

6. 評価と考察

提案手法の有効性を示すため，文字入力の速度と精度を評価した．実験には，以前提案した使用頻度のみに基づいた領域割当て *(*図 *3)*，入力特性のみに基づいた領域割当て *(*図 *9)*，両者を考慮した領域割当て *(*図 *11)* の *3* 種類に加えて，均等な領域を割り当てるパターンの *4* 種類を用いたパイメニューを用意した．均等な領域割当てでは，全ての項目に均等な領域を与えるために，領域数 *3* の領域を各項目に与え，時計回りにアルファベット順に配置した．また，従来手法として，*1* ストロークで複数文字を入力する *Quikwriting*[7] とも比較を行った．

被験者は，ランダムに表示される英単語を入力する．提案手法に必要となる文字の使用頻度は，あらかじめ用意しておいた現代英語の最頻出語 *2000* 語を用いて計算した．各被験者には，それぞれ *5* つの手法で *20* 単語入力させ，*1* 文字あたりの平均入力時間と入力精度を記録した．この実験を *1* 人につき *1* 日 *1* 回ずつ *14* 日間行わせ，被験者の習熟度を評価した．被験者は *22* 歳から *24* 歳までの大学生 *6* 名である．図 *12*，図 *13* にそれぞれ，各手法

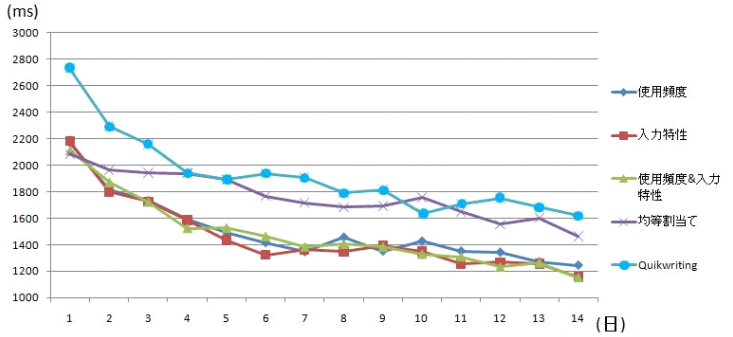

図 12 平均文字入力時間
Fig. 12 Evaluation result on speed.

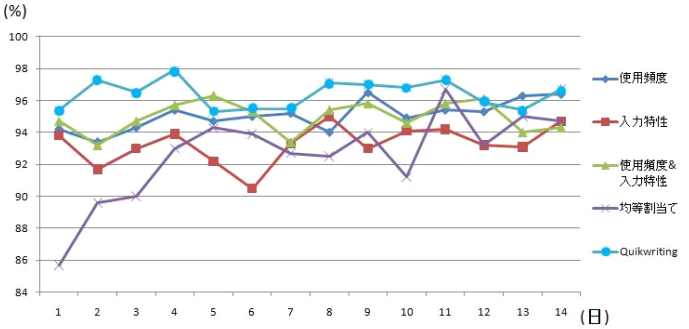

図 13 文字入力精度
Fig. 13 Evaluation result on accuracy.

ごとの平均入力時間と入力精度を示す．

図 *12* より，*1* 日目の実験では，均等な領域割当てを用いたときが入力速度が最も速いことがわかる．これは，均等な領域割当てでは，項目をアルファベット順に時計回りに配置したことで，被験者は文字が配置されている場所を覚える必要がなかったためだと考えられる．しかし，*2* 日目の実験以降では *3* 種類の領域割当て手法の方が入力速度は速くなった．よって，領域割当て手法の項目の配置を覚えるためにかかる時間は少ないといえる．また，

領域割当てアルゴリズムを用いた *3* 種類の手法で比較すると，*5*，*6* 日目で入力特性に基づいた手法が入力速度が最速になったが，最終的には使用頻度と入力速度両方に基づく手法とほぼ同程度の入力速度になることがわかる．さらに，提案手法は *3* 種類とも，従来手法である *Quikwriting* よりも速い入力が可能だといえる．

図 *13* より，*1* 日目の実験では，均等な領域割当てを用いたときが明らかに精度が低いことがわかる．この結果から，選択項目を増やすために *1* つ *1* つの項目を小さくすると精度が悪くなることをがわかる．*2* 日目以降，実験を重ねるごとに精度が良くなっているのは，ジョイスティックとパイメニューの入力操作に慣れたためだと考えられる．また，使用頻度に基づく手法と，使用頻度と入力特性に基づく手法の方が，入力特性に基づく手法よりも精度が高いといえる．これは，入力特性に基づく領域割当てでは，使用頻度が低い項目でも，ある程度の領域を割り当てられているため領域の大きさに大きな差がなかったためだと考えられる．

以上の結果から，*3* 種類の領域割当てアルゴリズムを用いた手法のうち，入力速度が速いのは，入力特性に基づく手法と，使用頻度かつ入力特性に基づいた手法であり，入力精度が高いのは，使用頻度に基づく手法と，使用頻度かつ入力特性に基づいた手法である．よって，使用頻度かつ入力特性に基づいた領域割当てが，最も入力速度と精度のバランスが良く最適な領域割当て手法だといえる．

7. ま と め

本研究では，アナログジョイスティックを用いたパイメニュー方式の文字入力インタフェースを提案し，設計と実装を行った．提案する文字入力インタフェースは，アナログ値を用いてジョイスティックを傾け手を離すという *1* つの単純な動作で，素早くて正確な文字入力が行える．小型のジョイスティックを使用したことで，スペースが限られた環境での入力インタフェースとして有効に利用できると考えられる．また，項目の使用頻度とパイメニュー内でのアナログジョイスティックの入力特性を考慮した領域割当てを提案した．項目使用頻度の比に応じて，入力しやすい領域を入力しやすい位置に割り当てることで，正確で素早い入力が実現できた．

今後の課題としては，被験者を更に増やして，長期的な実験を行うことや，予備実験である入力特性の評価を更に詳しく行うことで，より最適な領域割当てを実現する．

8. 謝　　辞

本研究の一部は，科学研究費補助金基盤 *(A)(20240009)* および特定領域研究 *(21013034)* の支援によるものである．ここに記して謝意を表す．

参 考 文 献

1) D. Hopkins: The design and implementation of pie menus, Dr. Dobb's Journal, *Vol. 16, No. 12, pp. 16–26, 1991.*
2) J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman: An empirical comparison of pie vs. linear menus, Proceedings of the SIGCHI Conference on Human Factors in Computing systems (CHI' 88), *pp. 95–100, 1988.*
3) D. Venolia and F. Neiberg: T-Cube: A fast, Self-disclosing pen-based alphabet, Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI' 94), *pp. 265–270, 1994.*
4) 堀井真吾, 菊池 猛, 千葉玉三, 赤池英夫, 角田博保: 速記型ペン入力方式の検討, 情報処理学会研究報告 (ヒューマンインタフェース研究会 *95-HI-59), pp. 1–8, 1995.*
5) G. Kurtenbach and W. Buxton: The limits of expert performance using hierarchic marking menus, Proceeding of the ACM conference on Human factors in computing systems, *pp. 482–487, 1993.*
6) F. Guimbretiere and T. Winograd: FlowMenu: Combining command, text, and data entry, Proceedings of ACM User Interface Software and Technology (UIST' 00), *pp. 213–216, 2000.*
7) K. Perlin: Quikwriting: Continuous stylus-based text entry, Proceedings of the ACM Symposium on User Interface Software and Technology (UIST' 98), *pp. 215–216, 1998.*
8) J. Mankoff and G. Abowd: Cirrin: A word-level unistroke keyboard for pen input, Proceedings of User Interface Software and Technology (UIST' 98), *pp. 213–214, 1998.*
9) Paul M. Fitts: The information capacity of the human motor system in controlling the amplitude of movement, Journal of Experimental Psychology, *Vol. 47, No. 6, pp. 381–391, 1954.*
10) 宮本雅勝, 村松邦彦, 寺田 努, 塚本昌彦: アナログジョイスティックに適したパイメニュー型インタフェースの設計と実装, ヒューマンコンピュータインタラクション研究会 *(SIGHCI), pp. 165–170, 2008.*
11) 宮本雅勝, 寺田 努, 塚本昌彦: 多項目パイメニューのための領域割当てアルゴリズム, ヒューマンインタフェースシンポジウム *2009, pp. 15–22, 2009.*