インターフェイスの街角 (15)
ウェアラブル・コンピュータのテキスト入力法

増井俊之

“いつでもどこでも計算機” へ向けて

最近、計算機を身につけて人間の能力を広範囲に補強しようという**ウェアラブル・コンピューティング** (Wearable Computing) が注目を集めています。1998 年 11 月には、NICOGRAPH98 の会場で Wearables Tokyo[1]が開催され、ファッション・ショーもおこなわれました。

従来、ウェアラブル・コンピューティングを実現するうえで最大の難問はディスプレイの大きさや重さだと考えられていましたが、ごく小さな眼鏡型ディスプレイなどが開発され、表示装置の問題は解決に向かっているようです。

一方、入力装置についてはあまり大きな進展はみられません。IBM が 1998 年 9 月に発表した超小型のウェアラブル PC[2]の入力には音声認識が採用されていました。音声認識による入力には、キーボードなどのかさばる装置が不要というメリットはありますが、騒音のあるところでは使えませんし、人前で音声コマンドを発行したり、メールの文章を読み上げるのは抵抗があるでしょう。たとえ使う側に違和感がなくても、電車内などで音声認識を使ってメールを書いたりすれば、白い眼でみられるか、「周りのお客様のご迷惑になりますので、音声認識のご利用はご遠慮ください」などとアナウンスされるにちがいありません。したがって、〝いつでもどこでも計算機を使う〟というウェアラブル・コンピューティングの精神からは遠いものになってしまいます。

ウェアラブル・コンピューティングの普及のためには、小型で静かな入力装置の開発が不可欠です。そこで、今回はウェアラブル・コンピューティングで使用可能な各種の小型入力装置を紹介します。

1 http://www.nikkei.co.jp/events/ccn/wearables/

2 http://www.ascii.co.jp/ascii24/issue/980911/hard03.html

図 1　Wearables Tokyo ファッション・ショー

Twiddler

ウェアラブル・コンピューティングの研究でもっともひろく使われている入力装置は、おそらく HandyKey[3]の「Twiddler」でしょう。Twiddler は片手で持って操作できる小さな装置ですが、一般の PC で使われている PS/2 マウスとフルキーボードの機能をすべて備えています。

Twiddler には、親指で押す 6 個のボタンと、人差し指〜小指で選択する 4 × 3=12 個のボタンがあります。親指のボタンにはマウスボタンやコントロールキーが、人差し指〜小指のボタンには英数字キーが割り当てられています。12 個のキーですべての英数字キーの動作をさせるため、2 つのキーを同時に押すと、通常のキーボードの 1 つのキーを押したのと同じ効果が得られる仕組みになっています。このように、複数のキーを同時に押すことによって多くの機能を表現する方式を一般に〝コードキーボード (Chord Keyboard)〟と呼びます。キーの組合せは、プロ

3 http://www.handykey.com/

図 2　Twiddler

図 3　Twiddler と Private Eye をつけた Rhodes 氏

グラムで自由に変えられます。Twiddler では、本体を傾けてマウスカーソルを動かすことも可能です。

MIT Media Lab[4]は、ウェアラブル・コンピューティングの研究に力を入れている組織の 1 つです。MIT が提案したウェアラブル・コンピューティングの標準仕様である Lizzy[5]というシステムでは、入力装置に Twiddler が使われています。

MIT では、研究するだけではなく、四六時中計算機を身につけて生活をしている研究者も多いようです。図 3 は、MIT で ``Remembrance Agent''[6]の研究をおこなっている Bradley Rhodes 氏が Twiddler と Private Eye[7]を装着しているところです。以前、Rhodes 氏が私の所属する研究所に来たとき、1 日中これらの装置をつけたまま行動していたのには驚かされました。

Twiddler をすこし使ってみましたが、指を曲げて 2 つのキーを押すのはやや難しいようです。ただし、Rhodes 氏はかなりの速度で入力していたので、慣れればギターを弾くように使えるのかもしれません。マウスの機能も備えているため、通常のアプリケーションを変更せずに使える点はたいへん魅力的です。

4 http://lcs.www.media.mit.edu/projects/wearables/

5 http://lcs.www.media.mit.edu/projects/wearables/lizzy/index.html

6 編集中のものにもっとも近いファイルをつねに表示して、過去におこなった類似の作業を簡単に思い出せるようにするシステム (http://rhodes.www.media.mit.edu/people/rhodes/RA/)。

7 回転鏡と LED で構成された安価な表示装置で、任天堂のバーチャルボーイでも同様の技術が使われていました。

図 4　BAT キーボード

図 5　カスタネット・キーボード

BAT

BAT キーボードは、Infogrip[8]が開発したコードキーボードです (図 4)。Twiddler には全部で 18 個のキーがありましたが BAT には 7 個しかありません。少ないキーでたくさんの機能を実現するには複雑な指づかいが必要になりますが、Infogrip によれば 1〜2 時間で英数字をタイプできるようになるそうです。

カスタネット・キーボード

富士通研究所の杉本正勝氏は、片手で文章の入力が可能な「カスタネット・キーボード」[1] を考案しました[9]。

カスタネット・キーボードは、20 個強のキーで構成されています (図 5)。片手で持ちやすい形を追求した結果、カスタネット型になったそうです。

1 月号で紹介した T9 キーボード[10]と同様、カスタネット・キーボードでは複数の文字が 1 つのキーに割り当てられています。たとえば、``Y''と ``M''は同じキーにマップされているため、``や''も ``ま''も同じキー操作になり、それだけでは「安い」と「増井」の区別がつきません。そこで、``曖昧解消キー''を使って、目的の単語を候補から選べるようになっています。

カスタネット・キーボードは執筆時点 (1998 年 12 月) ではまだ市販されていませんが、PS/2 インターフェイス

8 http://www.onehandkeyboard.com/

9 http://www.fujitsu.co.jp/hypertext/news/1997/May/28-3.html。SHK (Single Hand Keys) とも呼ぶようです。

10 プッシュホンのボタンに似た、``ABC''や ``DEF''などの複数の文字が割り当てられたテンキーを用いて文字を入力するキーボード。自然言語辞書を利用して、曖昧性を解消しながら文章を作成していくことができます。

図 6　PalmPilot 用 SHK

図 7　CUT Key

をもつものや、PalmPilot に装着できるもの（図 6）などが試作されています。PalmPilot 版は、図 5 とはすこし異なる 18 個のキーから構成され、1999 年春に発売予定だそうです[11]。

CUT Key

ミサワホームは、パーソナル・コンピュータのテンキーに似たキーボードで日本語が入力できる「CUT Key (Compact Universal Terminal Keyboard)」[12]を販売しています。CUT Key は、図 7 のようなキー配列になっています。日本語のローマ字入力を念頭において開発したそうで、上側の母音キーと下側の子音キーが明確に区別されています。

CUT Key では、子音と母音の組合せで五十音を入力します。たとえば「は」と入力する場合は ˋH″のキー（ ˋ,″ キー）を押したあとで ˋA″のキー（ ˋ7″キー）を押します。 ˋB″ ˋP″も ˋH″と同じキーに割り当てられていますが、「ば」と入力する場合はこのキーを 2 回押してから、「ぱ」と入力する場合は 3 回押してから ˋA″のキーを押します。H/B/P、K/G などのように清音/濁音/半濁音を構成する子音が同じキーに割り当てられているので憶えやすい配列といえます。

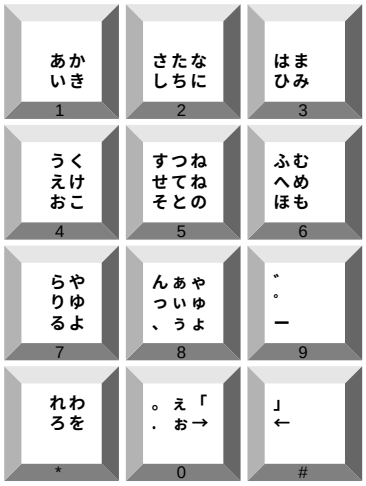
小野式配列

日本 NCR の小野勝康氏も、テンキーのようなキーボードを使って五十音を入力する手法を提案しています [2]。

11 http://www.fujitsu.co.jp/hypertext/flab/News/1998/May/980527/980527.html

12 http://misawa01.misawa.co.jp/CUTKEY/

図 8　小野式キー配列



2 ストロークで五十音を入力するのは CUT Key と同じですが、最初に押したキーの位置と、その次に押したキーと最初のキーとの位置関係によって入力する文字を選択します。

小野氏が考案したキーボード配列（図 8）では、各キーに最大 9 個の ˋかな″が割り当てられています。たとえば「す」と入力する場合には、まず「す」が刻印されている ˋ5″キーを押します。そして、「す」が ˋ5″キーの左上にあることを示すために ˋ5″キーの左上にある ˋ1″キーを押します。このように、キートップに対応した 2 つのキーを押すことによって文字を入力していきます。最初に押すキーの位置とその次のキーとの相対位置関係を利用しているという点で、1 月号で紹介したペンによる文字入力システム Quikwriting、T-Cube などのキーボード版と捉えることもできます。

Half-QWERTY キーボード

カナダの Matias[13]では、普通の ASCII キーボード（QWERTY 配列）を片手でも使えるようにした Half-QWERTY キーボードを開発しています [3]。

Half-QWERTY キーボードは図 9 のような配列で、普通の ASCII キーボードをまっぷたつにしたような構造になっています。 ˋF″や ˋD″などの左側のキーは普通のキーボードの場合と同様にして入力しますが、スペースキーを押しながら ˋF″キーや ˋD″キーを押すと、ちょうど

13 http://www.dgp.toronto.edu/matias/

図 9　Half-QWERTY キーボード配列

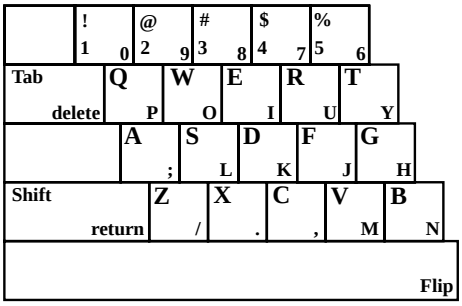


図 10　Half-QWERTY キーボードを使っている様子

図 11　FingeRing

図 12　無線版 FingeRing

左右反転した位置にある〝J〟〝K〟などのキーを押したのと同じことになります。

FingeRing

NTT の福本雅朗氏は、キーボードを使わず、指輪の振動を利用して文字を入力する「FingeRing」を提案しています [4, 5]。指輪には加速度センサーが付いており、机などを叩いたときの振動から各指の打鍵を検出します。数年前に開発された FingeRing（図 11）では各指輪にコードを接続する必要がありましたが、最近のもの（図 12）では、指輪内のコンデンサがバッテリーとして働き、指輪と腕時計型通信装置とのあいだの誘導電磁界通信によりコードレスで動作します。

福本氏は、FingeRing の 2 ストロークで文字を入力する手法も考案しています。

片手で日本語入力

1998 年 4 月号で、ペン計算機用の文章入力手法 POBox[14]を紹介しました。ペンの代わりに片手キーボードを利用すれば、ウェアラブル・コンピューティングでの入力に同じ手法が使えます。POBox の入力では、ペンで単語の読みを指定して次の入力単語の候補リストを作成し、そのなかから入力したい単語をペンで選択する操作を繰り返しますが、読みの指定による単語の絞り込みと候補の選択を片手キーボードでおこなえば、POBox と同様に効率的に文章を入力していくことができます。

片手キーボードでの文字入力には、正確な読みを指定する方法（Twiddler、BAT、CUT Key など）と、曖昧性を許容しながら指定していく方式（T9 やカスタネット・キーボードなど）があります。どちらがよいかは、正確に読みを指定するためにどのくらいの手間がかかるかによりますが、ここでは T9 や SHK よりはるかに曖昧度の大きな入力装置を使う方法として、私が開発した JunKey (Japanese UNi-handed Keyboard) を紹介します。

片手日本語入力手法 JunKey

JunKey では、左手の 5 本の指だけを使って日本語を入力します。Twiddler や BAT などのコードキーボードではなく、普通のキーボードの A、S、D、F の各キーとスペースキー（左手のホームポジション位置のキー）だけを使います。

ローマ字の入力では、Q や L などを除く 20 個あまりのキーが使われますが、JunKey ではこれらを完全には区別しません。人差し指を使うキー（M、U、Y など）を〝F〟に、中指を使うキー（E、D、I など）を〝D〟に、薬指を使うキー（S、W、O など）を〝S〟に、小指を使うキー（A、Z、P など）を〝A〟に割り当てます。たとえば、「増井」という文字列を得るには、通常は M、A、S、U、I と入力しますが、JunKey では M の代わりに F を、U の代わりに F を、I の代わりに D を使い、〝F A S F D〟と入力します。

このようなマッピングでは、「安い」(YASUI) や「橋」(HASHI) も同じ〝F A S F D〟という入力になり、曖昧

14 http://www.csl.sony.co.jp/person/masui/POBox/

性が生じます。しかし、現実には単語の使用頻度に偏りがあるので、複数の候補を頻度順に表示するようにすれば、少ない入力から検索した場合も目的の単語が候補に含まれる可能性はかなり高くなります。

使用例

JunKey を起動すると、以下のように入力単語の候補が表示されます。使用頻度の高い〝が〞や〝は〞などの助詞は、最初に表示されています。

```
% junkey

は が か の を
```

ここで「増井」と入力するために、M に対応する〝F〞キーを入力すると、画面は次のようになります。

```
は が の も と
```

〝は (HA)〞や〝が (GA)〞〝の (NO)〞など、ローマ字表現の最初の文字が人差し指に対応しているものはそのまま残り、人差し指に対応しない〝か (KA)〞などは候補から消えます。

続いて〝A〞キーを押します。

```
は が な やった やって
```

さらに〝S〞キーを押すと、画面は以下のように変化します。

```
増井 ます ました ません 場所
```

私の辞書では「増井」がかなり上位に位置しているので、この時点で候補リスト内に出現しています。ここでスペースキー（候補選択キー）を押すと最初の候補が選択され、画面は以下のように変化します。

```
[増井]
ます ました ません 場所 ますが
```

もう 1 回スペースキーを押すと、以下のように次の候補が選択されます。

```
[ます]
ました ません 場所 ますが マシン
```

〝増井〞を選択した状態で〝F〞キーを押すと、〝増井〞に続いて出現する可能性が高く、かつ人差し指に対応した文字で始まる単語が候補として表示されます。

```
増井
    俊之 の は が も
```

ここでスペースキーを 2 回押すと〝の〞が選択され、画面は以下のようになります。

```
増井 [の]
     は が も と に
```

ここで〝F〞キーを押すと、

```
増井の
       か で 研究 繰り返し データ
```

となり、続けてスペースキーを 3 回押すと〝研究〞が選択されて「増井の研究」という文字列が入力されます。

このように、自分がよく使う文字列は容易に入力できますが、それ以外の単語の入力もそれほど難しくはありません。たとえば、〝A S A F D D〞と入力すると候補として〝旭川市 旭区 淡路町〞が表示され、〝F A D A F D F F〞では〝中目黒 高根村〞などが表示されます。〝ASAHIKAWASHI〞や〝NAKAMEGURO〞のように正しく入力しなくても、曖昧なキー指定で、かつ単語の途中まで入力するだけで、たいていは十分な候補リストが得られます。

この手法で日本語を入力する場合、漢字を 1 分あたり約 60 文字入力できます。

実装

JunKey の実装はきわめて簡単です。読みとして A、S、D、F だけを含む特別な辞書を使う、キーボード版 POBox として実現できます。プログラムは 200 行ほどと小さいものです（末尾のリスト 1）。基本的には、パターンが入力されるたびに辞書を検索し、マッチする単語を候補として表示してスペースキーで選択できるようにするだけです。

〝F A S F D〞の入力から文字列「増井」が検索できるように、あらかじめ図 13 のような辞書を作成しておきます。

図 13　JunKey の単語辞書

パターン	単語
fa	は
fa	が
da	か
.....	.....
fasfd	増井
fdff	メール
dsfddafa	これから
faffa	やった
.....	.....
sfdfddff	している
fadff	ファイル
fddff	ている
dsfsfa	ことが
.....	.....

この種の辞書は、POBox の場合と同様の手法で簡単に作れます。

おわりに

携帯電話やウェアラブル・コンピュータ上での文章入力手法に関して、いろいろな企業が主導権を握ろうと切磋琢磨しています。誰もがどこでも簡単に高速に文章を入力できるような万能薬的手法はまだみつかっていないようですが、今後の発展に期待したいところです。

（ますい・としゆき　ソニー CSL）

[参考文献]

[1] 杉本正勝「片手操作キーカード (SHK) による日本語入力」、情報処理学会モーバイルコンピューティング研究会研究報告、Vol.97、No.54、pp.1–6、1997 年 5 月

[2] 小野勝康「テンキーによる日本語入力 ── 2 タッチ 50 音配列を実現」、情報処理学会ヒューマンインタフェース研究会研究報告 98-HI-77、Vol.98、No.22、pp.49–54、1998 年 3 月

[3] Edgar Matias, I. Scott MacKenzie and William Buxton, Half-qwerty: Typing with one hand using your two-handed skills, In *CHI'96 Conference Companion*, pp.51–52. ACM Press, April 1996

[4] 福本雅朗、平岩 明、曽根原 登「ウェアラブルコンピュータ用キーボード fingering」、電子情報通信学会論文誌 A、Vol.J79-A、No.2、pp.460–470、1996 年

[5] Masaaki Fukumoto and Yoshinobu Tonomura, "Body Coupled FingeRing: Wireless wearable keyboard", In *Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI'97)*, pp.147–154, Addison-Wesley, April 1997

リスト 1　JunKey ソース

JunKey は短いプログラムですが（もっとも小さな日本語入力プログラムの 1 つかもしれません）、これだけで片手キーボードによる日本語入力が実現できます。A、S、D、F の各キーによる読み指定とスペースキーによる次候補選択に加え、C キー（パターンのクリア）、X キー（前候補）、V キー（全マッチ指定）、Z キー（改行）が使えます。プログラムおよび辞書、辞書作成プログラムは私の Web ページ (http://www.csl.sony.co.jp/person/masui/UnixMagazine/) で公開していますのでお試しください。

```
#include <stdio.h>
#include <string.h>
#include <termios.h>
//
//  端末設定 (POSIX)
//
int cbreak(int fd)
{
  struct termios buf;
  if (tcgetattr(fd, &buf) < 0) return -1;
  buf.c_lflag &= ~(ECHO | ICANON);
  buf.c_cc[VMIN] = 1;
  buf.c_cc[VTIME] = 0;
  if (tcsetattr(fd, TCSAFLUSH, &buf) < 0) return -1;
  return 0;
}
//
```

```
//  VT100カーソル制御
//
#define reverse()  printf("\033[7m")
#define normal()  printf("\033[m")
#define bol()    printf("\r")
#define erase()    printf("\033[K")
#define fullerase()  printf("\r\033[K")
#define up()    printf("\033[A")
#define right(i)  if((i)>0) printf("\033[%dC",i)
//
//  単語辞書/例文辞書
//
#define WORDDIC "word.pp"
#define MAXWORDS 40000
unsigned char *word[MAXWORDS],*yomi[MAXWORDS];
int nwords = 0;
#define PHRASEDIC "phrase.pp"
#define MAXPHRASES 50000
unsigned char *pcontext[MAXPHRASES],*pword[MAXPHRASES],*pyomi[MAXPHRASES];
int nphrases = 0;

#define MAXLINELEN 1000
#define MAXWLEN 100

void readdic()
{
  FILE *f;
  char linebuf[MAXLINELEN];
  char context[MAXWLEN],y[MAXWLEN],w[MAXWLEN];

  if((f = fopen(WORDDIC,"r")) == NULL) exit(0);
  while(fgets(linebuf,MAXLINELEN,f) && nwords < MAXWORDS){
    sscanf(linebuf,"%s %s",y,w);
    word[nwords] = strdup(w);
    yomi[nwords] = strdup(y);
    nwords++;
  }
  fclose(f);

  if((f = fopen(PHRASEDIC,"r")) == NULL) exit(0);
  while(fgets(linebuf,MAXLINELEN,f) && nphrases < MAXPHRASES){
    sscanf(linebuf,"%s %s %s",context,y,w);
    pcontext[nphrases] = strdup(context);
    pword[nphrases] = strdup(w);
    pyomi[nphrases] = strdup(y);
    nphrases++;
  }
  fclose(f);
}
//
//  候補
//
#define MAXCANDS 50
unsigned char cands[MAXCANDS][MAXWLEN];
int ncands = 0;
int curcand = -1;  // 現在選択されている候補のインデックス

char outbuf[MAXLINELEN] = "";  // 出力文字列
int outbuflen = 0;
//
//  パターン/検索
```

```
//
char pat[100] = "";
int exact = 0;

void resetpat()
{
  strcpy(pat,"");
  exact = 0;
  curcand = -1;
}
void search()
{
  int i,j;
  int plen,patlen;
  outbuflen = strlen(outbuf);
  patlen = strlen(pat);
  ncands = 0;
  for(i=0;i<nphrases && ncands < MAXCANDS-1;i++){
    plen = strlen(pcontext[i]);
    if(outbuflen >= plen &&
       strncmp(pcontext[i],outbuf+outbuflen-plen,plen) == 0 &&
       (exact ? strcmp(pat,pyomi[i]) : strncmp(pat,pyomi[i],patlen)) == 0){
      for(j=0;j<ncands;j++)
        if(strcmp(pword[i],cands[j])==0) break;
      if(j < ncands) continue;
      strcpy(cands[ncands++],pword[i]);
    }
  }
  for(i=0;i<nwords && ncands < MAXCANDS-1;i++){
    if((exact ? strcmp(pat,yomi[i]) : strncmp(pat,yomi[i],patlen)) == 0){
      for(j=0;j<ncands;j++)
        if(strcmp(word[i],cands[j])==0) break;
      if(j < ncands) continue;
      strcpy(cands[ncands++],word[i]);
    }
  }
}
//
//  確定文字列/候補の表示
//
void display()
{
  int i,col = strlen(outbuf);

  printf("\n");
  fullerase();
  right(col);
  for(i=curcand+1;i<ncands && i<=curcand+5;i++){
    if((col += (strlen(cands[i])+1)) > 78) break;
    printf("%s ",cands[i]);
  }
  up();
  fullerase();
  printf("%s",outbuf);
  if(curcand >= 0){
    reverse();
    printf("%s",cands[curcand]);
    normal();
  }
  fflush(stdout);
}
```

```
//
//  メインルーチン
//
int main(int argc, char **argv)
{
  char s[2] = " ";

  cbreak(0);
  readdic();
  search();
  display();
  while(read(0,s,1)){
    switch(*s){
    case 'c': // パターンのクリア
      resetpat();
      search();
      break;
    case 'z': // 確定/改行
      strcat(outbuf,cands[curcand]);
      resetpat();
      display();
      printf("\n");

      strcpy(outbuf,"");
      search();
      break;
    case 'v': // 完全一致検索指定
      if(! exact){
        exact = 1;
        search();
        curcand = 0;
      }
      break;
    case 'x': // 前候補/後退
      if(curcand >= 0){ // 前候補
        curcand--;
      }
      else { // 後退
        outbuflen = strlen(outbuf);
        if(outbuflen > 1) outbuf[outbuflen-2]='\0';
        resetpat();
        search();
      }
      break;
    case ' ': // 次候補
      curcand++;
      break;
    case 'a': case 's': case 'd': case 'f':
      if(curcand>=0 && strlen(pat) > 0){ // 前の候補を確定
        strcat(outbuf,cands[curcand]);
        resetpat();
      }
      strcat(pat,s);
      search();
      break;
    }
    display();
  }
}
```