ファジィルールを用いたウィンドウの自動再配置

正 員 津 原 進 (日 立)

Automatic Window Rearrangement Using Fuzzy Rules

Susumu Tsuhara, Member (Hitachi, Ltd.)

Today, it's remarkable that the operating systems attach the "window system" as a user interface on the small computers such as personal computer, workstation, and so on.

In these window systems, as you know well, the "overlapping multi-window" method is used as a display control method. Although there are much merit in it, we can not disregard some demerits. One of the most undersirable situations is that when a window is enlarged or opened, some other windows may be hidden completely under the active window and become quite invisible. To conquer this situation, several improved methods have been proposed. But, it seems that these are not focused on the center of the problem.

In this paper, we define it as a "fuzzy control" problem. Namely, the problem of recovering the windows which fell into a "disadvantageous" status by the window manipulation to the most "advantageous" status. And we propose a new method of automatic window rearrangement using fuzzy inference. Finally, we show the usefulness of the proposed method by several experiments.

キーワード：小形情報機器，ウィンドウシステム，オーバラッピングマルチウィンドウ，操作性向上，ウィンドウ自動再配置，フジィ推論

1. ま え が き

小形情報機器の主流 OS とされている UNIX オペレーティングシステム^{*1}，OS/2^{*2}，MS-DOS^{*3}にウィンドウシステム搭載の動きが顕著である。UNIX には OSF/Motif^{*4}や OPEN LOOK^{*5}，OS/2 には Presentation Manager^{*3}，MS-DOS には MS-Windows^{*3}が，それぞれ登場してきている。

これらのウィンドウシステムに採用されている制御方式は「オーバラッピングマルチウィンドウ」方式である。これは，ウィンドウ重ね置き方式とも呼ばれているように，表示スクリーン上に複数のウィンドウを相互に重なり合いを許して同時表示する方式であり，完成度の高い方式として注目を集めている。

しかし，上記方式に問題がないわけではなく，ユーザがウィンドウを大きくしたり，オープンしたりしたときに，ユーザに全く見えないウィンドウが発生する場合もあるなど，操作の流れを阻害するような現象も見受けられる。

これを改良すべく各種の方式が提案されているが，いずれも，ただ単にユーザに見えないウィンドウに注目しているだけであり，他のウィンドウについては配慮がなされておらず，断片的改良方式であるように思われる。また，ウィンドウがユーザに「見える」，「見えない」というのは「あいまいさ」を含んだ定性的な概念であり，これを定量的に処理できることが必要である。

本論文では，これをヒューマンインタフェース向上の問題として総合的，定量的に処理することを試みる。具体的には，ウィンドウ操作の結果，「ユーザにとって不利な状態に陥ったウィンドウを，最も有利な状態へと回復する」問題として定義し，解決のための一手法としてファジィルールを用いたウィンドウの自動再配置手法を提案する。更に，実験により本手法の

*1 UNIX オペレーティングシステムは，UNIX システムラボラトリーズ社が開発し，ライセンスしている。

*2 OS/2 は，アメリカ・IBM 社の登録商標である。

*3 MS-OS/2，MS-DOS，Presentation Manager，MS-Windows は，アメリカ・マイクロソフト社の登録商標である。

*4 OSF/Motif は OSF の登録商標である。

*5 OPEN LOOK はアメリカ・AT&T 社の商標である。

有効性を検証する。

2. 現行ウィンドウシステムの概要

〈2・1〉 登場の背景 最も歴史の長い UNIX オペレーティングシステム⁽¹⁾では、1986年に4.3 BSD (Berkeley Software Distribution) が発表され、この版でMITで開発された X Window System^{*6}が搭載された。現在、IBM社を中心とする OSF (Open Software Foundation) グループが推進する OSF/Motif と、AT&T社を中心とする UI (Unix International) グループが推進する OPEN LOOK とに分かれている。

OS/2⁽²⁾は、マイクロソフト社および IBM 社の戦略商品として1987年4月に発表された。当初ウィンドウシステムは搭載されなかったが、同年末ごろからマイクロソフト社より V 1.1 版が開発者用に配布され始めた。この版では、PM (Presentation Manager) というウィンドウシステムが標準装備されている。引き続き V 1.2 版が配布された。V 2.0 版も予定されている。

MS-DOS は、マイクロソフト社により1981年に登場、その後バージョンアップを重ね、現在 V 4.0 となっている。この MS-DOS をベースとしたウィンドウシステムとして1985年に MS-Windows V 1.0 が登場した。その後、MS-Windows V 2.0、MS-Windows/386 の登場が相次ぎ現在、MS-Windows V 3.0 が発表されるに至っている。

上記のようなウィンドウシステム登場の動きに大きな影響を与えたのは、Xerox 社の Star⁽³⁾であると考えられる。Star の製品発表は1981年である。Star では、ビットマップ表示装置とマウスを用いた GUI (Graphical User Interface)、デスクトップメタフォ、マルチウィンドウ、ダイレクトマニピュレーション、アイコン、オブジェクト指向、モードレス、WYSIWYG などの有用でかつ斬新な設計コンセプトが採用された。これらの設計コンセプトが市場に受入れられ、これが契機となって、上記ウィンドウシステムの登場を促進したものと思われる。

〈2・2〉 ウィンドウの構成と操作法 代表例として、OS/2 PM におけるオーバラッピングマルチウィンドウの表示例を図1に示す。

Look & Feel は、OSF/Motif や MS-Windows と同一であるが、UI の OPEN LOOK とは多少異なるようである。同図下端に示した小さな正方形は、アイ

図1 オーバラッピングマルチウィンドウの表示例

Fig.1. An example of the overlapping multi-window.

図2 ウィンドウの標準構成

Fig.2. Standard structure of the window.

コンと呼ばれており、閉じたウィンドウとみなすことができる。しばらく不用なウィンドウはユーザの指示により、このようにアイコン化することができる。このアイコンを元の開いたウィンドウに復帰させるには、マウスでクリックするだけでよい。ウィンドウやアイコンは、それぞれ表示優先度を持ち、優先度の高いものほど画面手前に表示される。画面の最も手前に表示されているウィンドウはアクティブ（またはフォーカス）ウィンドウと呼ばれる。当然、アクティブウィンドウの表示は他のいかなるウィンドウによってもダメージは受けない。

標準的な構成のウィンドウを図2に示す。

構成要素として、システムメニューボックス、タイトルバー、ミニマイズボックス、マキシマイズボックス、メニューバー、スクロールバー、サイジングボーダ、クライアントウィンドウがある。以下、各々につき説明する。

(1) システムメニューボックス システムメニュー（ウィンドウの移動、サイズ変更、最大化、最小

*6 X Window System は、アメリカ・マサチューセッツ工科大学の商標である。

化、クローズなど)を表示するためのボックスである(マウスで本ボックスをクリックすればシステムメニューが表示される)。

(2) タイトルバー ウィンドウ上で走行する Application Program (以下、AP と略記) の名称を表示するエリアである。マウスを、このタイトルバーに位置付けドラッグすればウィンドウを移動させることができる。

(3) ミニマイズボックス ウィンドウを、アイコンと呼ばれる小さなシンボルにする(アイコン化する)ためのボックスである。システムメニューの「最小化」と同等の機能をもつ。しばらく不用なウィンドウは、本ボックスをマウスでクリックすることによりアイコン化することができる。

(4) マキシマイズボックス ウィンドウの大きさを、全画面サイズにする(フルスクリーン化する)ためのボックスである。システムメニューの「最大化」と同等の機能をもつ。ウィンドウがフルスクリーン化されると、マキシマイズボックスは消えて、代わりにリストアップボックスが同じ場所に表示されるので、これをマウスでクリックすればウィンドウを元の状態に復帰させることができる。

(5) メニューバー ユーザが指示可能な AP の動作を、コマンドメニューとして列挙・表示するためのエリアである。列挙・表示されたコマンドメニューの中から、プルダウン式に目的のコマンドをマウスで選択・指示することにより、AP の動作を制御することができる。

(6) スクロールバー ウィンドウ内の AP の表示データを、上下左右に動かすためのエリアである。ウィンドウ内に表示されていないデータを参照したいときに使用する。

(7) サイジングボダ ウィンドウの幅や高さを変更するためのエリアである。システムメニューの「サイズ変更」と同等の機能をもつ。マウスをこのサイジングボダへ位置付け、ドラッグすることによってこれを行う。上・下辺はウィンドウの高さを、左・右辺はウィンドウの幅を、それぞれ変更するのに使用され、ウィンドウの四隅は、幅と高さを同時に変更するのに使用される。

(8) クライアントウィンドウ AP の表示データを表示するためのエリアである。

3. 操作性向上のための問題提起

現行ウィンドウシステムでは、ウィンドウは簡単に操作できるように配慮されている。例えば、ウィンド

ウの大きさを変更するには、サイジングボダにマウスカーソルを位置付け、ドラッグすればよい。また、オープンするには AP 名やアイコンをマウスでクリックすればよい。詳細は(2・2)節で述べたとおりである。このように、頻繁に発生する操作はほぼワンタッチで行えるよう設計されており、完成度の高いシステムとして注目を浴びている。

しかし、上記のようなウィンドウ操作の結果、他のウィンドウの表示が大きなダメージを受ける場合もあることも事実である。例えば、ノンアクティブウィンドウを包含するようにアクティブウィンドウを配置したときなど、ノンアクティブウィンドウはユーザに全く見えなくなる。また、ユーザにはわずかに見えるが、見える部分が極めて少なく、表示している効用がないということもありうる。

このような問題に対するユーザの反応は、大別すると次の二つとなるようである。

(1) 現状肯定 たとえ、ノンアクティブウィンドウの表示が大きなダメージを受けても、それはユーザの指示であり、ノンアクティブウィンドウをシステムが勝手に変更すべきではない。許可なく勝手に変更するとユーザは混乱する。

(2) 現状改良 ノンアクティブウィンドウを、ユーザに有利なように自動的に配置し直すべきである。そうしなければ、常にノンアクティブウィンドウを意識しなければならず、ユーザの負担が増大し、操作がわずらわしくなる。また、同時複数表示というマルチウィンドウの効用もなくなる。

上記(1)、(2)項のうちどちらを選択すべきかの判断は容易ではないが、本論文では(2)項の立場に立つ。すなわち、ユーザによるアクティブウィンドウの操作と連動して、ノンアクティブなウィンドウをユーザにとって有利なように自動的に配置し直す(これを自動再配置と呼ぶことにする)。ユーザは最初のうちは当惑するかもしれないが、慣れてくれば、これが自然であると考えられるようになることを期待している。

次章では、上記のようなノンアクティブウィンドウの自動再配置の一方式を提案する。

4. ファジールールを用いたウィンドウ自動再配置方式

提案するウィンドウ制御方式では、ユーザによるアクティブウィンドウの操作と連動して、ノンアクティブウィンドウを自動的に再配置する。

ノンアクティブウィンドウは一般に複数存在するので、表示優先度の高いものから順に逐次一つずつ再配

図 3 ウィンドウ自動再配置の処理手順
Fig. 3. Procedures of the automatic window rearrangement.

位置する。再配置においては、単にウィンドウの表示位置を決定するだけでなく、その表示様式も決定する。

図 3 にノンアクティブウィンドウ自動再配置の手順を示す。図示のごとく下記の 7 ステップで行う。

(1) 配置候補領域列挙 再配置すべき対象ウィンドウを、現在の位置でなく別の位置に表示するときの候補となり得る、すべての長方形領域を列挙するステップである。以下にその方法を示す。

(i) 複合長方形計算 対象ウィンドウよりも表示優先度の高いすべてのウィンドウ/アイコンで形成される複合長方形を計算する〔図 4(a)参照〕。対象ウィンドウをこの複合長方形に重ならないように配置できれば対象ウィンドウは 100% ユーザに見え、高水準の満足度が得られるものと考えられる。その意味でこの複合長方形は、対象ウィンドウの配置禁止領域である。

(ii) 水平分割 表示スクリーン上の上記複合長方形以外の領域を、平面走査法により最小数の長方形に水平分割する〔図 4(b)参照〕。

(iii) 垂直方向への拡大 上記で得られた水平分割長方形の各々につき、幅を変えないで上辺は上へ、

(a)

(b)

(c)

図 4 配置候補領域列挙の方法
Fig. 4. Method of enumerating the possible rectangles.

下辺は下へと可能な限りシフトして高さ最大の長方形を得る〔図 4(c)参照〕。

(iv) 垂直分割 (ii)と同様に、表示スクリーン上の複合長方形以外の領域を、平面走査法により最小数の長方形に垂直分割する。

(v) 水平方向への拡大 (iii)と同様に、上記で得られた垂直分割長方形の各々につき、高さを変えないで左辺は左へ、右辺は右へと可能な限りシフトして幅が最大の長方形を得る。

(vi) 不要候補の削除 前記(iii)と(v)とで得られた候補領域の中から、不要な候補を削除する。すなわち、一方が他方に完全に含まれるような候補もあり得るので、そのときには含まれるほうの候補を削除し

て、相互に包含関係にない配置候補領域を得る。

(2) 配置候補領域への表示様式決定 (1)項で求めた配置候補領域へ対象ウィンドウを配置するとき、対象ウィンドウの表示様式を定めるステップである。許容する表示様式として、非表示/アイコン表示/通常表示の三種類を対象とする。配置候補領域の大きさは多様であり、「通常表示」に限らず、「非表示」や「アイコン表示」が適切なものもある。本ステップでは、複数の配置候補領域の各々につき、その大きさに注目して適切な表示様式を決める。

各配置候補領域の幅と高さを「小さい」、「中くらい」、「大きい」の三つの水準に区分けし、幅と高さのこれらの組合せに対して、適切な表示様式を対応させる。この考え方から以下のルールが導かれる。

if $w_i=PS$ and $h_i=ANY$ then $k_i=ND$
..... (1)

if $w_i=ANY$ and $h_i=PS$ then $k_i=ND$
..... (2)

if $w_i=PM$ and $h_i=PM$ then $k_i=IC$
..... (3)

if $w_i=PM$ and $h_i=PB$ then $k_i=DN$
..... (4)

if $w_i=PB$ and $h_i=PM$ then $k_i=DN$
..... (5)

if $w_i=PB$ and $h_i=PB$ then $k_i=DN$
..... (6)

ここに、 w_i : 配置候補領域 i の幅、 h_i : 配置候補領域 i の高さ、 k_i : 表示様式 (非表示/アイコン表示/通常表示)、 PS : 正で小、 PM : 正で中、 PB : 正で大、 ANY : 任意、 ND : 非表示、 IC : アイコン表示、 DN : 通常表示

ルール(1)は幅が小さいなら「非表示」、ルール(2)は高さが小さいなら「非表示」、ルール(3)は幅が中くらいで高さが中くらいなら「アイコン表示」、ルール(4)は幅が中くらいで高さが大きいなら「通常表示」、ルール(5)は幅が大きく高さが中くらいなら「通常表示」、ルール(6)は幅が大きく高さが大きいなら「通常表示」、をそれぞれ意味する。図5に上記ルールによる推論の様子を示す⁽⁴⁾。

推論結果は図示のごとく多峰性となるので、その最大値を与える k_i を読取ることで「非表示」か「アイ

図5 ウィンドウ表示様式の決定方法
Fig.5. Method of deciding the window style in each possible rectangles.

コン表示」か「通常表示」かを定める。

(3) 対象ウィンドウの表示様式決定 (2)項で、配置候補領域すべてについて、その表示様式は「非表示」が適切であると決定されたときには、対象ウィンドウの表示様式は、「非表示」とするのが適切である。逆に、「アイコン表示」や「通常表示」が適切であると決定された配置候補領域が一つ以上あれば、対象ウィンドウの表示様式は、その状況に応じて、ユーザにとって最も有利となるように決定すべきである。本ステップでは、このような考え方で対象ウィンドウの表示様式を決定する。図6に本ステップでの処理フローを示す。

まず、ステップ31で、すべての配置候補領域が「非表示」と決定されたか否かを判定する。いま、

N_{ic} : 「アイコン表示」と決まった配置候補領域の数

N_{dn} : 「通常表示」と決まった配置候補領域の数
とすると、 $N_{ic}+N_{dn}=0$ のとき、すべての配置候補領域が「非表示」と決定されたことになり、ステップ34で対象ウィンドウの表示様式を「非表示」とする。また、 $N_{ic}+N_{dn}>0$ のとき、対象ウィンドウの表示様

図 6 対象ウィンドウの表示様式の決定フロー
Fig. 6. Decision flow on the style of the objective window.

式を「アイコン表示」か「通常表示」のいずれかに決定すべく、ステップ 32 へ進む。ステップ 32 では「通常表示」の選択肢の有無を判定する。対象ウィンドウがユーザによって手動アイコン化されたものであれば、ユーザの指示を継承して「アイコン表示」とすべきであり、「通常表示」の選択肢はあり得ない。従って、対象ウィンドウが手動アイコン化されたウィンドウであれば、ステップ 35 で対象ウィンドウの表示様式を「アイコン表示」とする。また、対象ウィンドウが手動アイコン化されたウィンドウでなければ、ステップ 33 へ進む。ステップ 33 では、「通常表示」と決定された配置候補領域の有無を判定し、これがないとき、すなわち $N_{dn}=0$ のとき、ステップ 35 で対象ウィンドウの表示様式を「アイコン表示」とする。また、これがあるとき、すなわち $N_{dn}>0$ のとき、ステップ 36 で対象ウィンドウの表示様式を「通常表示」とする。

(4) 対象ウィンドウの現状評価 対象ウィンドウを仮に現在の位置に、現在の表示様式のままで表示するとしたときの、ユーザにとっての満足度を評価するステップである。後述の(5)項の特別な場合として取扱うので、ここでは説明は省略し、評価に用いるルールだけを以下に示す。

if $m_c=ANY$ and $da_c=PB$ then $s_c=PS$
..... (7)

if $m_c=PB$ and $da_c=ANY$ then $s_c=PS$
..... (8)

if $m_c=PS$ and $da_c=PS$ then $s_c=PB$
..... (9)

ここに、 $m_c(=0)$: 対象ウィンドウの移動距離、 da_c : ダメージを受ける部分の全体の面積に占める割合、 s_c : 対象ウィンドウの満

図 7 配置候補領域へ対象ウィンドウの配置規則
Fig. 7. Rules of arranging the objective window on the possible rectangles.

足度

ファジィ推論は、 $m_c=0$ として後述の(5)項の配置候補領域評価と同様に行い、対象ウィンドウの現状の満足度をデファジフィケーションされた値 cs_c として得る。

(5) 配置候補領域評価 対象ウィンドウを配置候補領域へ配置したときの、ユーザにとっての満足度を、個々の配置候補領域について評価するステップである。対象ウィンドウの表示様式は、(3)項で決められた表示様式である。また、対象ウィンドウを配置候補領域のどの位置へ配置するかは、対象ウィンドウの大きさと配置候補領域の大きさとの関係によって決める。図 7 にこれを示す。

(a)図は幅も高さも配置候補領域のほうが対象ウィンドウよりも大きい場合であり、この場合には対象ウィンドウが配置候補領域に含まれ、かつ移動距離が最小となるような位置を選択する。(b)図は幅は配置候補領域のほうが大きい、高さが小さい場合であり、この場合には対象ウィンドウの上端を配置候補領域の上端に合せ、対象ウィンドウの左右端が配置候補領域をはみ出さず、かつ移動距離が最小となるような位置を選択する。(c)図は高さは配置候補領域のほうが大きい、幅が小さい場合であり、この場合には対象ウィンドウの左端を配置候補領域の左端に合せ、対象ウィンドウの上下端が配置候補領域をはみ出さず、かつ

移動距離が最小となるような位置を選択する。(d)図は幅も高さも配置候補領域のほうが対象ウィンドウよりも小さい場合であり、この場合には対象ウィンドウの左上端点を配置候補領域の左上端点に合せる。その結果、対象ウィンドウが表示スクリーンをはみ出す場合には、表示スクリーン内に収まるように位置補正する。上記により対象ウィンドウを再配置すると一般にその位置が変わる。この現在位置からの移動距離が大きければユーザは混乱するであろう。そこで、この移動距離を満足度を左右する要因の一つと考える。また、対象ウィンドウを再配置したとしても、対象ウィンドウの表示は一般に、より表示優先度の高いウィンドウによってダメージを受けウィンドウ全体が100%ユーザに見えるとは限らない。対象ウィンドウの大きさが、配置候補領域の大きさを上回る場合もあるからである。このダメージを受ける部分の大きさが満足度を左右するもう一つの要因と考えられる。本ステップでは、これら二つの要因を総合して満足度を決定する。対象ウィンドウの移動距離および受けるダメージの大きさを、「小さい」、「大きい」の二つの水準に区分けし、これらの組合せに対して、対象ウィンドウの満足度を対応させる。この考え方から、以下のルールが導かれる。

if $m_i = ANY$ and $da_i = PB$ then $s_i = PS$
.....(10)

if $m_i = PB$ and $da_i = ANY$ then $s_i = PS$
.....(11)

if $m_i = PS$ and $da_i = PS$ then $s_i = PB$
.....(12)

ここに、 m_i :対象ウィンドウの移動距離、
 da_i :ダメージを受ける部分の全体の面積に
占める割合、 s_i :対象ウィンドウの満足度

ルール(10)は受けるダメージが大きいなら満足度は小さい、ルール(11)は移動距離が大きいなら満足度は小さい、ルール(12)は移動距離が小さく、受けるダメージも小さいなら満足度は大きい、をそれぞれ意味する。図8に上記ルールによる推論の様子を示す。

このようにして、対象ウィンドウを配置候補領域 i へ配置したときの、ユーザにとっての満足度をデファジフィケーションされた値 cs_i として得る。

(6) 配置候補領域選出 満足度が現状を大きく上回る配置候補領域を選出するステップである。前記 cs_i と cs_c との差をとり、これを「負で大」、「ほぼゼロ」、「正で大」の三つの水準に区分けし、これらに対して、「非選出」、「選出」を対応させる。この考え方

図8 対象ウィンドウの満足度の評価方法
Fig. 8. Method of evaluating the degree of satisfaction on the objective window.

図9 配置候補領域の選出方法
Fig. 9 Method of electing the possible rectangles.

から以下のルールが導かれる。

if $(cs_i - cs_c) = NB$ then $e_i = NS$ (13)

if $(cs_i - cs_c) = ZR$ then $e_i = NS$ (14)

if $(cs_i - cs_c) = PB$ then $e_i = SL$ (15)

ここに、 e_i :選出可否(非選出/選出)、
 NB :負で大、 ZR :ほぼゼロ、 PB :正で
大、 NS :非選出、 SL :選出

ルール(13)は満足度が現状よりかなり小さいなら「非選出」、ルール(14)は満足度が現状とほぼ同程度なら「非選出」、ルール(15)は満足度が現状よりかなり大きいなら「選出」をそれぞれ意味する。図9に、上記ルールによるファジィ推論の様子を示す。

図 10 実験結果
Fig. 10. Results of the experiment.

推論結果は図示のごとく多峰性となるので、その最大値を与える e_i を読取ることで「非選出」か「選出」かを定める。

(7) 対象ウィンドウの表示位置決定 前ステップでの選出の結果を受けて、対象ウィンドウを現状のままとするか否かを決定するステップである。選出された候補領域がなかった場合には、現状を大きく改善できないので、この場合には、対象ウィンドウは現状のまま、その位置と表示様式とを保持することとする。しかし、選出された候補領域があった場合には、現状を大きく改善できるので、それらの中で最大の満足度を有する配置候補領域の位置へ対象ウィンドウを再配置する。

以上、ユーザによるウィンドウの操作に連動して動作する他のウィンドウの自動再配置の一手法について述べた。

なお、ウィンドウのオープン時には、まずユーザに最も有利な位置にデフォルトまたは元の大ききでオープンし、その後、上記の自動再配置が動作するようにする。

このウィンドウオープン時の処理は上記の自動再配置とほぼ同様で、ここでは説明は省略するが、上記7ステップのうちで、(2)、(3)項を省略した形である。また、(1)項ではオープンするウィンドウ以外のすべてのウィンドウで複合長方形を形成し、それ以外の領域について配置候補領域を列挙する。列挙の方法は前記と同じである。更に、(4)、(5)項における対象ウィンドウの現在位置と大ききさは、デフォルトまたは元の位置と大ききとし、 da_c 、 da_i はオープンするウィンドウが、他にダメージを“与える”部分の全体の面積に占める割合とする。

5. 実験結果

ユーザによるウィンドウ操作には、ウィンドウのオープン、アクティベート、サイズ変更、フルスクリーン化、移動、クローズ、アイコン化があるが、結果の自明なフルスクリーン化を除いて、比較的良好に使用され、他のウィンドウの表示に大きなダメージを与える可能性のあるサイズ変更（拡大）とオープンについて実験を行った。

実験結果を現行方式と対比して図10に示す。上段がアクティブウィンドウのサイズ拡大、下段がウィンドウオープンである。

ノンアクティブなウィンドウが受けるダメージの量に注目すると、提案方式と現行方式とでその差が顕著であり、提案方式のほうがユーザにとって有利であることがわかる。また、ウィンドウの移動距離も一定の範囲内に収まっており、バランスのとれた再配置が行われていると考える。

6. むすび

以上、ユーザによるアクティブウィンドウの操作に連動するノンアクティブウィンドウの自動再配置の一手法として、ファジィ推論を用いた手法を提案し、その実験結果を現行方式と対比して示した。

なお、より有効なファジィ知識の獲得とその検証は、メンバシップ関数の調整も含めて今後の課題である。

コンピュータシステムの操作性向上は、ウィンドウシステムの登場が示すように、人間にとって本質的な課題であり、今後も、より人間に近づいたインタフェースを求めて活発な研究が展開されるものと思われる。

終りに、本研究の機会を与えていただいた当社マイクロエレクトロニクス機器開発研究所所長 春名公一博士、同副所長 遠藤武之博士ならびに、有益な御討論をいただいた関係各位に深甚の謝意を表します。

(平成3年3月4日受付)

文 献

- (1) 石田：「UNIX システムの歴史と最近の動向」，情報処理，27, 1341 (昭 61-12)
- (2) 石田：「OS/2 の現状と展望」，同上，31, 380 (平 2-3)
- (3) J. Johnson, et al.: "The Xerox Star: A Retrospective",

COMPUTER, 22, No. 9, 11 (1989)

- (4) 菅野：ファジィ制御 (平 2) 日刊工業新聞社

津 原 進 (正員)

昭和 45 年九州工業大学電気工学科卒業。同年(株)日立製作所中央研究所に入所。現在，同社マイクロエレクトロニクス機器開発研究所に勤務。小形情報機器の OS およびその応用システムに関する研究開発に従事。情報処理学会，電子情報通信学会会員。