1E2-3

ユーザの作業履歴を考慮した知的ワークスペース構築機構

An Intelligent Workspace Building Mechanism Based on User's Work Histories

吉田 圭佑 *1　大囿 忠親 *1　新谷 虎松 *1
Keisuke Yoshida　Tadachika Ozono　Toramatsu Shintani

*1名古屋工業大学大学院情報工学専攻
Department of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology

Event-transparent windows are useful for reference works. However, it makes window management difficult because users cannot move event-transparent windows easily. We developed an intelligent workspace building mechanism that automatically moves windows to appropriate positions based on user's work histories. The work history consists of a mouse event history, a key event history and an application-switch history. We described an algorithm to determine the optimal position of windows based on user's work history for reference works using event-transparent windows.

1. はじめに

本研究では，利用するアプリケーションの数および種類の変化に柔軟に対応する，知的ワークスペース構築機構を検討した．知的ワークスペース構築機構では，ユーザの作業履歴に基づいて，各アプリケーションウィンドウに表示されている情報の重要度を推定し，ディスプレイに表示する情報評価値が多くなるように位置を変更する．これにより，アプリケーションの数および種類の変化における，ユーザの手動によるワークスペース構築の負担を軽減する．

一般的に PC 上でのタスクは，一つのアプリケーションではなく，複数のアプリケーションを利用して行う．例として，Web ページの情報を閲覧しながら，テキストエディタで文書を編集するといったことが挙げられる．このように，複数のアプリケーションを利用するためには，ユーザはそれぞれのアプリケーションウィンドウを操作し，タスクが行いやすいように配置，すなわちワークスペースを構築する必要がある．

PC 上でのタスクを行う上で，利用するアプリケーションの数および種類は変化する．従来は，アプリケーションの数および種類が変化するたびに，ユーザの手動によるウィンドウマネジメントでのワークスペース構築が必要であった．また，PC の性能の向上およびモニターの画素数および枚数の増加に伴い，ウィンドウマネジメントはより複雑になった．

また，情報の反映作業などの支援のために，アプリケーションウィンドウへのクリックイベント透過性および透明度を変化させる方法がある．しかし，アプリケーションウィンドウのクリックイベント透過性および透明度の変化は，ユーザによるウィンドウマネジメントの負担が大きくなる．ユーザの負担を軽減するために，ウィンドウマネジメントの自動化は必要である．しかし，これまでのウィンドウマネジメント手法は，ユーザの作業状態を考慮していなかった．そこで，本研究では作業履歴を用いることで，ユーザの作業状態をウィンドウマネジメントに反映させる．

本稿では，有効なワークスペースの条件，および作業履歴に基づく表示情報の重要度，およびワークスペース構築の自動化について述べる．

連絡先: 吉田圭佑，名古屋工業大学大学院情報工学専攻，愛知県名古屋市昭和区御器所町，052(735)5584，kyoshida@toralab.org

2. 最適なワークスペースの構築

ユーザは複数のタスク遂行環境を保管し，並行して作業する機会が増えた．ゆえにワークスペースの構築には，利用アプリケーションの組み合わせおよびウィンドウマネジメントの二つを考慮する必要がある．本章では，ウィンドウマネジメントに着目してワークスペース構築について考察する．

2.1 ウィンドウマネジメントの自動化

ウィンドウマネジメントの自動化における重要点は，1) アプリケーションの表示内容，および 2) 直接的なアクティブウィンドウ切替の 2 点である．ここでの目標は，アクティブウィンドウの可読性を高めること，および簡潔にアクセス可能なアプリケーション数の増加を両立することである．

1) のアプリケーションの表示内容に関して，表示されている情報の量を表す特徴量として，特徴含有値マップ M_{wf} を導入する．特徴含有値マップ M_{wf} は，後述する特徴含有値の分布状態を表し，情報の存在を示す．これは，可視状態のアプリケーションウィンドウの数が，ワークスペースの有効性の指標である [Robertson 05] という知見に基づいている．すなわち，特徴含有値マップ M_{wf} は，あるアプリケーションウィンドウを表示する有効性を表すといえる．特筆すべき点として，特徴含有値マップ M_{wf} は，言語および計算機の実行環境に依存しない特徴量である点が優れている．

2) の直接的なアクティブウィンドウ切替に関しては，アプリケーションウィンドウを最小限のマウスクリックのみで切り替えられることを意味する．マウス操作によるアクティブウィンドウ切り替えの傾向として，タスクバーおよび Dock を用いる方法に比べ，マウスクリックによって切り替える方法が多くなされる傾向がある [Hutchings 04]．一方で，単にアプリケーションウィンドウが重ならないように並べるのでは，個々のアプリケーションウィンドウの表示内容の重要さが考慮されておらず不十分である．本研究では，アプリケーションウィンドウ中の各領域の重要さを，関心度マップ M_{wi} として表現する．ここでは，アプリケーションウィンドウ上においてユーザにとって重要な領域は，コンテンツの表示領域のみでは無い点に着目している．例えば，ユーザがアプリケーションウィンドウの一部分のみを，アプリケーションの選択のために可視可能となるように位置決めしている状態がある．このとき，アクティブウィンドウのコンテンツの表示領域の適正化，および可

視アプリケーションウィンドウ数の最大化を図っているともいえる．

2.2 ワークスペース構築

本研究では，1) および 2) を満たすワークスペースの構築を目指す．そこで，ある時点でディスプレイに表示されている情報にどの程度価値があるかを示す値として情報評価値 $M_{dev}(x, y, t_c)$ を導入する．情報評価値 $M_{dev}(x, y, t_c)$ は，M_{wf} および M_{wi} の和によって与えられる．つまり，情報評価値 M_{dev} は，実際に表示されている情報の量およびユーザの関心度を反映した値となる．情報評価値 $M_{dev}(x, y, t_c)$ の和の最大化が，1) および 2) を満たす，最適なワークスペース構築における条件である．式 (1) に，ワークスペースの有効性を示す評価値である $Evaluation$ を示す．

$$Evaluation(t_i) = \sum_{x}^{DW} \sum_{y}^{DH} M_{dev}(x, y, t_i) \quad (1)$$

式 (1) は，デスクトップに表示されている全体の表示内容を評価する．$Evaluation$ は，ディスプレイ表示全体の情報評価値の和を示し，この情報評価値の和が大きいほどディスプレイ表示領域を有効活用しているワークスペースとなる．$M_{dev}(x, y, t)$ はある時刻 t の，ディスプレイ表示の座標 (x, y) の情報評価値を示す．DW および DH はそれぞれディスプレイ表示の横幅および縦幅を示す．

ここで，この $Evaluation$ を増加させるために，自動化できるウィンドウマネジメントの操作値として位置，サイズおよび透明度が挙げられる．本研究では，作業履歴を考慮してアプリケーションウィンドウの位置決定の自動化を目指す．本システムでは，$Evaluation$ の値が大きくなるように，アプリケーションウィンドウを配置する．特筆すべき点として，アクティブウィンドウのみでなく，背後のウィンドウも逐次配置を変更する．これによって，利用アプリケーション数の増減および種類の変更に対して柔軟なウィンドウマネジメントが期待できる．

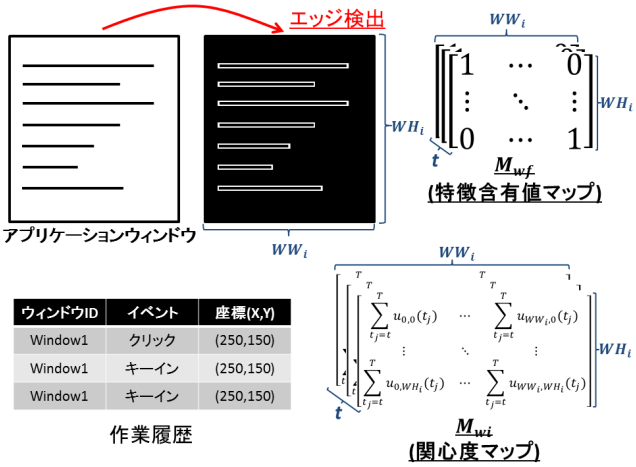
3. 章では，ユーザの関心を反映させた情報評価値 $M_{dev}(x, y, t)$ について述べる．

3. 作業履歴に基づく情報評価値

本章では，デスクトップ表示における情報評価値 $M_{dev}(x, y, z, t)$ について述べる．情報評価値とは，ユーザにとって価値のある情報がどれだけデスクトップ領域に表示しているかを示す．該当するウィンドウの表示位置において，その位置に識別できる情報があるかどうかおよびその位置にユーザが関心があるかどうかは，ユーザにとって価値のある情報の存在を示すために必要である．本研究では，情報評価値 $M_{dev}(x, y, z, t)$ を以下の式で定義する．

$$M_{dev}(x, y, z, t) = \delta(x, y)\left\{\alpha \sum_{t_j=t}^{T} M_{wi}(x, y, z, u(t_j)) + (1-\alpha) M_{wf}(x, y, z, t)\right\} \quad (2)$$

式 (2) は，アプリケーションウィンドウのデスクトップ表示座標 (x, y) に最前面で表示されているウィンドウの情報評価値 $M_{dev}(x, y, z, t)$ を示す．$\delta(x, y)$ は，(x, y) にアプリケーションウィンドウが最前面に存在する場合に 1 を，存在しない場合に



ウィンドウID	イベント	座標(X,Y)
Window1	クリック	(250,150)
Window1	キーイン	(250,150)
Window1	キーイン	(250,150)

図 1: 情報評価値計算のための M_{wi} および M_{wf} の概要図

0 を返す．α は M_{wi} および M_{wf} の値に重みを与える値である．$\sum_{t_j=t}^{T} M_{wi}(x, y, z, u(t_j))$ は，対象ウィンドウへのユーザによる操作履歴による関心度の和を求めている．$u(t_j)$ は，ある時刻 t_j のユーザの操作履歴である．$M_{wf}(x, y, z, t)$ は，ある時刻 t の対象ウィンドウの特徴含有値を示す．情報評価値 $M_{dev}(x, y, z, t)$ は，ウィンドウ表示位置に対応するユーザの関心度（$M_{wi}(x, y, z, u(t))$）および特徴含有値（$M_{wf}(x, y, z, t)$）の和によって与えられる．各アプリケーションウィンドウは，M_{wi} および M_{wf} が与えられる．M_{wi} および M_{wf} の概要図を図 1 に示す．M_{wi} および M_{wf} はそれぞれ関心度マップと特徴含有値マップを示す．$M_{wi}(x, y, z, u(t))$ はある時刻 t における，ユーザの操作結果から与えられる最前面から z 番目のウィンドウの座標 (x, y) の関心度（≥ 0）を示す．$M_{wf}(x, y, z, t)$ はある時刻 t における，最前面から z 番目のウィンドウの座標 (x, y) の特徴含有値（≥ 0）を示す．

M_{wi} は関心度マップであり，アプリケーションウィンドウ内の各座標に対する情報の関心度を示す．あるウィンドウに対して，積極的にキーボード入力イベントおよびマウス入力イベントなどが行われたときに，対象ウィンドウへの関心度が高いとする．また，関心度の計算に用いる作業履歴件数を制限することで，ユーザの関心の移り変わりをワークスペース構築に反映できる．本研究では，ユーザの作業履歴を用いることで，アプリケーションウィンドウ内の情報への関心をデスクトップ表示における情報評価値に反映するため，ユーザの作業履歴として，主要なインターフェースであるキーボードおよびマウスイベントを監視し操作履歴として取得する．関心度に反映させる操作および内容を，アプリケーション切り替え，マウスクリックイベントおよびキーボード入力イベントの三つに分類した．

アプリケーション切り替えではアクティブウィンドウの切り替わりを取得する．アプリケーションを切り替える方法では，ウィンドウを直接クリックする方法，Dock やタスクバーを用いる方法，およびキーボードショートカットによる方法が挙げられる．アプリケーション切り替え操作が行われた時，切り替え先のアプリケーションウィンドウ全体の関心度を上げる．マウスクリックイベントでは，マウス操作による作業として，アプリケーション切り替えが行われない場合のマウスクリックイベントを取得する．マウスクリックイベントが起こった座標の周辺の関心度を上げる．キーボード入力イベントでは，アプリ

ケーション切り替えが行われない場合のキーボード入力イベントを取得する．事前に入力したい箇所にクリックイベントが行われると仮定し，直前のマウスクリックイベントの座標周辺の関心度を上げる．本研究では作業履歴の，直近の一定量を関心度の計算に利用する．直近の一定量の履歴のみを利用することで，作業時に変化していくユーザのアプリケーションウィンドウ内の情報への関心の変化を反映できる．

M_{wf} は特徴含有値マップであり，アプリケーションウィンドウ表示の情報の識別のための情報の分布を示す．本研究では，アプリケーションウィンドウ表示に対してエッジ検出することで，文字や図形の境目，すなわち情報を区別するための情報をエッジとして抽出し特徴含有値とする．エッジ検出では，アプリケーションウィンドウ表示をグレースケール化し，ラプラシアンフィルタを用いて空間フィルタリングする．これにより，アプリケーションウィンドウ内の空き領域を有効活用できる．

4. ワークスペース構築機構

4.1 システム概要

ワークスペースを構築する上で，システムから変化させるウィンドウの操作値の候補として，位置，サイズおよび透明度が挙げられる．本研究では，特に位置を自動で変化させることで各アプリケーションウィンドウの可視性を向上させる．

本システムでは，位置を自動で変化させるタイミングを新規ウィンドウ生成時およびアプリケーション切り替え時とする．

Algorithm 1 新規ウィンドウ表示位置決定

1: **Input:** w_{new}, t_c
2: $evaluation \leftarrow Evaluation(t_c)$
3: $i \leftarrow 0$
4: **while** $isCoverd(w_{new}, w \in W)$ **do**
5: $\quad setPosition(evaluation, w_{new}, i)$
6: $\quad i \leftarrow i + 1$
7: **end while**
8: **for** $w' \in \{w | isOverlap(w_{new}, w), w \in W\}$ **do**
9: $\quad updatePosition(evaluation, w)$
10: **end for**

Algorithm 2 $setPosition(evaluation, w_{new}, i)$

1: **Input:** $evaluation, w_{new}, i$
2: $checkRow \leftarrow screenWidth/w_{new}.size.width$
3: $checkCol \leftarrow screenHeight/w_{new}.size.height$
4: $CA \leftarrow \langle\rangle$
5: **for** $j = 0$ to $checkRow$ **do**
6: $\quad$ **for** $k = 0$ to $checkCol$ **do**
7: $\quad\quad$ **for** $dir \in \{TL, TR, UL, UR\}$ **do**
8: $\quad\quad\quad w_{candidate} \leftarrow candidate(w_{new}, dir, j, k)$
9: $\quad\quad\quad w_{candidate}.score \leftarrow evalPortion(w_{candidate})$
10: $\quad\quad\quad CA \leftarrow append(CA, w_{candidate})$
11: $\quad\quad$ **end for**
12: $\quad$ **end for**
13: **end for**
14: $CA \leftarrow sort(CA)$
15: $w_{new}.pos \leftarrow CA[i].pos$

TL:左上	TR:右上	UL:左下	UR:右下

Algorithm 3 $updatePosition(evaluation, w_{target})$

1: **Input:** $evaluation, w_{target}$
2: $stopCount \leftarrow 0$
3: $current \leftarrow evaluation$
4: **while** $stopCount < 5$ **do**
5: $\quad distance \leftarrow random()$
6: $\quad CA \leftarrow \langle\rangle$
7: $\quad$ **for** $dir \in$ {T, U, L, R, TL, TR, UL, UR} **do**
8: $\quad\quad w_{candidate}.pos \leftarrow move(w_{target}, dir, distance)$
9: $\quad\quad w_{candidate}.score \leftarrow evalAll(w_{candidate})$
10: $\quad\quad CA \leftarrow append(CA, w_{candidate})$
11: $\quad$ **end for**
12: $\quad w_{candidate} \leftarrow MAX(CA)$
13: $\quad$ **if** $current > w_{candidate.score}$ **then**
14: $\quad\quad stopCount \leftarrow stopCount + 1$
15: $\quad$ **else**
16: $\quad\quad stopCount \leftarrow 0$
17: $\quad\quad w_{target}.pos \leftarrow w_{candidate}.pos$
18: $\quad$ **end if**
19: **end while**

T:上	U:下	L:左	R:右
TL:左上	TR:右上	UL:左下	UR:右下

タイミングを新規ウィンドウ生成時とするのは，新規ウィンドウの追加によりワークスペースが大きく変化するタイミングであり，ウィンドウマネジメントが必要になるためである．タイミングをアプリケーション切り替え時とするのは，切り替え前後のアプリケーションウィンドウの M_{wi} が大きく変化するタイミングであるためである．アプリケーションの併用時に切り替え前後のアプリケーションウィンドウの可視性を上げることで，情報の反映作業が支援できる．

新規ウィンドウを生成した時のウィンドウの表示位置決定アルゴリズムを，Algorithm 1 に示す．t_c は，ウィンドウ生成時を示し，$evaluation$ はウィンドウ生成時の情報評価値の和を示す．$isCoverd(w_{new}, w)$ は，w_{new} が $w \in W$（またはその逆）と表示が全て重なっている場合に 1，そうでない場合に 0 を返す．$isOverlap(w_{new}, w)$ は，w_{new} と $w \in W$ の表示が一部重なっている場合に 1，そうでない場合に 0 を返す．Algorithm 1 の 4 から 7 行目では，情報評価の合計が少ない位置，なおかつ新規ウィンドウが他のウィンドウと表示が完全に重ならないような表示位置に決定する．また，Algorithm 1 の 8 から 10 行目では，新規ウィンドウが一部分だけ重なったウィンドウの位置を更新する．これらは，マウス操作によるアプリケーションスイッチの傾向として，タスクバーを用いる場合に比べウィンドウに直接アクセスする場合が多い傾向があるため [Hutchings 04]，アプリケーションウィンドウが全て隠れないようにするためである．

アプリケーション切り替え時のウィンドウの表示位置決定アルゴリズムでは，アクティブ状態切り替え前後のアプリケーションウィンドウに対して，$updatePosition(evaluation, w_{target})$ によって表示位置を更新する．これは，切り替え前後の対象アプリケーションウィンドウの M_{wi} の合計値が高くなり，それ以外のアプリケーションウィンドウの M_{wi} の合計値が低くなるため，最新の作業に適したウィンドウマネジメントが期待できるためである．

情報評価値が少ない矩形領域を探索し表示位置を決定する $setPosition(evaluation, w_{new}, i)$ を Algorithm 2 に示す．

図 2: システム実行前

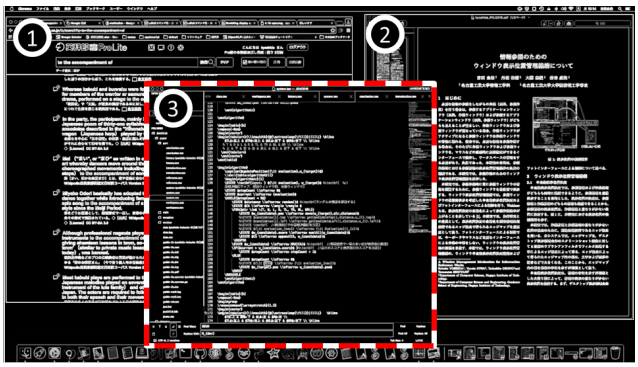


図 3: システム実行後

$setPosition(evaluation, w_{wi}, i)$ は，スクリーンの四隅（左上，右上，左下および右下）を始点として，新規ウィンドウのサイズでグリッド状に分割し，情報評価値が i 番目に少ない位置に表示位置を決定する．Algorithm 2 の 5 から 13 行目は，表示位置候補の評価をしている．$candidate(w_{new}, dir, j, k)$ は，始点 dir から j 行 k 列目の表示位置候補を返す．$evalPortion(w_{candidate})$ は，$w_{candidate}$ の位置の情報評価値の和を返す．$append(CA, w_{candidate})$ は，表示位置候補配列 CA に $w_{candidate}$ を要素として追加した候補配列を返す．Algorithm 2 の 14 および 15 行目では，$sort(CA)$ によって候補位置を情報評価値で並べ替え，i 番目に情報評価値が少ない位置に決定している．

アプリケーションウィンドウの位置を更新する $updatePosition(evaluation, w_{target})$ を，Algorithm 3 に示す．$updatePosition(evaluation, w_{target})$ は，対象ウィンドウを中心に 8 近傍にランダムの距離を移動させた位置を新しい表示位置候補とした山登り探索をする．$distance$ は，対象ウィンドウをどれだけ動かすのかを示し，$random()$ によってランダムな値を得る．Algorithm 3 の 7 から 11 行目では，$move(w_{target}, dir, distance)$ で表示位置候補を求め，表示位置候補に移動した場合のデスクトップ表示全体の情報評価値の和を $evalAll(w_{candidate})$ で求めている．求めた情報評価値の和が最大の表示位置候補が現在の情報評価値の和より大きい場合に位置を更新する．位置の更新が複数回連続で行われなかった場合に，探索を終了する．これにより，柔軟なウィンドウ表示位置の更新が期待できる．

4.2 システム実行例

システム実行前および実行後をそれぞれ図 2 および図 3 に示す．図 2 中の①，②，および③はそれぞれ Web ブラウザ，PDF ビューワーおよび新規ウィンドウのテキストエディタを示している．新規ウィンドウ生成時では，まず新規ウィンドウはエッジの重なりが少ない位置に表示される．次に，新規ウィンドウと一部分が重なる既存ウィンドウを対象に，エッジの重なりが少なくなるように移動する．その結果，実行後は実行前に比べて，既にあるウィンドウの配置が画面端に広がるように変更されている．このように，本ワークスペース構築機構では，ディスプレイに表示される情報量を増加させ，なおかつ他のウィンドウへの直接アクセスができるように自動的にウィンドウマネジメントを行う．

4.3 考察

本研究では，ワークスペースを構築する上で，位置をシステムから変化させるウィンドウの操作値とした．本稿で候補に挙げたサイズおよび透明度について考察を行う．

ウィンドウサイズを自動で変化させるためには，変化するアプリケーションの特徴を理解することが必要である．例えば，高機能エディタでは快適な利用のために大きなウィンドウサイズが求められる．つまり，ワークスペース内の可視状態のアプリケーションの増加のために，ウィンドウサイズを小さくすることは不適である．

透明度を自動で変化させるための指標として，3. 章で述べた M_{dev} が利用できる．例えば，情報評価値が高いアプリケーションウィンドウは不透明にし，少ないアプリケーションウィンドウは半透明にする．しかし，アプリケーションウィンドウを半透明にした時，背後のアプリケーションとの文字の重なりによって視認性の低下が起こる可能性がある．ゆえに，透明度の自動化にあたり，文字の重なりを防ぐような視認性の改善手法 [吉田 15] が必要になる．

5. おわりに

本研究では，ユーザの作業履歴を考慮した知的ワークスペース構築機構を検討した．知的ワークスペース構築に向けて，有効なワークスペースについて考察し，ユーザの作業履歴に基づいた情報の重要度を定義した．ワークスペース構築の自動化では，有効なワークスペースの条件およびユーザの作業履歴に基づいた情報の重要度に従い，各アプリケーションウィンドウについて有効な表示位置を推定および決定をした．本システムによって，ユーザの PC 上でのタスクを行う上でのワークスペース構築の負担の軽減が期待できる．

参考文献

[Hutchings 04] Hutchings, D. R., Smith, G., Meyers, B., Czerwinski, M., and Robertson, G.: Display space usage and window management operation comparisons between single monitor and multiple monitor users, in *Proceedings of the working conference on Advanced visual interfaces*, pp. 32–39, ACM (2004)

[Robertson 05] Robertson, G., Czerwinski, M., Baudisch, P., Meyers, B., Robbins, D., Smith, G., and Tan, D.: The large-display user experience, *Computer Graphics and Applications, IEEE*, Vol. 25, No. 4, pp. 44–51 (2005)

[吉田 15] 吉田圭佑, 丹羽佑輔, 大囿忠親, 新谷虎松 : 重畳 Web ブラウジングにおける視認性改善について, 人工知能と知能処理研究会, 第 115 巻, pp. 25–30 (2015)