3J-4

ＤＴＰにおけるイメージ編集機能

伊藤　精悟　　沢田　一夫　　堤　義直
(東芝　情報通信システム技術研究所)

1　はじめに

最近ＤＴＰに対する関心が高まり、高品位な文書出力が求められるようになってきた。ＤＴＰの要素技術として、ＣＰＵの高性能化はもとより、日本語フォントのアウトライン技術や低価格な高解像度のＬＢＰ技術が確立されつつあり、高品位な文書出力が可能になってきている。また、文字や図形について、高度なレイアウトを行うための使い易いＭＭＩが実現されてきている。　この様な状況下で、ＤＴＰのイメージデータを扱うイメージ編集機能についても、それにあった対応が必要となっている。つまり解像度の高いイメージデータの編集をＤＴＰの文字や図形のレイアウト機能の手軽さで行う機能の実現である。この様なニーズにあったイメージ編集機能をＷＳ上で開発（プロトタイピング）を行ったので報告する。

2　従来の編集機能

イメージ編集機能は、編集のベースとなるイメージデータ（ベースイメージデータ）をスキャナまたはイメージファイルから入力して、そのイメージパターンをＣＲＴ等のディスプレイ装置で表示しながら編集を行うものである。その機能として、移動、複写、拡大／縮小、回転、鏡像等がある。従来のイメージ編集機能において、これらのコマンドの実行には、先ずソース側の切出し範囲指定とそれを合成表示するデスティネーション側の場所を指定することが必要で、例えば、回転コマンドでは回転させる範囲と回転後に表示させる場所を指定する必要があった。コマンドの実行によりイメージデータの切出しから、回転の処理、合成（貼付け）までが一気に行われ、実行後には編集結果としてのベースイメージデータだけが残った（図１参照）。この方法では編集履歴が残されておらず、コマンドのアンドウが困難であり、原則的に失敗は許されないため、コマンドの実行前に実行後のレイアウトを正確に決めておく必要があった。

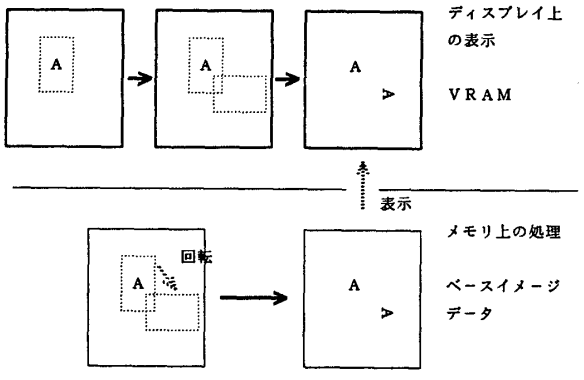

図１　従来例（回転コマンド）

3　ユーザフレンドリな編集機能

ＤＴＰの文字や図形のようにマウスを用いて試行錯誤しながら複雑なレイアウト編集が行えるようになると、当然イメージ機能でもその手軽さが求められることになる。

（１）マンマシンインターフェイス（ＭＭＩ）

イメージ編集において、文字や図形のように、描画後にレイアウトを変える等の編集のアンドウを実現するためには、変更する前のイメージデータをバックアップしておく必要がある。アンドウによりこのバックアップデータから一つ前の状態を得て、そこから再度やり直すという方法である。この方法は一つ前のコマンドに対してのアンドウにはよい方法であるが、ベースイメージのすべてのデータをバックアップする必要があり、扱うイメージデータの解像度が高くデータ量が多い場合は問題がある。特に複数のコマンドに股がるアンドウは事実上不可能である。また、一つ前のコマンドにたいするアンドウだけでは高度なレイアウトが難しく、さらにアンドウという形を取るため、文字や図形のレイアウト編集操作とは感じが違うものとなってしまう。そこで、よりユーザフレンドリな操作方法として文字や図形のように、手軽にレイアウト編集のできる方法を考えた（図２参照）。これは従来方法のようにコマンドの実行によりイメージの切出しから、編集処理、貼付けまでを一気に行うのではなく、これらの３つの処理を別々の

The Image Editor of Desk-Top Publishing

Seigo ITO　Kazuo SAWADA　Yoshinao TSUTSUMI

TOSHIBA　Information and Communication Systems Laboratory

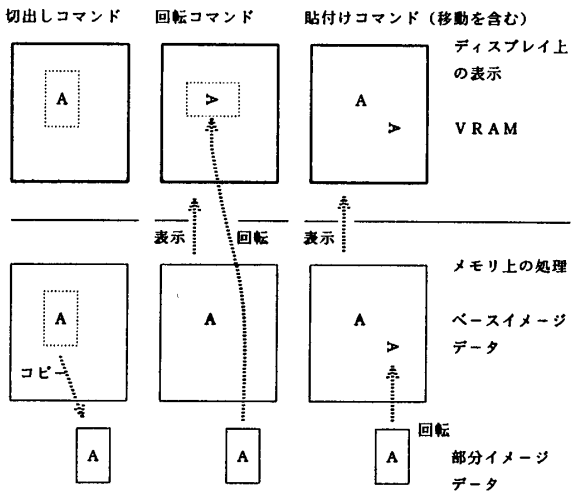

図2　本方式による編集例

コマンドに分けて行うようにしたものである。まず、切出しコマンドで指定したベースイメージ上の範囲のデータをコピーして、1つの独立したイメージデータ（部分イメージデータ）として管理を行う。次に、部分イメージデータを選択して編集コマンドを実行すると、編集処理はこの部分イメージデータに対して行われる。この時その処理結果はＶＲＡＭ上で反映され、ベースイメージデータに対して一切変更は行われない。ＶＲＡＭ上でこの見せ掛けの合成を行うことにより、編集途中（通常の編集）においてはもとのイメージを壊さずに残せるため、後から簡単に変更できるようになる。ベースイメージデータに対する実際の編集（部分イメージデータの合成）は最後に貼付けコマンドで行い、これまでＶＲＡＭにたいして行った一連の編集をベースイメージデータに対して一括して行う。

（2）編集操作スピードの向上

快適な操作を行うためにはその応答速度が速いことが必要である。編集途中はその対象となるイメージデータ全てを管理する必要があるが、そのデータ量は大きなもので、イメージデータは基本的にディスク上でファイル管理を行う必要がある。このため一寸した編集でもファイルアクセスが入り処理速度がダウンしてしまうことになる。これではＭＭＩが使いやすいとしてもその価値が半減してしまう。そこでメモリ上にイメージのキャッシュ領域を作り、メモリ上で扱う全体のデータ量を制限し、ある程度までのイメージデータをここで管理して、それを越える場合はアクセス時間が古いものからファイル上におとして管理する方法（ＬＲＵ）をとった。この方法により通常の編集ではファイルアクセスを意識せずに使える様になった。

またイメージの編集は対象となるイメージ全体をＣＲＴ等のディスプレイ上に表示して行うが、ディスプレイの解像度は一般に扱うイメージの解像度よりも低いため、イメージ全体がディスプレイ内に入るよう密度変換（縮小）して表示する必要がある。編集コマンドの実行によりＶＲＡＭを対象に合成表示が行われることになるが、この場合イメージデータはそのままの解像度でＶＲＡＭに書き込めなくなり、密度変換する必要が出てくる。密度変換はＶＲＡＭに書き込むイメージデータの解像度が高ければ高いほど時間がかかり、これでは快適な操作性は望めない。そこで、もとのイメージデータ（編集用イメージデータ）の他に、これをＣＲＴの解像度に密度変換した表示用のイメージデータを全てのイメージに対して管理するようにし、通常の編集の様にＶＲＡＭを対象にする編集は、表示用イメージデータ同士で素早く行うようにした（図3参照）。例えば回転コマンドの場合、コマンド実行時に回転処理を行うのは解像度の低い表示用イメージデータだけで、編集用イメージデータには一切処理を行わない。但し、その履歴は表示用及び編集用イメージデータを管理するデータ構造に登録しておく。実際の編集は貼付けコマンドの実行により行われ、ここで初めて編集用イメージデータに対して回転処理がなされる。以上の方法を取ることにより操作性のよいイメージ編集機能が実現できた。

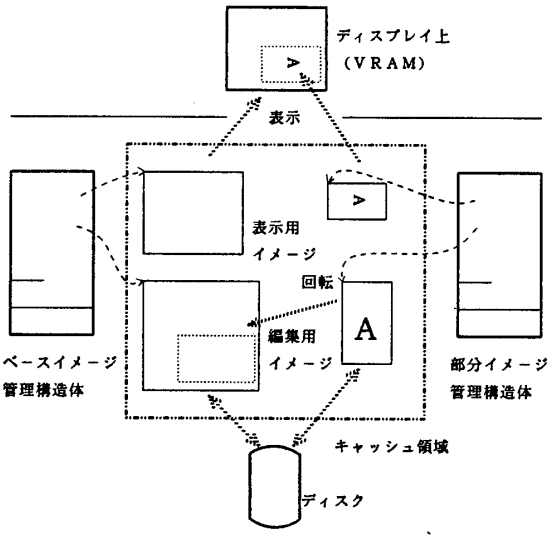

図3　イメージデータ管理方法

4　まとめ

すべてのイメージに対して編集用と表示用の2つのイメージデータを持ち、通常の編集作業を低解像度でデータ量の少ない表示用データを用いてＶＲＡＭ上で確認しながら行い、また、ベースイメージの編集用イメージに対する実際の編集は最後に一括して行うことで、一連の操作が迅速にでき、かつやり直しが簡単で正確なレイアウト決めができるイメージ編集が可能となった。