フルカラー画像を表示可能な8ビットフレームバッファ

4Q－9

滝澤 哲郎　前野 和俊
NEC C&C 研究所

1 はじめに

WebやCD-ROMの普及に伴い、PCでビデオ映像や高品位なグラフィックス画像を目にする機会が増えてきた。一方、PCの表示能力も進化し、低中解像度ではフルカラー画像を表示できるものが一般的になってきたが、高解像度では256色(8ビット)しか表示できないものがまだまだ多い。

表示色数を増やすにはフレームバッファ容量を増やすことで対応可能であり、最近ではメモリコストも下がってきたため以前よりも敷居は低くなってきたが、処理すべき情報量が増えることでグラフィックス性能が低下するという弊害もある。

そこで我々は、8ビットのフレームバッファでフルカラー画像を表示する手段として、DPCMに基づいたデータ圧縮/復元をリアルタイムで行なう方式COLORPACKを開発し、グラフィックスアクセラレータ(μPD72623)に実装した。

本稿では、COLORPACKのアルゴリズムの概要、実現にあたって課題となった領域判別手段、COLORPACKの有効な応用例について述べる。

2 COLORPACK

2.1 アルゴリズム

COLORPACKは、フレームバッファに隣接する画素との差分情報を格納し、表示の際に元の画素値を復元することで、少ない記憶容量で元画像の情報をできるだけ正確に再現することを試みたものである。

限られたビット数で差分情報を保持するため差分値を量子化するが、画像では差分値は小さいことの方が圧倒的に多いため、差分値が小さいほど量子化誤差も小さくなるような非線形量子化を施す。

画像を表現する色空間としては、フレームバッファで通常用いられるRGB色空間と、主にビデオデータで用いられるYUV色空間が代表的である。YUVでは、人間の目が敏感に反応する輝度(Y)はすべての情報を保持するが、人間の目には鈍感な色差(U、V)については情報量を削減するのが普通である。COLORPACKでは画像データを1画素8ビットで保持するため、RGBでは(R:3, G:3, B:2)ビット、YUVでは1画素を(Y:4, UまたはV: 4)ビット(U、Vは連続する2画素が同じ値)とした(YUV 4:2:2フォーマット)。

シミュレーションの結果、RGBでは量子化誤差が1画素あたり約2.2ビット、YUVでは約0.6ビットとなり、主観評価でもYUVの方が圧倒的に良い結果が得られたため、YUVを採用した。

さらに、差分を取る方向を横方向に限定することにより、ライン間での情報の受け渡しを不要とし、バッファをほとんど必要としない回路での実現を可能とした。また、これにより処理がライン内で閉じていることになり、ウィンドウのオーバラップなどにより複雑な形状での表示が要求されるフレームバッファへの適用が可能となった。

圧縮の手順の概要を以下に示し、処理例を図1に示す(図はYのみを例示)。

1. 各ラインの最初の画素は、1画素を16ビットで保持する(Y: 6, U: 5, V: 5)
2. 2画素目は、1画素目を重複して出力する(フレームバッファ上には保持しない)
3. 3画素目以降は、Yデータは毎画素4ビットの差分コードで保持し、UおよびVデータはそれぞれ交互に1画素置きに4ビットの差分コードで保持する。なお、量子化誤差の蓄積を防ぐため、差分は隣接画素の復元結果との間で取る
4. 1画素しか幅がない部分では、1画素を8ビットで保持する(Y: 4, U: 2, V: 2)

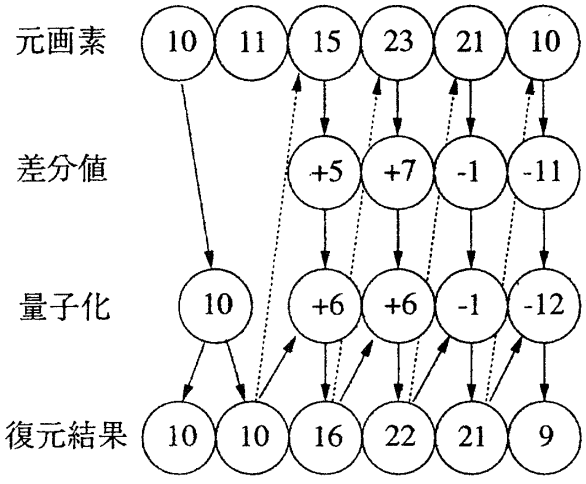

図1: COLORPACK処理の例

Full Color Images on an 8bpp Frame Buffer
Tetsuro Takizawa,
Kazutoshi Maeno, NEC C&C Research Laboratories

2.2 領域判別手段

COLORPACKでは、フレームバッファ上で通常のデータと圧縮したデータがそれぞれの表示位置に従って混在する。通常のデータはルックアップテーブル(LUT)を通ってDAコンバータ(DAC)へと入力されるが、圧縮したデータはCOLORPACKの復元部を通ってDACへと入力されるというように、フレームバッファからリードした後の処理が異なる(図2)。そのため、何らかの手段で両者の判別を行なう必要がある。

データの8ビットのうちの1ビットを判別用に使ってしまうと、通常のデータでは128色しか使えず、圧縮したデータでは画質が低下するので問題外である。そこで、フレームバッファ上に表示データとは独立した領域判別用のデータを取ることにした。

一般に、フレームバッファには空き領域(オフスクリーン)が存在するため、領域判別用データを保持することによるコストの増加はほとんど発生しない。オフスクリーンの空き具合によって、1画素あたり1～2ビット分の情報を確保し、表示時に参照することによって各画素が圧縮されているかいないかを判定する。オフスクリーンに1画素あたり1ビット分の容量しか確保できない場合は、互いに隣接した複数の画像を同時に扱うことができないが、2ビット分の容量を確保できれば、互いに隣接した画像があってもその境界を識別することが可能となるため、表示上の制限を一切なくすことができる。

例えば、画面の解像度が1280x1024の場合は、表示データが1.25 MBを占めるが、この解像度を表示可能なフレームバッファの容量は通常2 MBあるため、1画素あたり2ビットの領域判別用データ(320 KB)を保持することができる。

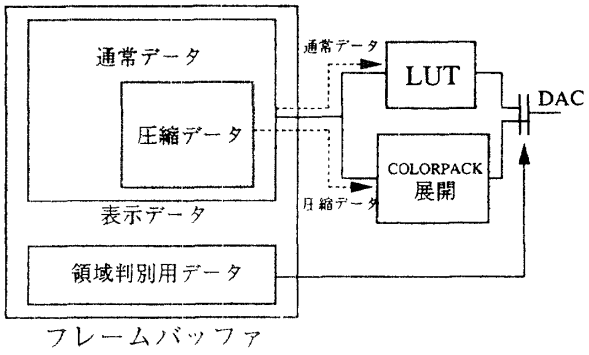

図 2: COLORPACKの表示プロセス

3 COLORPACKの応用と利点

3.1 ビデオ表示

ビデオ表示は256色表示での画質低下が大きく、フルカラー表示が特に要求される。グラフィックス表示が256色でも、ビデオ表示処理をグラフィックス表示処理とは別処理し、両者を合成表示することによりビデオ表示を常にフルカラー表示とする方法(オーバレイ)もあるが、複数のビデオを同時表示することが困難である(表1)。

例えば、ビデオ会議では複数のビデオ表示が要求される局面があるが、COLORPACKを使用すると容易に対応可能である。

3.2 画像表示

従来、8ビットのフレームバッファでのフルカラー画像表示では、ディザによる減色処理を施すか、使用頻度の高い色を優先的にLUTに割り当てるといった策が取られてきた。LUTを最適化する方法では、画像の種類によっては良い結果が得られるが、複数の画像を同時に表示すると画質が低下する(表1)。

例えば、Webブラウザでは複数の画像を同時に表示することが多いが、COLORPACKを使用するとこれらの画像をすべてフルカラー表示することが可能である。

表 1: 各方式の比較

	コスト	画質	複数画像	ビデオ
COLORPACK	○	○	○	○
オーバレイ	△	○	×	○
ディザ	○	×	○	○
LUT最適化	○	△	×	×
24bpp	×	○	○	○

4 おわりに

我々は、8ビットのフレームバッファでフルカラー画像を表示する方式COLORPACKを開発し、本稿ではCOLORPACKのアルゴリズム、課題とその解決策、応用例と利点について述べた。また、本方式をグラフィックスアクセラレータに実装し、ビデオ表示や画像表示に有効なことを確認した。

本方式はコンパクトな回路規模(圧縮: 1.5 Kゲート、復元: 2 Kゲート)で実装可能なため、PCはもちろんのことコストセンシティブなNCやInternet端末などにも向く。今後は、PCにおけるソフトウェア環境の整備とともに、PC以外の端末への応用も考えていきたい。