DRAM内臓マイクロコントローラを用いた画像・音響処理技術に関する研究

静岡大学博士論文

DRAM内蔵マイクロコントローラを用いた画像・音響処理技術に関する研究

１９９９年９月

坂本　直史

あらまし

マイクロプロセッサの領域では技術進歩が急激である。1971年に発表された最初のマイクロプロセッサは語長4ビット、集積トランジスタ数約2300のものにすぎなかった。その後、半導体の集積度の向上とともに、語長8ビットのもの、語長16ビットのものへと発展した。そして、語長128ビットのマイクロプロセッサが組み込まれたゲーム機の開発が既に発表され、家庭に近々登場する。

マイクロプロセッサと半導体の集積度の向上とともに、小規模なマイクロコンピュータを構成するCPU（Central Processing Unit）とメモリと周辺インタフェースのすべてを1個の集積回路で実現したシングルチップ・マイクロコントローラ（以下では単にマイクロコントローラと呼ぶ）も発展した。マイクロコントローラは、比較的低コストで簡単に使用できるため産業分野に浸透し、家庭用電気製品を始めとする民生機器や、ロボット等の産業用機器にも組み込まれ、使用されるようになった。近年のマイクロコントローラは、語長32ビットの高性能なものが出現し、従来の民生機器や産業用機器を制御する用途以外の新しい分野に適用されはじめた。それは、32ビット・マイクロコントローラの高性能でかつ低消費電力という特徴を活かし、携帯端末、デジタル・スチル・カメラやゲーム機といった、従来の家庭用電気製品に画像、音声や通信といった情報処理的な側面を持つ製品群の分野である。これらの製品群においてマイクロコントローラは、従来からの機器制御に加えて、画像や音声の圧縮伸張処理、液晶ディスプレイへのグラフィック表示、通信制御等の処理も担っている。特に、これらの製品のほとんどは、入力デバイスとしてCCD（Charge-Coupled Device）を、出力デバイスとして液晶ディスプレイやスピーカを持つものが多い。従って、32ビット・マイクロコントローラを用いたデータの入出力処理としての画像・音響処理技術は今後極めて重要な技術となっていくものと考えられる。

さらに、マイクロコントローラを用いた画像・音響処理技術の重要性は、半導体の高集積度化に伴い、今後ますます高くなっていくと考えられる。過去にマイクロコンピュータが半導体の集積度の向上に伴い1個の集積回路で実現されたように、マイクロコントローラは大容量のメモリと特定用途向けハードウェアと共に、大規模なシステムLSIに組み込まれ使用され始めている。このようなシステムLSIにおいては、マイクロコントローラを内蔵メモリと広い巾の内部バスで接続できる。そのため、大規模のデータを扱う画像・音響処理が一層高速に処理可能となる。従って、近く訪れるシステムLSI時代を念頭に置くと、大容量メモリを内蔵し広い内部バスでメモリと接続したマイクロコントローラを用いて実現する画像・音響処理技術の用途は広がり、その実現手法の検討は重要な課題であると考える。

ところが、マイクロコントローラには、コンピュータやワークステーションに使用されるマイクロプロセッサとは異なり、搭載されたハードウェア資源に制約がある。しかし、市場から求められる、画像およびオーディオデータの圧縮伸張処理やグラフィック処理といった画像・音響処理は、コンピュータやワークステーション上でおこなわれているのと同等のものである。そこで本論文では、処理性能の面でも劣り、ハードウェア

資源にも制約があるマイクロコントローラを用いて市場の要求する画像・音響処理いかに現実のものにするかという課題に取り組んだ。

大容量メモリを内蔵し広い内部バスでメモリと接続したマイクロコントローラを前提としても、従来の画像・音響処理技術をこのようなマイクロコントローラを用いたシステムにそのまま適用することはできない。適用するには、マイクロコントローラのもつハードウェア資源の制約を考慮にいれて、画像・音響処理の演算アルゴリズム、演算精度、メモリの使用方法等を工夫していく必要があると考えられる。従って、マイクロコントローラの持つ制約から生じる画像・音響処理実現上の問題を、製品仕様から特定できる境界条件を利用しながら、解決するための処理技術の開発を本論文の主要な目的とする。

そこで、マイクロコントローラの持つハードウェア資源の制約と画像・音響処理技術との関係について先ず検討する。特に、マイクロコントローラの持つハードウェア資源の活用、大容量メモリとして用いられるDRAMの特性を活かしたメモリ格納方法、広いバス巾を活用したメモリアクセス方法や人間の視聴覚モデルを用いた画像・音響処理の高速化等について検討する。そして、画像・音響処理技術をマイクロコントローラ上で実現する場合のマイクロコントローラ特有の開発手法を提案する。

提案するマイクロコントローラ特有の画像・音響処理技術の開発手法は、マイクロコントローラを用いて画像・音響処理技術を開発する場合に考慮すべき手法について概念的に説明したもので、各々の処理技術で重要となる課題も異なれば、課題解決のための適用手法も異なる。そこで、実際に市場要求のある４つの画像・音響処理技術、つまり、デジタル・スチル・カメラ用撮像信号処理、静止画像処理、３次元グラフィックス処理、オーディオ復号処理に対して、提案するマイクロコントローラ特有の開発手法が如何に適用できるかについて具体的に検討し、それぞれの画像・音響処理技術に対して課題を明らかにし、提案手法の適用の仕方を詳細に検討する。

これにより、市場要求を満たし、画像・音響処理技術の適用範囲を広げ、マイクロコントローラを用いた新たな製品群を創出することが可能になると考える。

目次

第１章　序論

1.1 本研究の背景

コンピュータは、命令の実行と計算を行うCPU（Central Processing Unit)、プログラムやデータを蓄えるメモリと周辺機器とで構成できる。そして、これらコンピュータの構成要素はバスと呼ぶ共通の信号線群で結合される。CPUはコンピュータの中心的役割を担い、その回路構成は複雑である。しかし、マイクロエレクトロニクスの進歩により、このCPUを１個の集積回路で実現することが可能になった。これをマイクロプロセッサと呼ぶ。

マイクロプロセッサの出現により、コンピュータを１個のマイクロプロセッサとメモリおよび周辺機器を接続するための周辺インタフェースとで簡単に構成できるようになった。これをマイクロコンピュータと呼ぶ。小規模なマイクロコンピュータの場合には図1.1に示すように、CPUとメモリと周辺インタフェースのすべてを１個の集積回路で実現することも可能である。これをシングルチップ・マイクロコントローラ（以下では単にマイクロコントローラと呼ぶ）と呼ぶ。

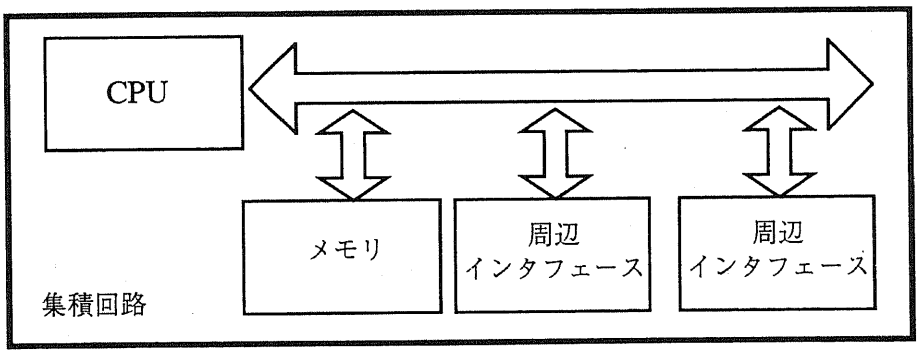


図1.1　シングルチップ・マイクロコントローラ

マイクロプロセッサの領域では技術進歩が急激である。図1.2にマイクロプロセッサ開発の流れを示す。1971年に発表された最初のマイクロプロセッサ、インテル社の4004は語長4ビット、集積トランジスタ数約2300のものにすぎなかった。その後、半導体の集積度の向上とともに、語長8ビットのもの、語長16ビットのものへと発展し、現在では百万を越えるトランジスタを集積した語長128ビットのものが開発されている[1)2)]。

このような語長や機能の拡大とともに、マイクロプロセッサはその応用分野を拡大した。そして、1976年頃からは、マイクロプロセッサを中心として、プログラムを記憶するROM、データを記憶するRAM、周辺入出力機器とのインタフェース回路等をすべて搭載したマイクロコントローラが登場した。これは、比較的低コストで簡単に使用できるため、産業分野に浸透し、家庭用電気製品を始めとする民生機器や、ロボット等の産業用機器にも組み込まれ、使用されるようになった。このマイクロコントローラも、語長8ビット、16ビットとビット数の拡大を続け、現在32ビットのものの使用が拡大してきている。

また、1979年からプロセッサの第3の流れとも言うべきプロセッサ群が現れた。信号処理の応用に特化した構成を持つプロセッサでDSP（Digital Signal Processor）と呼ばれるものである。基本的には、音声や画像のフィルタ等信号処理で必須となる積和演算を高速化することを目的とし、ハードウェアに工夫をこらしたものである。

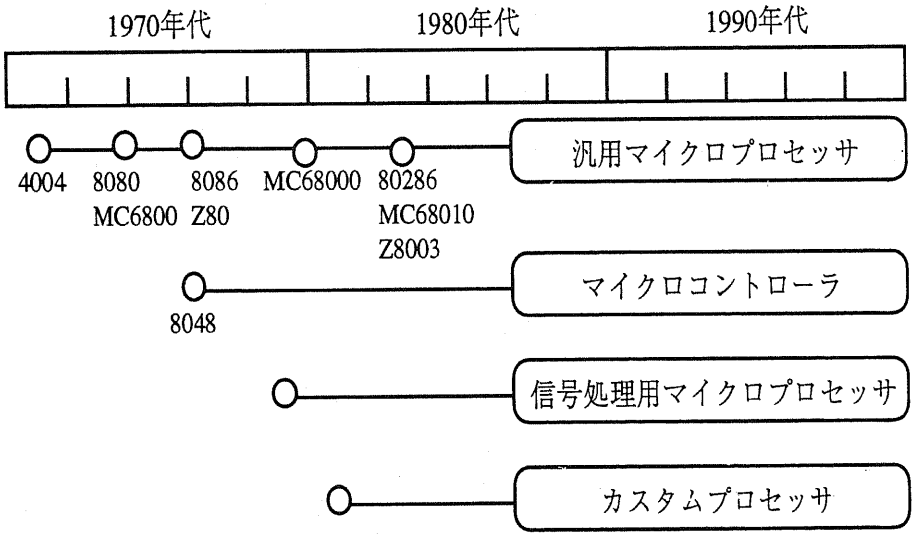


図1.2　マイクロプロセッサ開発の流れ

さて、民生機器や産業用機器に組み込まれ、使用されるようになったマイクロコントローラは、その使用目的から、コンピュータやワークステーションに使用されるマイクロプロセッサとは異なった要求を市場から求められている。マイクロコントローラでは、コンピュータやワークステーションに使用されるマイクロプロセッサと同じように高い性能も要求されるが、同時に安価な価格、低い消費電力、機器に実装しやすいパッケージ形態などが求められる。

安価な価格を実現するためには、マイクロコントローラのチップサイズを小さくし、安価なパッケージを使用する必要がある。チップサイズを小さくするためには、民生機器や産業用機器で使用されることの少ないハードウェア資源を搭載しないことが多い。例えば、浮動小数点演算器の実現には広いチップ面積が必要だが、民生機器や産業用機器の用途では浮動小数点演算を必要とすることが少ない。そのため、ほとんどのマイクロコントローラは浮動小数点演算器を搭載していない。また、乗算器の実現も広いチップ面積が必要だが、乗算は比較的使用することが多いため、入力データのサイズの小さな乗算器を搭載することが多い。一方、セラミック素材の高価なパッケージではなく、安価なプラスチック・パッケージを使用するには、動作時の発熱量が問題となる。この発熱量の観点から、キャッシュ・メモリ、浮動小数点演算器や乗算器の搭載に取捨選択が発生する。

このような市場からの厳しい要求に応えながら、近年のマイクロコントローラは、語長32ビットの高性能なものが出現し、従来の民生機器や産業用機器を制御する用途以外の新しい分野に適用されはじめた。それは、32ビット・マイクロコントローラの高性能でかつ低消費電力という特徴を活かし、携帯端末、デジタル・スチル・カメラやゲーム機といった、従来の家庭用電気製品に画像、音声や通信といった情報処理的な側面を持つ製品群の分野である。

これらの製品群においてマイクロコントローラは、従来からの機器制御に加えて、画像や音声の圧縮伸張処理、液晶ディスプレイへのグラフィック表示、通信制御等の処理も担っている。特に、これらの製品のほとんどは、入力デバイスとしてCCD（Charge-Coupled Device）を、出力デバイスとして液晶ディスプレイやスピーカを持つものが多い。従って、32ビット・マイクロコントローラを用いたデータの入出力処理としての画像・音響処理技術は極めて重要な技術である。

さらに、マイクロコントローラを用いた画像・音響処理技術の重要性は、半導体の高集積度化に伴い、今後ますます高くなっていくと考えられる。過去にマイクロコンピュータが半導体の集積度の向上に伴い1個の集積回路で実現されたように、マイクロコントローラは大容量のメモリと特定用途向けハードウェアと共に、大規模なシステムLSIに組み込まれ使用され始めている。このようなシステムLSIにおいては、マイクロコントローラを内蔵メモリと広い巾の内部バスで接続できる。そのため、大規模のデータを扱う画像・音響処理が一層高速に処理可能となり、その応用分野も広がると考える。従って、近く訪れるシステムLSI時

代を念頭に置き、大容量メモリを内蔵し広いバスでメモリを接続したマイクロコントローラを用いて実現する画像・音響処理技術の実現手法の検討は重要な課題であると考える。

1.2 本研究の目的

本論文の主要な課題は、大容量メモリを内蔵し広いバスでメモリを接続したマイクロコントローラの最高性能と価値を引き出す画像・音響処理技術を示すことにある。最高性能を引き出すためには、それらの処理がどのような応用に、そしてどのような製品企画の制限下で使われるかを理解することが重要である。そして、市場要求を満たすための課題を見い出し、その課題を克服する手段や方法を見つけ出す必要がある。

1.2.1 市場分野からの要請

現在32ビット・マイクロコントローラが使われている主要な製品分野を表1.1に示す。表1.1に示した製品群において、マイクロコントローラは1.1節に示したように従来の機器制御用途以外に、画像、音声や通信といった情報処理的な用途にも使用されている。これらの製品開発にあたり、マイクロコントローラの性能を活かしソフトウェアで実現することを期待されている画像・音響処理に次のようなものが挙げられる。

(1) CCDから入力したデータからRGBデータを復元する撮像信号処理
(2) カラー静止画像の圧縮伸長
(3) グラフィック機能
(4) オーディオデータの復号
(5) 音声処理
(6) 通信制御

表1.1　32ビット・マイクロコントローラが使われている主要な製品分野

製品分野	製品
カメラ関連	デジタル・スチル・カメラ、デジタル・ビデオ・カメラ、プリンタ
情報端末関連	カー・ナビゲーション・システム、携帯端末
ゲーム関連	ゲーム機

また、これらを製品に組み込み使用する場合の位置付けを図1.3に示す。これらの画像･音響処理はミドルウェアと呼ばれる形の汎用ライブラリとして実現し、応用プログラムに組み込まれ使用される。

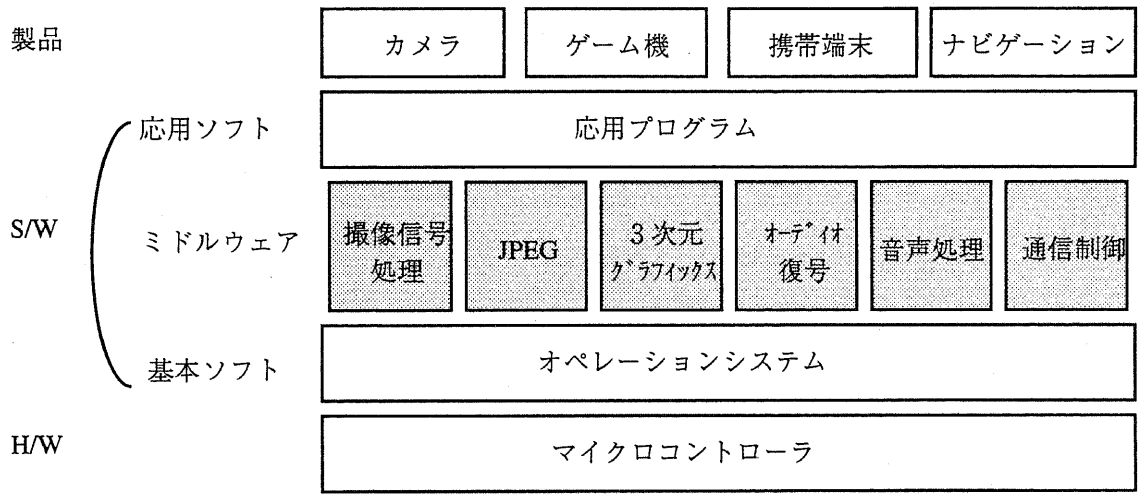


図1.3　ミドルウェアの製品での位置付け

(1)の撮像信号処理は、特にCCDを入力デバイスとして持つ機器で必要となる処理で、デジタル・スチル・カメラやデジタル・ビデオ・カメラ等で用いられる。

(2)のカラー静止画像の圧縮伸長は、カラー静止画像を取り扱う機器で必要となる処理で、デジタル･スチル･カメラやカラー静止画像の出力機器であるビデオ･プリンタや携帯端末等で用いられる。

(3)のグラフィック機能は、ユーザに直観的にわかりやすい画面を必要とする機器で必要となる処理で、カー・ナビゲーションやゲーム機等で用いられる。

(4)のオーディオ復号処理は、音楽を聞くための出力デバイスを持つ機器で必要となる処理で、携帯端末やゲーム機等で用いられる。

(5)の音声処理は、音声を用いた入出力手段としての音声合成処理や音声認識処理で、カー・ナビゲーション・システムやゲーム機等で用いられる。音声合成処理は、人間が音声を発声するメカニズムを実現し人工的に音声を生成する処理で、その方式は、録音編集方式、パラメータ編集方式と規則合成方式の３つに分類できる[3]。また32ビット・マイクロコントローラを用いたミドルウェアとして実現する際の手法として、プログラムと音声の波形データを含むデータを併せて500Kバイト程度の少ないメモリで実現する規則合成方式を用いた手法[4]や使用頻度の高い合成音に特化して良好な合成音を生成する手法[5]等が研究されている。また、音声認識処理は、音声波に含まれる意味内容に関する情報を抽出し判定する処理で、学習を行った特定の話者の音声のみを認識する特定話者型と、誰の音声でも認識できる不特定話者型に分類できる[3][6]。音声認識を用いたカー・ナビゲーション・システムでは、はじめは特定話者向け音声認識が用いられていたが、現在は不特定話者向けが主流である。

(6)の通信制御は、携帯端末を始めとするデータ転送が必要な機器で必要となる処理である。携帯端末等で利用可能な通信デバイスは多岐にわたる。アナログ電話回線とモデム、ISDN回線とターミナルアダプタ、携帯電話やPHS（簡易型携帯電話システム）等を使ったリモートアクセスやLAN接続やIrDA（Infrared Data Association）を使ったローカルアクセス等が現在普及している。これらの中で、PHSシステムを使ったPIAFS（PHS Internet Access Forum Standard）を用いたデータ通信や、モデムやIrDAを用いたデータ通信は32ビット・マイクロコントローラを用いたミドルウェアとして実現可能である[7]。また、今後の有望な通信手段であるスペクトラム拡散変調方式を用いたデジタル無線通信[8]も、2GHz帯程度の周波数帯の電波を使用する場合はミドルウェアとして実現可能である。例えば、スペクトラム拡散変調方式の応用システムとして、衛星航法に利用されているGPS（Global Positioning System）の復調処理のベースバンド部分を32ビット・マイクロコントローラを用いて実現する方式も検討されている[9][10]。

1.2.2 従来の画像・音響処理技術の課題

1.1節に述べたように、マイクロコントローラは、コンピュータやワークステーションに使用されるマイクロプロセッサとは異なり、搭載されたハードウェア資源に制約がある。しかし、市場から求められる、画像およびオーディオデータの圧縮伸張処理やグラフィック処理といった画像・音響処理は、コンピュータやワークステーション上でおこなわれているのと同等のものである。処理性能の面でも劣り、ハードウェア資源にも制約があるマイクロコントローラを用いて市場要求を実現するには、従来の画像・音響処理技術をマイクロコントローラに適応したときの課題を見い出し、その課題を克服する手段や方法を見つけ出す必要が

ある。本節では、1.2.1節に述べた、市場要求のある画像・音響処理の(1)から(4)に対して、従来の画像・音響処理技術を大容量メモリを内蔵し広いバスでメモリを接続したマイクロコントローラに適応したときの課題を述べる。

(1) デジタル・スチル・カメラ用撮像信号処理における課題

デジタル・スチル・カメラが一般に普及したのは、この3年程のことである。更に、デジタル・スチル・カメラ専用のCCDが開発されたのはこの1、2年のことである。現在では、高画質を要求されるデジタル・スチル・カメラのCCDには、ベイヤー型の原色フィルタアレイ[11]が採用されることが多い。

ベイヤー型の原色フィルタアレイは図1.4に示すような構造を基本としている。つまり、Gを市松に配列し、残りの部分におのおのRBを市松に配列する構造である。これにより、輝度信号の主要成分であるG信号の含まれる割合を多くし、残りの部分に比較的解像度が低くてよい他の色信号を配列している。

このCCDからの入力データからRGB 3つの色成分を復元するのが撮像信号処理である。その中でも特に、1画素あたりRGB 3つの色成分のうち1つの色成分しかないCCDの入力データから、残り2つの色成分を周囲の画素の色情報から生成する画素補間処理は、画質を決める上で極めて重要である。

画素補間処理は、幾何学的変換処理として従来研究されてきた技術を転用するのが普通である。しかし、各種の幾何学的変換方式を画素補間方式として用いたときの適応性の比較や、特にベイヤー型の原色フィルタアレイに適した画素補間方式の検討、ソフトウェアで画素補間を実現する場合に適した方式の検討、演算量と画質の関係など、検討されていない課題が多い。

R	G	R	G	R	G	
G	B	G	B	G	B	
R	G	R	G	R	G	
G	B	G	B	G	B	
R	G	R	G	R	G	
G	B	G	B	G	B	

図1.4　ベイヤー型原色フィルタアレイの構造

(2) 静止画像処理 JPEG への適用における課題

国際標準であるカラー静止画像の圧縮方式JPEG（Joint Photographic Experts Group）[12]は、デジタル・スチル・カメラ、デジタル・ビデオ・カメラの静止画像取り込み機能、プリンタ等に広く採用されている。この方式では種々の画像圧縮方式が規定されているが、応用製品のほとんどの場合は、その中でもカラー画像を１０分の１から２０分の１程度に圧縮しても画質の劣化が少ないことに特徴がある非可逆の圧縮方式を採用している。

デジタル・スチル・カメラの場合、シャッターを押してから記憶メディアに記憶するまでの間、人間が待っていて不快感を持たない時間は数秒程度と考えられる。そのためには、デジタル・スチル・カメラで標準的に用いられるVGA（Video Graphics Array）サイズの画像（640 x 480 画素）のJPEG圧縮を0.5秒程度で処理する必要がある。しかし、マイクロコントローラとDRAMを使用した場合には、この処理時間を達成するのは難しいという問題がある。この問題を解決するためには、メモリ使用方法や演算アルゴリズムを工夫した高速化手法の検討という課題がある。

(3) 3 次元グラフィックスへの適用における課題

1.1節に述べたように、マイクロコントローラには、浮動小数点演算器や32ビット×32ビットの入力データサイズを持つ乗算器は通常搭載されていない。3次元グラフィックスを使用するゲーム機では、浮動小数点演算器を搭載した高価なマイクロコントローラやマイクロコントローラとは別にグラフィックス用のアクセラレータが用いられている。ゲーム機の高級機は今後ともこのようなやりかたで実現されると考えられる。一方で、携帯型の安価なゲーム機の分野では、部品価格を低く抑えるため、特別なグラフィックス用のアクセラレータを用いず、浮動小数点演算器を搭載しない通常のマイクロコントローラを用いて3次元グラフィックスを実現したいという市場要求がある。また、カー・ナビゲーション・システムや携帯端末でも同様な3次元グラフィックスに対する市場要求がある。

しかし、浮動小数点演算をソフトウェアでエミュレーションしたのでは3次元グラフィックスは全く性能がでない。従って、浮動小数点演算を用いず、しかも小さなサイズの乗算器を用いて3次元グラフィックスを実現するために、整数演算を用いた演算アルゴリズムを工夫した高速化手法の検討という課題がある。

(4) オーディオ復号への適用における課題

携帯機器において、画像による出力と同様に重要な出力手段としては音による出力がある。音による出力が特に重要な携帯機器として一般に普及しているもの

に音楽再生用の携帯機器がある。近年の音楽（オーディオ）データのデジタル化やインターネットの普及に伴い、オーディオデータをパーソナルコンピュータ上で、更には携帯機器で再生して楽しむようになってきた。このためには、オーディオデータを従来よりも高い割合で圧縮し、しかも従来と同程度の再生時の音質を実現する必要がある。従来よりも高いデータの圧縮率を実現するためには、多くのフィルタ処理が必要となり、その実現には膨大な処理量が必要である。また、従来と同程度の音質を実現するためには、16ビット以上のデータを用い、最終的に16ビット以上の演算精度が必要である。

上述のようなオーディオデータ復号処理をデータサイズの小さな乗算器しか搭載しないマイクロコントローラを用いて実現するためには、乗算のサイズと演算精度を工夫した演算アルゴリズムの開発とフィルタ処理の際のデータや係数を格納するメモリの使用方法を工夫した高速化手法の検討という課題がある。

1.2.3 本研究で用いたマイクロコントローラ

1.1節で述べたように、近年では語長32ビットの高性能なマイクロコントローラが従来の民生機器や産業用機器を制御する用途以外の新しい分野に適用され始めた。32ビット・マイクロコントローラの高性能でかつ低消費電力という特徴を活かし、従来の家庭用電気製品に画像、音声や通信といった情報処理的な側面を持つ製品群の分野である。

本研究の主要な課題は、システムLSIに組込まれたマイクロコントローラを想定し、大容量メモリを内蔵し広いバスでメモリを接続したマイクロコントローラの最高性能と価値を引き出す画像・音響処理技術を示すことにある。従って、研究結果の画像・音響処理技術を実際に汎用的な32ビット・マイクロコントローラ上に実現し、その処理性能や効果を確認することが重要である。表1.2に本研究で主として用いた32ビット・マイクロコントローラM32R/D[13)14)]の諸元を、図1.5にそのハードウェア構成を示す。また、図1.6と図1.7にこのマイクロコントローラを搭載した評価ボードの構成図と装置写真をそれぞれ示す。更に、図1.8に本研究で用いたマイクロコントローラのチップ写真を示す。

表 1.2　本研究で主として用いた 32 ビット・マイクロコントローラの諸元

CPUコア	32ビットRISCアーキテクチャ
VAX MIPS	52.4MIPS @ 66.7MHz (Dhrystone V2.1)
内蔵メモリ	2Mバイト DRAM、2Kバイト キャッシュ
周辺回路	32ビット×16ビット積和演算器、メモリコントローラ等
外部バス	アドレス24ビット、データ16ビット
クロック周波数	内部66.7MHz、外部16.67MHz
電源電圧	3.3V
消費電力	700mW(typical) / 2mW(stand-by)
チップサイズ	153.7mm^2
CPUサイズ	5.7mm^2
プロセス	0.45 μ mCMOS、4 層ポリシリコン、2 層メタル

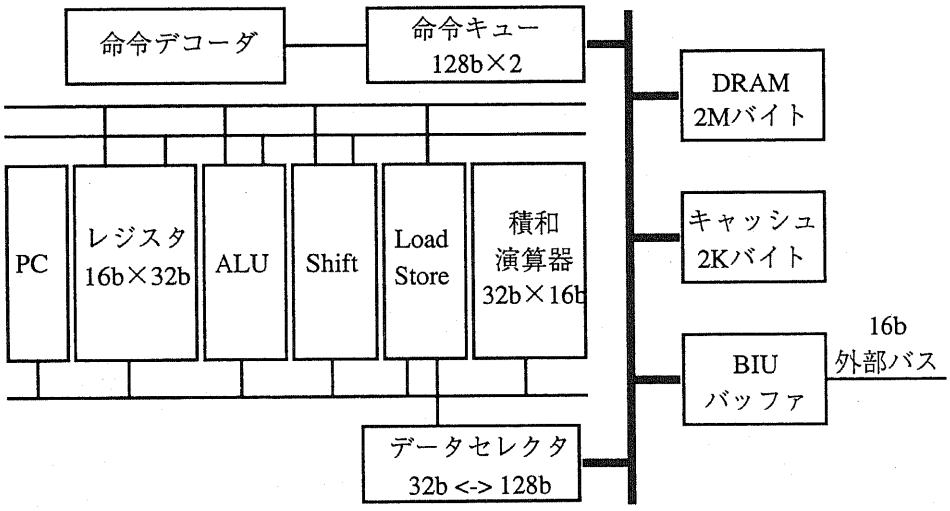


図 1.5　32 ビット・マイクロコントローラ M32R/D のハードウェア構成

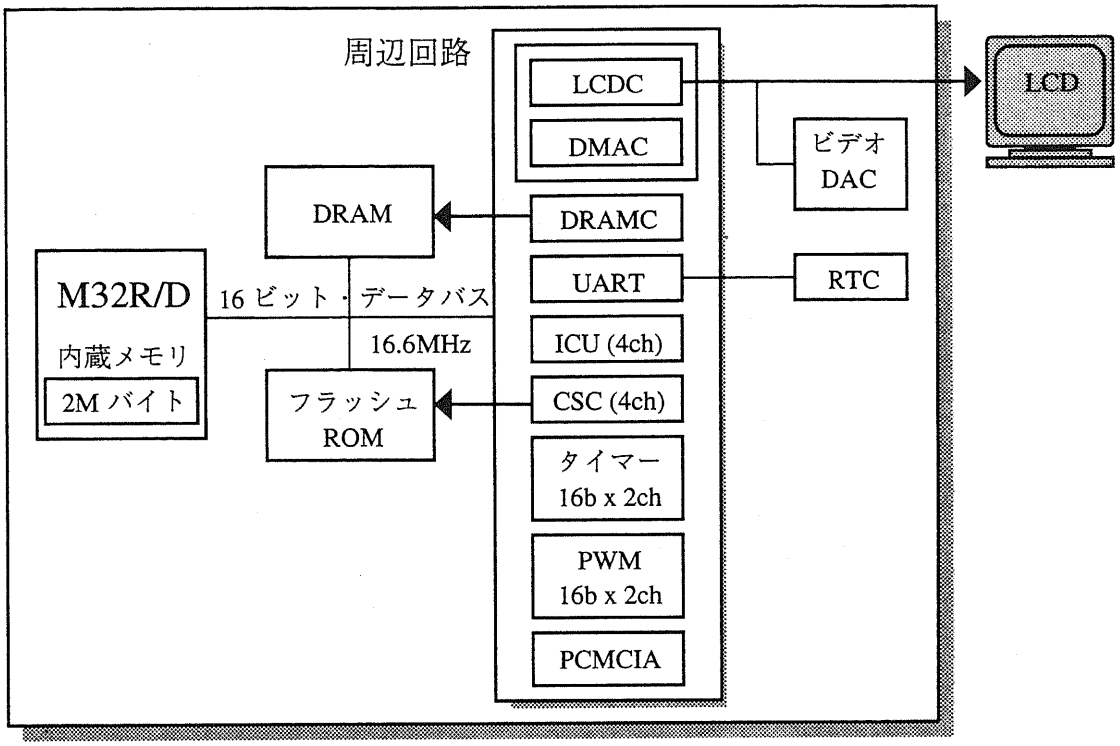


図 1.6　評価ボードの構成図

本研究で主として用いた32ビット・マイクロコントローラM32R/Dは、近年の高性能マイクロプロセッサで多く見られる単純な命令セットとパイプライン・データバスを持つRISC(Reduced Instruction Set Computer)アーキテクチャ[15)16)]を採用している。また、2Mバイトという大規模なDRAMをチップに内蔵しているという特徴を持つ。そしてCPUと内蔵メモリを128ビットという広い巾の内部バスで接続することにより、大規模なメモリの効率的なアクセスが可能となっている。つまり、広い巾の内部バスがCPUと大規模な内蔵DRAMを接続することで、頻繁なメモリ・アクセスを要求する画像・音響処理を効率的に実現することが可能になる。従って、本研究で用いたマイクロコントローラは、マイクロコントローラと内蔵メモリを接続した広い巾の内部バスを利用した画像・音響処理の検討に最適なものの１つである。

1.2.4 本研究で用いた画像処理技術評価用画像

画像処理技術は本研究の主要な研究テーマの１つである。本研究の成果として開発した画像処理技術を実際の画像に適用する場合の、画質や処理速度等を評価するために用いた画像を図1.9と図1.10に示す。図に示した画像の出典は次の２つである。

(1) 画像処理技術標準化委員会　監修
　財団法人　日本企画協会　　発行
　　高精細カラーディジタル標準画像データ(ISO/JIS-SCID)、1995年
(2) テレビジョン・システム評価用テストチャート改善委員会編、
　(社) テレビジョン学会（当時）発行
　　テレビジョン・システム評価用デジタル標準画像、1985年

高精細カラーディジタル標準画像データのうち本論文で用いたものを図1.9に示す。画像の識別記号N1～N5は自然画像の評価で、S1は解像度の評価で使用した。

また、テレビジョン・システム評価用デジタル標準画像の５チャートを図1.10に示す。これらの５チャートも自然画像を用いた評価で使用した。なお、論文中では、ITE肌色チャートをチャート1、ITE標準絵柄 - ヘアーバンドの女性 - をチャート2、ITE標準絵柄 - 天気予報 - をチャート3、スイスの山村をチャート4、チューリップをチャート5とそれぞれ引用している（ITEはテレビジョン学会の英文略称）。

いずれも画像処理技術の評価に標準的に用いられているものである。

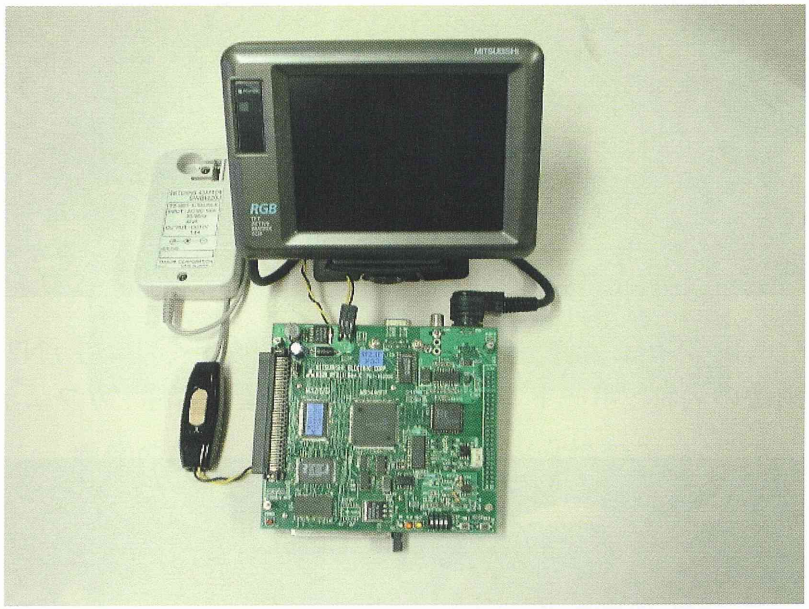


図 1.7　32 ビット・マイクロコントローラ M32R/D を搭載した評価装置

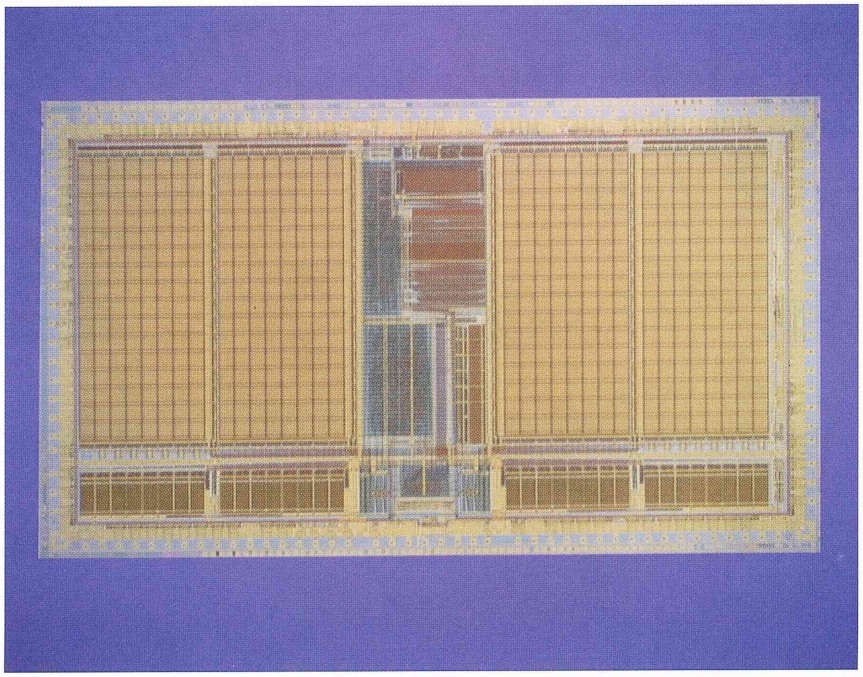
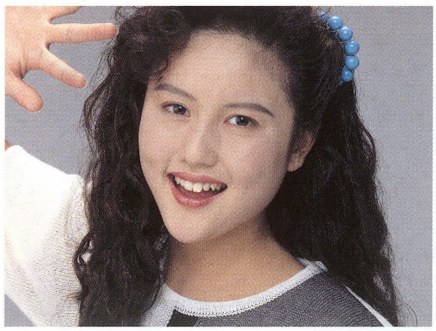


図 1.8　32 ビット・マイクロコントローラ M32R/D のチップ写真



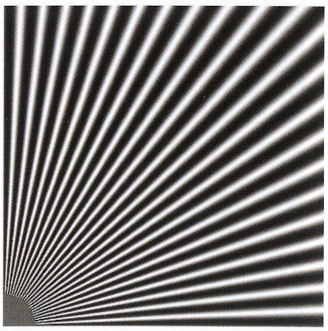
画像識別記号：N1
画像の名称：ポートレート

画像識別記号：N2
画像の名称：カフェテリア

画像識別記号：N3
画像の名称：果物かご（篭）

画像識別記号：N4
画像の名称：ワインと食器

画像識別記号：N5
画像の名称：自転車



画像識別記号：S1
画像の名称：解像度チャート

図 1.9　高精細カラーディジタル標準画像データ

画像の名称 : ITE 肌色チャート（女性のアップ）

画像の名称 : ITE 標準絵柄 - ヘアーバンドの女性 -

画像の名称 : ITE 標準絵柄 - 天気予報 -

図 1.10　テレビジョン・システム評価用デジタル標準画像（その１）

画像の名称：スイスの山村

画像の名称：チューリップ

図 1.10　テレビジョン・システム評価用デジタル標準画像（その２）

1.2.5 DRAM内蔵マイクロコントローラを用いた画像・音響処理技術開発の目的とその要点

半導体の集積度向上とともに、今後は様々な機能が1つのチップにシステムLSIとして集積して実現されていくと考えられる。集積される機能の1つとして32ビット・マイクロコントローラがある。システムLSIでは、組み込んだマイクロコントローラと内蔵メモリを広い巾の内部バスで接続することにより、大規模なメモリを効率的にアクセスできるようになる。従って、1チップ内にRISCアーキテクチャを採用したCPUと大規模なDRAMを搭載し、広い巾の内部バスでこれらを接続する構成をとる32ビット・マイクロコントローラを用いて画像・音響処理を効率的に実現する技術を考察することは、システムLSI化という今後の半導体の動向に合致し、非常に重要である。そのため本論文では、このようなアーキテクチャを採用したDRAM内蔵32ビット・マイクロコントローラを用いることで、大容量メモリを内蔵し広いバスでメモリを接続したマイクロコントローラを用いて実現できる画像・音響処理技術の検討をその目的とした。

4つの分野において従来の画像・音響処理技術をマイクロコントローラに適応したときの課題を1.2.2節で述べた。その結果判明したことは、いずれをとっても、従来技術をそのまま適用することはできず、マイクロコントローラのもつハードウェア資源の制約を考慮にいれて、画像・音響処理の演算アルゴリズム、演算精度、メモリの使用方法等を工夫していく必要があるということである。

画像・音響処理の演算アルゴリズムや演算精度の検討に際しては、マイクロコントローラの持つハードウェア資源の制約から、従来の演算アルゴリズムを見直し、アルゴリズムの適用方法の工夫・改善を検討した。そして、その結果を実用に適した演算アルゴリズムとして提案した。

デジタル・スチル・カメラ用撮像信号処理の検討では、拡大縮小などの幾何学的変換で研究された方式を画素補間処理にどのように適用するかを検討した。各画素にRGB 3つの色成分があることを前提とする従来方式を、CCDの構造からRGB 3つの色成分のうち1つの色成分しか各画素に存在しないデータに適用するためのアルゴリズムを提案した[17]。また、局所的な色の相関を用いた画素補間方式についても検討した。従来方式では、局所的な色の相関が成り立たない、色信号が急激に変化する領域を検出し、補間式を動的に変更したり、重み付け係数を用いることによってアルゴリズムの適用を工夫している。しかし、エッジ検出や重み係数の計算は演算量が多く、マイクロコントローラを用いたシステムで実現するには適さないため、少ない演算量で局所的な色の相関が成り立たない箇所に対応するアルゴリズムを提案した[18]。

また、静止画像処理JPEG、3次元グラフィックスやオーディオ復号の検討においても、マイクロコントローラのアーキテクチャとハードウェア資源に適した

離散コサイン変換の高速アルゴリズムの検討[19]、適用可能な画面サイズを制限し、乗算の実現方法を工夫することで固定小数点を用いた実現アルゴリズムの検討[20]、演算式の係数の対称性を用いることで演算量を削減するアルゴリズムの検討[21]等をそれぞれ実施し、実用に適した演算アルゴリズムとして提案した。

一方、画像・音響処理で重要なメモリの使用方法の検討に際しては、大容量メモリとして利用されるDRAMの特性と広い内部バスでCPUと接続されたアーキテクチャを考慮して、演算アルゴリズムの実現方法を工夫・改善し、実用に適した演算アルゴリズムとして提案した。

静止画像処理JPEGの検討では、離散コサイン変換後のエントロピー符号化の際に行われるジグザグ走査や連続する零係数の検出について、DRAMの特性と広い内部バスを考慮し、メモリへのランダムなアクセスをなくし、連続的するメモリ・セルへアクセスしながら処理可能な実現方法や、メモリ参照の回数を削減する実現方法を提案した[22]。

また、オーディオ復号の検討では、JPEGでの提案と同様に、変形離散コサイン変換やその後の窓掛け等の処理で、メモリへのランダムなアクセスをなくし、連続的するメモリ・セルへアクセスしながら処理可能な実現方法を提案した[21]。

本論文では、近く訪れるシステムLSI時代を想定し、大容量メモリを内蔵し広いバスでメモリを接続したマイクロコントローラを用いて画像・音響処理技術を実現する手法を提案する。これにより、市場要求を満たし、画像・音響処理技術の適用範囲を広げ、マイクロコントローラを用いた新たな製品群を創出することが可能になると考える。

1.3 本論文の構成

本論文では、第1章：序論、第2章：マイクロコントローラを用いた画像・音響処理技術、第3章：デジタル・スチル・カメラ用画像処理技術、第4章：静止画像処理JPEG、第5章：3次元グラフィックスにおける画像処理技術、第6章：オーディオ復号における音響処理技術、および、第7章：結論から構成される。これらの章はいずれも、DRAM内蔵マイクロコントローラ用いた画像・音響処理技術に関して述べている。図1.11に本論文の構成と各章の位置付けを示す。まず第2章では、一般的な画像・音響処理技術をDRAM内蔵マイクロコントローラ上で実現する場合に考察すべき特殊性とマイクロコントローラ特有の開発手法を明らかにする。そして、そこで述べた一般的概念を実際の画像・音響処理技術に如何に適用し、マイクロコントローラを用いて実現する上での課題を克服するかを第3章以降で述べる。以下では順次、各章の概略を述べる。

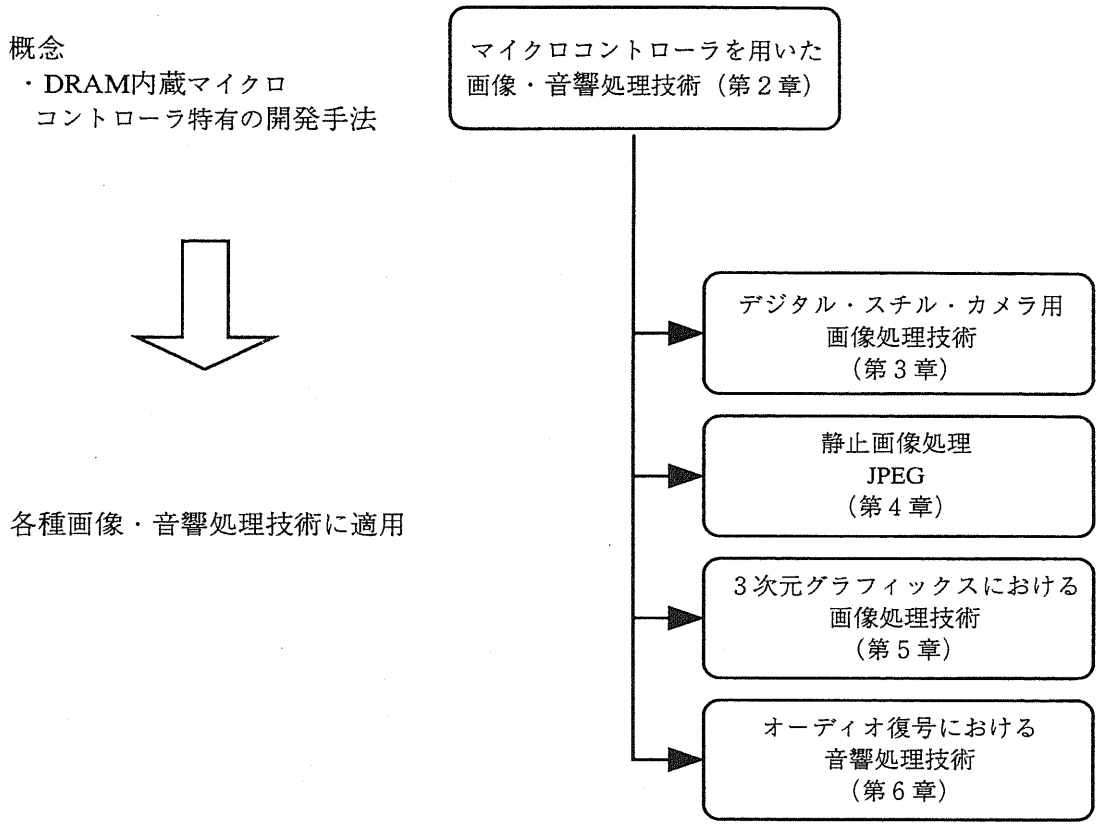


図 1.11　本論文の構成

本研究に関わる画像･音響処理技術では、コンピュータやワークステーションに使用されるマイクロプロセッサとは異なり、搭載されたハードウェア資源に制約があるマイクロコントローラを用いて、実用的な画質や音質あるいは処理速度を実現することが重要である。そこで第２章では、DRAM 内蔵マイクロコントローラの持つハードウェア資源の制約と画像･音響処理技術との関係について検討する。特に本章では、マイクロコントローラの持つハードウェア資源の活用、DRAMの特性を活かしたメモリ格納方法や人間の視聴覚モデルを用いた画像･音響処理の高速化等について検討する。そして、画像・音響処理技術をマイクロコントローラ上で実現する場合のDRAM内蔵マイクロコントローラ特有の開発手法を提案する。

第３章以降では、第２章の結果を1.2.1節に述べた市場要求のある４つの画像･音響処理技術に対して如何に適用できるかについて述べる。第２章で提案したマイクロコントローラ特有の画像･音響処理技術の開発手法は、DRAM内蔵マイクロコントローラを用いて画像･音響処理技術を開発する場合に考慮すべき手法に

ついて概念的に説明したもので、各々の処理技術で重要となる課題も異なれば、課題解決のための適用手法も異なる。そこで、第3章以降でそれぞれの画像・音響処理技術に対して課題を明らかにし、提案手法の適用の仕方を詳細に検討する。

第3章では、マイクロコントローラを用いたデジタル・スチル・カメラ用撮像信号処理の課題とその解決手法について考察する。特にベイヤー型の原色フィルタアレイに適した画素補間方式について、ソフトウェアで実現する場合の課題を検討する。そして、演算量と画質の関係を詳細に検討することで、マイクロコントローラを用いた画素補間方式の実現手法を考察する。

第4章では、マイクロコントローラを用いたカラー静止画像圧縮方式JPEGの課題とその解決手法について考察する。JPEG方式を用いた画像圧縮をソフトウェアで実現する場合の最大の課題はその処理速度にある。そのため、JPEG方式のアルゴリズムや実行した場合の処理時間の内訳を分析し、高速化のための課題を明らかにする。そして課題を解決するために、メモリ使用方法や演算アルゴリズムを検討する。

第5章では、マイクロコントローラを用いた3次元グラフィックスの課題とその解決手法について考察する。3次元グラフィックスをマイクロコントローラで実現する上での最大の課題は、マイクロコントローラの多くが浮動小数点演算器を搭載しないということである。従って、実用的な処理速度の3次元グラフィックスを実現するためには、浮動小数点演算ではなく整数演算を用いて実現しなければいけないという課題がある。そのためには、整数演算を用いて3次元グラフィックスに必要な演算精度を保ち、しかも高速に処理する手法を検討する。

第6章では、マイクロコントローラを用いたオーディオ復号の課題とその解決手法について考察する。オーディオ復号処理をマイクロコントローラで実現する上での最大の課題は、小さいサイズの乗算器を用いながら要求される音質を実現し、同時にリアルタイム処理可能なように処理量を抑えることである。それらの課題を解決するためには、各種フィルタの高速化アルゴリズム、フィルタ処理で多く用いられる乗算の実現手法、フィルタ処理に用いるデータや係数の格納方法について検討する。

第7章では、第2章から第6章までの本研究の成果を総括し、結論とする。

第1章の参考文献

1) K. Kutaragi and et al.; "A Microprocessor with a 128b CPU, 10 Floating-Point MACs, 4 Floating-Point Dividers, and an MPEG2 Decoder", IEEE International Solid-State Circuits Conference, pp. 256-257, (1999)

2) F. Michael and et al.; "A High Bandwith Superscalar Microprocessor for Multimedia Applications", IEEE International Solid-State Circuits Conference, pp. 258-259, 466, (1999)

3) 吉井; "音響・音声工学", 近代科学社, (1992)

4) Y. Ishikawa, Y. Kisuki, T. Sakamoto and T. Hase; "Speech Synthesis Software for a 32-bit Microprocessor", IEEE Transactions on Consumer Electronics, Vol. 44, No. 3, pp. 1173-1182, (1998)

5) Y. Ishikawa, Y. Kisuki, T. Sakamoto and T. Hase; "Speech Synthesis Method based on Application-Specific Units and its Implementation on a 32-bit Microprocessor", IEEE Transactions on Consumer Electronics, Vol. 45, No. 3, pp. 980-985, (1999)

6) L. Rabuner, B. Juang; "音声認識の基礎", NTTアドバンステクノロジ, (1995)

7) 坂本、森田、佐藤、長谷; "組込み用マイクロコントローラを用いたPHSデータ通信機能の全ソフトウェア化", 映像情報メディア学会誌, Vol. 53, pp. 901 - 904

8) 横山; スペクトル拡散通信システム, 科学技術出版社, (1988)

9) 浅井、坂本、長谷; "Software Solution for GPS baseband Processing Using a 32-bit Embedded Microprocessor", 映像情報メディア学会誌, Vol. 53, pp. 738 - 745, (1999)

10) 浅井、影本、坂本、長谷; "32ビット組込み型マイクロプロセッサを用いたGPS衛星捕捉処理の高速化", 電気情報通信学会論文誌, 1999年11月号に掲載予定

11) B. E. Bayer; Color imaging array, U. S. Patent, 3971065

12) Joint Photographic Experts Group; ISO/IEC JTC 1/SC 29/WG 10, Digital compression and coding of continuous-tone still images - part 1: Requirements and guidelines, (1994)

13) S. Iwata, T. Shimizu, J. Korematu, K. Dosaka, H. Tsubota, and K. Saitoh; "Performance Evaluation of a Microprocessor with On-chip DRAM and High Bandwidth Internal Bus", Custom Integrated Circuits Conference, pp. 269-272, (1996)

14) 奥村、澤井、岩田、佐藤、那須、布村、橘高、平野、山崎、堂阪; "16MビットDRAM内蔵32ビットマイクロプロセッサ", 信学技報, ICD96-7, pp.49-55, (1996)

15) D. A. Patterson and J. L. Hennessy; "Computer architecture: A quantitativeapproach", Morgan Kaufman Publishers, Inc., (1990)
16) S. B. Fuber; "VLSI RISC Architecture and Organization", Marcel Dekker, Inc., (1989)
17) T. Sakamoto, C. Nakanishi and T. Hase; "Software Pixel Interpolation for Digital Still Camera for a 32-bit MCU", IEEE Transactions on Consumer Electronics, Vol. 43, No. 3, pp. 1342-1352, (1998)
18) 中西, 坂本, 長谷; "組込み形マイクロプロセッサに適した画素補間方式", 映像情報メディア学会誌, Vol. 53, No. 4, pp.141-149, (1999)
19) T. Sakamoto and T. Hase; "Software JPEG for a 32-bit MCU with Dual Issue", IEEE Transactions on Consumer Electronics, Vol. 44, No. 4, pp. 1334-1341, (1998)
20) K. Yoshida, T. Sakamoto and T. Hase; "A 3D Graphics library for a 32-bit micro-processors for embedded systems", IEEE Transactions on Consumer Electronics, Vol. 44, No. 3, pp. 1107-1114, (1998)
21) T. Sakamoto, M. Taruki and T. Hase; "A Fast MPEG-Audio Layer III algorithm for a 32-bit MCU", IEEE Transactions on Consumer Electronics, Vol. 45, No. 3, pp. 986-993, (1999)
22) T. Sakamoto and T. Hase; "JPEG Software Solution for a 32-bit MCU", IEEE Transactions on Consumer Electronics, Vol. 43, No. 3, pp. 410-417, (1997)

第2章　マイクロコントローラを用いた画像・音響処理技術

2.1 はじめに

本論文に関わる画像・音響処理技術は、コンピュータやワークステーションに使用されるマイクロプロセッサとは異なり、搭載されたハードウェア資源に制約があるマイクロコントローラを用いて実現することを前提としている。従って、ハードウェア資源の制約を受けながらも、画質や音質あるいは処理速度の面で実用的な範囲を実現するための手段や方法を検討する必要がある。

画像・音響処理技術は長年の間研究を重ねられ、その成果が実用化されてきたことは言うまでもない。つまり、画像・音響処理技術は専用装置として、あるいは半導体の進歩に伴い画像・音響処理用の専用の集積回路として実現され、装置に組み込まれてきた。また画像・音響処理技術は、コンピュータの発展とともに、大型計算機やパーソナルコンピュータのソフトウェアとしても実現され、多くの分野で利用されてきた。そして近年の、家庭で使用するような製品に組み込むことを前提とした32ビット・マイクロコントローラの高性能化に伴い、より広範囲の身近な製品に適用できる可能性がでてきた。しかし、より身近な製品に画像・音響処理技術をソフトウェアで実現して適用するには、従来の大型計算機やパーソナルコンピュータを想定した画像・音響処理技術では課題が多い。実用化の面でより踏み込んだ研究が必要である。

そこでこの問題に対し、従来の画像・音響処理技術をマイクロコントローラを持つハードウェア資源の制約と製品仕様から決まる処理の仕様に関する限定から、従来技術を適用する上での問題点を総合的に解析する必要があると考えた。

本章では、マイクロコントローラを持つハードウェア資源の制約と画像・音響処理技術との関係について検討する。特に本章では、DRAM内蔵マイクロコントローラの持つハードウェア資源の活用、DRAMの特性を活かしたメモリ格納方法や人間の視聴覚モデルを用いた画像・音響処理の高速化等について検討する。そして、画像・音響処理技術をDRAM内蔵マイクロコントローラ上で実現する場合の特有の開発手法を提案する。

2.2 マイクロコントローラを用いた画像・音響処理技術の特殊性

マイクロコントローラは、コンピュータやワークステーションに使用されるマイクロプロセッサとは異なり、搭載されたハードウェア資源に制約がある。通常のマイクロコントローラの持つこれらの制約を列挙すると次のようなものがある。

・浮動小数点演算器がない
・乗算器のサイズは16ビット×16ビットあるいは32ビット×16ビット程度
・特定用途のアクセラレータを内蔵しない

浮動小数点演算器がなければ、特に3次元グラフィックスの実現手法は検討が必要である。また、乗算器のサイズが小さいと、16ビットを越えるオーディオデータを取り扱うオーディオ復号や座標データを取り扱う3次元グラフィックスの処理性能に大幅な影響を与える。

また、マイクロコントローラに搭載されたハードウェア資源から発生する制約以外にも、マイクロコントローラが組み込まれる製品から受けるいろいろな制約もある。それは、補助記憶領域として使用するメモリに関する制約と、製品仕様から限定される処理対象画像の限定や処理時間に関する制約である。

マイクロコントローラを組み込んで使う製品では、製品の部品価格を低くとどめるためや、消費電力を抑え電池を使用しても長時間動作させるために、補助記憶領域としてDRAMを使用することが多い。大量のデータを扱う画像処理では、一般に画像データを保存しているメモリへのアクセスが多い。その保存用の領域としてDRAMを使用することでメモリのアクセス時間が多くかかり、画像処理全体の処理性能を低下させる可能性がある。また、リアルタイムでの復号処理を要求されることが多い音響処理では、メモリアクセス時間が多くかかることにより、演算処理量を減らしたにもかかわらず、リアルタイム処理できない可能性がある。

また製品仕様から、画像・音響処理についての外部仕様に関する境界条件が決められる。例えばデジタル・スチル・カメラの場合、製品に採用されたCCDの大きさから、処理すべき画像のサイズが決まる。また、画像のフォーマットやシャッターを押してから記憶メディアに記憶するまでの時間から割り出される画像処理に割り当てられる処理時間が決まる。コンピュータやワークステーション上の画像・音響処理の場合、取り扱う画像のサイズやフォーマットあるいは音のサンプリングレートに制限を設けず適用できるように、汎用性を考えて設計されている。しかし、マイクロコントローラを使って行う場合、上述のような製品仕様から決まる境界条件の範囲内で仕様や処理時間を限定できるという特徴もある。

以上のように、マイクロコントローラを用いた画像・音響処理は、マイクロコントローラのもつハードウェア資源や補助記憶用メモリに制約がある。一方で、製品仕様面から処理対象画面の大きさや種類、また要求される処理時間を限定できるという利点もある。つまり、マイクロコントローラを用いた画像・音響処理では、マイクロコントローラを用いることで発生する制約を考慮しながら決められた機能を実現する手法を検討しなければいけないという特殊性がある。

2.3 マイクロコントローラを用いた画像・音響処理技術開発手法

マイクロコントローラを用いた画像･音響処理技術の開発には､2.2節で述べたように､マイクロコントローラを用いることで発生する制約を考慮しながら実現手法を検討する必要がある。その実現手法の検討の際には、マイクロコントローラのもつハードウェア資源の活用、DRAMの特性を活かしたメモリ格納方法や人間の視聴覚モデルを用いた画像・音響処理の高速化等について検討する必要がある。本節では、DRAM内蔵マイクロコントローラを用いた画像・音響処理技術を開発するにあたり重要なこれらの開発手法について述べる。

2.3.1 定数のテーブル化

フィルタ係数としてsin関数やcos関数の値を使用することが多い。コンピュータやワークステーション上の画像･音響処理の場合、多機能で汎用的なものを開発するため、フィルタ係数も画像・音響処理実行時にsin関数やcos関数を使用して初期化し、その後で画像・音響処理を実行する。sin関数やcos関数といった算術関数のライブラリ関数は、通常倍精度の浮動小数点を用いて計算しており、マイクロコントローラで実行すると処理が遅いという問題がある。またプログラムの他の部分で浮動小数点を用いていない場合、この係数の計算のためだけに浮動小数点演算ライブラリが必要になり、プログラムのコードサイズが増えるという問題も発生する。

しかし､マイクロコントローラを組み込んだシステムの場合､2.2節にも述べたように､製品仕様から処理すべき画像のサイズやフォーマットあるいは画像・音響処理機能が決まるため、画像・音響処理を実行するたびに動的に係数を計算する必要がない場合が多い。このような場合には、使用できるROMに余裕があれば、事前に計算した値を定数としてテーブル化し、このテーブルを参照する処理を検討する必要がある。

2.3.2 固定小数点化

数値データの標準的表現方法には､固定小数点表現と浮動小数点表現がある[1]。固定小数点表現とは、小数点の位置が固定されている数値データの表現である。例えば、整数表現は、最下位ビットの直後に小数点が固定されている固定小数点表現とみなすことができる。実際の計算で固定小数点表現を使用する場合には、固定小数点表現を使用する者が位取り計算を行う必要がある。それに対して、浮動小数点表現とは、使用する者が位取り計算を行う必要がないように考案された小数点表現である。数値計算で扱う数のとる範囲が広い場合は､位取りを行う必要のある固定小数点表現では不便であるため､浮動小数点表現が使用されることが多い。

数値データを固定小数点表現で表わすと、マイクロコントローラが命令として持つ整数演算を用いて各種演算を実現できる。しかしその際、小数点位置を考慮して計算を実現する必要がある。一方、数値データを浮動小数点表現で表わすと、位取りが自動的に行われるため小数点位置を考慮することなく計算できる。しかし、浮動小数点演算器を

搭載していないマイクロコントローラで浮動小数点演算を行うためには、浮動小数点演算をソフトウェアで実現したライブラリ関数を使用する必要がある。浮動小数点演算のソフトウェア・ライブラリは、扱える数の範囲が広く、しかも精度の高い計算と、位取り計算を行うため、種々の表現形式の変換や、オーバーフローやアンダーフローのチェックを内部で実施している。そのため、計算に必要な処理時間は通常非常に大きい。

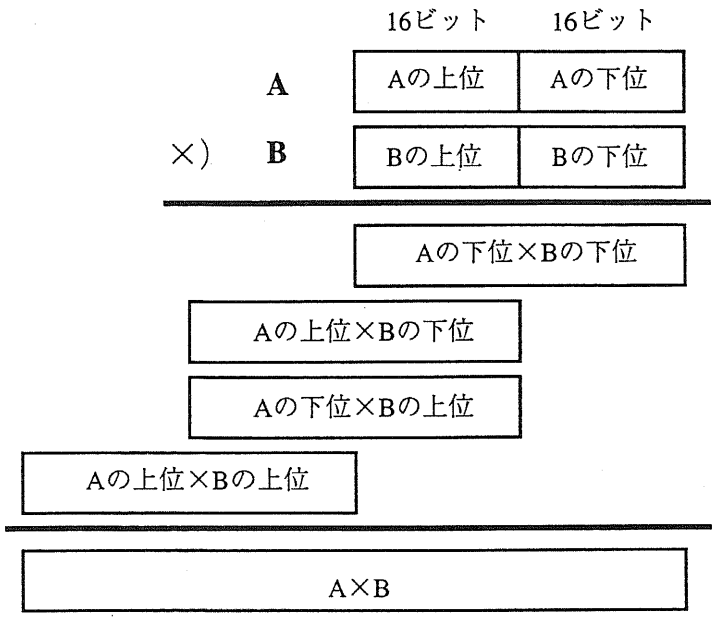
マイクロコントローラを用いた画像・音響処理では、これらの固定小数点表現および浮動小数点表現の特徴を把握し、処理内容によってこれらを使い分ける必要がある。浮動小数点表現の使用が適した応用は、取り扱う数値データの値のとる範囲が広く、固定小数点表現では表すことが難しく、しかも高速に処理する必要がない場合である。それ以外の場合は、固定小数点表現を使用する必要がある。画像・音響処理プログラムでは各種画像圧縮で採用されている離散コサイン変換、色空間の変換や3次元グラフィックス等で浮動小数点演算が使用される。離散コサイン変換、色空間の変換や3次元グラフィックスでは、いずれも浮動小数点演算をソフトウェアでエミュレーションしたのでは全く性能がでない。従って、マイクロコントローラを用いて実現するためには、固定小数点表現を採用し、整数演算を用いて実現することが必須である。離散コサイン変換や色空間の変換の場合、取り扱う数値データの値のとる範囲が狭く、固定小数点表現を用いて整数演算で実現することは比較的容易である。しかし、3次元グラフィックスの場合は、取り扱う数値データの値のとる範囲が広いため、固定小数点表現にすることは難しい。これについては、後述の手法を組み合わせて実現方法を検討する必要がある。

2.3.3 演算精度の検討

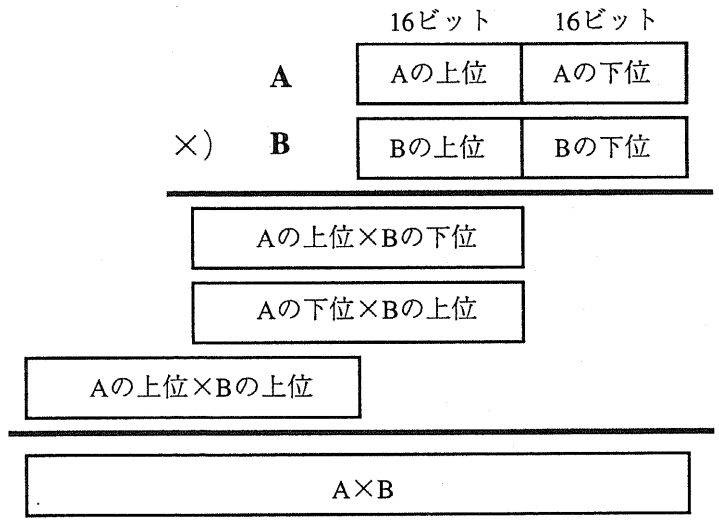
2.3.2節では固定小数点表現について述べたが、数値データを固定小数点表現に表すためには、小数点の位置を決める必要がある。そのためには、取り扱う数値データのとる値の範囲やそのデータを用いた演算途中に発生する桁上がりや桁下がりを考慮して、数値データの語調や小数点位置を決定する必要がある。これらは、演算結果として必要な精度や要求される処理速度を考慮して決定する必要がある。本節は、固定小数点表現を用いたデータの小数点位置を決めるのに重要な要因となる演算精度について、処理の高速化のために検討すべき項目について述べる。

(1) 乗算と演算精度

データを用いた演算処理に乗算がある場合、マイクロコントローラに搭載されている乗算器のサイズを考えると、語調は短い方が処理速度の高速化には有利である。しかし演算精度の観点からは、語調は短い方が精度は悪くなる。従って、乗算の用いられる場所に応じて、求められる演算精度と処理速度に合った乗算のサイズを決定する必要がある。図2.1に16ビット×16ビットの乗算器を搭載したマイクロコントローラを用いた場合の、乗算の種類に応じた計算方法を模式的に示す。図では、16ビット×16ビットの乗算命令を用いて32ビット×32ビットを実現する場合と32ビット×32ビットの乗算のうち下位16ビット同士の乗算を省略した場合を示した。その他にも、32ビット×16ビットの乗算や16ビット×16ビットの乗算も図から容易に推測できる。



(1) 32ビット×32ビットの乗算



(2) 下位16ビット同士の乗算を省略した
32ビット×32ビットの乗算

図2.1　32ビット×32ビットの乗算の実現手法

例えば文献2では、マルチチャンネル・デジタル・オーディオ符号化手法の1つであるDolby Digital[3]の復号処理を16ビット×16ビットの乗算器を用いて実現することを検討した。従来、出力データの演算精度として、16ビットから20ビット程度を要求されるオーディオ復号処理では、20ビット×20ビットの乗算器[4]あるいは24ビット×24ビットの乗算器[5][6][7][8]を用いて積和演算を行ってきた。しかし、20ビットや24ビットといった特殊なサイズの乗算器を搭載したプロセッサは、主としてオーディオ用途のDSPで、汎用のマイクロコントローラはこのような特殊なサイズの乗算器を搭載しない。そこで文献2では、16ビット×16ビットの乗算器を用い、図2.1の(2)に示した乗算方法を採用す

ることで、出力データにおいて20ビットの演算精度を実現できることを示した。また、これにより全体の処理量を約1割削減できたことも同様に示した。これらからわかるように、高い演算精度を実現するほど、演算が複雑になり、それに必要な処理時間が増える。これらの乗算の実現方法やそれにかかる処理時間について、使用するマイクロコントローラで検討したうえで、求められる演算精度と処理速度に合った乗算のサイズを決定する必要がある。

また、積和演算命令とアキュムレータの丸め命令を持つマイクロコントローラの場合、積和演算後に丸め処理を行う必要があれば、積和演算結果の小数点位置がアキュムレータの丸め命令の仕様に合うように、データの小数点位置を決定する必要がある。

(2) 算術関数と演算精度

2.3.1節でsin関数やcos関数といった算術関数を用いる場合の問題点を述べた。しかし、処理アルゴリズム上、算術関数を用いなければいけない場合もある。このような場合には、算術関数の結果として得られる計算結果に必要な演算精度を検討する必要がある。算術関数を実現したライブラリ関数は、通常倍精度の浮動小数点を用いて計算し、演算結果を倍精度の浮動小数点で返す[9]ため重い処理になる。倍精度の浮動小数点という演算精度が必要ない場合は、単精度の浮動小数点あるいは固定小数点を用いて算術関数を実現することを検討する必要がある。

例えば文献１０では、人工衛星から受信した情報に基づいて、現在位置を表す経度および緯度を計算するプログラムにおいて、処理速度の高速化を目的に算術関数の必要な演算精度を検討している。そこでは、浮動小数点の表現形式として、マイクロコントローラが搭載する乗算器のサイズを考慮に入れた、倍精度と単精度の浮動小数点表現の中間的な精度を持つ表現形式を採用している。また、マクローリン展開を用いて算術関数を実現する際の展開の次数についても、用途に応じた演算精度の観点から検討している。

(3) 人間の視聴覚特性と演算精度

画像や音声のデータ圧縮では、人間の視聴覚特性を利用することが多い。このデータ圧縮の手法を演算精度を決める際にも利用できる。

画像処理の場合は、人間の視覚特性を利用して視覚上問題のない範囲で画質を落とすことができる。つまり、画像の持つ情報のうち、表示したときに視覚特性上知覚しにくい部分を省略してデータを圧縮する。このことは、演算精度を決める際にも利用できる。すなわち、表示したときに視覚特性上知覚しにくい範囲で演算精度を落とすことができる。また、演算精度を落とすことで処理を簡単化でき、高速化できる場合もある。

また代表的な音響処理であるオーディオ圧縮方式でも人間の聴覚特性を積極的に利用している。つまり周波数帯域で聞こえない成分を取り除くことでデータ圧縮をする。オーディオ圧縮の基本アルゴリズムは、こうした聴覚特性をうまく利用したサブバンド符号化あるいは変換符号化である。聴覚特性を用いたオーディオ符号化の基本アルゴリズムを図2.2に示す。まず、入力したオーディオ信号を時間領域から周波数領域へ変換し、周波数領域で帯域分割する。この際に用いるのがサブバンド符号化あるいは変換符号化で

ある。これと並行して、聴覚マスキング効果や最小可聴限特性のモデルを入力信号に当てはめ、各帯域の出力信号のビット数割り当てを決める。聴覚マスキング効果や最小可聴限特性のモデルを図2.3に示す。聴覚マスキング効果とは、大きな振幅の音（例えば図2.3のA点）によって周波数軸上で隣接する小さな振幅の音（例えば図2.3のB点）の音が聞こえなくなる現象である。また最小可聴限特性は、聴き取れる振幅の最小値が周波数によって異なるという特性で、この人間の聴覚で聴き取れなくなるような信号のレベルの周波数の関数をマスキングしきい値と呼ぶ。この特性のために、図2.3において同じ振幅の音でもD点の音は聴こえるが、C点の音は聴こえない。これら聴覚特性を利用してデータ圧縮を行う。これらの方式を用いて圧縮する場合、人間に聴き取れないとされた音のデータは、省略され零とみなされることが多い。こういった圧縮の特徴を利用して、演算量や演算精度を落とし、高速化できる場合もある。

以上のように、演算精度を検討する場合、人間の視聴覚特性も考慮に入れて検討する必要がある。

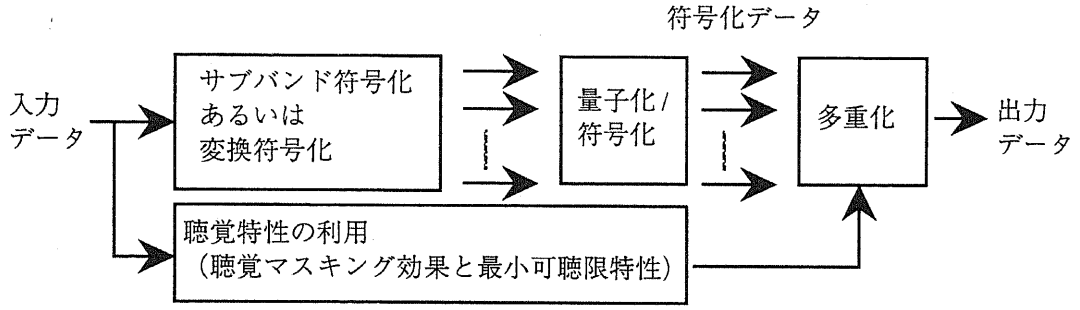


図2.2　聴覚特性を用いたオーディオ符号化

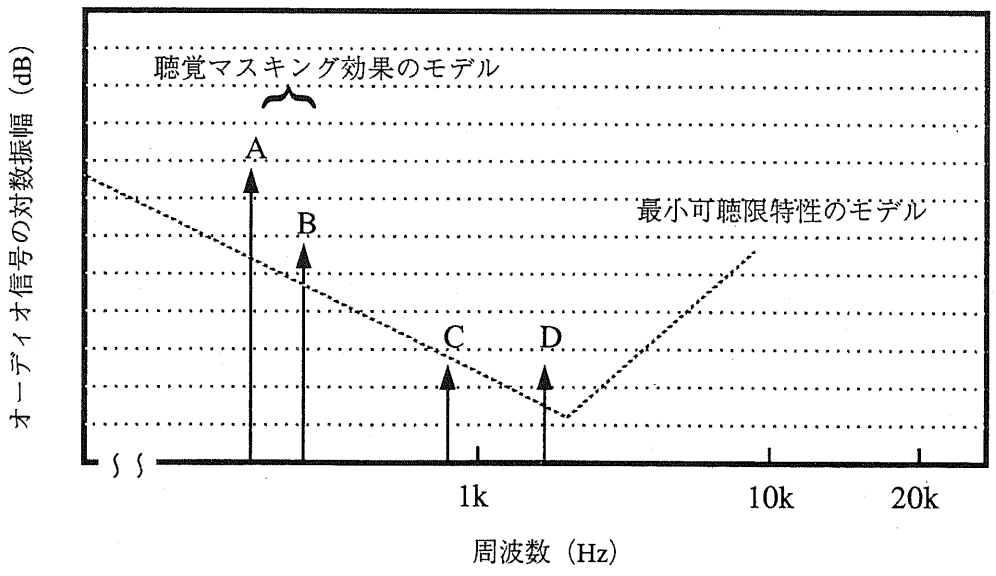


図2.3　聴覚マスキング効果と最小可聴限特性のモデル

2.3.4 メモリ・アクセスを考慮した高速化手法

2.2節で述べたように、マイクロコントローラを使用する製品は補助記憶領域としてDRAMを使用することが多い。また最近はDRAMを内蔵し、CPUとDRAMを広いバスで接続したマイクロコントローラもある。図2.4にDRAMの標準的な構成を示す[11]。

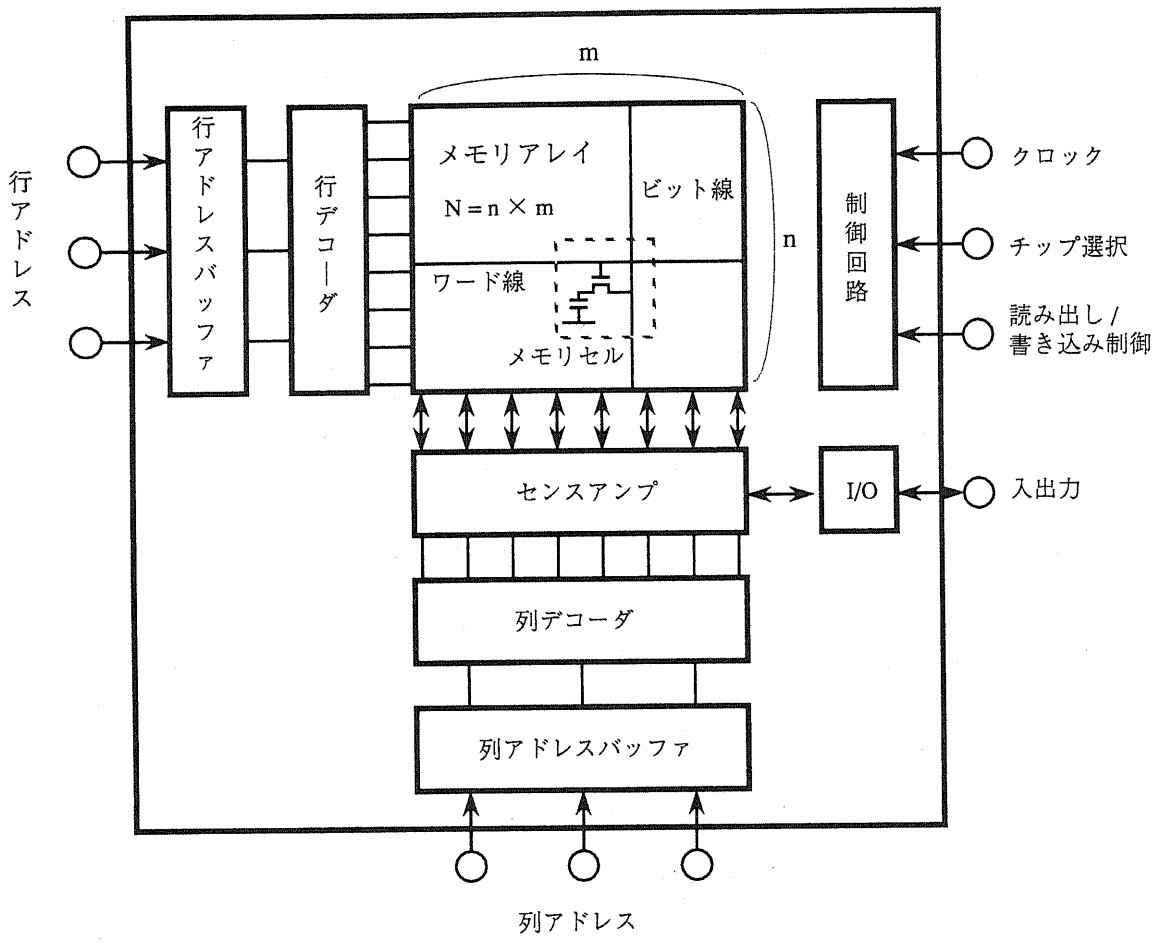


図2.4　DRAMの構成

DRAMでは、データはN（n×m）ビットのメモリアレイに蓄積される。読み出しや書き込みを行おうとするメモリセルの情報は、行及び列に保存され、行デコーダによる特定のワード線の選択によって（n本のワード線のうち1本）mビットのメモリセルがセンスアンプに結合される。次に列デコーダによる特定のビット線（m本のビット線のうち1本）の選択によって、その中の1個のセンスアンプが入出力回路（I/O）に結合され、制御回路の指令に従って読み出し、あるいは書き込みが行われる。このDRAMのメモリ読み出しや書き込みの動作のうち、ワード線の選択やメモリセルのセンスアンプに特に時間がかかる。従って、同一ワード線上のメモリへの連続的なアクセスは比較的効率良く実行できるが、異なるワード線へのアクセスは効率が悪い。言いかえると、DRAMへのメモリ・アクセスは、その構造上、連続したアドレスのメモリ・アクセスは比較的効

率良く実行できるが、ランダムなアドレスのメモリ・アクセスは効率が悪い。従って、画像・音響処理プログラムの開発にあたっては、連続したアドレスのメモリ・アクセスを行いながら処理を行うように検討する必要がある。特にDRAMと広いバス巾でCPUを結合できる場合、ワード線上のメモリを一度で転送可能なシステム構成を取ることが考えられる。この場合は、ワード線のビット巾も考慮したメモリ・アクセスも検討する必要がある。

2.4 むすび

本章では、従来の画像・音響処理技術をマイクロコントローラを搭載した製品群に適用する上での問題点を、搭載されたハードウェア資源の制約と製品仕様から決まる処理の仕様に関する限定から、総合的に解析した。そして、ハードウェア資源の制約を受けながらも、画質や音質あるいは処理速度の面で実用的な範囲を実現するための手段や方法を検討した。特に、DRAM内蔵マイクロコントローラの持つハードウェア資源の活用、DRAMの特性を活かしたメモリ格納方法や人間の視聴覚モデルを用いた画像・音響処理の高速化等について検討した。その結果以下のような手段や方法を提案した。

(1) 定数のテーブル化
(2) 固定小数点化
(3) 乗算の実現方法と演算精度の検討
(4) 算術演算の精度検討
(5) 人間の視聴覚特性を考慮した演算精度の検討
(6) メモリ・アクセスを考慮した高速化手法

これらの手段や方法は、画像・音響処理全般に有効で、マイクロコントローラを対象とした画像・音響処理技術を開発するための一般的な開発手法として適用できる。これらの手法を実際の画像・音響処理に適用し、最大限の効果を得るためには、これらを基本的な手法として、適用の仕方を個別に工夫する必要がある。これに関しては、次章以降で具体的に述べる。

第2章の参考文献

1) 森、名取、鳥居; "岩波講座　情報科学-18　数値計算", pp.1-27,　岩波書店, (1982)

2) T. Okabe, T. Sakamoto and T. Hase; "Full Audio Software Solution for a 16-bit DSP Core for Digital Audio Decoder LSI", IEEE Trans. on Consumer Electrics, Vol. 44, No. 1, pp40-47, February, (1998)

3) ATSC A/52; "Digital Audio Compression (AC-3) Standard ", United States Advanced Television Systems Committee

4) M. Hashimoto, H. Iwabuchi, R. Chiba, E. Loker, and V. Fruchter; "Digital Signal Processor for Dolby AC-3 Decoding", AES 東京コンベンション95予稿集, pp116-119, (1995)

5) L. Bergher, X. Figari, F. Frederiksen, M. Froidevaux, J. M. Gentit, O. Queinnec; "MPEG Audio Decoder for Consumer Applications", CICC, pp413-416, (1995)

6) W. Lau and A. Chwu; "A Common Transform Engine for MPEG & Ac3 Audio Decoder", IEEE Trans. on Consumer Electronics, Vol. 43, No. 3, pp559-566, August, (1997)

7) M. Yasuda; "MPEG2 Video Decoder and AC-3 Audio Decoder LSIs for DVD Player", IEEE Trans. on Consumer Electronics, Vol. 43, No. 3, pp462-468, August, (1997)

8) L. Bergher; "Dolby AC-3 and MPEG-2 Audio Decoder IC with 6-channels Output", IEEE Trans. on Consumer Electronics, Vol. 43, No. 3, pp567-574, August, (1997)

9) ANSI X3,159-1989; "American National Standard for Information Systems - Programming Language - C", pp. 112-118, American National Standards Institute, Inc., (1989)

10) 郭, 浅井, 樽木, 坂本, 長谷; “FPUを持たない32bit MCU用GPSナビゲーションプログラムの高速化”, 映像情報メディア学会誌, Vol. 53, 10月号掲載予定

11) 伊藤; "超LSIメモリ", pp.79-110, 培風館, (1982)

第３章　デジタル・スチル・カメラ用画像処理技術

3.1　はじめに

マルチメディアの進展とともに簡易な画像入力手段としてデジタル・スチル・カメラが爆発的に普及している。デジタル・スチル・カメラには当初、家庭用ビデオ・カメラ用に開発されたCCDイメージセンサが使用されていた。しかし、デジタル・スチル・カメラではビデオ・カメラよりも高い解像度が要求される。そのため、最近、デジタル・スチル・カメラ用に新たに開発されたCCDイメージセンサでは、色信号の感度が良く、色再現性が良いベイヤー型原色方式[1]の色フィルタアレイを採用することが多くなってきた。

一般的なベイヤー方式のCCDのフィルタ配列を図1.4に示した。ベイヤー方式では、高解像度が必要な輝度信号に寄与する割合の高いG画素（本章ではR、G、Bの各ドットを画素と呼ぶ）を市松状に配置し、残りの部分にそれぞれGの半分の画素数にあたるR、Bをさらに市松状に配列している。デジタル・スチル・カメラやビデオ・カメラでは、通常CCDイメージセンサを１個だけ使う単板式を採用しているため、各画素（例えばG画素）において欠落している色成分（例えばRおよびB画素）を周囲の画素から補間する必要がある。欠落している残りの２色をこの周囲の画素から補間して生成する処理を画素補間と呼ぶ。デジタル・スチル・カメラは静止画撮像が目的となるため、ビデオ・カメラに比較して色の再現性などの高画質化が重視される。画素補間方式は画質に大きく作用するため、高画質化の重要なポイントとなる。

本章では、第２章で述べたマイクロコントローラを用いた画像・音響処理技術の開発手法を、デジタル・スチル・カメラ用画像処理に適用する場合について、具体的な高速化手法を示し、その性能を評価し、有効性を検証する。

3.2　画素補間方式の課題

画素補間方式としては、一般には拡大縮小などの幾何学的変換で研究された方式が転用される。これら幾何学的変換で研究された方式を利用した画素補間方式は、すべての画素に対して全色成分がある場合の方式である。一方、デジタル・スチル・カメラ用のCCDから得られるデータは、前述のように１画素当たりR、G、B３色のうち１色しかないのが普通である。このようなCCDのデータに対して、各種の幾何学的変換方式を画素補間方式として用いたときの適応性の比較や、特にデジタル・スチル・カメラのCCDで採用されることの多いベイヤー型の原色フィルタアレイに適した画素補間方式の検討は未だ十分に研究されていない。また、ソフトウェアで画素補間を実現する場合に適した方式や、演算量と画質の関係など、検討されていない課題が多い。

そこで、標準的な組込み型のマイクロコントローラとDRAMを用いてソフトウェアで実現することを前提として、ベイヤー型の原色フィルタアレイに適した画素補間方式を検討する。CCDのデータから静止画像を生成するために画素補間を使用するには、再生画像の画質が重要である。一方で、マイクロコントローラを用いソフトウェアで実現した画素補間方式を実際の民生機器に適用するためには、処理時間が課題となる。つまり、画素補間方式で得られる再生画像の画質と画素補間に必要になる演算量を両立させる画素補間方式を検討する必要がある。この一見相矛盾する両者を両立させるためには、第2章で述べた、人間の視覚特性を考慮して演算精度を工夫していくことが重要になる。

さて画素補間処理では、補間対象色と同一色の画素だけを用いて補間処理を行うのが一般的である。また近年の研究では、特に色信号の高解像度化に関する試みがなされている。例えば、色の相関を算出するため、局所的な色の相関を利用し画素信号の補間処理を行う方式や、水平方向や2次元のLPF（Low Pass Filter）を用いる方式などが報告されている[2)3)4)5)]。以下では、先ず補間対象色と同一色の画素だけを用いて処理を行う画素補間方式について検討し、次に色の相関を用いて画素補間方式の改良を検討する。そして、それらの性能を評価し、有効性を検証する。

3.3 同一色を用いた画素補間方式

本節では、同一色を用いた画素補間方式のうち代表的な8種類の画素補間方式をベイヤー型原色方式のCCDに適応したときの再生画像の特徴を、解像度および色の再現性の観点から調べる。先ず3.3.1節で、本節で議論する8つの画素補間方式について述べ、3.3.2節でその画素補間方式をCCDからのデータへ適応する方法について述べる。3.3.3節章では、各方式の特徴を調べた結果と若干の考察をする。これにより、デジタル・スチル・カメラに適した同一色を用いた画素補間方式を示す[6)]。

3.3.1 画素補間方式の概要

本節では、幾何学的変換で用いられている画素補間方式の中から今回検討する方式について説明する。

本節では、もっとも単純な補間方法として「最近傍法」、周囲の4個の画素を用いて線形補間を行う「線形補間法」、さらに精度の高い補間を行う方法で、周囲の16個の画素を用いて3次式による補間を行う方法である「3次補間法」、そして「周囲の画素の相関を考慮して線形補間を行う方法」の4つの方法を検討対象とした。以下では各々の方式について簡単に説明する。

なお本節では、補間を行う画素を(x, y)、補間に用いる画素を(x_k, y_l)とし、(x, y)の濃度値c(x, y)を以下の式によって求める。ここで濃度値とは、画素の濃淡を表す離散的な整数値である。

$$c(x, y)=\sum_{k}\sum_{l} c(x_k, y_l)h(|x - x_k|)h(|y - y_l|) \tag{3.1}$$

h(x) は画素(x_k, y_l)に対応する濃度値$c(x_k, y_l)$に掛け合わせる重み係数と呼ぶ。

(1) 最近傍法

もっとも近い画素の濃度値を補間する画素の濃度値とする。つまり、例えば図3.1に示す非格子点(x', y')での画素補間を考えると、非格子点(x', y')の周囲4個の格子点(x+1,y+1)、(x+1,y+2)、(x+2,y+1)、(x+2,y+2)のうち最も近い格子点(x+1, y+2)の濃度値を補間する画素の濃度値とする。

近傍の点が均等な距離にある場合は、そのうちの1つの画素を選択し、その濃度値を補間する画素の濃度値とする。重み係数は以下の式で与えられる。

$$h(x)=\begin{cases} 1 & 0 \leqq x \leqq 0.5 \\ 0 & 0.5 < x \end{cases} \tag{3.2}$$

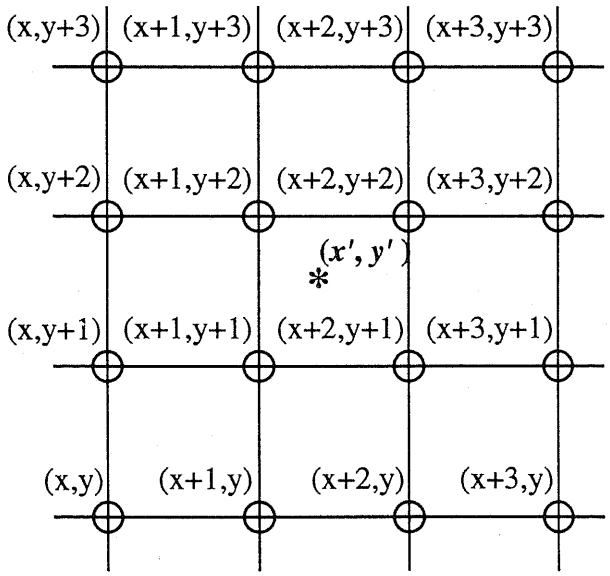


図3.1 画素補間と格子点

(2) 線形補間法

周囲の4個の画素の濃度値を用いて線形補間を行う。つまり、例えば図3.1に示す非格子点(x', y')での画素補間を考えると、非格子点(x', y')の周囲4個の格子点(x+1,y+1)、(x+1,y+2)、(x+2,y+1)、(x+2,y+2)の濃度値を用いて線形補間を行う。重み係数は以下の式で与えられる。

$$h(x)=\begin{cases}1-x & 0\leqq x\leqq 1\\ 0 & 1<x\end{cases} \tag{3.3}$$

(3) 3次補間法

周囲の16個の画素における濃度値に対し、3次式で求めた重み係数を用いて補間を行う。つまり、例えば図3.1に示す非格子点(x', y')での画素補間を考えると、図に示した16個の格子点の濃度値を用いて補間を行う。3次式は理論的には最適値であるsinc関数に近い値が得られるように決める。

3次式の代表的なのものは、以下の式で表されるもので、以下では1変数3次補間法と呼ぶ。

$$h(x)=\begin{cases}(a+2)x^3-(a+3)x^2+1 & 0\leqq x<1\\ ax^3-5ax^2+8ax-4a & 1\leqq x<2\\ 0 & 2\leqq x\end{cases} \tag{3.4}$$

変数aとしては、a = -1[7]、a= 0.5[8]、a=0.75[9]などが報告されている。

また、2変数を用いて重み係数を求める3次式を表す方式がある[10]。その3次式を以下に示す。この方式を以下では2変数3次補間法と呼ぶ。

$$h(x)=\begin{cases}\frac{1}{6}(-9b-6c+12)x^3+\frac{1}{6}(12b+6c-18)x+\frac{1}{6}(-2b+6) & 0\leqq x<1\\ \frac{1}{6}(-b-6c)x^3+\frac{1}{6}(6b+30c)x^2+\frac{1}{6}(-12b-48c)x+\frac{1}{6}(8b+24c) & 0\leqq x<2\\ 0 & 2\leqq x\end{cases} \tag{3.5}$$

変数(b, c)として、(b, c) = (0.33, 0.33)、(b, c) = (1.5, -0.25)などが報告されている[11]。前述した1変数3次補間法も、この式で与えられる。たとえば、(b, c) = (0, -c)とすれば、1変数3次補間法と同じ式になる。

また、3次B-splineと呼ばれる方式も報告されている[12]。3次B-splineは、以下の式で与えられる。

$$h(x)=\begin{cases}\frac{3}{6}x^3-x^2+4 & 0\leqq x<1\\ -\frac{1}{6}x^3+x^2-2x+\frac{8}{6} & 1\leqq x<2\\ 0 & 2\leqq x\end{cases} \tag{3.6}$$

(4) 相関を考慮した線形補間法

周囲の4個の画素における濃度値を用いて線形補間を行う。ただし、周囲の画素の縦、横の相関を調べ、相関の強い方向（上下または左右方向の画素それぞれの差分を求め、差分の少ない方向）の2画素の濃度値を用いて線形補間を行う。

2画素の線形補間は、以下の式で与えられる。

$$c(x,y)=\sum_k c(x_k, y_l)h(|x-x_k|) \tag{3.7}$$

または、

$$c(x,y)=\sum_l c(x_k, y_l)h(|y-y_l|) \tag{3.8}$$

重み係数は以下の式で与えられる。

$$h(x)=\begin{cases}1-x & 0\leqq x\leqq 1\\ 0 & 1<x\end{cases} \tag{3.9}$$

また、相関が同じ（上下、または左右方向の画素の差分が等しい）場合は、(2)で述べた線形補間法を用いる。

3.3.2 画素補間方式のデジタルカメラへの適応

本節では3.3.1節で述べた画素補間方式をデジタル・スチル・カメラへ適用する方法について説明する。

3.1節で述べたように、最近のデジタル・スチル・カメラで採用されている撮像方式における色成分の補間方式を検討対象とした。つまり、ベイヤー型原色方式の色フィルタアレイを採用したCCDイメージセンサを単板式で用いた場合の補間方式を検討対象とした。

以下の節では補間方式の適用の仕方について説明する。

(1) 最近傍法

上下、または左右のなかの1点の濃度値を補間する点の値とする。どの画素を選択してもよい。以下、この方式をnrstと略す。

(2) 線形補間法

Gの画素は、上下左右にある4個の画素の濃度値の平均値を補間画素の濃度値とする。RまたはBの画素については、図3.2に示したような上下、または左右にRまたはBの画素がある場合（G画素位置上での補間）は、その2画素の濃度値の平均値を補間画素の濃度値とする。たとえば、図3.2(a)の場合、G画素位置上におけるBの濃度値は、上下のB画素の濃度値の平均値、Rの濃度値は、左右のR画素の濃度値の平均値となる。また、図3.3に示したような上下左右にRまたはBの画素がない場合（RあるいはB画素位置上での補間）は、補間画素の斜め上の2画素と斜め下の2画素の計4画素の濃度値の平均を補間画素の値とする。たとえば、図3.3(a)の場合、R画素位置上のBの濃度値は、斜め上のB画素2個と、斜め下のB画素2個の濃度値の平均値となる。以下、この方式をlnrと略す。

G	B	G
R	G	R
G	B	G

(a) 左右にRがある場合

G	R	G
B	G	B
G	R	G

(b) 左右にBがある場合

▨：補間する画素位置を示す

図3.2 G画素位置上でのR/Bの補間

B	G	B
G	R	G
B	G	B

(a) Bの補間

R	G	R
G	B	G
R	G	R

(b) Rの補間

▨：補間する画素位置を示す

図3.3 R/B画素位置上でのB/Rの補間

(3) 3次補間法

求めたい画素(例えばG画素)の周囲n×n配列の中の同色画素（G画素）のみに注目する。同色画素Xiに対し、(3.1)を適応して、以下の演算により補間画素の濃度値を求める。

$$c(x,y)=\sum_{k} c(x_k,y_k)h(|x-x_k|)h(|y-y_k|) \qquad (3.10)$$

ただし、Xiの座標を(x_i, y_i)とし、重み係数は3.3.1節の(3)で説明した3次式で求める。

画素配列が図1.4に示したベイヤー方式のCCDでは、同色画素に注目すると16画素すべてが存在するわけではない。そのため、実際に濃度値を求めるときに用いられた重みづけ係数の和（以下、直流ゲインと呼ぶ）が1にならない。直流ゲインが1にならない場合、正しい濃度値の大きさが得られない。そこで、直流ゲインが1になるように係数を正規化する。

正規化を行うため、1変数3次補間法では、3次式の変数aを異なる値にしても同じ重みづけ係数となる。よって、本節では、1変数3次補間法では、a = -1 の場合のみ評価する（以下、この方式をcbと略す）。また、2変数3次補間法では、3次式の変数(b, c)を(0.33, 0.33)としたもの（以下この方式をctと略す）、(1.5, 0.25)としたものの（以下この方式をctwと略す）2通りの方法を評価する。また、3次B-spline方式（以下この方式をbspと略す）についても評価する。

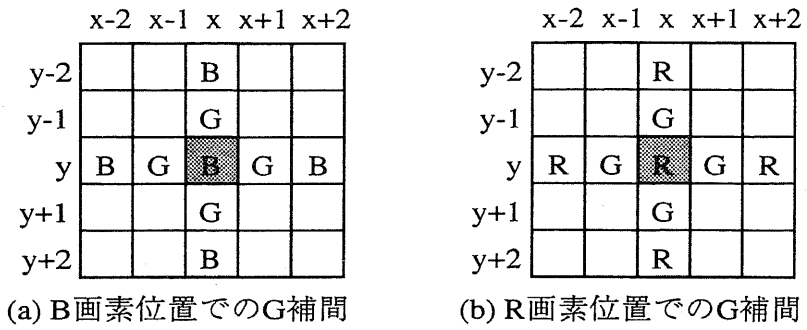
(4) 相関を考慮した線形補間法

Gの画素に対してのみ、相関を考慮した線形補間法を用いる。R/B画素は、同色画素が少ないので、(2)で説明したR/B画素の補間方式を用いる。以下、G画素の補間方式について説明する。

自然画像では、RGB成分には強い相関関係があることが報告されている[12]。補間したい画素の周辺の相関を求める場合、次の2種類の方法が考えられる。

まず、同色のみに注目した場合（以下、rltと略す）について説明する。補間したい画素（以下、補間画素と呼ぶ）位置の上下または、左右方向の画素はGの画素である。そこで、上下左右それぞれの差分を求め、差分の少ない方向の2画素の濃度値の平均を補間画素の濃度値とする。差分が同じ場合のみ、上下左右の4画素の濃度値の平均を補間画素の濃度値とする。

次に、補間したい画素位置上の色に注目し、その色に対して上下または左右の相関を求める方式について説明する（以下、crlと略す）。たとえば、図3.4(a)の(x, y)位置のB画素（以下、B(x, y)と記述する）におけるGの濃度値を求める場合、色Bの縦方向、横方向の相関を求め、相関の強い方向にあるGの画素を利用して求める。具体的には、B(x-2, y)、B(x+2, y)の濃度値の平均とB(x, y)の濃度値の差分、B(x, y-2)、B(x, y+2)の濃度値の平均とB(x, y)の濃度値の差分を求め、差分の少ない方向にある2個のG画素の平均を求める濃度値とする。仮に、縦方向の相関が強い場合、すなわちB(x, y-2)、B(x, y+2)の濃度値の平均とB(x, y)の濃度値の差分が少ない場合、G(x, y-1)、G(x, y+1)の濃度値の平均がB(x, y) 画素位置上のGの濃度値となる。図3.3(b)も同様である。ただし、差分が同じ場合は、上下左右のG画素の平均値が補間画素位置のGの濃度値となる。



(a) B画素位置でのG補間　　(b) R画素位置でのG補間

▩ : 補間する画素位置を示す

図 3.4 相関を考慮した線形補間法（補間したい画素上の色に注目した場合）

3.3.3 評価結果

本節では、3.3.2節で説明した画素補間方式の特徴を考察するために行った評価結果をまとめる。評価は、解像度、色の再現性、組込み型マイクロコントローラにおける処理に要したサイクル数とS/N比の関係の3つについて行った。

CCDから得られる画像データに基づいて補間方式を比較する場合、図1.4のサンプリング方式によって得られるデータが基本となるため、画素ごとの各色成分の基準値が参照できず、比較および評価が難しいという問題がある。そこで、評価を行うにあたって、各画素に対し全ての色成分がそろっている画像を利用し、図1.4の方式に従って、画素毎にRGB成分のいずれか1つを取り出すことによりCCDから得られる画像データを作成した。

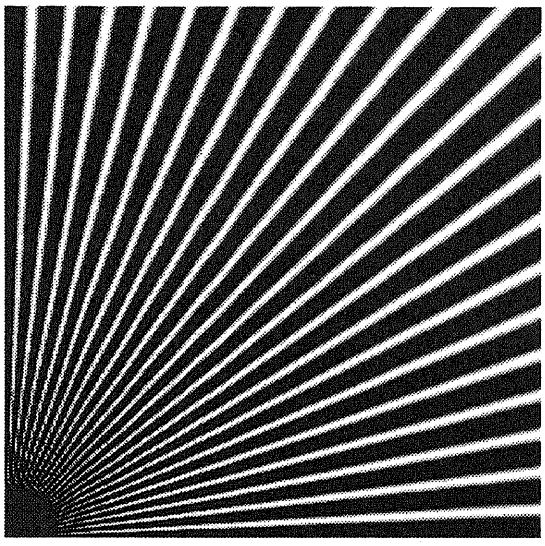
また、解像度、色再現性については、客観的、定量的に評価する方法は知られていない。以下で行った評価は、すべての色成分がそろっている画像（以下、原画像と呼ぶ）と各補間方法で得られた画像（以下、再現画像と呼ぶ）を用い、その違いを主観的または定量的に比較することによって行った。

評価を行った補間方式をまとめておく。補間方式は、以下の8種類である。括弧内に以下での略称を示す。

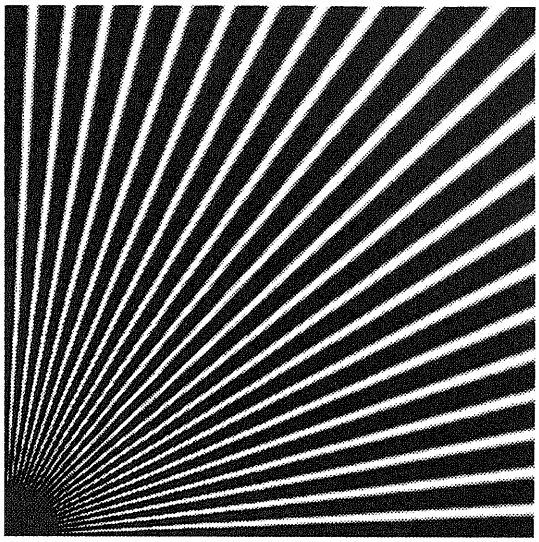
1. 最近傍法(nrst)
2. 線形補間法(lnr)
3. 1変数3次補間法(cb)
4. 3次B-spline法(bsp)
5. 2変数3次補間法、(b, c) = (0.33, 0.33) (ct)
6. 2変数3次補間法、(b, c) = (1.5, 0.25) (ctw)
7. 同色の画素に対する相関を考慮した線形補間法(rlt)
8. 補間する画素位置上の色に対する相関を考慮した線形補間法(crl)

(1) 解像度

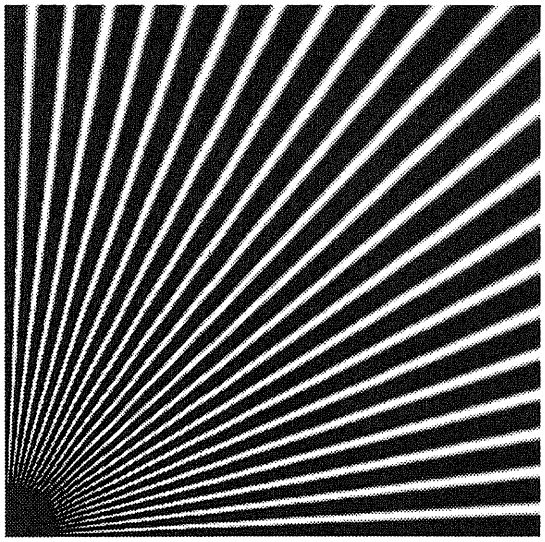
各補間方式における解像度の違いを評価した結果を示す。解像度の評価に用いた画像は、1.2.4節に示した高詳細カラーデジタル標準画像データ（以下、SCIDと略す）の多値解像度チャートである。図3.5に各画素補間の適用結果を示す。最近傍法（nrst）以外では、解像度という観点から絵を見る限り特に大きな違いが見られなかった。



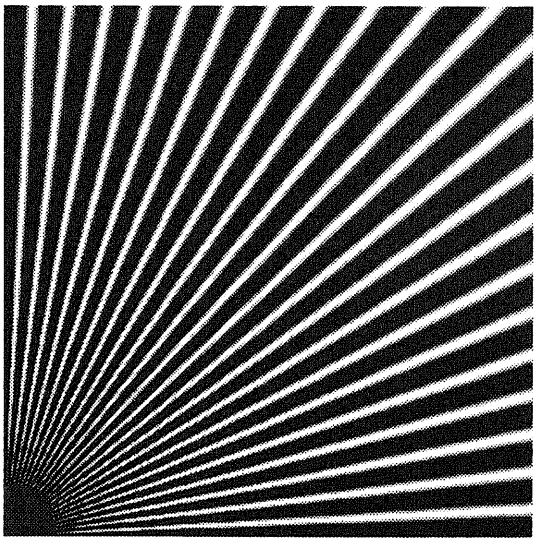
(a) nrst方式の適用結果



(b) lnr方式の適用結果

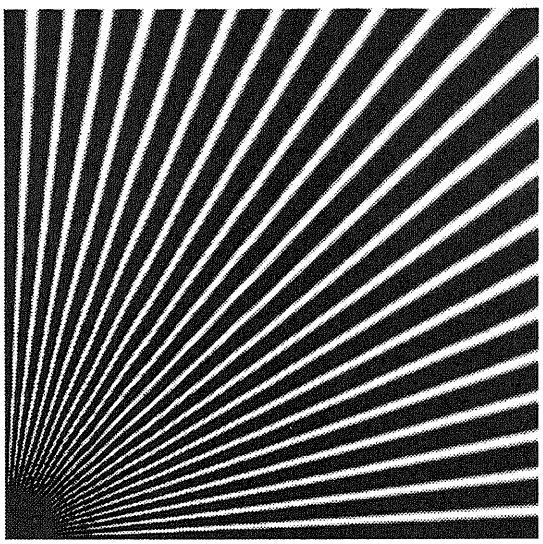


(c) cb方式の適用結果

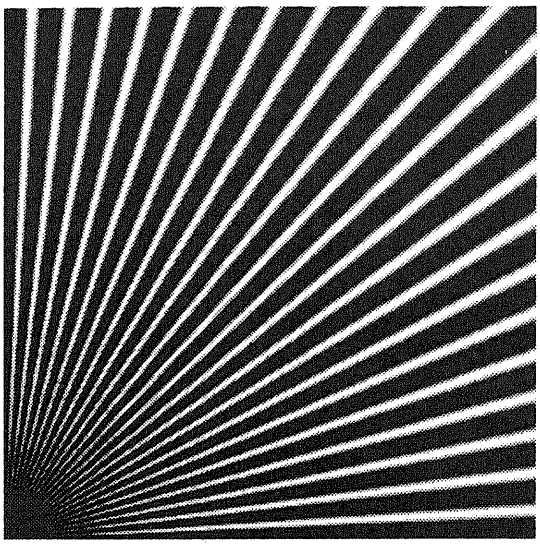


(d) bsp方式の適用結果

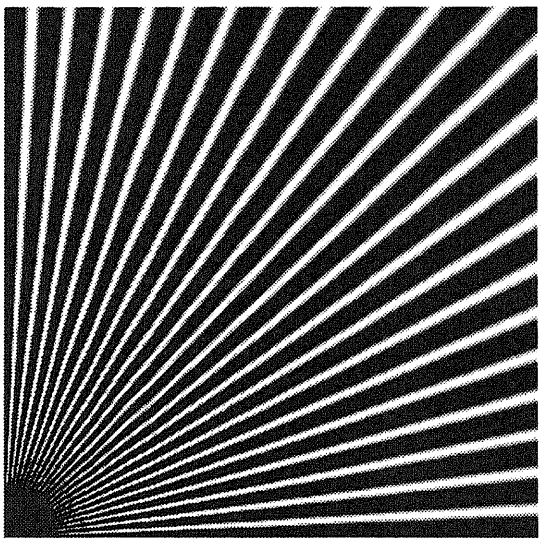
図3.5　解像度の比較（nrst、lnr、cb、bsp方式を適用）



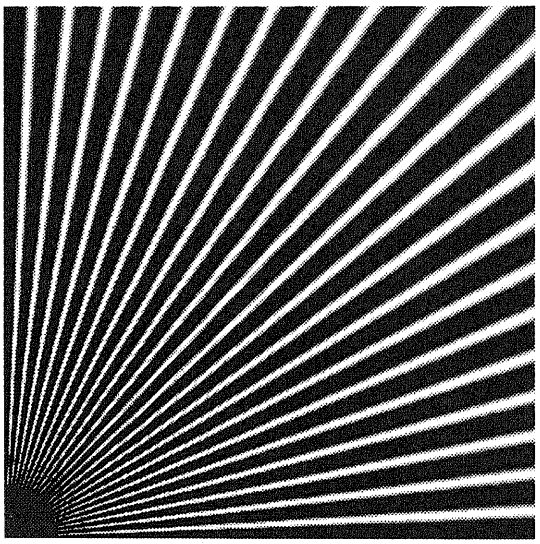
(e) ct 方式の適用結果



(f) ctw 方式の適用結果



(g) rlt 方式の適用結果



(h) crl 方式の適用結果

図 3.5　解像度の比較（ct、ctw、rlt、crl 方式を適用）

(2)　色の再現性

各画素補間方式における色の再現性を評価した結果を示す。

(1)の解像度の評価に用いた図3.5を見ると、各画素補間方法による違いは、主に高周波数領域で見られる偽色の発生の大きさであることがわかる。CCDから得られる入力データは、拡大縮小などの幾何学的変換と異なり、すべての画素に対してRGBいずれかの画素の値は存在する。そのため、再現画像の質は解像度の劣化よりも、サンプリング方式が異なることや補間する画素が色ごとに異なることによって発生する偽色の視覚への影響の方が大きいことがわかる。

この(1)での結果に基づき、色の再現性を評価するため評価画像としてCircular Zone Plate (CZP)を用いた。CZPは計算式

$$f(x,y)=C_1\times\cos\left(\frac{\pi}{320}x^2+\frac{\pi}{240}y^2+\pi\right)+C_2 \tag{3.11}$$

を用いて作成した。画像サイズは640×480画素（VGAサイズ）で、画面の中心を(0,0)とし、(±320,0)と(0, ±240)がナイキスト周波数になるように作成している。また、各色成分はf(x,y)とした。

CZPを用いたのは、周波数の違いによる色再現性の違いを評価するためである。評価は、周波数の変化に伴って、再現画像の各色の濃度値と原画像のその画素における各色の濃度値の差がどのように変化するかによって行った。評価を行った領域は、画素の配列を考慮し、図3.6に示した水平方向と斜め45度方向の2領域を選択した。図3.7と図3.8に評価結果を示す。縦軸は濃度値の差、横軸は円の中心からの距離を現す。中心から離れるにしたがって、周波数が高くなるため、周波数の増加と濃度値の差の関係が分かる。

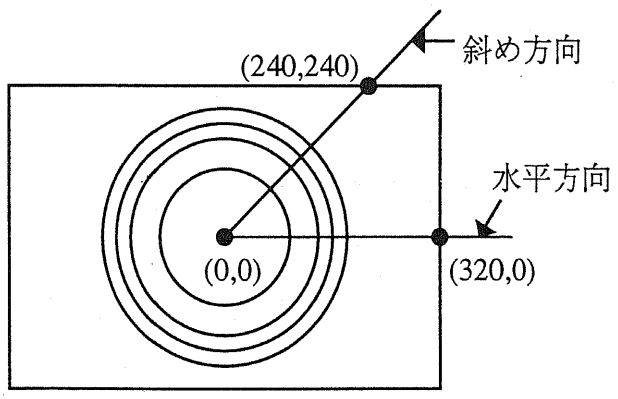


図3.6　評価領域

図3.7(a)は、Gの色の水平方向の色再現性を示したものである。図の縦軸は原画像と再生画像の濃度差を表すため、値が小さい程、色の再現性が良いことを示している。CZPでy座標が零の水平方向の場合、画素の縦方向の相関が強いため、rlt方式とcrl方式が最も良い。また、隣接画素の濃度値が近いため、広い領域を用いて補間を行う方式よりも、隣接画素の平均をとるlnr方式で良い色再現性が得られている。

図3.7(b)は、Gの斜め方向の色再現性を示したものである。図からもわかるように、この領域では各方式の特徴の違いが大きい。nrst方式は、周波数が高くなるにしたがって、徐々に再現性が悪くなっている。一方、rlt方式とcrl方式は低周波数の領域では再現性がよいが、中心からの距離が160のあたりから突然再現性が悪くなっている。また、ctw方式やbsp方式は高周波領域での再現性が良いが、低周波数領域での再現性が他の方式に比べて悪い。また、nrst方式以外の全補間方式おいて中心からの距離が133～155のあたりで再現性がよくなっている。これは、その領域のデータが、補間画素の隣接2画素の平均を取るという方式に適合するデータになっているためである。

図3.8は、RまたはBの画素の濃度値の差を示したものである。図3.8には色再現性で特徴が見られた2通りの場合を示した。図3.8(a)は水平方向で、その線上に補間する色の画素が存在する行の濃度値の差を現したもの、図3.8(b)は斜め方向で、補間する色の画素が存在しない線上の濃度値の差を現したものである。図3.8(a)からもわかるようにctwやbspは、高周波数の領域における差の絶対値は少ないが、低周波数の領域における差が他の方式よりも大きいので全体的には悪くなっている。また、図よりnrst以外の他の方式は、似たような傾向を示していることがわかる。図3.8(b)も図3.8(a)と同じような傾向が見られる。

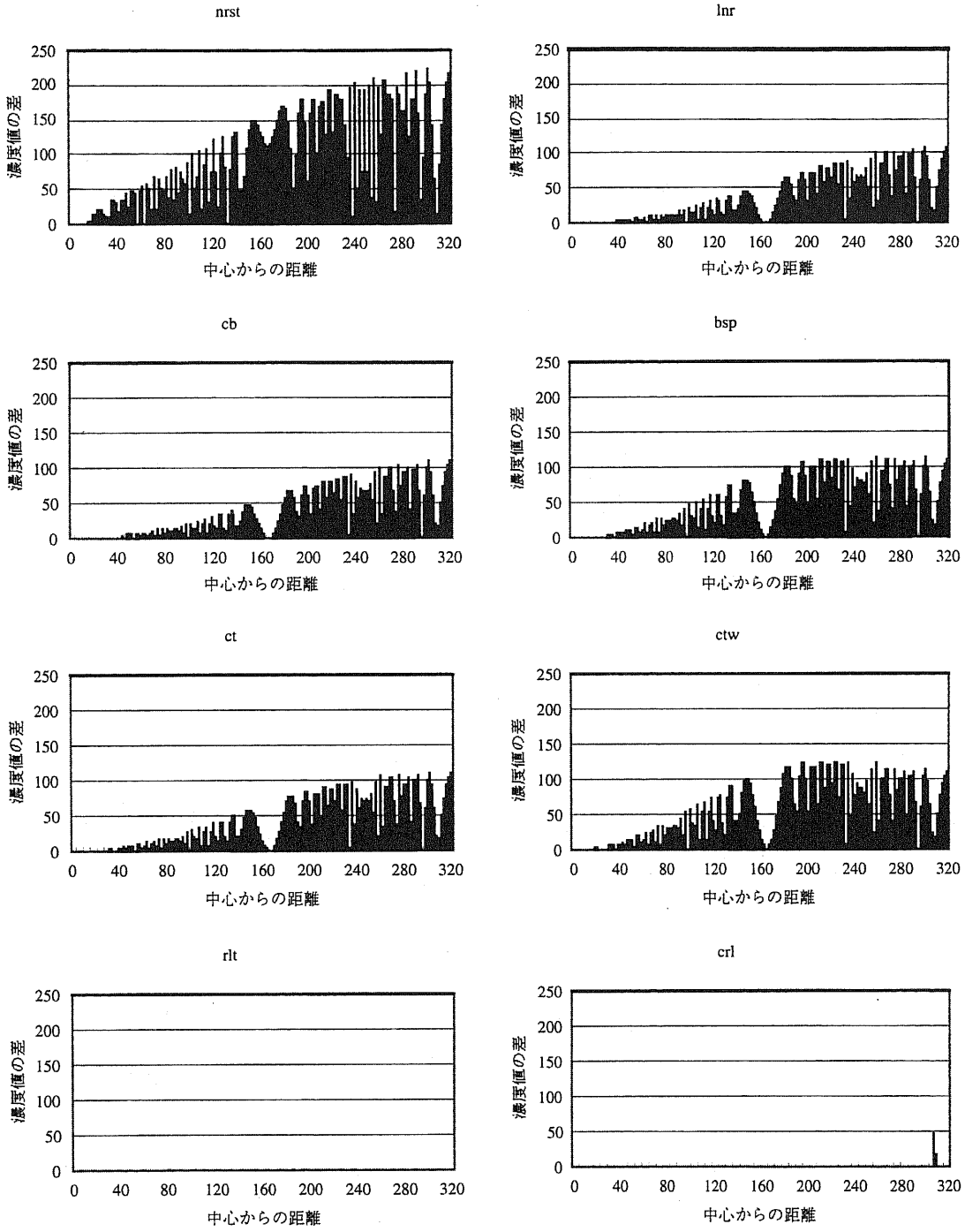


図 3.7(a)　水平方向の G 画素の濃度値の差

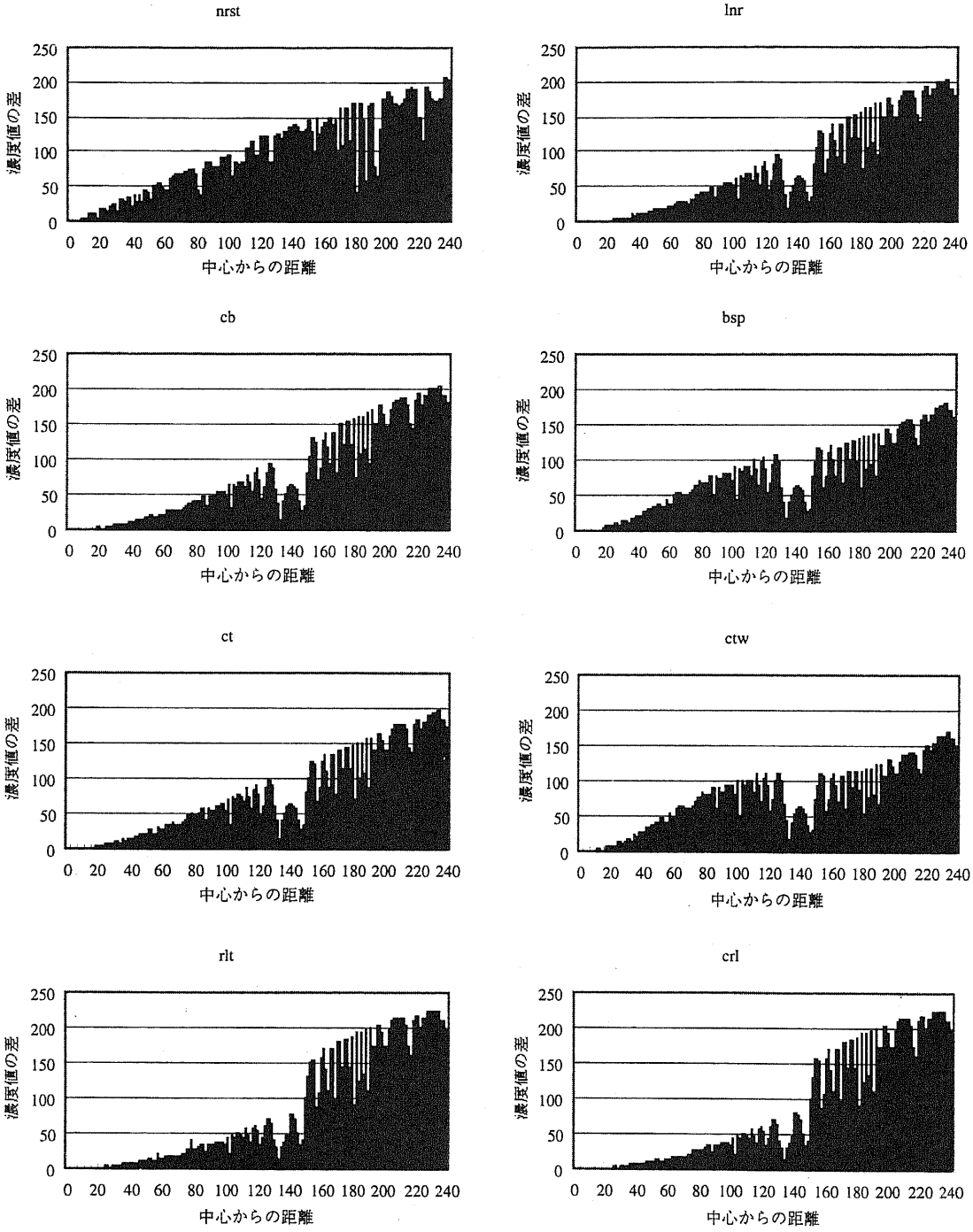


図 3.7(b)　斜め方向の G 画素の濃度値の差

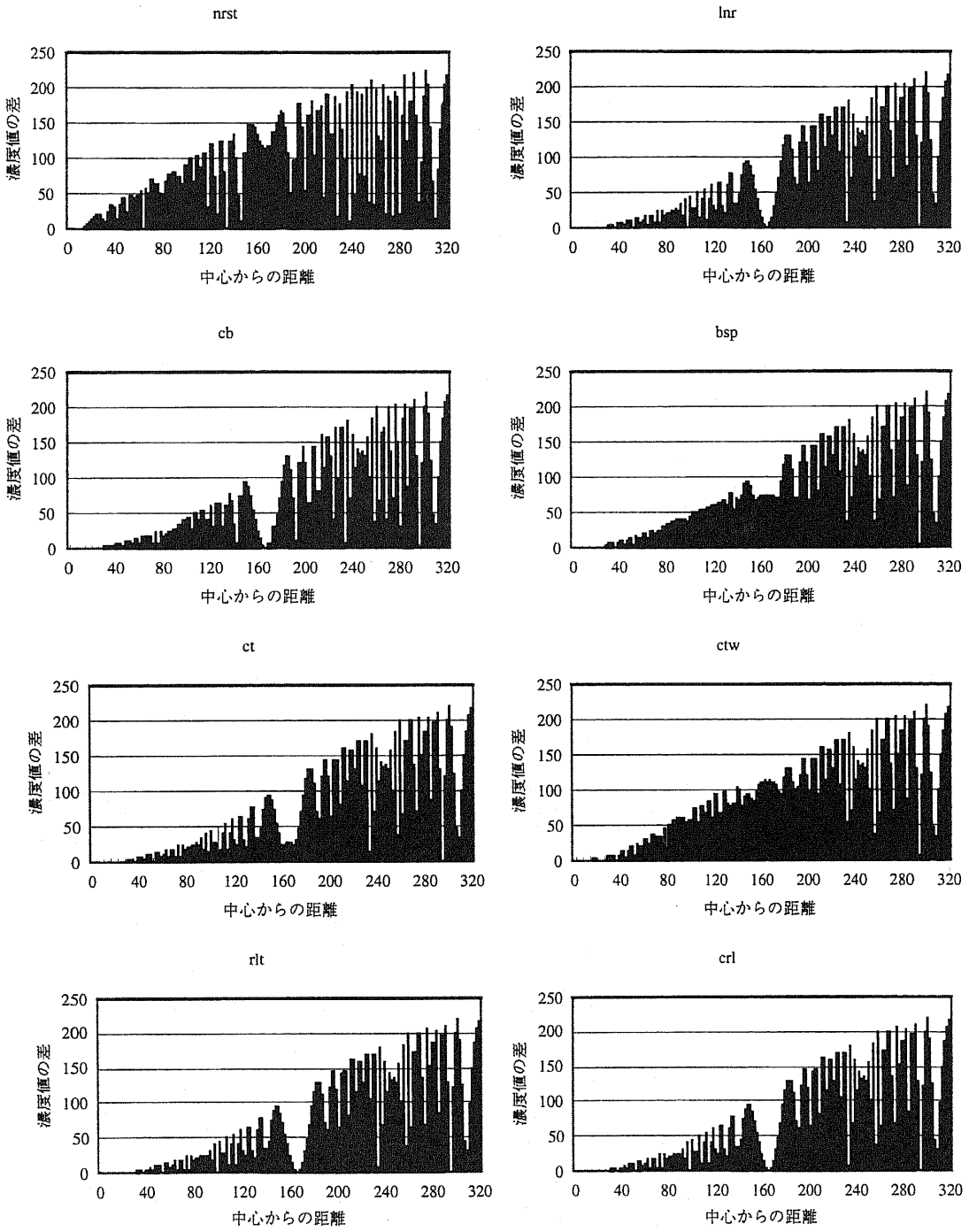


図 3.8(a)　補間する色の画素が存在する水平方向のＲまたはＢ画素の濃度値の差

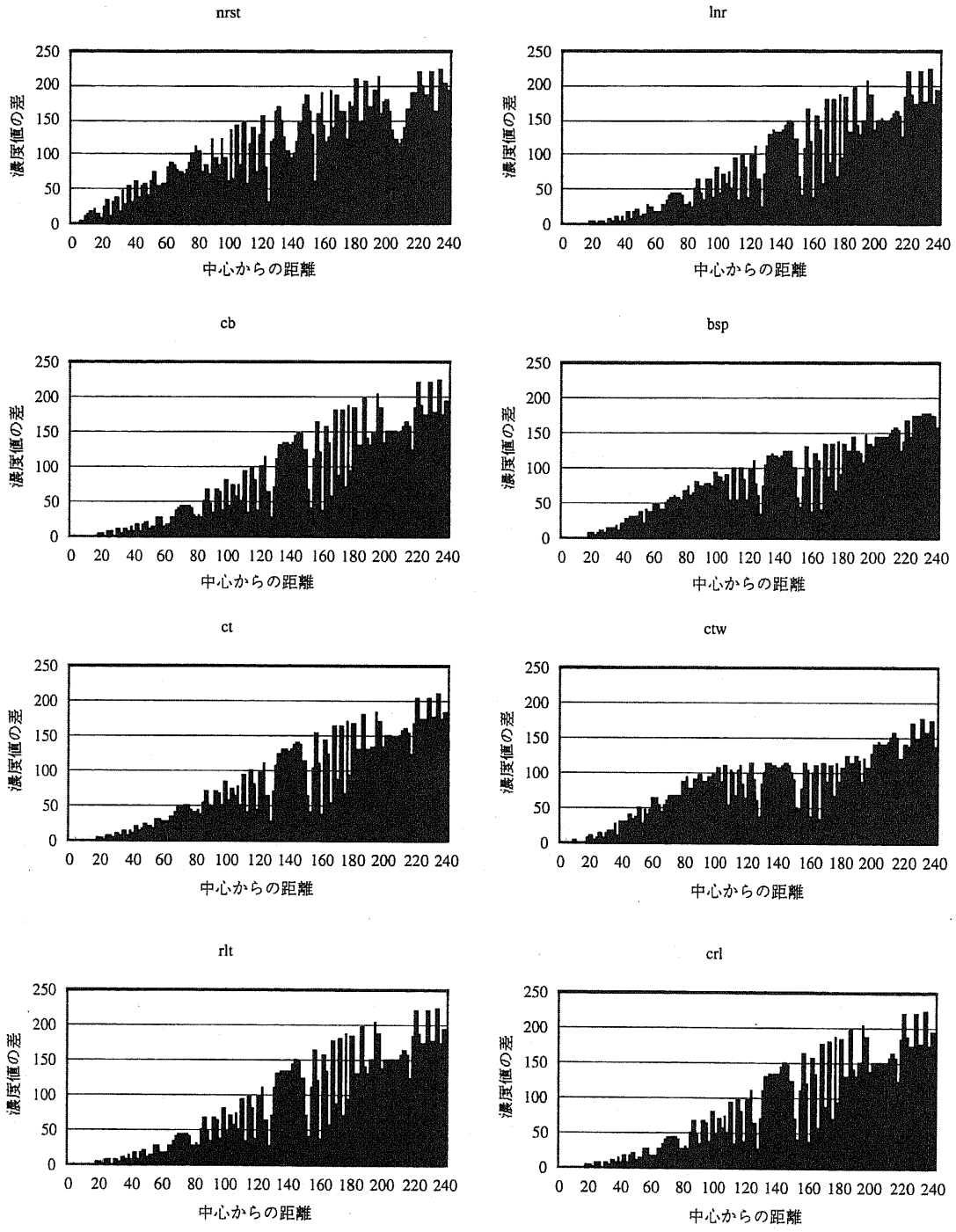


図 3.8(b)　補間する色の画素が存在しない斜め方向の R または B 画素の濃度値の差

(3) 演算量とS/N比

画素補間方式を実際の民生機器に適用するには、実現した時のコストや処理速度も重要である。そこで、各画素補間方式が安価な実現方法で実用的な演算量であるかを考察するため、実際に組込み型マイクロコントローラを用いて各方式をアセンブラ言語で実現し演算量を求めた。また、画質を客観的に比較するため、以下のS/N式[13]を用いた。

$$SNR = -10 \log_{10}\left[\frac{\frac{1}{KL}\sum_{k=0}^{K-1}\sum_{l=0}^{L-1}\{f(k,l)-f'(k,l)\}^2}{255^2}\right][dB] \qquad (3.12)$$

ただし、原画像全体の大きさをK×Lとして、f(k, l)は原画像の濃度値、f' (k, l)は再生画像の濃度値とする。

演算量は実行サイクル数を求めることによって算出した。実行サイクル数は正確なサイクル数を計算できるソフトウェア・シミュレータを用いて求めた。

評価に用いた画像は、(2)で用いたCZPにそれぞれ異なる色をつけた7種類の画像と1.2.4節に示したテレビジョン・システム評価用デジタル標準画像5チャートをVGAサイズに加工したもの使用した。

図3.9にCZPを用いた場合の評価結果を、図3.10に5種類の標準画像の評価結果を示す。図の縦軸はS/N比を示し、値が大きいほど原画像に近く良い画質を再現していることを表す。また、図の横軸は画素補間処理に要するサイクル数を示す。サイクル数に動作周波数（66MHz）の逆数をかけることで実行時間が求められるため、サイクル数が大きくなるほど（つまり、右方向にいくに従い）、画素補間に要する処理時間が大きくなることを表す。

図3.9および図3.10からわかるように、nrst方式、lnr方式、rlt方式、crl方式、およびcb方式、bsp方式、ct方式、ctw方式の3次補間方式の順に演算量が多くなっている。この結果から、動作周波数66MHzのマイクロコントローラを用いて処理した場合、最大0.22秒程度で処理可能である。これにより、演算量はどの方式も実用的な範囲と言える。一方、S/N比は、図3.9においてはctw方式がもっとも良く、rlt方式、crl方式は、演算量が多いにも係わらずあまり良いS/N比が得られていない。また、lnr方式は、演算量が少ない割にはある程度よいS/N比が得られている。

S/N比に関しては、自然画とCZPとは異なった傾向が見られた。図3.10の場合、rlt方式、crl方式において良い画質が得られている。この理由は次節で考察する。

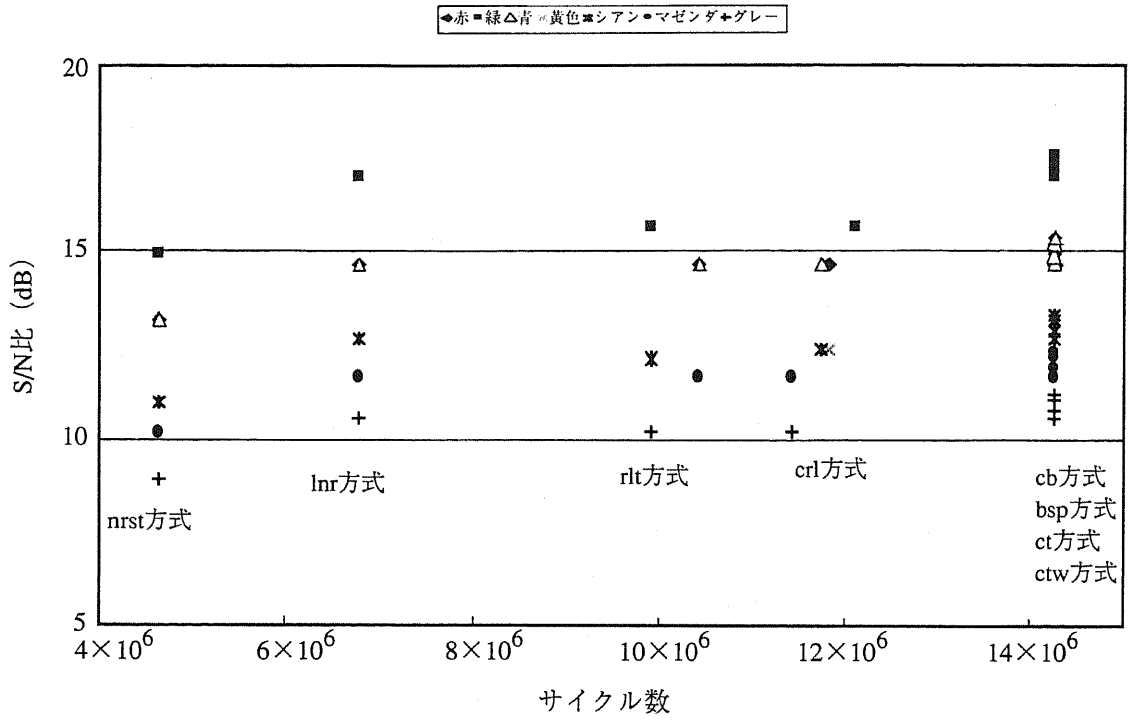


図 3.9　演算量と S/N 比（CZP 使用時）

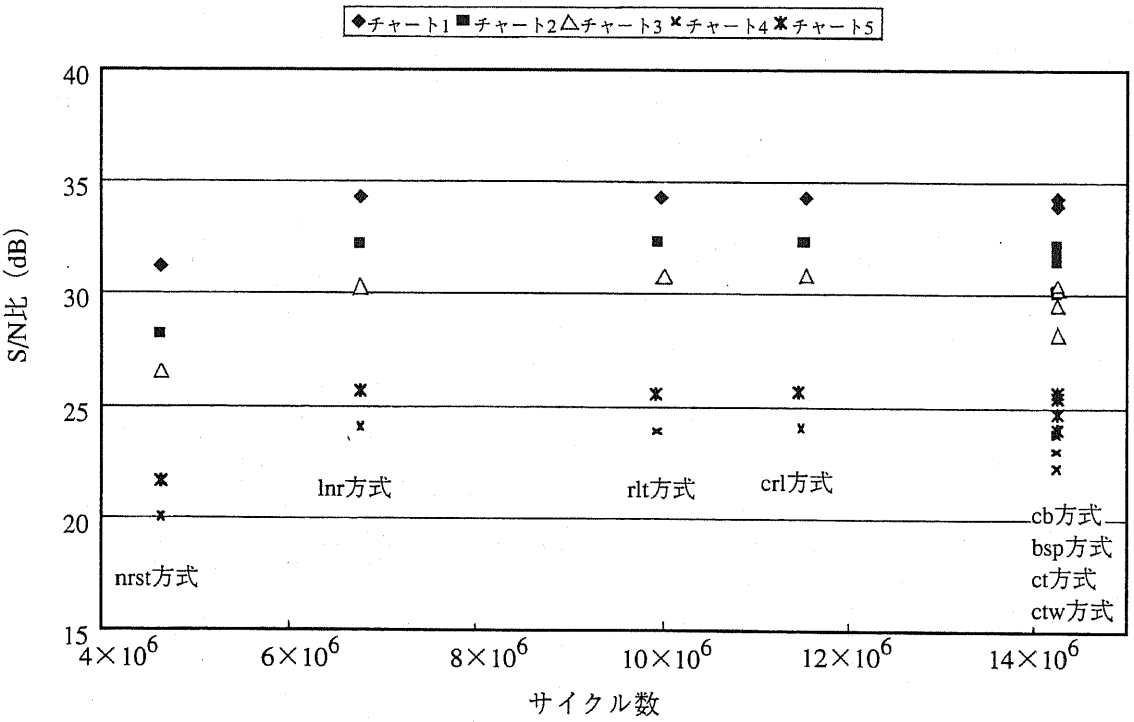


図 3.10　演算量と S/N 比（テレビジョン・システム評価用デジタル標準画像使用時）

3.3.4 考察

本節では、3.3.3節の評価結果から、デジタル・スチル・カメラに適応する補間方式について考察する。

演算量とS/N比の両方を考慮すると、lnr方式がもっともコストパフォーマンスが良い。しかし、デジタル・スチル・カメラが撮像する静止画においては、画質に重点をおく傾向がある。この観点からすると、必ずしもlnr方式がよいとはいえない。デジタル・スチル・カメラで撮像されるのは自然画が多いことを考慮に入れると、3.3.3節の(3)の図3.9の評価結果からrlt方式またはcrl方式を適応するのが良いと考えられる。

3.3.3節の(3)の評価で自然画とCZPのS/N比において特性の違いが見られている原因としては、自然画では画素間の相関が強いこと、画像全体において低周波数で構成される領域の占める割合が高いことが考えられる。これは、図3.7(a)からわかるように、相関が強い場合はrlt方式とcrl方式の色再現性が特に良いこと、また、図3.7(a)、(b)からわかるように、ctw方式においては、高周波領域の色再現性は良いが低周波領域の色再現性が悪いという特徴がある一方、rlt方式やcrl方式は、高周波領域の色再現性は悪いが、低周波領域の色再現性が良いという特徴があることからも確認できる。これらの色再現性の特徴が画像全体のS/N比へ影響し、自然画においては、演算量の多い3次補間方式よりも相関を考慮した線形補間法の方がよいS/N比が得られている。

以上を考慮すると、今回比較した方式の中では、デジタル・スチル・カメラに適応する方式はcrl方式であると結論づけられる。

3.4 色の相関を用いた画素補間方式

3.3節では、ベイヤー型原色方式のCCDに最適な同一色を用いた画素補間方式について検討した。得られる再生画像の特徴、組込み用途の32ビット・マイクロコントローラを用いて補間を実現するための演算量と、その結果得られる画質の観点から各画素補間方式について考察した。その結果、相関を考慮した線形補間法が、特に相関を求めるのに補間を行う画素位置上の色に注目し、その色に対して上下または左右の相関を求める方式が最適であることがわかった。本節では、この結果を更に押し進め、色の相関を用いた画素補間方式をベイヤー型原色方式のCCDに適応したときの再生画像の特徴を、解像度、色の再現性の観点から調べる。先ず3.4.1節で、局所的な色の相関を用いた補間方式の基本的な考え方を説明し、3.4.2節で評価を行う各補間方式の具体的な実現方法について説明する。3.4.3節章では、各方式の特徴を調べた結果と若干の考察を行う。これにより、色の相関を用いることでより高い画質の再生画像が得られることを示す[14]。

3.4.1 局所的な色の相関を用いた補間方式の概念

ベイヤー型原色フィルタによって得られる画像データにおいては、局所的な範囲では3色の色信号間に極めて強い相関が存在することが報告されている[15]。

局所的な色の相関を用いた補間方式の基本的な考え方として、以下の2通りの場合に分類できる。

(1) 各色信号の変化量が等しいと仮定する場合

5×5画素程度の範囲では例えばGの変化量とRの変化量が等しいと仮定できる[4]。

この仮定に基づけば、例えば文献4の提案によると図3.11(a)に示すように、ライン状に並んだX_1～X_5の位置にR_1、G_2、R_3、G_4、R_5が配置し、中央のX_3でのG_3を補間する場合に、Gの予測誤差ΔをRの変化量を用いて算出できる。つまり、(3.13)式のように、両側のGだけでなく、R放物線のカーブの変化量$f(R_1,R_3,R_5)$を利用する。

$$G_3 = G_3' + \Delta = \frac{G_2+G_4}{2} + f(R_1,R_3,R_5) \tag{3.13}$$

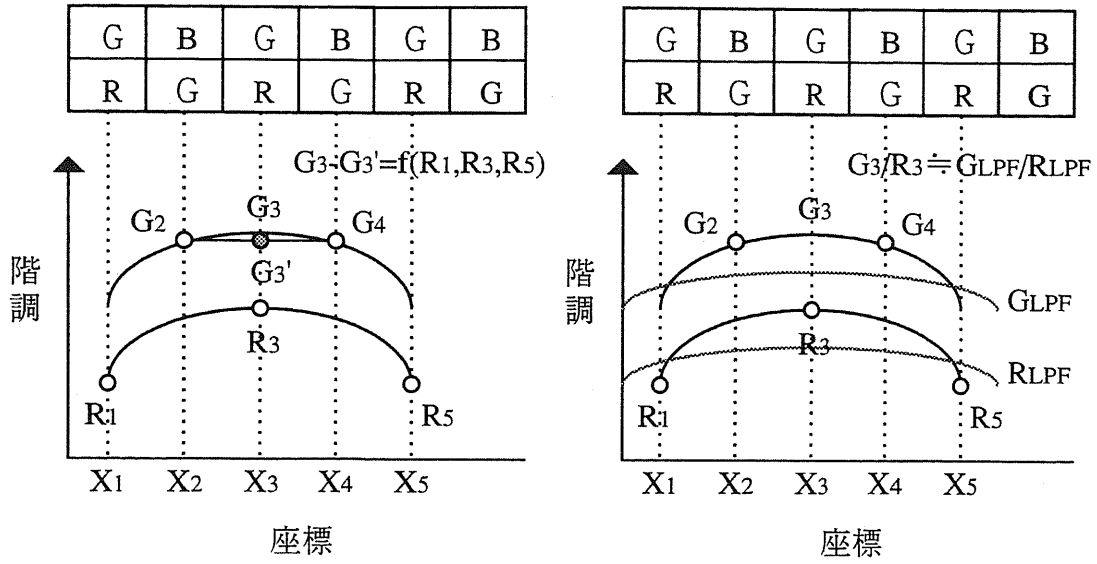
文献4では、事前に画像中のエッジやストライプなどを検出し、補間式を動的に変更し、演算に用いる画素を変更している。これらの処理はホストコンピュータ上のソフトウェアによって行っている。

(2) 各色信号の低周波成分の比率が等しいと仮定する場合

局所的な領域では、各色信号の変化は相似形に近く、その周辺において任意の2つの色信号が持つ低周波成分はほぼ一定の比率になると仮定できる[5]。この仮定に基づけば、例えば、図3.11(b)に示すように、X_3でのGとRの信号の相関は、X_3におけるGの低周波成分($G_{LPF}3$)とRの低周波成分($R_{LPF}3$)の相関と等しいと仮定し、低周波成分の比を補間位置の信号に掛け合わせて補間信号を算出できる。具体的には、以下の式となる。

$$G_3 : R_3 = G_{LPF}3 : R_{LPF}3 \quad \text{すなわち、} \quad G_3 = R_3 \times \frac{G_{LPF}3}{R_{LPF}3} \tag{3.14}$$

文献5では、この方式をハードウェアで実現しており、低周波成分は3ラインの画素信号を記憶し、水平方向のLPF、垂直方向のLPFを用いて求めている。さらに垂直方向の低周波成分を用いる方式と、水平方向の低周波成分を用いる方式を、重み付け係数を用いて併用することによって高画質化を行っている。



(a) 変化量を利用する場合　　(b) 低周波成分を利用する場合

図 3.11　原色型での補間処理方法

3.4.2　補間方式のマイクロコントローラへの適用

3.4.1節で述べた各方式の比較検討をマイクロコントローラ上で実現できる形で具体化する。G信号とRおよびB信号ではCCDのフィルタの配列が異なるため、本節ではまず、G画素の補間方式を説明し、次にRおよびB画素の補間方法を説明する。

(1)　G画素の補間方式

3.4.1節で述べた仮定は、色信号が急激に変化する領域では成り立たない。そのため、従来はパターンマッチングによるエッジの検出を行い補間式を動的に変更したり[4]、重み付け係数を用いる[5]ことによって対応していた。

しかし、このようなエッジ検出や重み係数の計算は演算量が多く、マイクロコントローラで実現するには適さない。そこで演算量を削減するため以下の方法を取ることにした。

1)　水平方向、垂直方向の変化が共にあるしきい値以下である場合は、補間位置の上下左右の信号値(補間対象色と同一色)の平均を補間画素の信号値とする。

2)　1)以外で垂直方向の相関が強い場合(水平方向の変位＞垂直方向の変位)は、垂直方向の画素に対し、3.4.1節で説明した局所的な色の相関を用いた補間方式を適用する。

3)　1)、2)以外の場合は、水平方向の画素に対し、3.4.1節で説明した局所的な色の相関を用いた補間方式を適応する。

この方法は、自然画像においては、水平もしくは垂直方向のいずれかに相関がある

場合が多いと言われており、色信号の変化が少ない場合は演算を簡略化しても画質に及ぼす影響が少ないことに基づいている。

図3.12を用いて、以上の方法を具体的に説明する。図3.12の水平i、垂直jの位置(i,j)のG信号を求める。(i,j)における水平方向、垂直方向の変位は3.3節のcrl方式により、以下の式で求めた差分とする。図3.12の例では(i,j)の画素はRであるが、Bの場合も同様に求められる。

$$\Delta V=\{R(i,j-2)+R(i,j+2)\}-2R(i,j) \quad (3.15)$$

$$\Delta H=\{R(i-2,j)+R(i+2,j)\}-2R(i,j) \quad (3.16)$$

1)の場合($\Delta V<th$ かつ $\Delta H<th$);

$$G(i,j)=\frac{G(i-1,j)+G(i+1,j)+G(i,j-1)+G(i,j+1)}{4}$$

2)の場合($th \leqq \Delta V<\Delta H$ または $\Delta V<th \leqq \Delta H$);

$$G(i,j)=g\{G(i,j-1),G(i,j+1),R(i,j-2),R(i,j),R(i,j+2)\}$$

3)の場合($th \leqq \Delta H \leqq \Delta V$ または $\Delta H<th \leqq \Delta V$);

$$G(i,j)=g\{G(i-1,j),G(i+1,j),R(i-2,j),R(i,j),R(i+2,j)\}$$

ここでthはしきい値であり実験的に定める。また、ここで示された関数gは以下に示す(a)、(b)のどちらかとする。

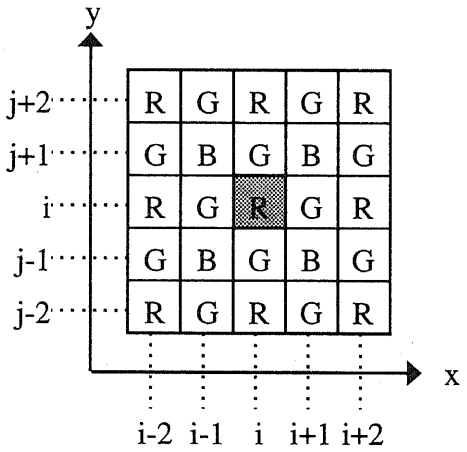


図3.12　R画素位置上でのG信号の補間

(a) 3.4.1節の(1)で述べた各色信号の変化量が等しいと仮定した場合

この場合の補間式としては(3.13)式におけるGの予測誤差$\Delta(=f(R_1,R_3,R_5))$に(3.15)式および(3.16)式を用いたものを採用した。補間式を以下に示す。

垂直方向　：$g_v(G_2,G_4,R_1,R_3,R_5)=\frac{G_2+G_4}{2}-\frac{\Delta V}{2}$

水平方向　：$g_h(G_2,G_4,R_1,R_3,R_5)=\frac{G_2+G_4}{2}-\frac{\Delta H}{2}$

(b) 3.4.1 節の(2)で述べた各色信号の低周波成分の比率が等しいと仮定した場合

(3.14)式における$G_{LPF}3$、$R_{LPF}3$を以下の式で求める。ここで、周波数成分を求めるのに使用する画素を5×5の画素範囲に限定したのは、3.4.1節の(1)で述べた変化量が等しいと仮定する方式と比較する際に、参照する画素範囲を統一するためである。

$$G_{LPF}3=\frac{1}{2}G_2+\frac{1}{2}G_4$$

$$R_{LPF}3=\frac{1}{4}R_1+\frac{1}{2}R_3+\frac{1}{4}R_5$$

補間式を以下に示す。

$$g(G_2,G_4,R_1,R_3,R_5)=R_3\times\frac{G_{LPF}3}{R_{LPF}3}$$

(2) RおよびB画素の補間方式

RおよびB画素はG画素と異なり、補間する画素の位置によって参照できる画素の位置が変化する。よって、補間する画素の位置により、補間式を変更しなければならない。そのため、G画素の補間で行った方法を適用すると演算が複雑化してしまい演算量も多くなる。そこで、参照する画素範囲を3×3に限定し、まず、その範囲内にある補間対象色の信号値と同位置での補間後のG画素信号値を用いて色信号間の関係を求める。これらの関係を用いて、補間位置のG画素の信号値から補間位置の信号値を算出する。

ここで、色信号間の関係として、

(a)差の変化が少ないと仮定した場合

(b)比の変化が少ないと仮定した場合

の2通りが考えられる。以下ではこの2つの場合について記述する。なお、以下の式において、G画素の値は入力値、または補間後の値である。

(a) 信号間の差の変化が少ないと仮定した場合

G画素の位置においては、RおよびB信号は必ず、左右または上下の隣接画素に存在する(例えば、図3.13(a))。よって、(3.17)式、(3.18)式が成り立つ。

$$R(i,j)-G(i,j)\fallingdotseq R(i-1,j)-G(i-1,j)\fallingdotseq R(i+1,j)-G(i+1,j) \quad (3.17)$$

$$B(i,j)-G(i,j)\fallingdotseq B(i,j-1)-G(i,j-1)\fallingdotseq B(i,j+1)-G(i,j+1) \quad (3.18)$$

従って以下の式を用いることで、図3.13(a)においては(i,j)におけるRおよびB信号を、図3.13(b)においてはR信号を求めることができる。

$$R(i,j)=G(i,j)+\frac{R(i-1,j)-G(i-1,j)+R(i+1,j)-G(i+1,j)}{2}$$

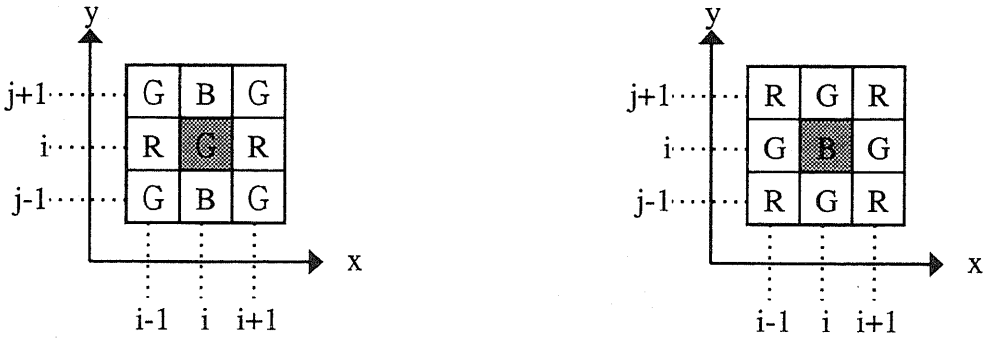
$$B(i,j)=G(i,j)+\frac{B(i,j-1)-G(i,j-1)+B(i,j+1)-G(i,j+1)}{2}$$

また、RおよびB画素の位置においては、斜め上下左右に補間したい色の信号が存在する(例えば、図3.13(b))。よって、(3.19)式が成り立つ。

$$
\begin{aligned}
R(i,j)-G(i,j) &\fallingdotseq R(i-1,j-1)-G(i-1,j-1)\\
&\fallingdotseq R(i+1,j-1)-G(i+1,j-1)\\
&\fallingdotseq R(i-1,j+1)-G(i-1,j+1)\\
&\fallingdotseq R(i+1,j+1)-G(i+1,j+1)
\end{aligned}
\tag{3.19}
$$

従って、以下の式を用いてRおよびBの(i,j)における信号を求める。

$$
\begin{aligned}
R(i,j)=G(i,j)+\frac{1}{4}\{&R(i-1,j-1)-G(i-1,j-1)+R(i+1,j-1)-G(i+1,j-1)\\
&+R(i-1,j+1)-G(i-1,j+1)+R(i+1,j+1)-G(i+1,j+1)\}
\end{aligned}
$$



(a) G画素位置でのR及びB信号の補間　　(b) B画素位置でのR信号の補間

図3.13　RおよびB信号の補間

(b) 信号間の比の変化が少ないと仮定した場合

図3.13(a)の場合は(3.20)式および(3.21)式が成り立ち、図3.13(b)の場合は(3.22)式が成り立つ。

$$R(i,j)/G(i,j)\fallingdotseq R(i-1,j)/G(i-1,j)\fallingdotseq R(i+1,j)/G(i+1,j) \tag{3.20}$$

$$B(i,j)/G(i,j)\fallingdotseq B(i-1,j)/G(i-1,j)\fallingdotseq B(i+1,j)/G(i+1,j) \tag{3.21}$$

$$
\begin{aligned}
R(i,j)/G(i,j) &\fallingdotseq R(i-1,j-1)/G(i-1,j-1)\\
&\fallingdotseq R(i+1,j-1)/G(i+1,j-1)\\
&\fallingdotseq R(i-1,j+1)/G(i-1,j+1)\\
&\fallingdotseq R(i+1,j+1)/G(i+1,j+1)
\end{aligned}
\tag{3.22}
$$

従って以下の式を用いることで、図3.13(a)においては(i,j)におけるRおよびB信号を、図3.13(b)においてはR信号を求めることができる。

$$R(i,j)=G(i,j)\times\frac{R(i-1,j)+R(i+1,j)}{G(i+1,j)+G(i-1,j)}$$

$$B(i,j)=G(i,j)\times\frac{B(i,j-1)+B(i,j+1)}{G(i,j+1)+G(i,j-1)}$$

$$R(i,j)=G(i,j)\times\frac{R(i-1,j-1)+R(i+1,j-1)+R(i-1,j+1)+R(i+1,j+1)}{G(i-1,j-1)+G(i+1,j-1)+G(i-1,j+1)+G(i+1,j+1)}$$

3.4.3 評価結果と考察

本節では、(1)で評価モデルの説明を行い、次に(2)で評価結果をまとめた後、(3)で若干の考察をする。

(1) 評価モデル

評価対象とする補間方式は、基本方式、2通りのG信号の補間方式、2通りのRおよびB信号の補間方式を評価項目とし、表3.1に示した7つのモデルを想定する。

3.3節では、各種の補間方式として最近傍法、線形補間法、3次補間法、相関を考慮した線形補間法等を、各補間方式で得られる再生画像の観点から評価した。その結果、補間したい画素位置上の色に注目し、その色に対して上下または左右の相関を求め、その結果相関が強いと判断した方向にある補間したい色から補間する方式（crl方式）が、他の方式より良いことが判明した。この方式を本節では基本方式として採用した。

表3.1 評価モデル

RおよびB信号の補間方式 ＼ G信号の補間方式	基本方式	変化量が等しいと仮定	低周波数成分の比率が等しいと仮定
基本方式	基本モデル	差モデル	比モデル
差の変化が少ないと仮定		差 - 差モデル	比 - 差モデル
比の変化が少ないと仮定		差 - 比モデル	比 - 比モデル

(a) RおよびB画素の位置におけるG信号の補間

図3.12を用いて説明する。上下左右の相関は、3.4.2節の(1)で説明した方式と同様、(3.15)式、(3.16)式で求めた差分を利用し、以下の式に基づいて線形補間を行う。

(i) $\Delta V < th$ かつ $\Delta H < th$ の場合;

$$G(i,j)=\frac{G(i-1,j)+G(i+1,j)+G(i,j-1)+G(i,j+1)}{4}$$

(ii) $th \leqq \Delta V < \Delta H$ または $\Delta V < th \leqq \Delta H$ の場合;

$$G(i,j)=\frac{G(i,j-1)+G(i,j+1)}{2}$$

(iii) $th \leqq \Delta H \leqq \Delta V$ または $\Delta H < th \leqq \Delta V$ の場合;

$$G(i,j)=\frac{G(i-1,j)+G(i+1,j)}{2}$$

ここでthはしきい値であり、今回の評価では256階調の信号に対しthを値を3とした。

(b) G画素の位置におけるRおよびB信号の補間

図3.13(a)を用いて説明する。RおよびB画素は3×3の画素範囲にある信号を用い線形補間を行う。

よって、G画素の位置(i,j)においてはR信号に関しては左右の画素を用いて線形補間し、B信号に関しては上下の画素を用いる。

$$R(i,j)=\frac{R(i-1,j)+R(i+1,j)}{2}$$

$$B(i,j)=\frac{B(i,j-1)+B(i,j+1)}{2}$$

(c) RおよびB画素の位置におけるBおよびR信号の補間

図3.13(b)を用いて説明する。この場合も3×3の画素範囲にある信号を用い線形補間を行う。図3.13(b)の場合は、B画素の位置でのR信号を求める例であるが、R画素位置でB信号を求める場合も同様に求めることができる。

$$R(i,j)=\frac{R(i-1,j-1)+R(i+1,j-1)+R(i-1,j+1)+R(i+1,j+1)}{4}$$

(2) 評価結果

評価は、色の再現性、組込み型マイクロコントローラにおけるサイクル数と(3.12)式で定義したS/N比の関係について行った。

CCDから得られる画像データに基づいて補間方式を比較する場合、各画素に全色成分のデータがそろっていないため、画素ごとの各色成分の基準値を参照できず、比較および評価が難しいという問題がある。そこで3.3.3節と同様に、評価を行うにあたって、各画素に対し全ての色成分がそろっている画像を利用し、原色フィルタの形式に従って、画素毎にRGB成分のいずれか1つを取り出すことによりCCDから得られる画像データを作成した。

(a) 色の再現性

評価では、すべての色成分がそろっている画像(以下、原画像と呼ぶ)と各補間方法で得られた画像(以下、再現画像と呼ぶ)を用い、その違いを定量的に比較することによって各方式の特徴を調べた。

各モデルにおける色の再現性を評価した結果を示す。評価には、3.3.3節の(2)で用いたCZPを用いた。CZPの画像サイズは640×480画素(VGAサイズ)で、水平垂直とも端がナイキスト周波数になるように作成している。また、各色成分の値は同値である。

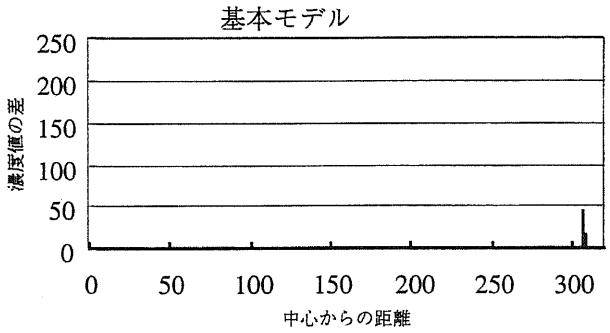
CZPを用いたのは、周波数の違いによる色再現性の違いを評価するためである。評価は、周波数の変化に伴って、再現画像の各色の濃度値と原画像のその画素における各色の濃度値の差がどのように変化するかによって行った。評価を行った領域は、画

素の配列、相関を考慮し、図3.6に示した水平方向と斜め方向の2領域を選択した。図3.14、図3.15、図3.16に評価結果を示す。縦軸は濃度値の差、横軸は円の中心からの距離を表す。中心から離れるにしたがって、周波数が高くなるため、周波数の増加と濃度値の差の関係が分かる。

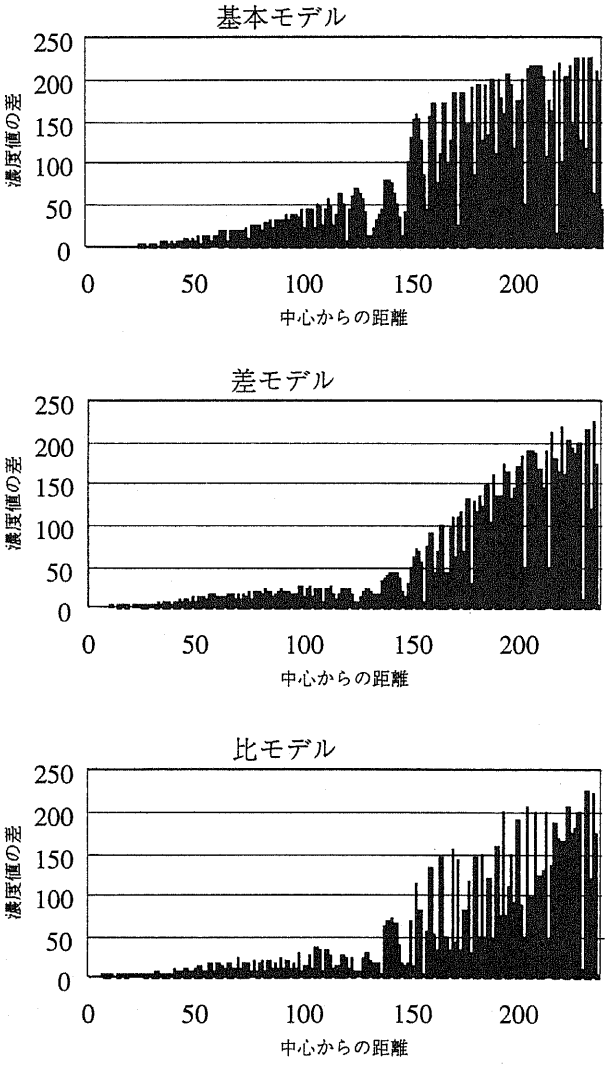
図3.14に3通りの方式（基本モデル，差モデル，比モデル）を用いた場合の再現画像と原画像のG信号の濃度値の差を示す。図3.14(a)は水平方向の、図3.14(b)は斜め方向の濃度値の差を示したものである。図3.14(a)では、基本モデルにおける水平方向の濃度値の差を示した。他の差モデルおよび比モデルでも同様の結果が得られ、水平方向については3通りの方式とも色の再現性に大きな差が見られないことがわかった。また、斜め方向については、比モデルがもっとも良い再現性が得られていることがわかった。

図3.15に水平方向のRおよびB信号の濃度値の差を示す。RおよびB信号については、行によって画素配列が異なる。図3.15(a)は、補間する画素の色がサンプリングされていない行の濃度値の差を、図3.15(b)は、補間する画素の色がサンプリングされている行の濃度値の差をそれぞれ示したものである。図3.15では、基本モデルと差-差モデルにおける水平方向の再現性を示した。基本モデルおよび差-差モデル以外のモデルについては差-差モデルと同様の結果が得られた。これにより、水平方向に関しては、基本モデル以外は良い再現性が得られていることがわかった。

図3.16に斜め方向のRおよびB信号の濃度値の差を示す。図3.15と同様、図3.16(a)は、補間する画素の色がサンプリングされていない線上、図3.16(b)は、補間する画素の色がサンプリングされている線上の色再現性を示したものである。図3.16(a)では、基本モデルと差-差モデルにおける斜め方向の再現性を示した。基本モデルおよび差-差モデル以外のモデルについては差-差モデルと同様の結果が得られた。図3.16(a)および(b)より、基本モデルと比較すると、信号間の差が等しいと仮定した場合も信号間の比が等しいと仮定した場合も良い色再現性が得られていることがわかった。また図3.14を用いた評価から判明した、Gの色再現性が良い、Gに関して比モデルを採用した比-差モデル、比-比モデルの色再現性も良いこともわかった。

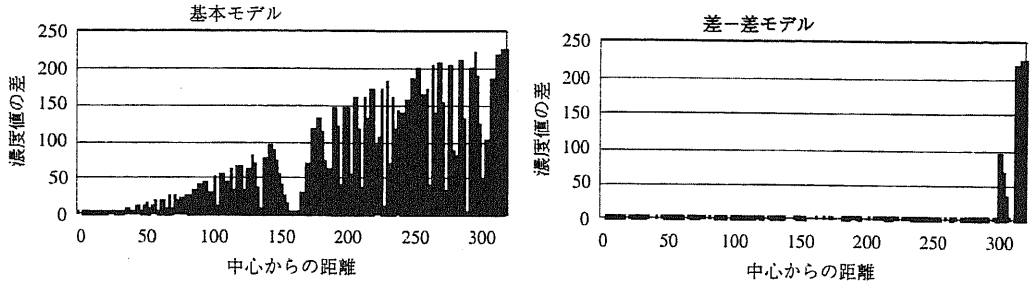


(a) 水平方向の濃度値の差

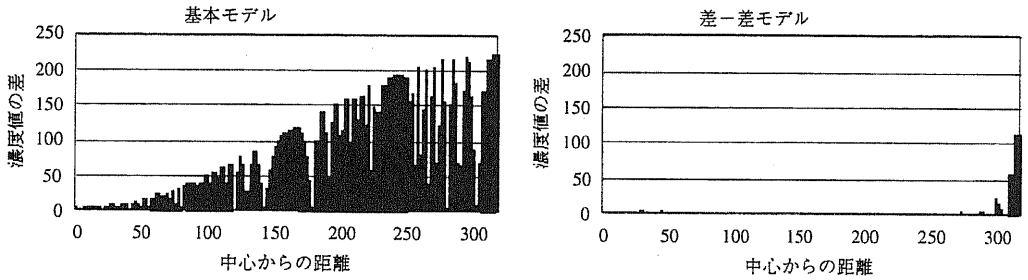


(b) 斜め方向の濃度値の差

図 3.14　G 信号の濃度値の差

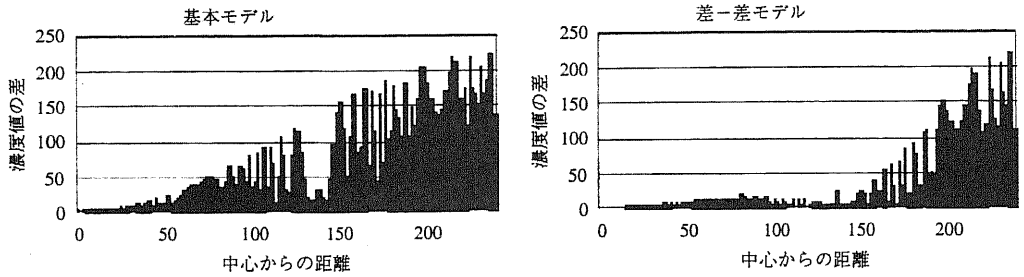


(a)　補間する画素の色がサンプリングされていない行の場合

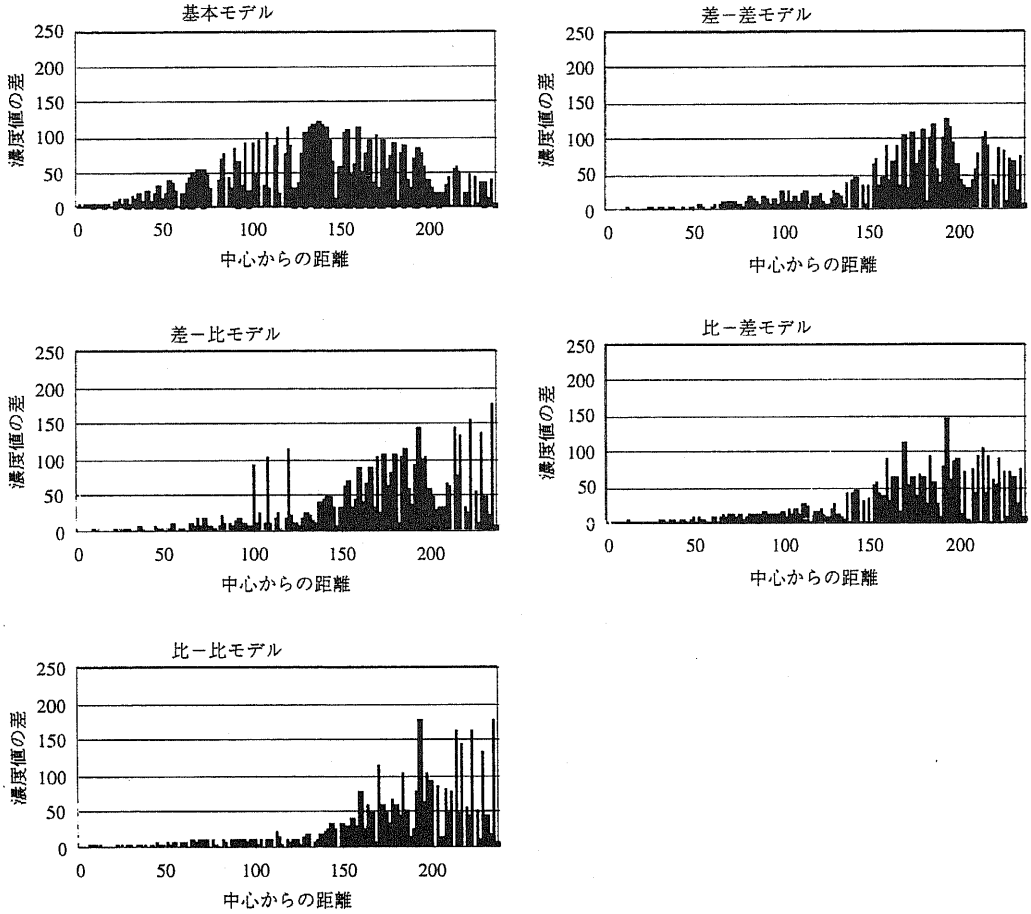


(b)　補間する画素の色がサンプリングされている行の場合

図 3.15　R および B 信号の水平方向の濃度値の差



(a)　補間する画素の色がサンプリングされていない行の場合



(b)　補間する画素の色がサンプリングされている行の場合

図 3.16　RおよびB信号の斜め方向の濃度値の差

(b) S/N比と演算量

各補間方式の演算量を見積るため、実際に組込み型マイクロコントローラを用いて、各補間方式をアセンブラ言語で実現し演算量を求めた。また、画質を客観的に比較するため、S/N比を用いた．評価に用いたマイクロコントローラの動作周波数は66MHzである。

演算量の考察には、実行サイクル数を正確に把握できるシミュレータを用いた。

評価に用いた画像は、CZPと1.2.4節に示したSCIDの5種類の自然画像を1280×960画素に加工したものを使用した。

図3.17にSCIDの評価結果を，図3.18にCZPの評価結果を示す。図の縦軸はS/N比を示し、値が大きいほど原画像に近く良い画質を再現していることを表す。また、図の横軸は画素補間処理に要するサイクル数を示す。サイクル数に動作周波数(66MHz)の逆数をかけることで実行時間が求められるため、サイクル数が大きくなるほど（つまり、右方向にいくに従い)、画素補間に要する処理時間が大きくなることを表す。図3.17における評価結果の値は5種類の画像のS/N比の平均値である。図3.17および図3.18に示したサイクル数は、画像全体に対して画像補間を行うのに必要なサイクル数である。またS/N値は、R、G、B全ての信号を考慮して全信号値の誤差の自乗平均を採った。

図3.17および図3.18より、基本モデル、差モデル、比モデル、差-差モデル、比-差モデル、差-比モデル、比-比モデルの順に演算量が多くなっていることがわかる。また図3.17から、差-差モデルが演算量に対するS/N比が最も良く、次に比-差モデルが良いことがわかる。しかし，図3.18では図3.17と異なり、比-差モデルが演算量に対するS/N比が最も良く、次に比-比モデルが良いことがわかった。

また、RおよびBの補間方式に相関を考慮した場合の方式を適用すると、S/N比が大きく向上し、特に色信号間の関係として差の変化量が少ないと仮定した方式のS/N比が良いことがわかった。

S/N比において、SCIDとCZPとは異なった傾向が見られている。この理由は次節で考察する。

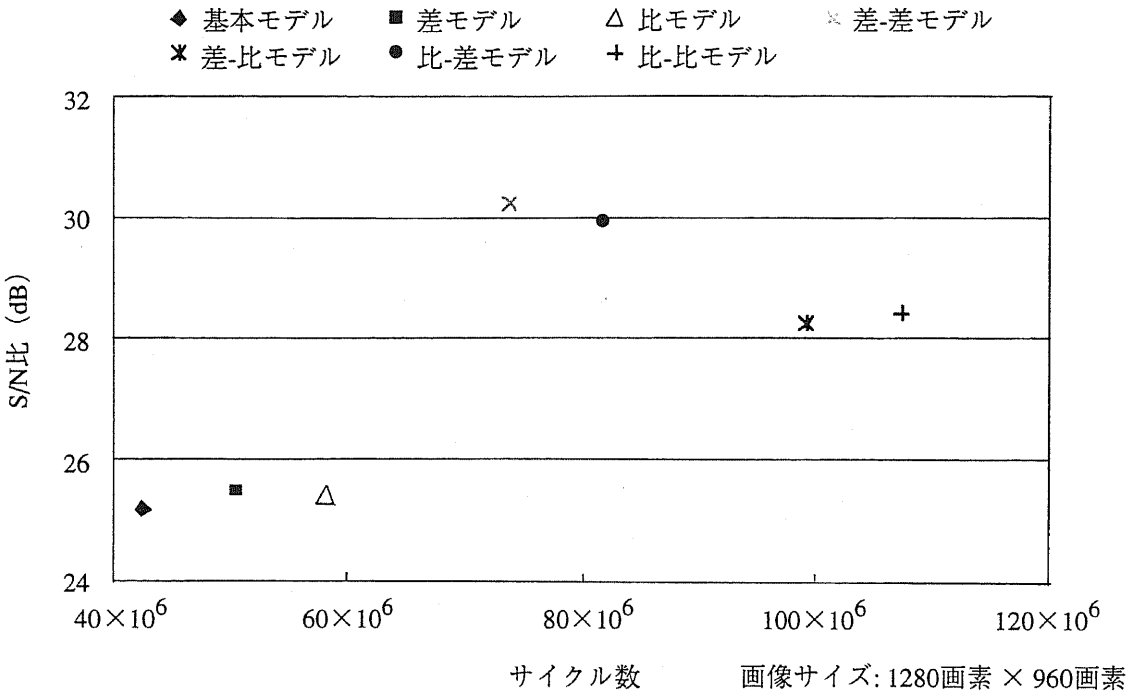


図 3.17　S/N 比と演算量（SCID を使用した場合）

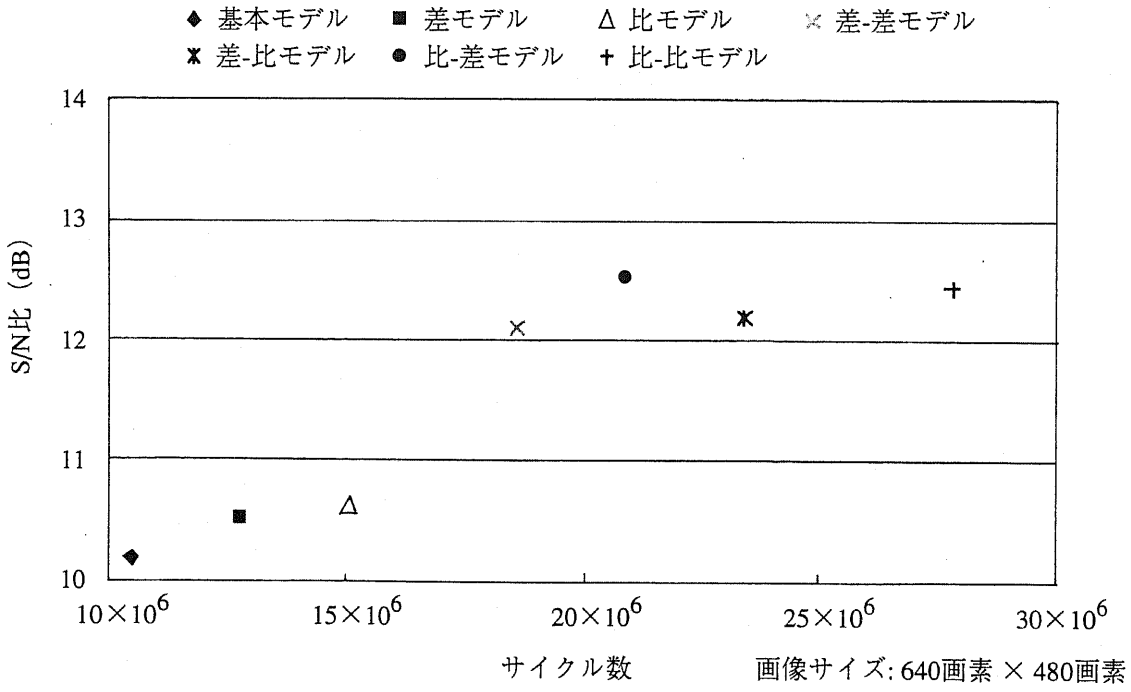


図 3.18　S/N 比と演算量（CZP を使用した場合）

(3) 考察

S/N比に注目すると、CZPではG信号の補間においては低周波成分の比率が等しいと仮定した場合の方がよく、RおよびB信号の補間においては信号間の比の変化が少ないと仮定した場合の方式の方がよい。しかし，自然画であるSCIDでは、G信号の補間においては、変化量が等しいと仮定した場合の方がよく、RおよびB信号の補間においては信号間の差の変化が少ないと仮定した場合の方式の方がよい。

このような違いが出た原因を考察するため、大きな違いがある画素に注目し、その画素周辺の画像の特徴を調べてみた。その結果を図3.19に示す．図3.19は、CZPにおいて、低周波成分の比率が等しいと仮定した場合の再現性が特に良い画素周辺の画像信号の概略図である。縦軸は信号値を表す．図3.19(a)においてX_3におけるGの信号値は、低周波成分の比較が等しいと仮定した場合

$$G_3=\frac{R_3\cdot G_{LPF}3}{R_{LPF}3}=\frac{10\cdot 60}{130}\fallingdotseq 5$$

信号間の変位が少ないと仮定した場合

$$G_3=G_3'-\frac{\Delta}{2}=60-120\fallingdotseq 0$$

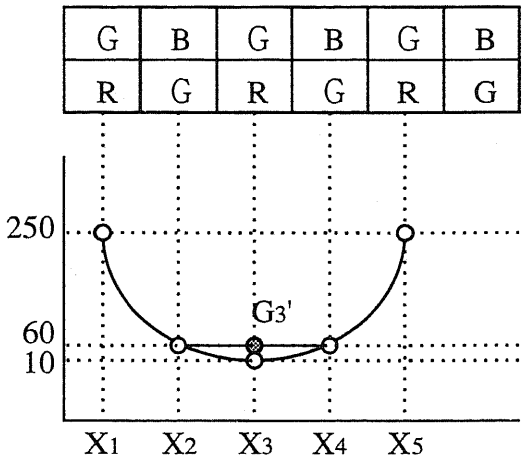
となる．図3.19(b)においても同様に各々

$$G_3=\frac{R_3\cdot G_{LPF}3}{R_{LPF}3}=\frac{10\cdot 220}{35}\fallingdotseq 60$$

$$G_3=G_3'-\frac{\Delta}{2}=220-25\fallingdotseq 195$$

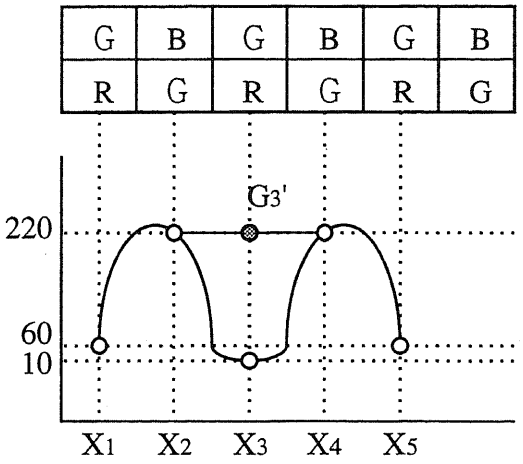
となる。また，$G_3(=R_3)$は、原画像におけるGの信号値である。CZPの性質上、図3.19(a)や(b)のように5×5の範囲において左右対称な画素パターンが多く、低周波成分を用いる補間方式に適合するため、低周波成分の比率が等しいと仮定した場合の方が良い結果が得られたと考えられる。RおよびBにも同様のことが言える。しかし、自然画においては、このような特徴をもった画像面積が少なく，CZPとは異なった結果が得られたと考えられる。

以上から、画像の特徴を調べ、変化量が等しいと仮定する方式，低周波成分の比率が等しいと仮定する方式を併用すればさらに良いS/N比が得られる可能性があると考えられる。



(注)同じ画素における色信号は同値

(a)ケース 1



(注)同じ画素における色信号は同値

(b)ケース 2

図 3.19　低周波成分の再現性が良い場合の画像の特徴

3.5　まとめ

本章では、デジタル・スチル・カメラ専用のCCDイメージセンサの画素構造として多く用いられているベイヤー型原色方式に適用する補間方式について考察している。画素補間処理では、補間対象色と同一色の画素だけを用いて補間処理を行うのが一般的である。また近年の研究では、特に色信号の高解像度化に関する試みがなされている。例えば、色の相関を算出するため、局所的な色の相関を利用し画素信号の補間処理を行う方式や、水平方向や2次元のLPFを用いる方式などが報告されている。そこで、先ず補間対象色と同一色の画素だけを用いて処理を行う画素補間方式について検討し、次に色の相関を用いて画素補間方式の改良を検討した。

先ず、同一色を用いた画素補間方式のうち代表的な8つの補間方式について検討を加えた。そして、画素補間により得られる再生画像の特徴と組込み用途の32ビット・マイクロコントローラを用いて、補間を実現するための演算量と、その結果得られる画質について考察した。

検討の結果、各補間方式の善し悪しは、解像度が高いかどうかよりも、高周波領域における偽色の発生が重要な要因になることがわかった。そこで、偽色が少なく、色の再現性がよいという観点では、相関を考慮した線形補間法が良いことがわかった。

次に画質と演算量を定量的に比較するために、実際にマイクロコントローラの演算量とS/N比の関係を調べた。演算量に関しては、VGAサイズの画像の補間処理を、動作周波数66MHzのマイクロコントローラで実行したところ何れの方式も0.2秒程度で実現できた。その結果、S/N比の観点においても線形補間方式が、演算量の割には良い画質が得られ、ディジタル・スチル・カメラのような民生用途では適していると考え

られる。

次に、この結果を更に押し進め、色の相関を用いた画素補間方式をベイヤー型原色方式のCCDに適応したときの再生画像の特徴を、解像度、色の再現性の観点から調べた。

色の再現性に関してCZPを使って解析した結果、G信号については各色成分の周波数成分の比率が等しいと仮定した場合の再現性が良く、RおよびB信号については色信号間の関係として比の変化が少ないと仮定した場合の再現性が良いことがわかった。

また演算量とS/N比の関係について、CZPとSCID中の5種類の自然画を使って解析した結果、G信号について各色成分の変化量が等しいと仮定し、RおよびB信号については色信号間の関係として差の変化が少ないと仮定した場合の補間方式が演算量に対するS/N比が良いことがわかった。

第3章の参考文献

1) B. E. Bayer; Color imaging array, U.S. Patent, 3971065

2) 小沢; "CCM補間処理による単板カラーカメラの色モアレ抑圧", テレビ雑誌, 46, 2, pp.210-216, (1992)

3) 米山, 山本, 谷添, 佐々木, 岡山; "高画質単板デジタルカメラ方式", 1994年テレビ年次大, 8-5, pp. 119-120, (1994)

4) 清水, J.E.Adams, C.M.Smith, 接待, 宮野, 大塚; "新開発の圧縮及び補間処理を用いたデジタルカメラ", SPSTJ Symposia on Fine Imaging, D-4, pp. 69-72, (1995)

5) 小沢; "原色型単板カラーカメラの偽色抑圧処理", 映情学技報,22,22,pp.1-6, (1998)

6) T. Sakamoto, C. Nakanishi and T. Hase; "Software Pixel Interpolation for Digital Still Camera for a 32-bit MCU", IEEE Transactions on Consumer Electronics, Vol. 43, No. 3, pp. 1342-1352, (1998)

7) S. S. Rifman and D.M. McKinnon; "Evaluation of Digital Correction Techniques for ERTS Images - Final Report", Report 20634-6003-TU-00. TRW Systems, Redondo Brach, Calif., (1974)

8) R. G. Keys; "Cubic Convolution Interpolation for Digital Image Processing", IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-29, pp. 1153-1160, (1981)

9) K.W.Simon; "Distal Image Reconstruction and Resampling for Geometric Manipulation", Proc. IEEE Symp. on Machine Processing of Remotely Sensed Data, pp. 3A-1-3A-11, (1975)

10) D. P. Mitchell and N. N. Arun; "Reconstruction Filters in Computer Graphics", Computer Graphics, (SIGGRAPH ユ88 Proceedings), Vol22, no.4, pp. 221-228, (1988)

11) W. F. Schreiber and E. T. Donald; "Transformation Between Continuous and Discrete Representations of Image: A Perceptual Approach", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. PAMI-7, no. 2, pp. 178-186, (1985)

12) H. S. Hou and C. A. Harry; "Cubic Splines for Image Interpolation and Digital Filtering", IEEE Trans. Acoust., Speech, Signal Process., vol. ASSP-26, pp. 508-517, (1987)

13) Pratt, K. William; "Digital Imgae Processing", Second Edition, Wiley-Interscience, New York, (1991)

14) 中西, 坂本, 長谷; "組込み形マイクロプロセッサに適した画素補間方式", 映像情報メディア学会誌, Vol. 53, No. 4, pp. 141-149, (1999)

15) 小寺, 中, 金森; "色の相関を利用した単色画像からのフルカラー画像の表示方式", 昭和63画電学会大, 16, 20, pp. 83-86, (1988)

第4章　静止画像処理JPEG

4.1　はじめに

デジタル・スチル・カメラでは、従来JPEG方式を用いて画像データを圧縮伸張するために専用の集積回路が使用されてきた。しかし、マイクロコントローラの高性能化に伴い、JPEG方式を用いた画像データの圧縮伸張をソフトウェアで行うことが実用レベルになってきた。マイクロコントローラを用いることで製品の低価格化、低消費電力化や高付加機能化の実現ができ、これらの面での利点もあることから、ソフトウェアによる画像の圧縮伸張の実用化に対する要求が強い。

JPEG方式は、ISO/IEC 10918-1でその処理方式は定義されている[1]。JPEG方式では、ベースライン・システム、拡張システム、および可逆符号化システムの3つの方式が定められている。その中でも離散コサイン変換（DCT : Discrete Cosine Transform）[2]を用いたベースライン・システムが広く普及しており、JPEG方式といえば、ベースライン・システムを指すと考えてもよい。ベースライン・システムで処理の中核をなすDCT演算やハフマン符号化については、従来から高速アルゴリズムが研究されている[2]。しかし、マイクロコントローラを用いてベースライン・システムによる画像の圧縮・伸長を高速にソフトウェアで実現するためには、DCT演算やハフマン符号化といった部分単独のアルゴリズムの高速化だけでは不十分で、それらを跨って工夫する必要がある。

本章では、第2章で述べたマイクロコントローラを用いた画像・音響処理技術の開発手法を、画像データの圧縮伸張処理に適用する場合について、具体的な高速化手法を示し、その性能を評価し、有効性を検証する。

4.2　ベースライン・システムによる画像圧伸処理

図4.1にJPEGベースライン・システムによる画像データの圧縮・伸長のフローを示す。画像データの圧縮では、8ビットのピクセル画像は8×8ピクセルのブロックに分割され、DCT演算、量子化の後、ハフマン符号化され圧縮されたデータとなる。一方画像データの伸長は、圧縮の逆で、ハフマン復号、逆量子化、逆DCT演算という処理手順で復号される。

本節では、ベースライン・システムをソフトウェアで高速に実現するための課題を後に示すために、DCT演算、ベースライン・システムを用いた画像データの圧縮および伸長の処理を詳細に示す[1][3]。

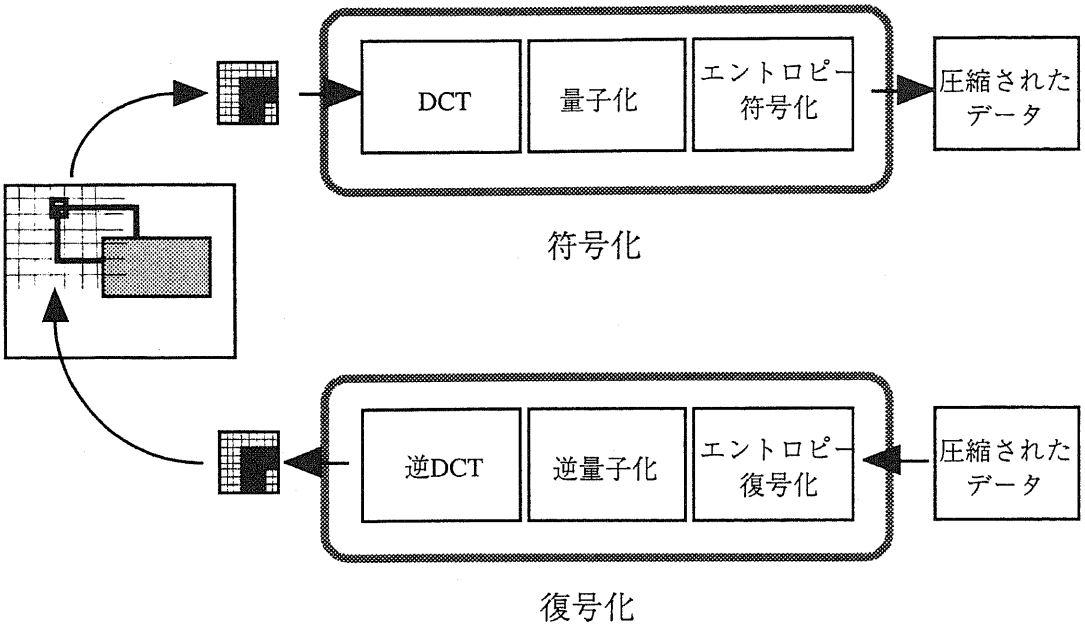


図 4.1 ベースライン・システムによる画像データの圧縮・伸長

4.2.1 DCT 演算

DCT演算部では、8×8ピクセルのブロックに対して2次元ＤＣＴ演算を実施する。一般にN点1次元DCT演算と逆DCT演算は、以下のように定義される。

$$F(u)=\sqrt{\frac{2}{N}}\,C(u)\sum_{x=0}^{N-1} f(x)\cos\left\{\frac{\pi\, u(2x+1)}{2N}\right\} \tag{4.1}$$

$$f(x)=\sqrt{\frac{2}{N}}\,C(u)\sum_{u=0}^{N-1} C(u)F(u)\cos\left\{\frac{\pi\, u(2x+1)}{2N}\right\} \tag{4.2}$$

ただし、

f(x) ：各N点の入力
F(u)：N点の変換値

$$C(i)=\begin{cases}\dfrac{1}{\sqrt{2}} & (i=0)\\ 1 & (i\neq 0)\end{cases} \tag{4.3}$$

2次元のDCT演算は、入力画像をN×N画像（ベースライン・システムでは8×8画素）のブロックに分割する。この各ブロック・データ f (x,y) に対して2次元DCT演算を実行し、N×Nの変換値 F (u, v) を得る。N×N 2次元DCT演算と逆DCT演算は、以下のように定義される。

$$F(u,v)=\frac{2}{N}C(u)C(v)\sum_{x=0}^{N-1}\sum_{y=0}^{N-1}f(x,y)\cos\left\{\frac{\pi u(2x+1)}{2N}\right\}\cos\left\{\frac{\pi u(2y+1)}{2N}\right\} \quad (4.4)$$

$$f(x,y)=\frac{2}{N}\sum_{u=0}^{N-1}\sum_{v=0}^{N-1}C(u)C(v)F(u,v)\cos\left\{\frac{\pi u(2x+1)}{2N}\right\}\cos\left\{\frac{\pi u(2y+1)}{2N}\right\} \quad (4.5)$$

2次元ＤＣＴ演算は、1次元ＤＣＴ演算に分解でき、計算量も少なくなる[3]。これは、(4.4)式を以下のように変形することから分かる。

$$F(u,v)=\sqrt{\frac{2}{N}}C(v)\sum_{y=0}^{N-1}\left[\sqrt{\frac{2}{N}}C(u)\sum_{x=0}^{N-1}f(x,y)\cos\left\{\frac{\pi u(2x+1)}{2N}\right\}\right]\cdot\cos\left\{\frac{\pi u(2y+1)}{2N}\right\} \quad (4.6)$$

(4.6)式の内側の級数は(4.1)式に示したＮ点１次元ＤＣＴ演算を表し、x方向の１次元ＤＣＴ演算に相当するのに対し、外側の級数は同様にｙ方向の１次元ＤＣＴ演算に相当する。従って、ベースライン・システムで用いられる８点の２次元ＤＣＴ演算は、以下に示す手順で実現できる。

(1) 縦方向あるいは横方向に、１列あるいは１行に並んだ８個の画素を１つの計算単位として積和演算を実施し、その結果を１列あるいは１行に書き込む１次元ＤＣＴ演算を行う

(2) (1)の演算を先ず片方向に８回繰り返す

(3) さらに、(2)の結果を使って他方向に同様の演算を実施する。

このように１次元ＤＣＴ演算を繰り返すことで、２次元ＤＣＴ演算を実現できる。図4.2に１次元DCT演算を用いた２次元DCT演算の実現フローを示す。

なお、画像の伸長時に使用する逆DCT演算に関しても同様の議論が出来ることは上式から明らかで、DCT演算と同様の手法で実現できる。

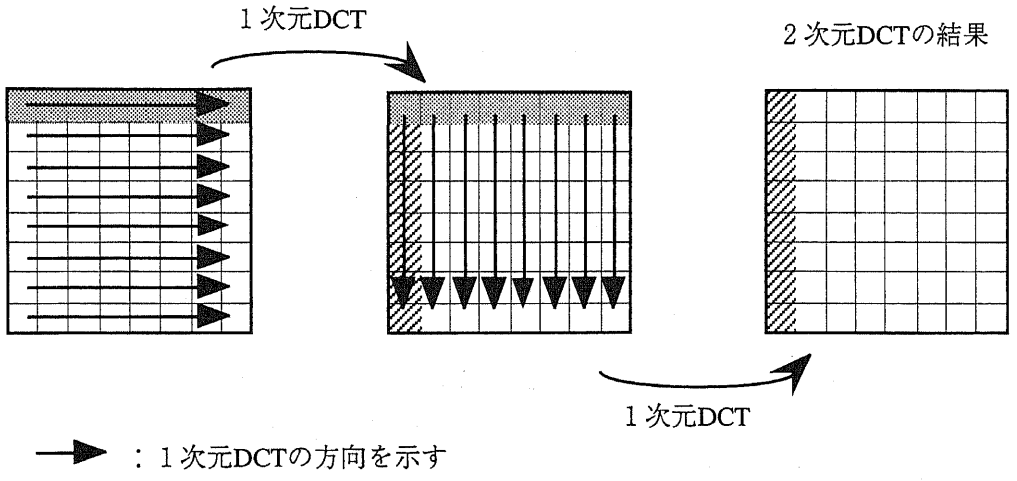


図4.2　１次元DCT演算を用いた２次元DCT演算の実現

4.2.2 画像データの圧縮

本節では、ベースライン・システムを用いた画像データの圧縮処理について述べる。図4.3に圧縮処理の詳細を示す。画像データの圧縮処理は以下の手順で実施される。

(1) 輝度（Y）、色差（Cb, Cr）の3つのコンポーネントから構成される8ビットのピクセル画像を8×8ピクセルのブロック（JPEGではminimum coded unit (MCU)と呼ばれる）に分割する。

(2) 8×8ピクセルに分割された各ブロックに対して、4.2.1節で述べたDCT演算を実施する。演算結果として64個のDCT係数が得られ、特に図4.3に示した1番目のDCT係数をDC係数、残りの63個をAC係数と呼ぶ。

(3) 64個のDCT係数に対して、量子化を実施する。量子化は、人間の視覚の高周波成分には鈍感であるという特性を利用し、視覚感度の低い周波数成分の情報量を減らし、画像データ量を減らす効果を持つ。

(4) 量子化されたDCT係数をエントロピー符号化する。エントロピー符号化の方式はハフマン符号化方式を用いる。AC係数は図4.4に示した順に従ってジグザグ走査することで1次元に並び直し、連続する零の係数の長さを示すラン長と零以外の係数の値の組を用いて符号化する。

図4.4に示したジグザグ走査の順番は簡単な計算式で表現できない。そのため、ジグザグ走査をソフトウェアで実現する際は、ジグザグ走査を実現したインデックス変換用の配列（以下では配列zig_zag[64]と呼ぶ）を予め用意し、配列の内容を参照して得た値をDCT係数を格納した配列（以下では配列coeff[64]と呼ぶ）のインデックスとしてDCT係数を読み出すことで実現する。つまり、i番目のAC係数を得るために配列coeffのzig_zag[i]番目の要素 coeff[zig_zag[i]] を読み出すことでジグザグ走査を実現する。

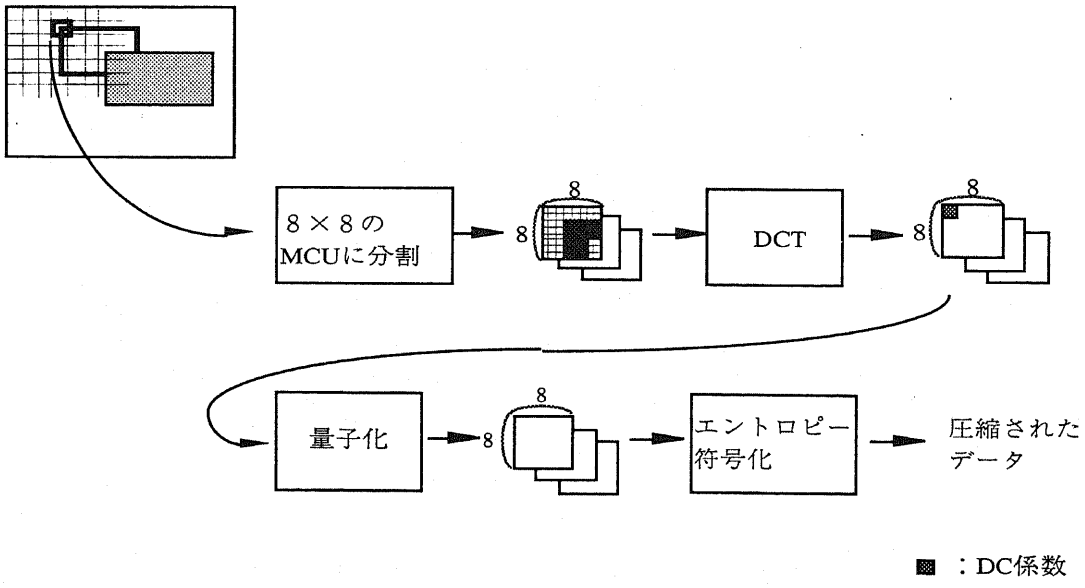


図4.3 圧縮処理

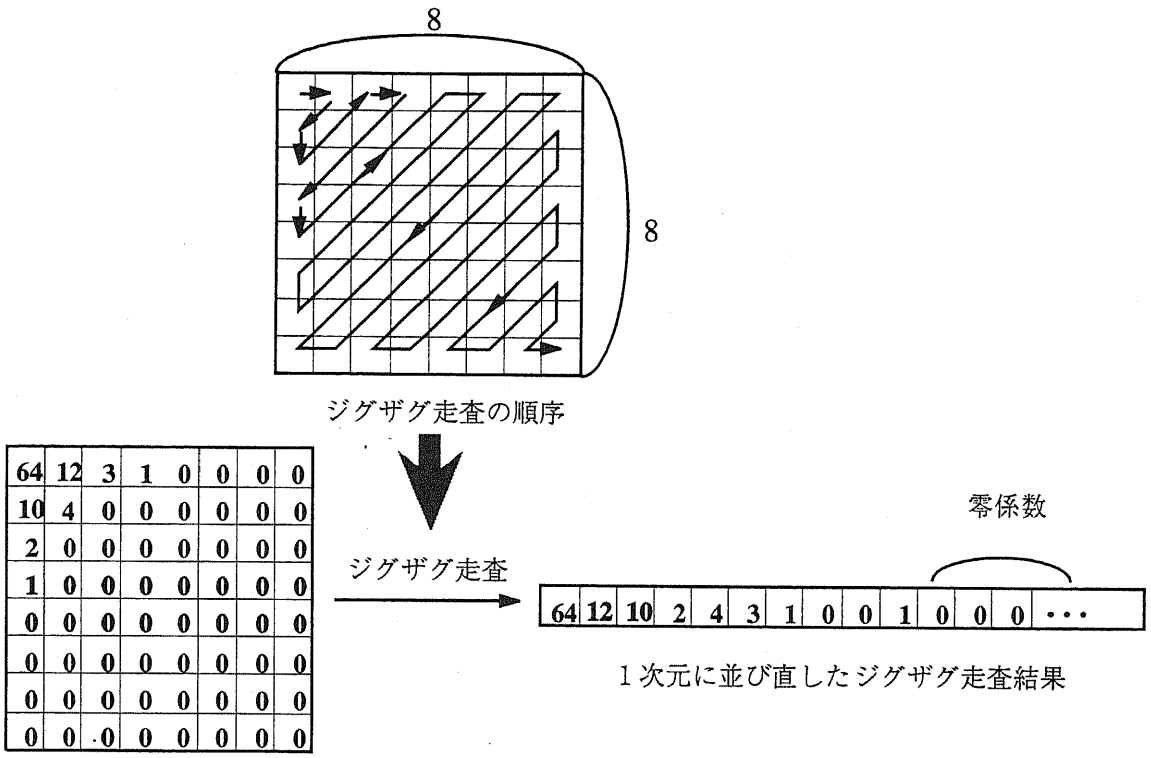


図4.4　ジグザグ走査

4.2.3　画像データの伸長

伸長処理は4.2節に述べたように、圧縮処理の逆で、ハフマン復号、逆量子化、逆DCT演算という処理手順で復号される。従って、ここではベースライン・システムを用いた画像データの伸長処理をソフトウェアで実現する場合に特徴的な部分についてのみ詳細を述べる。

圧縮の際の量子化により、量子化されたDCT係数の多くが零になる。特に色差に対応する量子化されたDCT係数ではその傾向が顕著である。逆DCT演算をソフトウェアで実現する場合、この傾向を利用して逆DCT演算を簡単化することが多い。つまり、4.2.1節で述べた8点1次元逆DCT演算の式において、7つの量子化されたDCT係数 $F(i)$ $(i=1,2,\ldots,7)$ が零ならば、8回の積和演算が、1回の乗算で済ますことができることに着眼する。そして、通常の8回の積和演算をする前に、連続する7つの量子化されたDCT係数が零であるかを確認し、もしそうなら1回の乗算で1次元逆DCT演算を実現するわけである。

4.3 高速化する上での課題

ベースライン・システムをソフトウェアで実現する場合、DCT演算やハフマン符号化の部分を単独で高速化しても、まだ高速化の余地がある。特に組込み用途のマイクロコントローラとDRAMを組み合わせて、実用的な処理時間で画像の圧縮や伸長を実現するには、DCT演算やハフマン符号化の部分を単独で高速化するだけでは不十分である。実際にJPEG方式による画像の圧縮伸張プログラムをマイクロコントローラ用に開発して速度性能を計測し、内部の処理時間を分析することで、処理性能に大きな影響を与える処理としては以下のものがあることがわかった。

(1) JPEG の処理単位である MCU のデータに対するロードおよびストア
(2) ハフマン符号化時のジグザグ走査
(3) ハフマン符号化時の零の AC 係数の長さを示すラン長の計算
(4) 逆 DCT 演算時に、連続する 7 つの DCT 係数が 0 であることの判定
(5) DCT 演算

これらの処理内容を分析してみると、(1)から(4)はメモリ・アクセスが主として問題になる処理で、(5)はメモリ・アクセスではなく、乗算を含めた演算が多い処理である。つまり、(1)から(4)はメモリ使用法の観点からメモリ・アクセスを考慮した工夫[4]が、(5)はアルゴリズムの観点からの工夫[5]が必要である。以下では、この 2 つの観点から課題を整理し、解決手法を示し、その有効性を評価する。

4.4 メモリ・アクセスを考慮した高速化手法

本節では、4.3節で述べた(1) ～ (4) の課題の詳細を分析し、その解決手法を示し、そして従来手法に対する提案手法の効果について評価する。なお、本節で述べる演算の計算量については、デジタル・スチル・カメラ等で用いられることの多い、VGA サイズ（表色系 YCbCr）でサンプリング比 Y:Cb:Cr=4:2:2 の画像を想定して算出した。算出結果は表4.1にまとめた。またこの時の MCU の数は、YCbCr の 3 つのコンポーネントがそれぞれ

Y : 640 × 480 / 64 = 4800 MCU
Cb : (640 × 480 / 2) / 64 = 2400 MCU
Cr : (640 × 480 / 2) / 64 = 2400 MCU

であり、合計 9600 個である。

4.4.1 従来手法の課題と演算見積もり

(1) MCU のデータに対するロードおよびストア

画像データの圧縮時には、4.2.2 節で述べたように各 MCU に対して以下に述べるロードおよびストアは必要となる。

(a) 8ビットのピクセル画像を8×8ピクセルのMCUに分割する際、ピクセル画像のMCUへのストア

(b) 8×8ピクセルに分割されたMCUに対してDCT演算を実施する際、8ビットのピクセル画像のロードと、DCT演算結果のストア

(c) 量子化時のDCT係数のロードと、量子化結果のストア

(d) エントロピー符号化の際、量子化されたDCT係数のロード

つまり、1つのMCUに対して3回のロードおよび3回のストアが必要になる。また、圧縮処理の逆の処理を行う画像の伸長時も、圧縮時と全く同じ回数のロードおよびストアが必要となる。それぞれのロードおよびストアの回数は、MCUの数（9600）と1MCU内のデータ数（64）の積、つまり614400回である。

(2) ハフマン符号化時のジグザグ走査

4.2.2節で述べたように、ジグザグ走査をソフトウェアで実現する際は、

i番目のAC係数 = coeff[zig_zag[i]]

というようにジグザグ走査を実現したインデックス変換用の配列zig_zagを予め用意し、テーブルの内容を参照して得た値をDCT係数を格納した配列coeffのインデックスとしてDCT係数をロードすることで実現する。

そのためには、メモリの参照が2回と配列coeffのアドレス計算は少なくとも必要である。特にAC係数をロードするためのアドレス計算を行うには、配列zig_zagからのロード結果を使用するため、ロードの際にキャッシュ・ミスやメモリ・ウェイトが入ると、ロードの完了を待つ必要がある。最悪時には、マイクロコントローラによっては、AC係数のロードに10サイクル以上かかる場合もあり、ジグザグ走査の速度性能に与える影響は大きい。さらに、この直後にAC係数の零判定を実施するため、AC係数のロードの際にも同様の状況が発生する可能性がある。従って、ジグザグ走査のための2回のロードに必要となるサイクル数は、そのまま全体の処理時間に影響する。なお、ジグザグ走査はAC係数に必要なため、その回数は、MCUの数（9600）と1 MCU内のAC係数の数（63）の積、つまり604800回である。

(3) ハフマン符号化時の零のAC係数の長さを示すラン長の計算

零のAC係数の長さを示すラン長の計算は、4.2.2節に述べたジグザグ走査でAC係数をロードした直後に、ロードした値が零かどうかで判定し、ラン長を計算する。その際、値が零かどうかで、分岐が発生する。この零判定が、実行される回数も多く（ジグザグ走査と同じ604800回）、値に応じて分岐が発生するため、回数を減らす工夫が必要である。

(4) 逆DCT演算時の連続する7つのDCT係数が0であることの判定

4.2.2節で述べたように、逆DCT演算をソフトウェアで実現する場合、7つの量子化されたDCT係数が零ならば逆DCT演算を簡単化する手法を採用することが多い。そ

のためには、7個のDCT係数のロード、それらのビットごとの論理和をとり（6回）、そのあと零判定をする必要がある。この処理は1次元逆DCT演算当たり8回で、2次元目の逆DCT演算にも行うと、1MCU当たり16回必要で、合計でMCUの数（9600）と1 MCU内のデータ数（64）の積、つまり153600回である。

表4.1　演算量見積り

処理内容	演算数の見積り
MCUのデータに対するロードおよびストア	9600×64×(3×LD+3×ST)
ジグザグ走査	9600×63×(2×LD+2×ADD)
ラン・レングスの計算	9600×63×(LD+CMP)
連続する7つのDCT係数の零判定	9600×16×(7×LD+6×OR)

注）ST / LD / ADD / CMP / OR は、それぞれストア / ロード / 加算 / 零比較 / ビットごとの論理和にかかる実行サイクル数を表す

4.4.2　高速化手法

組込み用途のマイクロコントローラとDRAMを組み合わせたシステムを考えた場合、4.4.1に述べた課題を解決するためには、メモリとのロードおよびストアを減らす、あるいはメモリに連続して配置されたデータをアクセスしながら処理を行うことが重要となってくる。上述した問題点を解決するために、適用できる可能性のある高速化手法には次のようなものが挙げられる。

(a)　レジスタ上のデータを効率的に利用することで、冗長なロードおよびストアを削減する

(b)　ハーフワードのデータを32ビットにパックして2回の演算を1度で済ます

(c)　ハーフワードのデータ2個分のロードをワードサイズのロード命令で実施する

(d)　1ワードの零判定で、ハーフワードのデータ2個が共に零であることを判定する

以下では、これらの一般的な高速化手法を具体的にどのように効率的に適用するかを述べる。

(1) MCUのデータに対するロードおよびストアの削減

4.4.1節の(1)で述べたように、圧縮および伸長の両方で、JPEGの基本処理単位であるMCUのデータに対するロードおよびストアが何度か発生する。例えば圧縮では、画像データのMCUへの分割→DCT演算→量子化→エントロピー符号化とアルゴリズム通り処理を個別に実行すると、それらの処理間でロードとストアが発生する。これらの処理を適切な範囲でまとめて処理をすれば中間的なロードとストアを削減できる。具体的には、画像データからMCUの1列あるいは1行に相当する画像データをレジスタに取込み1次元目のDCT演算を実行し、次に2次元目のDCT演算の1列あるいは1行が完了した時に、1列あるいは1行単位で量子化を行い、MCUへ格納することで、始めの3つの処理をまとめて実現できる。この場合、表3.1の9600×64×3回のロードと9600×64×3回のストアが、9600×64回のロードと9600×64回のストア、つまりロードおよびストアを3分の1に削減することができる。

(2) ハフマン符号化時のジグザグ走査の削減

4.4.1節の(2)で述べたように、ジグザグ走査が効率的に実現できない大きな理由は、ジグザグ走査は2回のメモリの参照の結果得られたAC係数を零判定する処理を行うが、その間に実行可能な他の処理がないため、ロードの際にキャッシュミスやメモリウェイトが入っても、その間に実行すべき処理がなくその完了を待つ以外ないことにある。これを緩和する方法として、ジグザグ走査時にインデックス変換用の配列参照をなくす方法が有効である。つまり、図4.5に示すようにDCT演算あるいは量子化の結果をMCUに格納する際にジグザグ走査後の位置に格納する。このことにより、エントロピー符号化時にはインデックス変換用の配列を参照することなくMCUを連続的にアクセスしながら符号化できる。これにより、ジグザグ走査時の課題である2回のメモリ参照のうち1回のメモリ参照は不要となり、課題の半分は解消できる。それと同時に、連続的にメモリをアクセスしながら符号化できるという利点も得られる。これはDRAMへのアクセスという面から効率的なメモリ使用法である。さらに、この連続的にメモリをアクセスしながら処理を進められるという利点は後述の手法でさらに有効活用できる。

(3) パックされた2個のデータのロード

4.2.1節で述べたように、2次元DCT演算は、1次元DCT演算の処理に分解でき、1次元目のDCT演算を縦あるいは横のどちらかに実行し、その結果を同じ行に戻し、2次元目のDCT演算を1次元目とは反対の方向に実行する。JPEGで用いる2次元DCT演算の場合、1次元目のDCT演算結果のデータ長はハーフワードで精度的に問題ない。従って、2個のデータのロードを同時にワードデータとしてロードすることが考えられる。しかし、ハーフワードの入力データを2つ同時に入力するためには、DCT演算を1次元目も2次元目も同じ横方向に実施する必要がある。そのためには1次元目のDCT演算を横方向に行い、その結果を格納する作業領域を使用し、1次元目のDCT演

算の結果を縦方向に格納する。これにより、2次元目のDCT演算も横方向に実行できる。特に、この方法は逆DCT演算の時に更に効果的である。それは、後述する連続する7つの量子化されたDCT係数が零である判定が効率的に1次元目も2次元目もできるからである。

この手法によりDCT演算のためのデータのロードは半分に削減することができる。

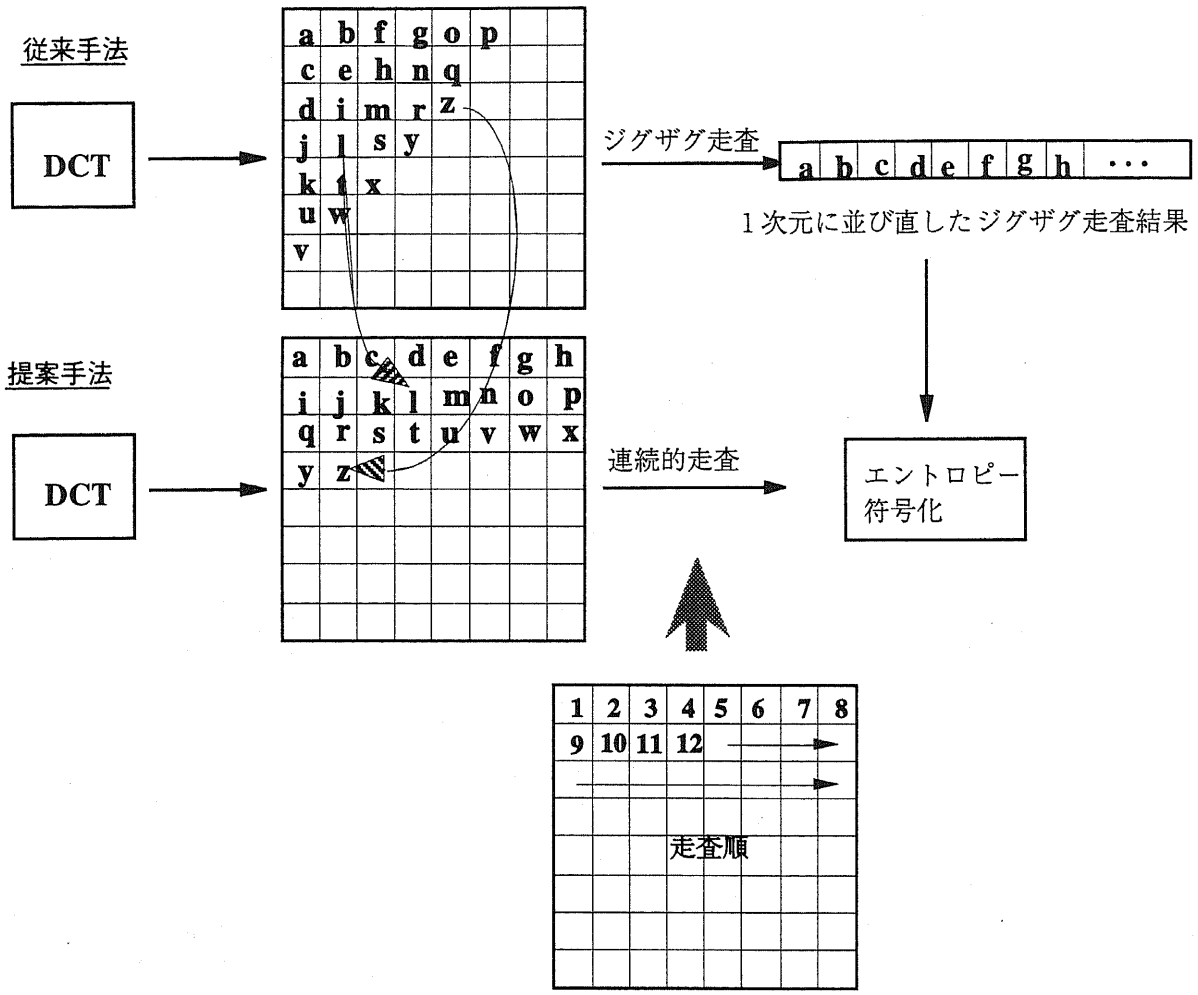


図4.5　ジグザグ・スキャンの削減方法

(4)　パックされた2個のデータの零判定

32ビット・マイクロコントローラには、32ビット長のレジスタ値が零であることを判定する命令がある。一方で、DCT係数はハーフワード・16ビットであるため、2つのDCT係数を1つのレジスタにパックしてロードして、レジスタ値が零であることを判定することで、連続した2つのDCT係数が零であることがわかる。連続したDCT係数の零判定の頻度が高い場合には、本手法が有効になると考えられる。以下では、こ

の手法の有効な適用方法を述べる。

(a) ハフマン符号化時の零判定

図4.6に、標準的な量子化テーブルを用いて画像圧縮した際のDCT係数の値の分布を示す。評価データとしては、1.2.4節に示したテレビジョン学会(当時)のテレビジョン・システム評価用デジタル標準画像5チャート(画素サイズ752×480のオリジナルデータを、サンプリング比Y:Cb:Cr=4:2:2のデータに変換)を使用した。図からわかるように91%のDCT係数が零であり、しかも15個以上零が連続する場合が78%もあることがわかる。つまり、DCT係数の零判定を行った場合、ほとんどの場合が零と判定されることがわかる。従って、32ビットのデータに対して零判定の命令を使用して、2つのDCT係数が零であることを1度で判定することは効果的だと判断できる。

この手法は、本節の(3)で述べたエントロピー符号化時にMCUを連続的にアクセスできる手法と合わせて初めて効率的に活用できる。

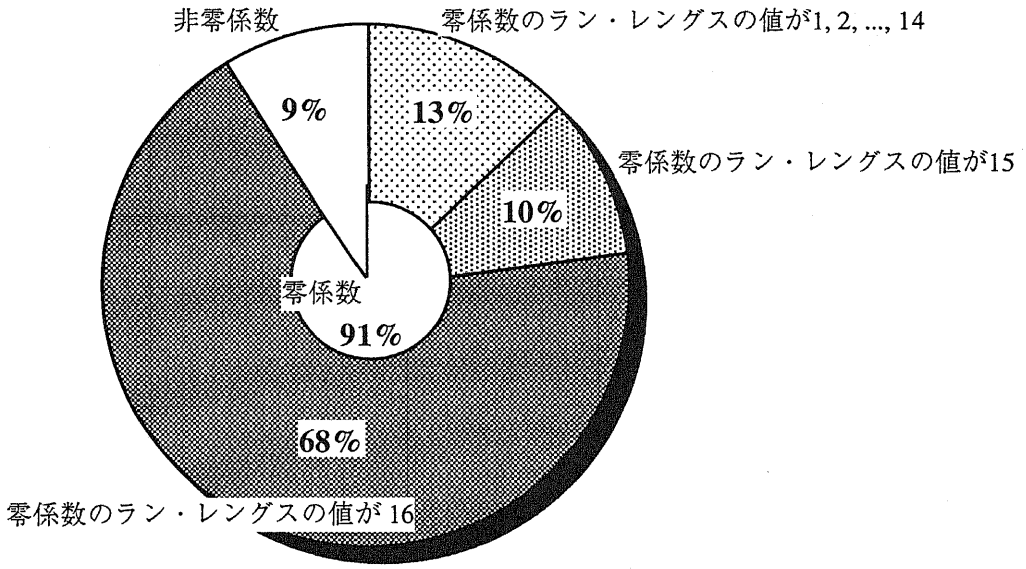


図4.6 DCT係数値の分布

(b) 逆DCT演算時の連続する量子化されたDCT係数の零判定

4.4.1節の(4)で述べたように、逆DCT演算をソフトウェアで実現する場合、7つの量子化されたDCT係数が零であることを判定するため、7個のDCT係数のロード、それらのビットごとの論理和をとる必要がある。本節の(3)で述べた1次元目も2次元目も同じ横方向に実施する手法と組み合わすと、連続した2つのDCT係数を1命令でロードすることで、DCT係数のロードが4回、ビットごとの論理和が3回に削減することができる。

4.4.3 評価結果と考察

(1) 演算量の比較

従来手法を用いた場合と4.4.2節で提案した手法を用いた場合の演算量の比較を表4.2に示す。表からわかるように、提案した手法を用いることで演算量がおおよそ半分になることがわかる。

表4.2 演算量の比較

処理内容	従来手法による演算数の見積り	提案手法による演算数の見積り
MCUのデータに対するロードおよびストア	9600×64×(3×LD+3×ST)	9600×64×(LD+ST)
ジグザグ走査	9600×63×(2×LD+2×ADD)	9600×63×(LD+ADD)
ラン・レングスの計算	9600×63×(LD+CMP)	9600×32×(LD+CMP)
連続する7つのDCT係数の零判定	9600×16×(7×LD+6×OR)	9600×16×(4×LD+3×OR)

注) ST / LD / ADD / CMP / OR は、それぞれストア / ロード / 加算 / 零比較 / OR演算にかかる実行サイクル数を表す

(2) 速度性能の評価

図4.7に4.4.2節で提案した手法を用いた場合の速度性能結果を示す。評価には1.2.4節に示したテレビジョン・システム評価用デジタル標準画像5チャート（画素サイズ752×480のオリジナルデータを、サンプリング比Y:Cb:Cr=4:2:2のデータに変換）を用いてY/Cb/CrからJPEGファイル生成までの時間を測定した。その結果、0.46秒から0.55秒で圧縮ができることがわかる。この結果をVGAサイズ換算すると0.5秒以下で圧縮ができることになる。これの結果を同じプロセッサに移植したIndependent JPEG Groupのプログラム(Version 6a)と比較すると約2倍の処理性能がでていることがわかる。

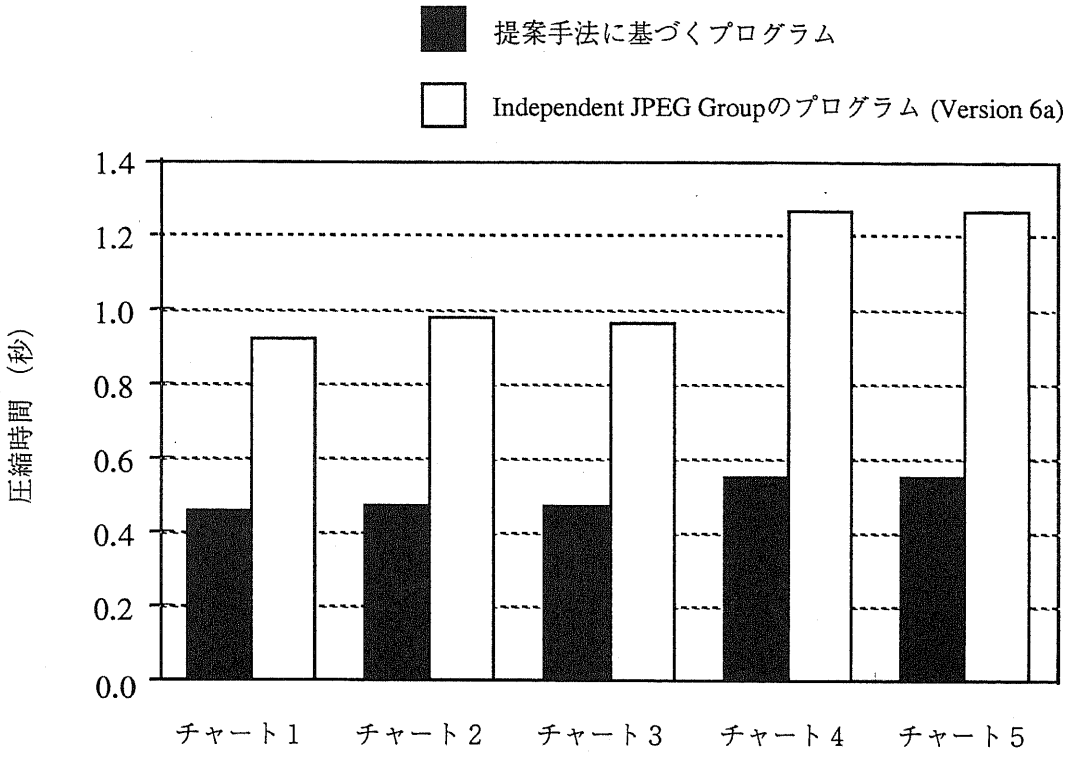

提案手法に基づくプログラム
Independent JPEG Groupのプログラム (Version 6a)
圧縮時間（秒）
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
チャート1
チャート2
チャート3
チャート4
チャート5
テレビジョン・システム評価用デジタル標準画像
画像サイズ: 752画素 × 480画素
画像フォーマット: YCbCr
サンプリング比: Y:Cb:Cr = 4: 2::2

図 4.7　速度性能比較

4.5 アルゴリズムを考慮した高速化手法

4.2節に述べたように、ベースライン・システムによる画像データの圧縮・伸長処理は、大きく分けハフマン符号化とDCT演算の2つの処理から構成される。ハフマン符号化の主な処理は、テーブル参照や参照した値による条件分岐処理で、この処理を通常のマイクロコントローラで高速にするのは難しい。一方、DCT演算は分岐処理のない演算処理で、種々の高速化アルゴリズムも提案され、マイクロコントローラでの処理方法を工夫することで高速化が期待できる。そこで本節では、マイクロコントローラの搭載しているハードウェア資源からの制約を考慮に入れてDCT演算の高速アルゴリズムを処理速度に関して評価する。4.5.1節では代表的なDCT演算の高速化アルゴリズムを先ず示す。次に4.5.2節では、これらの高速化アルゴリズムを従来からある並列実行機能を持たないマイクロコントローラを用いて実現する場合の処理速度を評価する。特に、乗算を加減算で置き換える高速化手法の有効性に着目して評価する。さらに4.5.3節では、4.5.2節の制約条件に命令の並列実行機能が加わった場合の処理速度を評価する。そして4.5.4節では4.5.2節と4.5.3節の議論をもとに、命令の並列実行機能を持ったマイクロコントローラを用いた時のベースライン・システムによる圧縮処理の処理性能を評価する。

4.5.1 DCT演算の高速化アルゴリズム

多くのDCT演算の高速化アルゴリズムが提案されているが[3]、ここでは、本研究で評価に使用した4種類の代表的な高速化アルゴリズムについて概要を説明する。

Chen[6]のアルゴリズムは最初に提案された高速化アルゴリズムで、8点1次元DCT演算に必要な乗算の数は16回と多いものの、規則性が高いのと、積和命令を使用するのに適していることから本論文での評価対象とした。

Hou[7]は再帰アルゴリズムを提案した。計算の最終段以外は規則性の高いアルゴリズムで、8点DCT演算に必要な演算は乗算が12回、加減算が29回である。他の高速化アルゴリズム、例えばVetterli[8]やLee[9]の方法と同じ数の演算回数が必要だが、規則性の高いことと乗算に必要な定数の数が少ないことがプログラムとして実現する際に有利になるため本論文での評価対象とした。評価時には、関数コールのオーバヘッドを避けるために再帰でないプログラムを作成した。

Loeffler[10]のアルゴリズムは、8点DCT演算に必要な乗算が11回（加減算はHou他の方法と同様の29回）という特徴がある。

Arai[11]のアルゴリズムは、DCT演算結果に係数（simple scalingsと呼んでいる）を乗じることで、DCT演算の演算を簡単にするという特徴をもつ。そのため、必要な演算としては乗算が5回、加減算が29回である。DCT演算の結果のうち6つの演算結果に係数を乗じるため乗算回数はLoefflerの方法と変わらないが、JPEGではDCT演算の後に量子化を行うため、JPEGに実装する際はこの係数を量子化係数に前もって掛けておくことで、この6回の乗算を省くことが可能になる。一方で、Araiの方法は演算式に

規則性や対称性がないため、演算精度に問題がある。

4.5.2 マイクロコントローラ上で実現した高速アルゴリズムの評価

(1) 高速化アルゴリズムをマイクロコントローラ上に実現する際の課題

組み込み用マイクロコントローラを用いて高速化アルゴリズムを実現する際には、マイクロコントローラの搭載するハードウェア資源からの制約を受ける。そのためDCT演算を実現する場合は、搭載するハードウェア資源と合った高速化アルゴリズムを選択する必要がある。

8点DCT演算を実現する際に問題になるハードウェア資源を下に挙げる。

(a) 乗算器の有無
(b) 積和やアキュムレータ操作に関連した命令の有無
(c) 汎用レジスタの数
(d) メモリのアクセス速度

例えば、乗算器を持たないマイクロコントローラでは、乗算の実行に必要なサイクル数は加減算と比べて10倍以上大きい。そのため、加減算数を増やしても乗算数をなるだけ減らすアルゴリズムの方が有利になる。一方で、乗算器を搭載している最近のマイクロコントローラでは、乗算が加減算と同じサイクル数で実行できる。そのため、乗算の数を減らすことよりも、8点とDCT演算のポイント数が少ないことを活かしてレジスタを有効活用することが重要になってくる。レジスタを有効に活用し、メモリの使用を最小限にするためには、計算の途中に保持しなければいけないデータの数、乗算に必要な定数の数等の方が、乗算の数を減らすことよりも処理速度という面では重要になってくる。

(2) 評価に用いるマイクロコントローラのハードウェア資源

今回の評価に際して前提としたマイクロコントローラは、RISCアーキテクチャを採用した32ビット・マイクロコントローラで、表4.3に示すようなハードウェア資源を搭載しているものとした。この前提は、簡単な信号処理機能を取り込もうとしている最近のRISCタイプのマイクロコントローラでは妥当なものである。16ビット×16ビットの乗算や積和を加減算と同様の1サイクルで実行可能と想定した。アキュムレータに対する丸めや転送命令も1サイクルでの実行可能とした。汎用レジスタは32ビット幅で16本だが、1本はスタックポインタとして使用するため実質15本をコーディングに使用可能とした。またメモリのアクセス速度に関しては、キャッシュを搭載しているため、常にキャッシュにヒットし、1サイクルでのデータが読み出せるという前提で検討した。

表4.3　前提としたマイクロコントローラのハードウェア資源

乗算器	32ビット×16ビットあるいは16ビット×16ビット
乗算器に関連した操作命令	積和命令、アキュムレータ値の丸め命令、アキュムレータと汎用レジスタ間のデータ転送命令 積和命令は1サイクルで実行可能
汎用レジスタ	32ビット幅で16本。ただし、1本はスタックポインタとして使用するため実質15本。
キャッシュ	あり
メモリのアクセス速度	常にキャッシュ・ヒットし、1サイクルで読み出し可能と仮定

(3)　高速化アルゴリズムの評価結果

(2)に述べたハードウェア資源から受ける制約を考慮して、4.5.1節に述べた4つの代表的な高速化アルゴリズムをアセンブリ言語で記述して、処理速度を評価した。なお、ベースライン・システムで使用する2次元DCT演算は、4.2.1節で述べたように、1次元DCT演算に分解できるため、評価は8点の1次元DCT演算で行った。8点の1次元DCT演算を実現するのに必要な各高速化アルゴリズムのステップ数を表4.4に示す。

表4.4　1次元の8点DCT演算に必要なステップ数

高速化アルゴリズム	ステップ数	アルゴリズムの特徴
Chen	76	積和演算を使用するのに適している。 乗算回数16回。
Hou	99	演算式の規則性が高い。 乗算12回、加減算29回。
Loeffler	117	乗算11回、加減算29回。
Arai	78	乗算5回、加減算29回。ただし、演算結果のうち6つに係数を乗じる必要あり。

表4.4から分かるように、乗算器を搭載した積和命令を持つマイクロコントローラ向けのDCT演算は、Chenのアルゴリズムを用いると効率よく実現できる。しかもChenのアルゴリズムはデータに対する乗算が1回ですむために演算精度の劣化の問題がない。Chenのアルゴリズムが他のアルゴリズムより効率よく実現できるのは、アルゴリズムに表れる4回の積和、つまり7回の加算と乗算が、積和命令を使うことで4命令

で実行できるためである。つまり、見かけ上、加算がなくなったような効果がある。反対に、Chenのアルゴリズム以外のものでよく使用する式、a+bとa-bを計算するには、aの値を1度他のレジスタに転送する必要が有り、見かけ上2回の加減算の実現に3命令が必要になる。そのため、Chenのアルゴリズム以外のものは効率的に実現できない。

4.5.3 並列実行を用いた場合のDCT演算実現方法

本章では、4.5.2節に述べたハードウェア資源から受ける制約条件のもとで、並列実行という新しい演算方法が加わった場合に、最適なDCT演算の高速化アルゴリズムの選択に変化が生じるかを評価する。

(1) 並列実行の定義

コンピュータやワークステーションで使用されるプロセッサでは一般的な並列実行技術が、組込み用途のマイクロコントローラでも最近取り入れられている[12]。DCT演算の高速化の一手法として、このような命令の並列実行機能を持ったマイクロコントローラを採用し、その特長を活かしたDCT演算を実現するという方法が考えられる。

今回の評価に際して前提とした並列実行は、全ての命令の組合せが並列実行可能ではなく、組込み用途という意味でハードウェア量を小さくするという観点から、2命令の並列実行が可能で、乗算/アキュムレータ操作/ロード/ストア/分岐といった命令は1ステップの命令実行で1回しか実行できない（つまり、2命令の並列実行を想定しているため、片側での実行のみ可能）といった制限付きを想定した。具体的には下のような並列実行を仮定した。

(a) 算術演算/論理演算/転送/シフト/nopは2つ並列に実行可能

(b) 乗算/アキュムレータ操作/ロード/ストア/分岐は片側での実行のみ可能で、定義された並列に実行可能な他の命令と合わせて並列実行する必要がある

(c) 並列実行の際に同一レジスタを双方で使用する時は、並列にレジスタ値のコピーが両方の命令に渡される

(2) 並列実行を用いた演算の課題

(1)で述べたような並列実行を仮定した場合、4.5.2節で評価した高速化アルゴリズムに関連した下に述べる2点の演算に影響がでる。

(a) a+bとa-bの計算

高速化アルゴリズムでよく使用する、a+bとa-bの計算には、並列実行がない場合は、4.5.2節の(3)で述べたように、aの値を1度他のレジスタに転送する必要が有り、3命令が必要になる。ところが、a+bとa-bの計算を並列に実行すると、(1)に述べたように、並列実行の際にレジスタ値のコピーが両方の命令に渡される。従ってa+bとa-bの計算は、

b+a‖a-b（ここで‖は左右の命令を並列実行することを示すものとする）

と1ステップで実行できる。つまり、並列実行の導入で、a+bとa-bの計算が3ステップから1ステップで実現可能になる。

(b) 積和演算

加減算とは異なり、乗算や積和命令およびアキュムレータ操作命令は1ステップで1回しか実行できない。従って、これらの命令を効率的に使用するためには、同時に算術演算や転送等の命令を実行する必要がある。

(3) 並列実行を用いた場合の高速化アルゴリズムの評価結果

(2)に述べた並列実行を用いた場合の演算への影響を考慮して、4.5.2節と同様に、4.5.1節に述べた高速化アルゴリズムをアセンブリ言語で記述して、高速化アルゴリズムの処理速度を評価した。8点の1次元DCT演算を実現するのに必要な各高速化アルゴリズムのステップ数を表4.5に示す。

表4.5　1次元の8点DCT演算に必要なステップ数（並列実行を用いた場合）

高速化アルゴリズム	ステップ数	アルゴリズムの特徴
Chen	53	積和演算を使用するのに適している。 乗算回数16回。
Hou	65	演算式の規則性が高い。 乗算12回、加減算29回。
Loeffler	77	乗算11回、加減算29回。
Arai	48	乗算5回、加減算29回。ただし、演算結果のうち6つに係数を乗じる必要あり。

表4.5から分かるように、並列実行機能を持ったマイクロコントローラ向けのDCT演算は、AraiのアルゴリズムとChenのアルゴリズムを用いると効率よく実現できる。4.5.2節の結果と異なり、AraiのアルゴリズムがChenのアルゴリズムよりも効率が良くなった理由は、(2)で述べたようにa+bとa-bの計算が効率よく実現できることが挙げられる。更にAraiのアルゴリズムは、乗算に必要な定数の数が4個と少なく、必要とするレジスタ数も少なという特徴もある。実際に1次元DCT演算を2次元DCT演算に組み込む際には、Araiのアルゴリズムのこの特徴が生きてくる。つまり、4.2.1節に述べたように、2次元DCT演算は縦横の双方向に合計16回の1次元DCT演算を実施する。Araiのアルゴリズムの特徴より、乗算に必要な定数を予めレジスタに設定し、16回の1次元DCT演算でその値を共有することが可能となる。従って、表4.5のAraiのアルゴリズムの実現に必要となるステップ数は48から実質的には45になる。

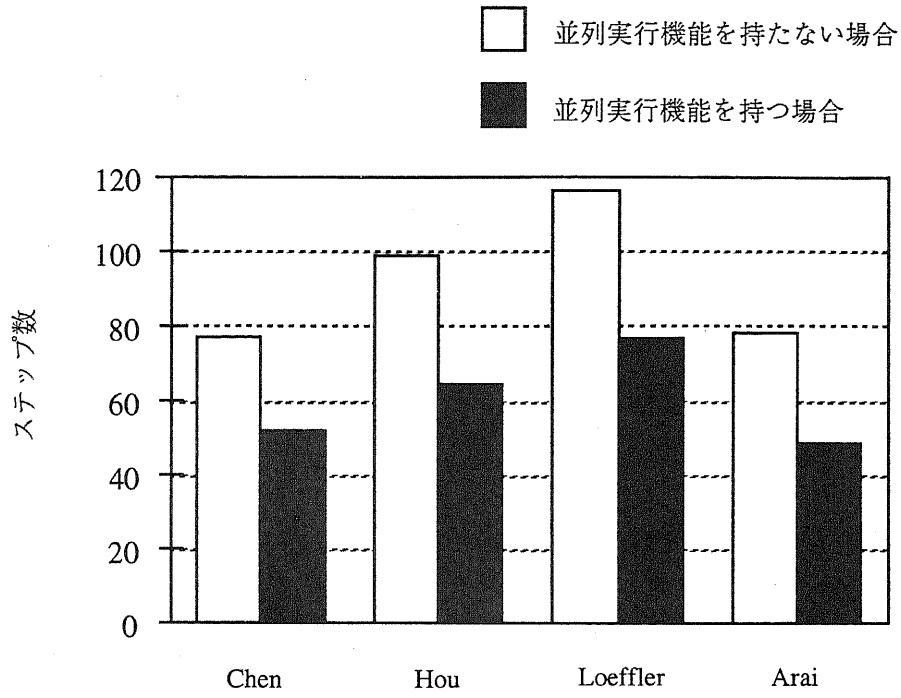


図 4.8　DCT 演算に対する並列実行の効果

また、表 4.4 と表 4.5 の比較結果を図 4.8 に示す。並列実行がなく、かつ乗算器を搭載したマイクロコントローラ向けのDCT演算は、Chenのアルゴリズムを用いて効率よく実現できた。これに並列実行機能が加わると、Chenのアルゴリズムのステップ数の減少率が一番低い。これはChenのアルゴリズムでは、積和命令の数がボトルネック化して、並列実行の効果を十分活かしきれない一方、Chenのアルゴリズム以外のものは、a + b と a - b の計算が効率よく実現できたことが原因である。その中でも Arai のアルゴリズムは、ステップ数の減少率が一番高い。

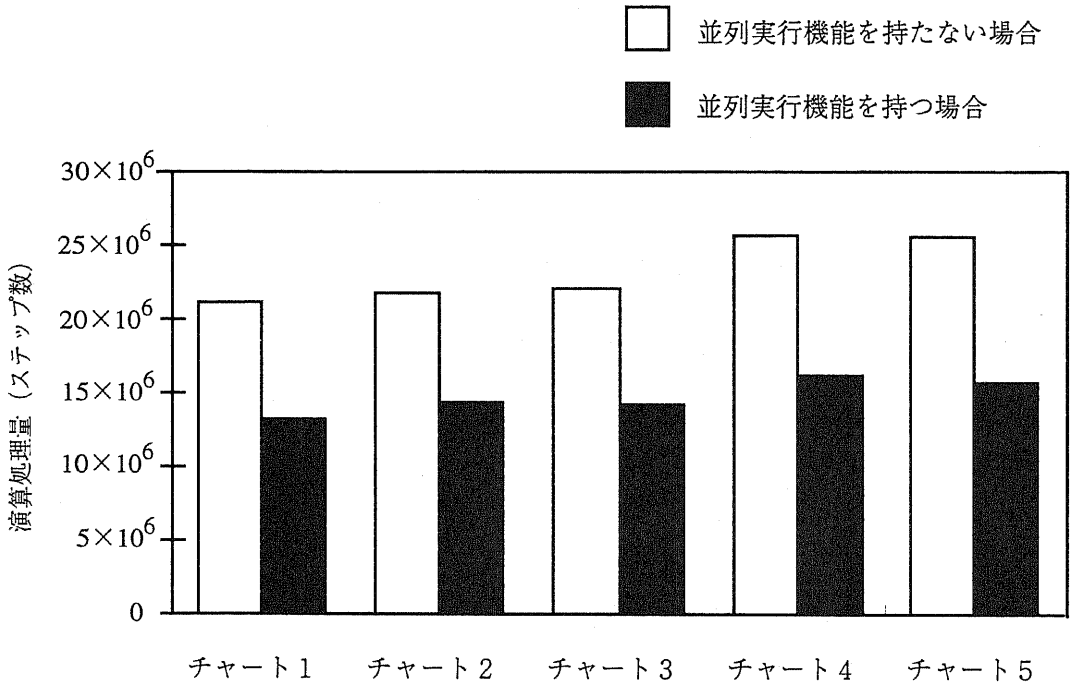
また、表4.4と表4.5の結果と比較することにより、並列実行機能が追加されることで、DCT 演算が約 40% 高速にできることが分かる。

4.5.4　並列実行を用いたソフトウェア JPEG の評価

図 4.9 に、4.5.2 節および 4.5.3 節で評価結果の良かった Chen および Arai の DCT 演算の高速化アルゴリズムをそれぞれ 2 次元DCT 演算に採用した場合の JPEG 処理全体の速度性能結果を示す。1.2.4 節に示したテレビジョン・システム評価用デジタル標準画像 5 チャート（画素サイズ 752 × 480 のオリジナルデータを、サンプリング比 Y:Cb:Cr=4:2:2 のデータに変換）を用いて Y/Cb/Cr から JPEG ファイル生成までの時間を測定した。その結果、並列実行機能を持たない従来型の MCU と、並列実行機能を持つマイクロコントローラを採用した場合のソフトウェアJPEGの処理速度とを評価した。その結果、前者のマイクロコントローラを採用した場合の約22Mステップ～約26Mステップに対して、後者のマイクロコントローラを採用した場合は約14Mステップ～約17M

ステップで圧縮ができることがわかった。つまり、前者のマイクロコントローラを採用した場合に比べ、後者のマイクロコントローラを採用した場合の方が、約35%高速にソフトウェアJPEGが実現できることがわかった。

またその際の処理速度は、並列実行機能のあるMCUを動作周波数100MHzで使用すると、ソフトウェアJPEGを用いて、VGAサイズで0.15秒以下で圧縮ができることに相当する。



テレビジョン・システム評価用デジタル標準画像

画像サイズ: 752画素 × 480画素
画像フォーマット: YCbCr
サンプリング比: Y:Cb:Cr = 4: 2:2

図 4.9　並列実行機能を用いた場合のJPEG処理全体の速度性能

4.6 むすび

本章では、第2章の議論を基に、静止画像処理JPEGをマイクロコントローラで実現する場合の高速化手法について述べた。

まず最初に、JPEGベースライン・システムの処理内容を分析し、全体の処理性能に大きな影響を与える処理を示した。そして、それらの処理内容からメモリ使用法の観点から高速化すべきものと、アルゴリズムの観点から高速化すべきものに分けた。

次に、メモリ使用法の観点から高速化すべき処理に対して、メモリ・アクセスを考慮した高速化手法について述べた。具体的には、MCUのデータに対するロードおよびストアの削減、ハフマン符号化時のジグザグ走査の削減、パックされた2個のデータのロード、そしてパックされた2個のデータの零判定に関する手法を述べた。そして、これらの手法を用いたJPEGプログラムを開発し、従来手法によるプログラムとの速度性能を評価した。その結果、提案した手法を採用することで2倍以上の処理性能が実現できることがわかった。

さらに、アルゴリズムの観点から高速化すべきDCT演算に対して、マイクロコントローラのアーキテクチャを活用した高速化手法について述べた。従来からのマイクロコントローラの場合、並列実行機能を持たない。この従来型の、乗算器を搭載し積和命令を持つマイクロコントローラの場合、積和演算を用いるChenのアルゴリズムを用いると効率的にDCT演算を実現できることがわかった。また一方で、最近のマイクロコントローラに取り入れられてきた並列実行機能を更に仮定すると、並列実行を効率的に活用できるAraiのアルゴリズムを用いると効率的にDCT演算を実現できることがわかった。これらの結果を用いたJPEGプログラムを開発し、速度性能を評価した。その結果、並列実行機能を採用することで、DCT演算は約40%、JPEG処理全体では約35%高速化できることが判明した。

このようにして、マイクロコントローラを用いたシステムに適した処理手法を用いることで、デジタル・スチル・カメラでよく用いられるVGAサイズの画像（サンプリング比 Y:Cb:Cr=4:2:2）の圧縮が、0.4秒（並列実行機能を持たない従来型のマイクロコントローラの場合、動作周波数66.6MHz）あるいは0.15秒（並列実行機能を持つのマイクロコントローラの場合、動作周波数100MHz）程度で実現できることが判明した。これにより、マイクロコントローラを用いてソフトウェアでJPEG方式による画像圧縮が実用的に実現できることを示すことができた。

第4章の参考文献

1) Joint Photographic Experts Group; ISO/IEC JTC 1/SC 29/WG 10, Digital compression and coding of continuous-tone still images - part 1: Requirements and guidelines, (1994)

2) K. R. Rao and P. Yip; Discrete Cosine Transform - Algorithm, Advantages, Applications, Academic Press, Scan Diego, California, (1990)

3) G. K. Wallace; "THE JPEG STILL PICTURE COMPRESSION STANDARD", IEEE Transactions Consumer Electrics, Vol. 38, No.1, pp.xviii-xxxiv, (1992)

4) T. Sakamoto and T. Hase; "JPEG Software Solution for a 32-bit MCU", IEEE Transactions on Consumer Electronics, Vol. 43, No. 3, pp. 410-417, (1997)

5) T. Sakamoto and T. Hase; "Software JPEG for a 32-bit MCU with Dual Issue", IEEE Transactions on Consumer Electronics, Vol. 44, No. 4, pp. 1334-1341, (1998)

6) W. A. Chen, C. Harrison, and S. C. Fralick; "A Fast computational Algorithm for the Discrete Cosine Transform", IEEE Transactions on Communications, Vol. COM-25, No. 9, pp. 1004-1011, (1977)

7) H. S. Hou; "A Fast Recursive Algorithm for Computing the Discrete Cosine Transform", IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-35, No. 10, pp. 1455-1461, (1987)

8) M. Vetterli, H. Nussbaumer; "Simple FFT and DCT Algorithms with Reduced Number of Operations", Signal Processing (North Holland), Vol. 6, No. 4, pp. 267-278, (1984)

9) B. Lee; "A New Algorithm to Compute the Discrete Cosine Transform", IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. ASSP-32, No. 6, pp. 1243-1245, (1984)

10) C. Loeffler, A. Ligtenberg, and G. S. Moschytz; "Practical Fast 1-D DCT Algorithms with 11 Multiplications", Proceedings IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-1989, pp. 988-991, (1989)

11) Y. Arai, T. Agui, and M. Nakajima; "A Fast Recursive Algorithm for the Discrete Cosine Transform", Transactions IEICE E-71 (11):1095, (1988)

12) T. Shimizu; "M32Rx/D - A Single Chip Microcontroller with A High Capacity 4MB Internal DRAM", Hot Chips 10, pp. 37-48, (1998)

第5章　3次元グラフィックスにおける画像処理技術

5.1　はじめに

カーナビゲーションシステム等の民生用機器では、近年、3次元グラフィックスを用いて、ユーザに客観的に、より理解しやすい画面表示を工夫する傾向にある。そのため、民生用機器で容易に使用できる3次元グラフィックスに対する要求が強い。この要求を満足するために、組込み用32ビット・マイクロコントローラと安価なDRAMを用いて実現する3次元グラフィックスが必要となってきた。

従来、3次元グラフィックスを実現するには、浮動小数点演算器(FPU: Floating Point Unit)が存在し、さらに特別のグラフィックスエンジンを載せたビデオカードや、高性能なワークステーションのように3次元ラフィックス専用のハードウェアを必要とするというのが一般的であった[1]。

一方、一般的にマイクロコントローラでは、(1)FPUがない、(2)乗算器が小さい(例えば、16ビット×16ビットや32ビット×16ビット)、(3)特別なグラフィックスエンジンがない、等のハードウェア上の制限がある。そのため、マイクロコントローラを使用して3次元グラフィックス機能を実現するのは難しく、実現したとしても処理速度が遅く、実用的な描画性能が得られない。

本章では、マイクロコントローラとDRAMを用いて、FPUや特別な3次元グラフィックス・エンジンを持たなくても、マイクロコントローラを組み込んで使用する機器で使用可能な3次元グラフィックスをソフトウェアで実現する手法を検討する。そして、第2章で述べたマイクロコントローラを用いた画像・音響処理技術の開発手法を、3次元グラフィックスに適用する場合について、具体的な高速化手法を示し、その性能を評価し、有効性を検証する。

5.2　開発したアルゴリズム

本節では、まず5.2.1節で3次元グラフィックスの処理概要、5.2.2節で実現した処理手順、5.2.3節で実現した3次元グラフィックスの処理時間の内訳についてそれぞれ説明する。

5.2.1　3次元グラフィックスの処理概要

現実の物体を3次元のデータとしてコンピュータで扱うには、物体の形状、色、運動、目の位置(視点)や視野、光源の配置のデータを単純化・抽象化して表現する必要がある。この抽象化の処理をモデリングと呼び、できあがったデータをモデルと呼ぶ。具体的には物体の幾何形状、表面の色や質感の情報、光源の大きさや性質、物体

の運動と変化、視点位置および視線の方向と視野角がある。本章で考察する３次元グラフィックスでも、これら５つのモデルを取り扱う。

次に、モデルが完成したら、物体がどのように見えるかを計算してディスプレイ上に表示する処理が必要である。この操作をレンダリングと呼ぶ。このレンダリング処理では、座標点(X, Y, Z)の座標データをもとにした四則演算、平方根といった演算処理が中心である。通常、座標データを浮動小数点で表現し、これに基づいた演算処理も浮動小数点演算を用いて実現する。

レンダリングの過程は２段階の処理に分けることができる。第一段階は、陰面処理と呼ばれる物体のどの部分が見えるかを決定する処理で、第二段階は、色調計算と呼ばれる物体にどんな色を見つけるかを決める処理である。

陰面処理の代表的な手法には、Zバッファ法[2]、スキャンライン法[3]、レイ・トレーシング法[4]などがある。また物体の色を決定するための色調計算の代表的な手法には、ランバートの余弦則を適用したモデル[5]、Phongの提案したモデル[6]、Torrance-Sparrowの反射モデル[7]などがある。

本論文の陰面処理と色調計算では、この３次元グラフィックスの用途が、グラフィックス用にFPUやアクセラレータを持たない組込み用途ということを考慮し、計算量と除算回数の少ないことを最優先に考えた。そのため、陰面処理については、Zバッファ法、色調計算についてはランバートの余弦則を適用したモデルを採用した。

5.2.2 実現した処理手順

組み込み用マイクロコントローラ向けの３次元グラフィックス実現手法検討のために今回開発した処理手順について述べる。

図5.1に３次元グラフィックスを実現するための処理手順を示す。以下、各処理について説明する。

まず、第一段階ではユーザがデータを入力する。この時の入力データは、物体の幾何形状、表面の色や質感の情報等で、5.2.1節で述べた先の５つのモデルである。なお、物体の運動と変化のモデルについては、ある特定時間（一般的にキーフレーム[8]と呼ぶ）における物体の位置データを指定することとした。

次に第二段階として、第一段階でユーザが指定したキーフレームの位置データを元に、3次スプライン[9]を用いて補間値を求める方法により、残りのフレームにおける物体の運動と変化のモデルを生成する。

更に第三段階として、レンダリング処理を行う。つまり、第二段階で計算して求めた結果と第一段階で入力された各種モデルデータにより、画面上に表示する物体の形と色を計算する。

第四段階としてディスプレイ画面上に表示する。つまり、先の段階で計算して求めた描画情報を、レンダリング処理用バッファメモリからディスプレイ上に表示するためのフレームメモリへ転送する。

最後に、ユーザが指定した最終フレームデータであるかどうかを判定し、違う場合

は、次のフレームデータに対して第三段階から処理を続行する。もし、最終フレームデータであれば、処理を終了する。

次節では、上記で説明した3次元グラフィックスの各々の処理について、計測した処理時間の結果について述べる。

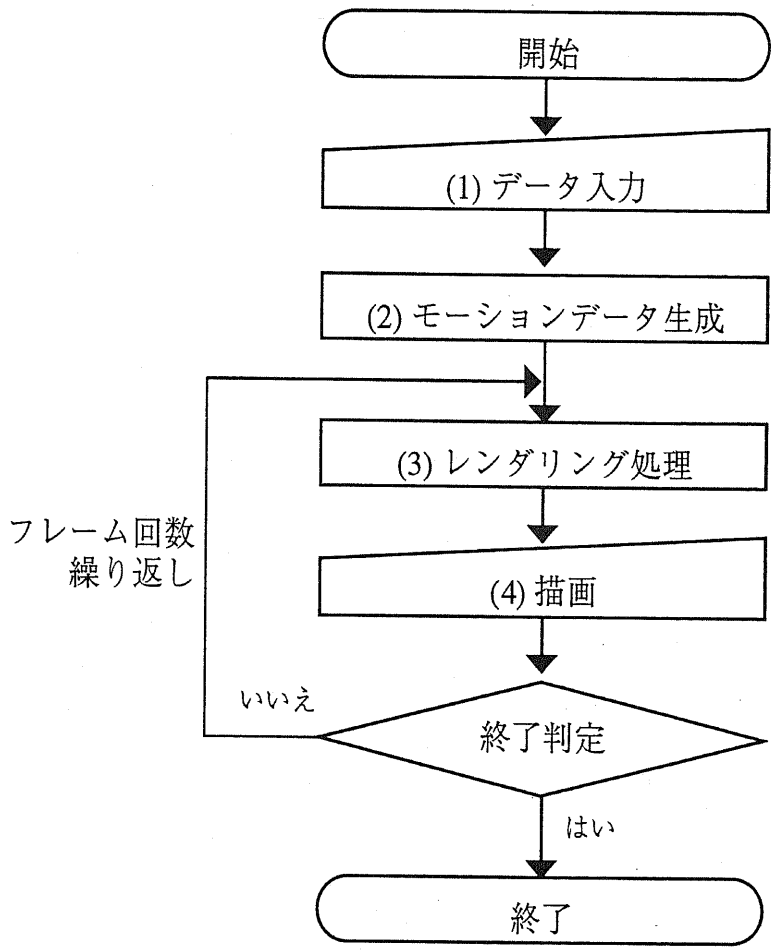


図5.1　3次元グラフィックスの処理手順

5.2.3　処理時間の内訳

5.2.2で述べた3次元グラフィックスの処理手順を浮動小数点を用いて実現し、各処理に要するサイクル数の割合を調べた。なお、1.2.3節で述べた32ビット・マイクロコントローラにはFPUが搭載されていないため、浮動小数点演算にはソフトウェア・パッケージを使用した。また計測には、図5.2で示す図形をY軸中心に時計回りに360度回転させた時の処理時間を計測した。

計測の結果、(3)のレンダリング処理に3次元グラフィックス処理全体の99%以上の時間がかかっていることがわかった。そこで、FPUがなく、乗算器が小さく、特別な

グラフィックスエンジンがないといった制限をもつハードウェア上で高速な３次元グラフィックス・ライブラリを実現するには、レンダリング処理の高速化が解決すべき最大の課題であると判断できる。

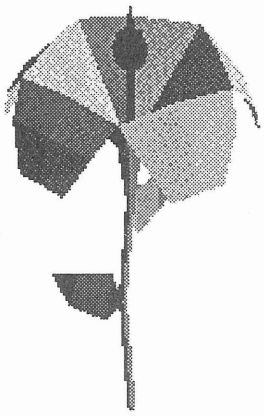


図 5.2　処理時間の計測に用いたグラフィックス・データ

次にこのレンダリング処理の内容を詳細に調べた。その結果、陰面処理部分と色調計算部分を合計したレンダリング処理における、四則演算を始めとする浮動小数点演算処理が占める割合が約93% であることが判明した。さらに、レンダリング処理で使用している浮動小数点演算の種類について調べた。その結果、浮動小数点演算として四則演算と平方根が多いことが判明した。図5.3にレンダリング処理における１フレーム当たりに実行された浮動小数点演算の四則演算および平方根の計算の回数を示す。

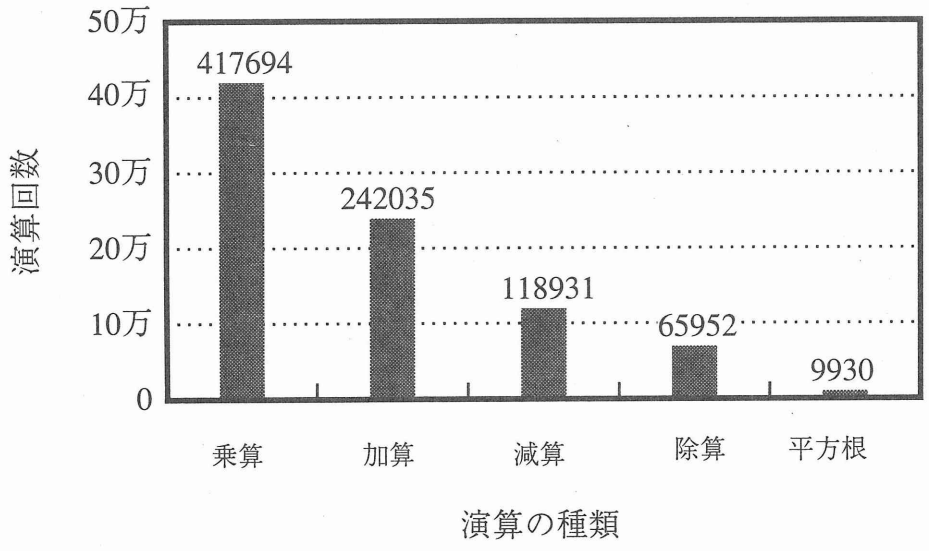


図 5.3　１フレーム当たりのレンダリング処理における浮動小数点演算の回数

5.3 高速化する上での課題

3次元グラフィックスをFPUを搭載しないマイクロコントローラで使用するには、その処理速度が最大の課題である。5.2節において処理速度を分析した結果、処理時間のほとんどをレンダリング処理が占め、そのレンダリング処理の約93%を浮動小数点演算が占めていることがわかった。これは、レンダリング処理の全体の処理に占める割合が高いこと、レンダリング処理では浮動小数点を用いた四則演算や平方根といった演算処理が非常に多く実行されることに起因している。

3次元グラフィックスを高速化するためには、レンダリング処理を高速化する必要がある。そのために有効な大きな手段として、浮動小数点演算処理を整数演算処理に置き換える方法とレンダリング処理の対象とする画素の数を減らす方法が考えられる。前者は、2章に述べた固定小数点化に当たる。図5.3よりレンダリング処理の演算としては乗算が多いことが判明しているため、固定小数点化の際には乗算の実現方法と演算精度の検討および算術演算の精度検討が重要になる[10]。また後者に関しては、2章に述べた人間の視覚特性を考慮して演算精度を検討する方法を適用することが考えられる。すべての画素においてレンダリング処理を実施することで現実に近い画面を再現しているが、そのレンダリング処理が多くの演算処理を必要としているため処理全体の速度が遅くなっている。従って、レンダリング処理と、レンダリング処理に代わる処理量の比較的少ない処理を併用し、レンダリング処理を実施する画素を減らして高速化する方法である。この際、レンダリング処理の一部を処理量の比較的少ない処理で代用するため、画質の劣化が発生する。そのため、この画質劣化を人間の視覚上問題のない範囲の劣化に留める代替処理を考案する必要がある。以下では、これら2つの観点から課題を整理し、解決手法を示し、その有効性を評価する。

5.4 固定小数点化による高速化手法

本節では、5.3節に述べた浮動小数点演算処理を整数演算処理に置き換える方法で3次元グラフィックスを高速化するための課題を分析し、その解決手法を示す。そして従来手法に対する提案手法の効果について評価する。浮動小数点演算を整数演算に置き換えるために、特に、小数点位置の工夫と、適応的な乗算演算を用いた固定小数点化および除算回数の削減についてを提案する。

5.4.1 固定小数点化による整数演算処理化

(1) 小数点位置の工夫

小数点位置を固定し、整数で数値を表現することにより、これまで浮動小数点演算で計算していた部分を整数演算に置き換えることができる。これにより、大幅な高速化を計ることができる。しかし一方では、表現できる値の範囲が狭まるという制限も生じ、演算精度も低下するという問題点がある。

今回検討している3次元グラフィックスでは、広く使用される民生機器を対象としている。そのため、対象となる画面サイズをカーナビゲーション等で普及している

1/4 VGA まで と限定した。そして、1/4 VGA の画面上で描画して実用上差し支えない X, Y, Z の範囲を経験的に求めた。その結果、X, Y 値については、小数点以下 4 桁、また、6 桁の整数を有効桁数とした。そのため、32 ビットのデータにおいて、X, Y 値については、1 ビット符号、19 ビット整数部、12 ビット小数部で表現した。Z 値については、通常取りうる値が X, Y 値と比べて 3 ～ 4 桁小さいため、32 ビットのデータにおいて、1 ビット符号、12 ビット整数部、19 ビット小数部として、表現することとした。

(2) 適応的な乗算手法

3 次元グラフィックスでは、レンダリング処理部分で乗算が多用されている。例えば、図5.3で示したケースで、1 フレーム当たり417694回の乗算が実行されている。(1) で述べた制約された画面サイズのもとでは、全ての乗算に対して 32 ビット × 32 ビットの演算精度が必要なわけではなく、データによっては、32 ビット × 32 ビットより低い演算精度でも十分な場合がある。

今回使用したマイクロコントローラは 32 ビット × 16 ビットの乗算器を搭載している。そのため、32 ビット × 32 ビットの乗算の実行には30サイクル、32 ビット × 31 ビットでは 11 サイクル、32 ビット × 16 ビットでは 1 サイクルがそれぞれ必要である。つまり、乗算の演算精度によって、必要なサイクル数が各々異なっている。

そこで、乗算の高速化のために、精度と処理時間の観点から処理内容に合わせて使用する乗算の演算精度を使い分けることにより、計算に要するサイクル数の削減を行った。具体的には、Z 値を求めるのには 32 ビット × 32 ビット、X, Y 値を求めるには 32 ビット × 31 ビット、色調を求めるには 32 ビット × 16 ビットというように使い分けた。

(3) 除算回数の削減

RISCアーキテクチャを採用したマイクロコントローラにおいては1ビットの除算に 1 サイクルかかり、これを繰り返し実行することで32ビットの除算を実行する。そのため、32 ビットの除算には、通常 40 サイクル程度かかり、除算回数を削減することは高速化の重要な手法である。

一般的に、レンダリング処理において、表示画面上の座標系への透視変換式[3]の関係より、

$$(x, y, z) \rightarrow (x/z, y/z, 1/z)$$

が示すとおり、同一の値（この場合 z）による除算が 3 回実行される。この変換は、頂点毎に実行される。例えば、図 5.2 の場合は 432 回の変換が行われている。

除算一回当たり 40サイクルかかると仮定する。そこで、上記座標系の変換式を実行するには、1/z, x/z, y/z の 3 回除算するので

$$40 + 40 + 40 = 120 \text{ [サイクル]}$$

必要である。

一方、上記演算をまず逆数（この場合 1/z）を求めてから、その結果を 32 ビット × 31 ビットの乗算で実現するように処理を変更することを考える。この場合の演算回数

は、逆数を求めるのに30サイクルかかるが、乗算はそれぞれ11サイクルを2回実行すればよく、

$$30 + 11 + 11 = 52 \text{ [サイクル]}$$

となる。

このように、上記の逆数を求める手段を用いれば1頂点（x , y , z）を計算する毎

$$120 - 52 = 68 \text{ [サイクル]}$$

の削減となった。

5.4.2 開発したソフトウェアの評価

(1) 評価に使用したハードウェアの概要

評価には1.2.3節に示した32ビット・マイクロコントローラ搭載の評価ボードを用いた。この評価ボードにはインターフェースとしてLCDインターフェースがあり、このLCDインタフェースに1/4 VGAのTFTの液晶ディスプレイを接続した。

(2) 評価結果

本節では、5.4節で述べた高速化手法の評価結果について述べる。

性能評価に際しては、5.2.3節で使用した図5.2の測定データを使用して計測した。

なお、5.2.3節で使用した、浮動小数点演算のソフトウェア・パッケージを用いて実現した3次元グラフィックスの処理性能を計測した場合、描画性能はおよそ100ポリゴン/秒であった。

(a) 固定小数点化の効果

図5.4に小数点位置と適応的な乗算演算を用いた時の速度性能を示す。

図5.4より、5.4.1節の(1)および(2)で述べた小数点位置の工夫と適応的な乗算手法を用いることにより描画性能を約13.8の高速化した。つまり、小数点位置の工夫により、元の描画処理性能に対して約9.2倍の高速化を実現し、適応的な乗算手法で更に約1.5倍の高速化を実現した。

(b) 除算回数の削減の効果

図5.5に除算回数の削減を用いた時の速度性能を示す。

図5.5より、5.4.1節の(3)で述べた除算の削減により、5.4.1節の(1)と(2)を適用した場合と比べ更に約7.4倍高速化することができた。

従って、5.4.1節に述べた手法を採用することで、描画性能は10,000ポリゴン/秒が得られ、もとの浮動小数点演算を用い、浮動小数点ソフトウェア・パッケージを用いて処理をした場合と比べて、およそ100倍の高速化が実現できた。

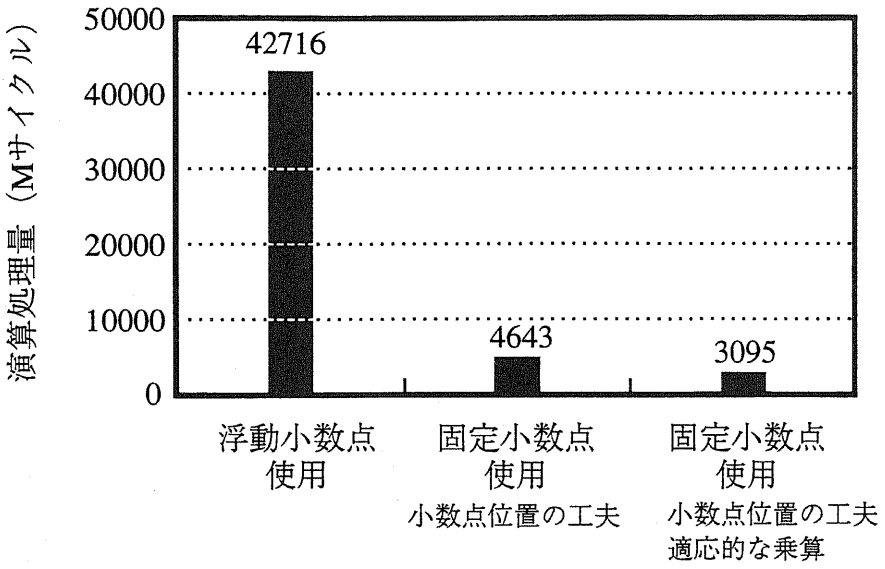


図 5.4　小数点位置と乗算の工夫による効果

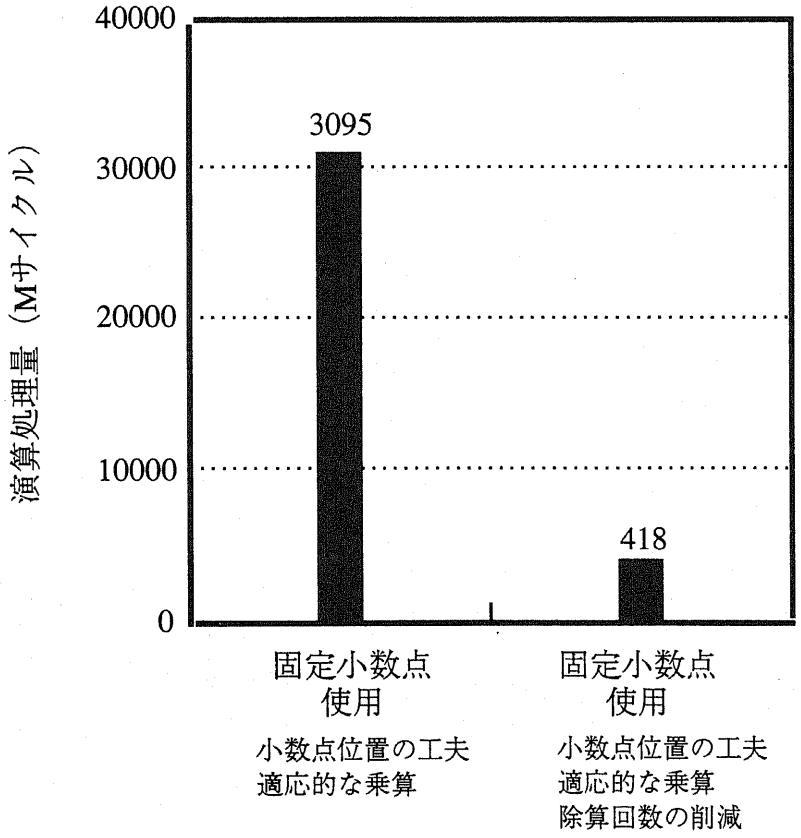


図 5.5　除算回数削減の効果

(3) 演算精度

性能評価に用いた測定データを用い、浮動小数点演算のソフトウェア・パッケージを用いて実現した3次元グラフィックスと、5.4.1節に述べた手法を採用した3次元グラフィックスそれぞれの描画結果を比較することで演算精度を確かめた。その結果、両者は同等の演算精度を持つことがわかった。つまり、浮動小数点を用いた場合と同じ演算精度を持ちながら、およそ100倍の高速化が実現できたことになる。

5.5 人間の視覚特性を考慮した高速化手法

本節では、5.3節に述べたレンダリング処理の対象とする画素の数を減らす方法で3次元グラフィックスを高速化するための課題を分析し、その解決手法を示し、そして従来手法に対する提案手法の効果について評価する。

5.5.1 画素補間処理を組み合わせたレンダリング処理

5.4節では、画面サイズをカーナビゲーション等で普及している1/4VGAまでと制限することで、浮動小数点演算を用いた3次元グラフィックスと同等の精度を持つ3次元グラフィックスを、マイクロコントローラとDRAMを用いて実現する手法を提案した。その結果、10,000ポリゴン/秒の描画性能が得られた。しかし、動きのある画像を描画する用途では、まだ処理速度の点で難点がある。

そこで、5.4節で提案した手法を用いた固定小数点演算を用いた3次元グラフィックス・ライブラリを使用し、図5.2のモデルデータをY軸中心に時計回りに360度回転させた時の処理時間を計測し、5.2.3節と同様に各処理で要するサイクル数の割合を調べた。図5.6にその結果を示す。図5.6より、5.4節で提案した手法を採用しても、レンダリング処理が処理全体の90%の時間がかかっていることがわかる。従って更に高速化し、動きのある画像を描画するためには、やはりレンダリング処理の高速化が解決すべき最大の課題であると判断できる。

そこで、レンダリング処理を高速にするため視覚上問題のない範囲で、画質の劣化を許容することにした。これを前提として、各画素ごとにレンダリング処理を用いて陰面処理と色調計算を行うのではなく、レンダリング処理と画素補間処理を並用し、レンダリング処理の回数を減らす手法を提案する。つまり、レンダリング処理を行う画素を減らすことで全体の処理の高速化を図ろうというものである。レンダリング処理を行わない画素では、周囲の画素値から補間処理で値を決定する。また同時にテクスチャーマッピングもあわせて高速化することを考える。

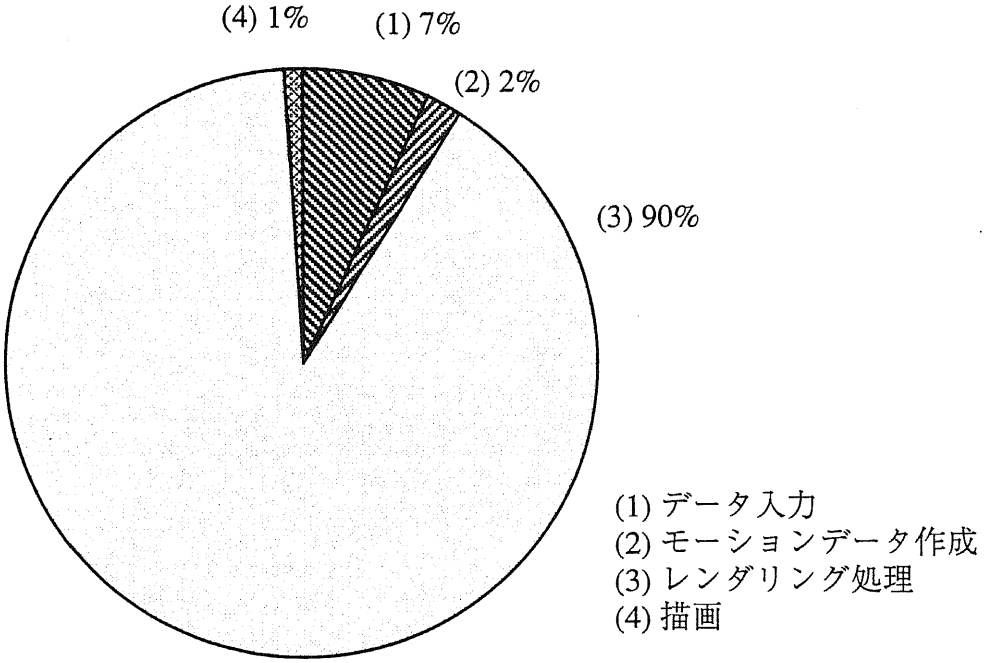


図 5.6　固定小数点演算を用いた 3 次元グラフィックス処理時間の内訳

シェーディング処理の一部として補間処理を用いるという観点から考察すると、Gouraud shading to phong shading といった、いわゆる interpolated shading[6]という手法がある。しかし、これらのシェーディング手法には多くの課題がある[6]。その中でも特に次の 2 点からこれらのシェーディング手法は著者らの用途に適しないと判断した。つまり、これらの手法はテクスチャーマップに適用できないという点と凸形の最小分割り距形にしか適用できないという 2 点である。前者はテクスチャーマップが 3 次元グラフィックスにとって必須機能で、高速化の必要があるからである。また後者はユーザーが指定したモデル記述のポリゴンを最小分割距形に分割する処理に時間がかかることと、分割した場合頂点数が多くなりシェーディング処理が長くなるからである。以下では提案した手法の詳細について述べる。

(1)　画素補間手法

図 5.7 を用いて今回用いた画素補間手法を説明する。座標点(x,y)、(x+3,y)、(x,y+3)、(x+3,y+3)の画素値はレンダリング処理を用いて計算する。これら以外の座標点、例えば(x+1,y+3)、(x+2,y+3)に関しては周囲の色調（この場合は座標点(x,y+3)と(x+3,y+3)の値）から補間する。画素補間方法としては、座標点(x,y+3)あるいは(x+3,y+3)の色調値をそのまま使用する方法や線形補間を用いる方法などが考えられる。しかし、3 次元グラフィックの場合、陰面処理を考慮しない通常の補間処理方法では、画質に与える影響が大きい。

5.2節に述べた3次元グラフィックでは、陰面処理として、Zバッファ法を採用しているので、各座標点はZ値を持っている。そこで、各座標点のこのZ座標を参照して、Z値の近い座標点の色の値を採用する補間手法を考案した。例えば座標点(x+1,y+3)の色の値を決定する場合に、この点でのZ値と、座標点(x,y+3)および(x+3,y+3)のZ値を比較し、差の小さい座標点の持つ色の値を採用する。このことにより、例えば、あるモデルの端の方に突然背景色が混じるような現象を少なくすることができる。

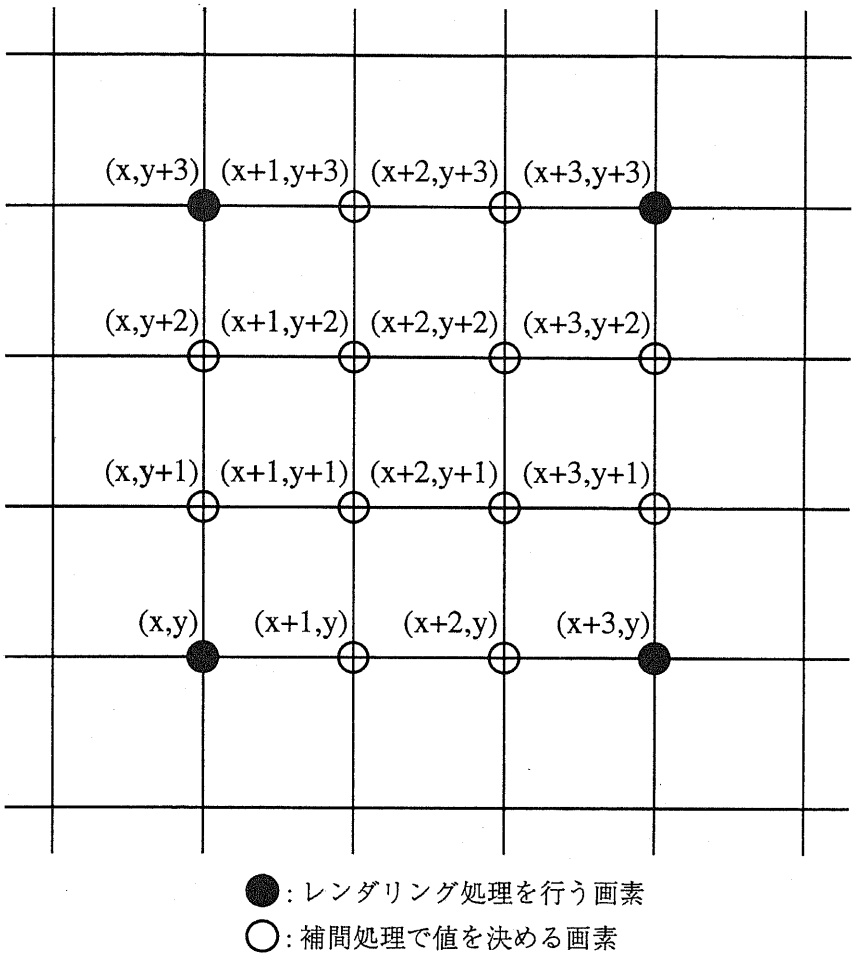


図5.7　画素補間

(2)　レンダリング処理による再計算

レンダリング処理と(1)に述べた補間処理を用いてレンダリング処理回数を減らしながら描画が可能となる。しかし、画質の観点から、(1)に述べた画素補間手法ですべての補間点を決定すると問題の生じる場合がある。その場合を図5.8と図5.9を用いて説明する。

5.2節に述べた3次元グラフィックでは、描画するモデルをX軸方向に左端から順次Zバッファ法を用いながら陰面処理を行い、色調計算を実施している。この陰面処理と色調処理計算を行う方向と直角の方向（つまりY軸方向）での補間処理において、補間のために参照する座標点の色の値がともに背景色の場合、補間値を背景色としてしまうと、描画するモデルの頂点が欠け、同時にその背景色が補間処理で伝播することで本来の画像とかなり異なる画像が生成されることがある。

例えば、図5.8において座標点(x,y+2)の補間値は、背景色を持つ座標点(x,y+1)と(x,y+4)から決定する。参照する座標点の値が共に背景色なのでこの場合の補間値も背景色となる。

その後先ず座標点(x+3,y+2)の値が座標点(x+3,y+1)と(x+3,y+4)の値から補間を用いて計算し、その後、座標点(x+1,y+2)および(x+2,y+2)の値を補間する。この場合座標点(x,y+2)、(x+1,y+2)、(x+2,y+2)、(x+3,y+2)のZ値の値によっては、最後の座標点を除く3点の色の値が背景色になる場合がある。その場合の補間結果を図5.9に示す。

前述のような場合を回避するために、(1)で述べた補間処理の際、Y軸方向の補間において、参照する座標値の色の値が共に背景色の場合は、補間で色調を決定せず、レンダリング処理でその値を計算して決定する必要がある。

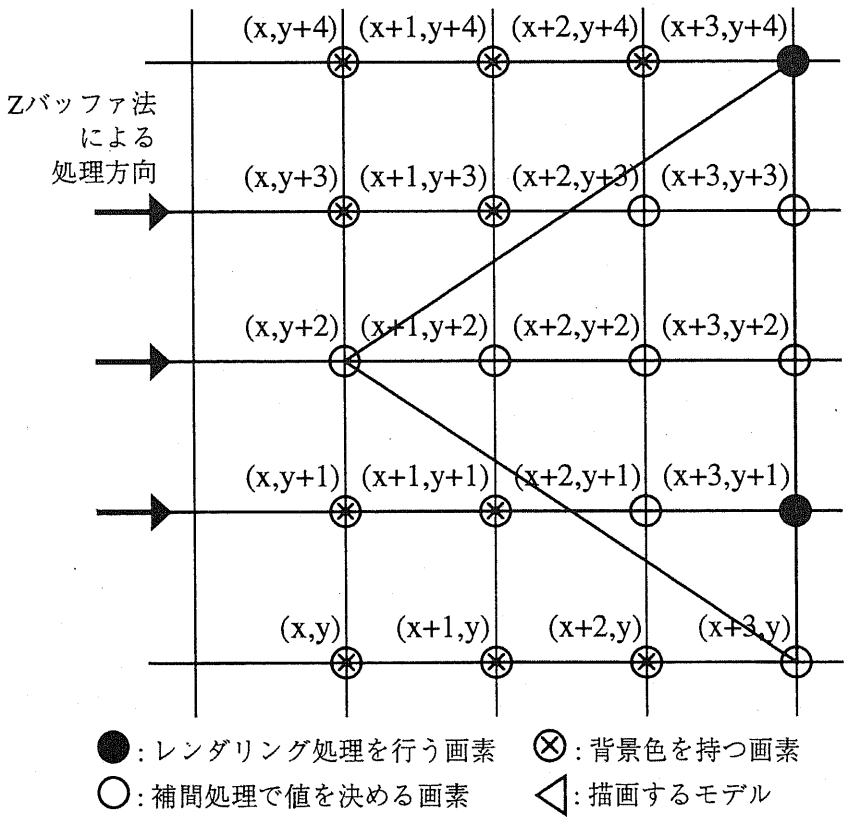


図5.8　レンダリング処理による再計算が必要なモデル例

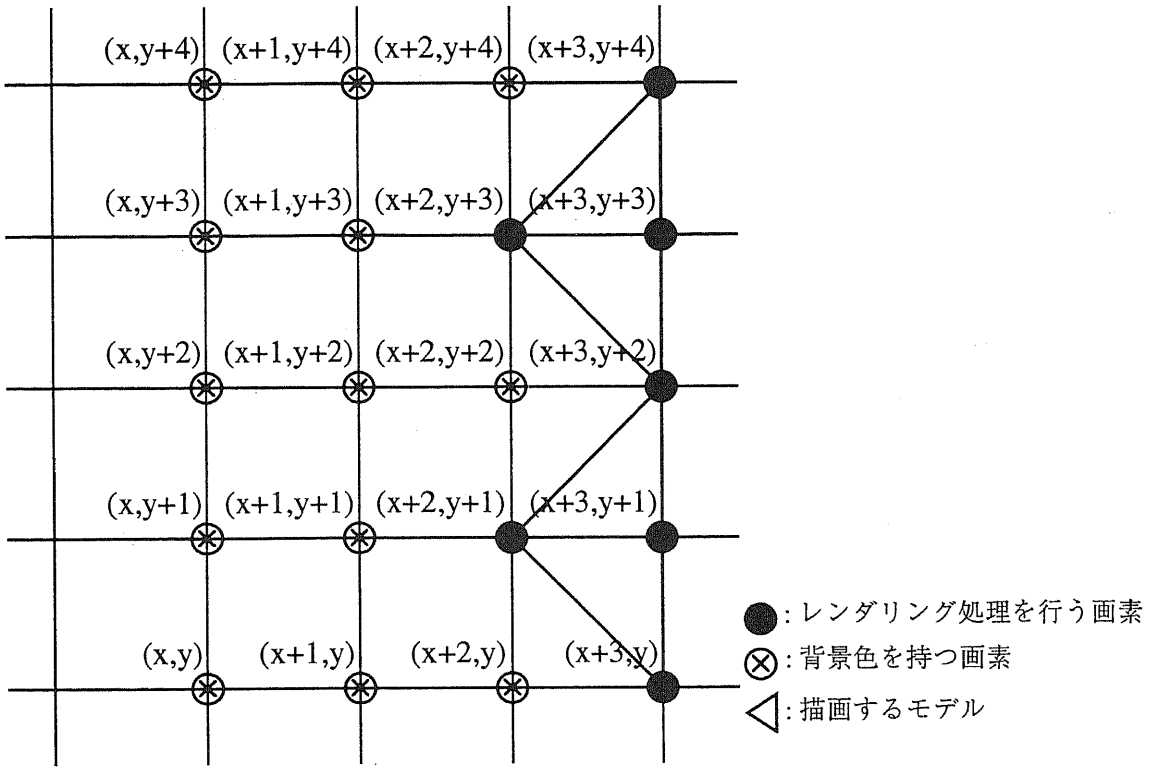


図 5.9　再計算を行わない場合のレンダリング結果

5.5.2　評価

(1)　評価方法

提案した手法は、例えば動きのある画像に対応するために、視覚的に問題のない範囲で画質を落としても、処理測度を上げるためのものである。従って、処理測度と画質は重要な要因である。今回の評価では、1.2.3節で述べた評価装置を使い、処理測度とS/N比を評価した。今回の評価では、X軸方向、Y軸方向ともに1画素ごとにレンダリング処理を行い、その間を補間する場合を判定した。また、S/N比の計算では、5.4節で提案した3次元グラフィックスで生成された画像を原画像とし、本節で提案した手法で生成した画像との比を次の計算式[11]を使い計算した。

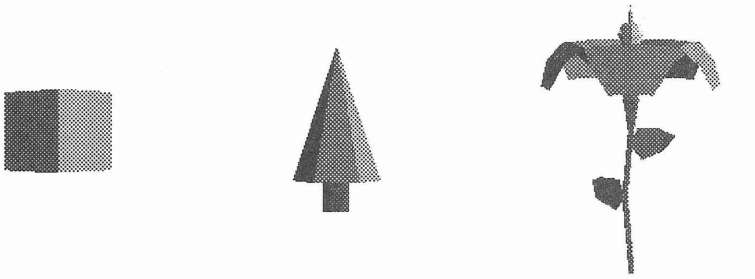
$$\text{SN比}=-10\log_{10}\left[\frac{\frac{1}{KL}\sum_{k=0}^{K-1}\sum_{l=0}^{L-1}\{f(k,l)-f'(k,l)\}^2}{63^2}\right] \quad [\text{dB}]$$

ただし、

KL　　：原画像全体の大きさ、

f(k,l)　：原画像の位置(k,l)における階調値（6ビット）、
f'(k,l)　：再生画像の位置(k,l)における階調値（6ビット）。

また、評価に使用したモデルは、大きなポリゴンから構成された立方体(図5.10の(a))、中くらいのポリゴンから構成された木(図5.10の(b))、小さいポリゴンから構成された花(図5.10の(c))の3種類を使用した。



ポリゴン数：6
全画素数：578724

ポリゴン数：26
全画素数：482270

ポリゴン数：432
全画素数：1504682

(a) 立法体　(b) 木　(c) 花

図5.10　評価に使用したモデル

さらに、本手法はテクスチャーマップにも適用できるためテクスチャーマップ機能とあわせて評価した。この評価に使用したモデルを図5.11に示す。図5.10に示した立方体の1面に花をテクスチャーとして付加したものと、ポリゴンで描画したものを比較した。処理性能の評価では各モデルを図5.10はY軸を、図5.11はZ軸を中心に時計回りに360°回転させた際の処理時間と、各フレームのS/N比を測定した。なお、フレーム数は見た目上なめらかに動作していると認識される0から20の21フレームとした。

ポリゴン数：362
全画素数：816295

ポリゴン数：6
全画素数：199952

(a) 基データ　(b) ポリゴンで描画したもの　(c) テクスチャーマップを使用したもの

図5.11　テキスチャーマップへの適用の評価モデル

(3) 評価結果

図5.10に示したモデルに対する処理時間の結果を図5.12に示す。図からも分かるように、モデルを構成するポリゴン数が小さく、画素値が大きいものほど、今回提案した手法の効果は大きく、約5倍の高速化が実現できたことがわかる。また、S/N比に関しては、S/N比の一番悪い花(図5.10(c))のデータでも36dB以上であり、画質の面では問題がなかった。

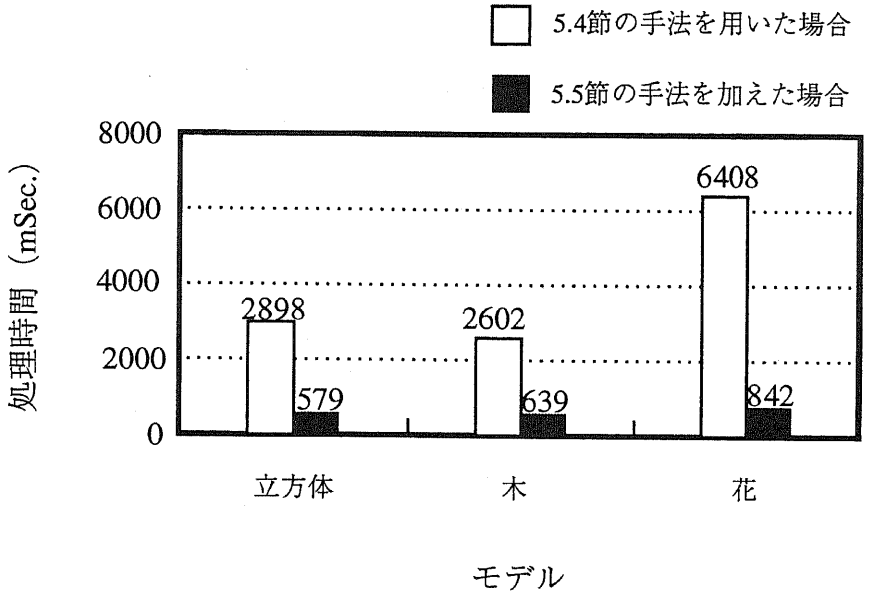


図5.12 評価結果

図5.11のテクスチャーマップを使用したモデルに適用した処理時間の結果を図5.13に示す。図から分かるように、提案した手法を用いることで、テクスチャーマップを用いた場合とポリゴンで描画する場合の速度比が大きくなり、テクスチャーマップが一層高速に実現できることがわかる。これは、テキスチャーのサイズが大きく、しかも提案した手法はポリゴン数が少なく、面積の大きなモデルに対して効果が大きいことが原因と考える。従って、大きなテキスチャーを付加する場合は、本手法は、テクスチャーマップの高速化手法として有効である。

また、S/N比に関しては、図5.10および図5.11(b)のモデルと比べて4dB程度低かった。S/N比のこの低下は図5.11(c)を見てもわかるが、動画としてみれば画質の面ではまったく問題がないと考える。

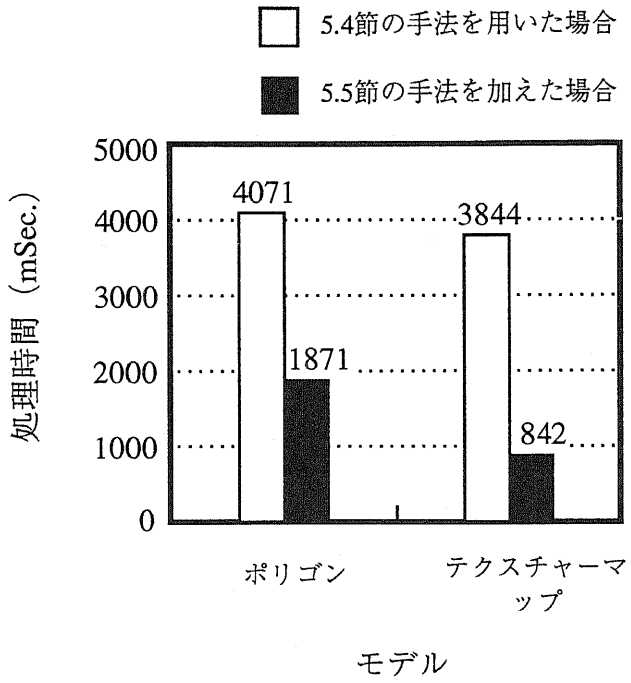


図5.13　テクスチャーマップを使用したモデルの評価結果

5.6 むすび

近年、組込み用32ビット・マイクロコントローラを用いて実現する3次元グラフィックス・ライブラリの要求が日増しに高まってきた。しかし、組込み用マイクロコントローラでは一般的に、浮動小数点演算器がない、乗算器が小さい、特別なグラフィックスエンジンがない、等のハードウェア上の制限があり、このようなマイクロコントローラでは3次元グラフィックス機能を実現することは難しかった。

本章では、32ビット×16ビットの乗算器を持った組み込み用マイクロコントローラとDRAMを用いて、たとえFPUや特別な3次元グラフィックス・エンジンを持たなくても、3次元グラフィックス機能をソフトウェアで実現する方法について提案し、実験機により評価した。

まず、3次元グラフィックスの全体の処理量を見積もった。その結果、高速化のためには、全体の処理時間のほとんどを占めるレンダリング処理を高速化する必要があることがわかった。さらに詳細に調べると、このレンダリング処理の約93%を占める浮動小数点演算を高速化することが、一番高速化のためには効果が大きいことがわかった。

そこで、浮動小数点演算処理を整数演算処理に置き換えたり、あるいは、RICSアーキテクチャを採用したマイクロコントローラにおいて多くの演算サイクルがかかる除算演算を削減することを検討した。この目的のために本章では、小数点位置の工夫と、適応的な乗算演算を用いた固定小数点と、除算回数の削減の3つを提案した。

(1) 1/4 VGAの解像度で実用上問題のない座標値（X, Y, Z）の固定小数点表現を採用した。

(2) 処理内容を吟味し、それぞれの演算で必要な精度に応じて3種類の乗算ビット数を選択する。

(3) 実行サイクル数が多い除算演算の回数を減らすために、まず逆数を求め、求めた逆数との乗算処理に変更する。

次に、実際に実験機を作り、その上に、上記のソフトウェアを実装して、その機能および性能の評価を行なった。

従来のソフトウェアパッケージを用いて浮動小数点演算部分を処理した場合は、およそ描画性能は100ポリゴン/秒であった。しかし、本章で提案した固定小数点の手段を用いることにより、従来と比べて約13.8倍の描画性能を高速化することができた。この固定小数点の工夫に加えて、先に提案した除算の削減を施すことにより、さらに約7.4倍高速化することができた。従って、これらの工夫により描画性能は10,000ポリゴン/秒が得られ、もとの浮動小数点演算部分においてソフトウェアパッケージを用いて処理をした場合と比べて、およそ100倍高速化を実現することができた。

また、レンダリング処理を高速にするため視覚上問題のない範囲で、画質の劣化を許容することにした。これを前提として、各画素ごとにレンダリング処理を用いて陰面処理と色調計算を行うのではなく、レンダリング処理と画素補間処理を並用し、レンダリング処理の回数を減らす手法を提案した。つまり、レンダリング処理を行う画素を減らすことで全体の処理の高速化を図ろうというものである。レンダリング処理を行わない画素では、周囲の画素値から補間処理で値を決定する。

この手法を評価した結果、提案した手法を用いてもS/N比から判断して画質の大きな劣化は見られないことがわかった。また、モデルを構成するポリゴンが大きい程、今回提案した手法の効果は大きく、約5倍の高速化が実現できたことがわかる。また、提案した手法はテクスチャーマップにも効果があり、大きなテクスチャーを付加する場合は、本手法は、テクスチャーマップの高速化手法として有効である。

参考文献

1) Silicon Graphics, Inc.; “Indigo Family Technical Report, 2.0”, pp.48-70, 1993.

2) J. D. Foley, A. Dam, S. K. Feiner and J. F. Hughes; “Computer Graphics Principles and Practice”, Addison-Wesley, pp. 668-672, (1993)

3) 同上; pp. 680-686

4) 同上; pp. 701-714

5) 同上; pp. 723-724

6) 同上; pp. 738-739

7) 同上; p. 764

8) N.Burtnyk and M.Wein; “Computer Generated Key Frame Animation”, SMPTE 80, pp.149-153, (1979)

9) 2)に同じ; pp. 511-514

10) K. Yoshida, T. Sakamoto and T. Hase;
“A 3D Graphics library for a 32-bit microprocessors for embedded systems”,
IEEE Transactions on Consumer Electronics, Vol. 44, No. 3, pp. 1107-1114, (1998)

11) Pratt, William K., “Digital Imgae Processing”, Second Edition, Wiley-Interscience, New York, (1991)

第6章　オーディオ復号における音響処理技術

6.1　はじめに

MPEGオーディオは、ISO/IECが定めた広帯域オーディオ信号符号化の国際標準である[1]。標準アルゴリズムは、サブバンド符号化に基づいたレイヤI/II、サブバンド符号化及び適応変換符号化に基づいたレイヤIII(以下MP3と呼ぶ)に分類される。

従来より、MPEGオーディオレイヤI/IIをハードウェアで実現する方式[2][3]や、DSPを使ってソフトウェアで実現する方法[4]はすでに報告されている。そこで、組込み用32ビット・マイクロコントローラとDRAMを使ったMP3デコーダの実現を試みた。

MP3では、レイヤI/IIにも採用されているサブバンド符号化、心理聴覚モデルを用いたビット割り当て等に加え、符号化品質向上のために適応変換符号化やエントロピー符号化が導入されている。この規格に従えば、CDなどの音楽を再生品質を保ちながら、ファイルサイズを約10分の1にまで圧縮でき、インターネット上でのファイルのやり取りが極めて容易にできるようになる。しかしながら、レイヤI/IIに比べ処理がより複雑になるため、汎用のマイクロコントローラを使ってMP3デコーダを実現する場合、その実現方法が課題となる。

本章では、32ビット・マイクロコントローラとDRAMを使ってMP3復号処理をリアルタイムに実行するための高速化手法とその評価結果について述べる[5]。先ず、MP3復号アルゴリズムについて述べ、さらに、1.2.3節で述べたDRAM内蔵32ビット・マイクロコントローラ上で効率的に実現する手法について述べる。そして、第2章で述べたマイクロコントローラを用いた画像・音響処理技術の開発手法を、MP3に適用する場合について、具体的な高速化手法を示し、その性能を評価し、有効性を検証する。

6.2　MPEGオーディオレイヤIII復号化方式の概要

本節では、MP3のデコードアルゴリズムの概要を先ず述べ、そのアルゴリズムの詳細を順次述べていく。

6.2.1　MP3復号処理アルゴリズムの概要

図6.1にMP3復号処理アルゴリズムを示す。図6.1に示す処理のうち多くの演算量を必要とするのは、逆量子化部、IMDCT（Invert Modified DCT）および窓掛け部、サブバンド合成部である。本節ではこれらの処理で行われている演算の定義式を記す[1]。

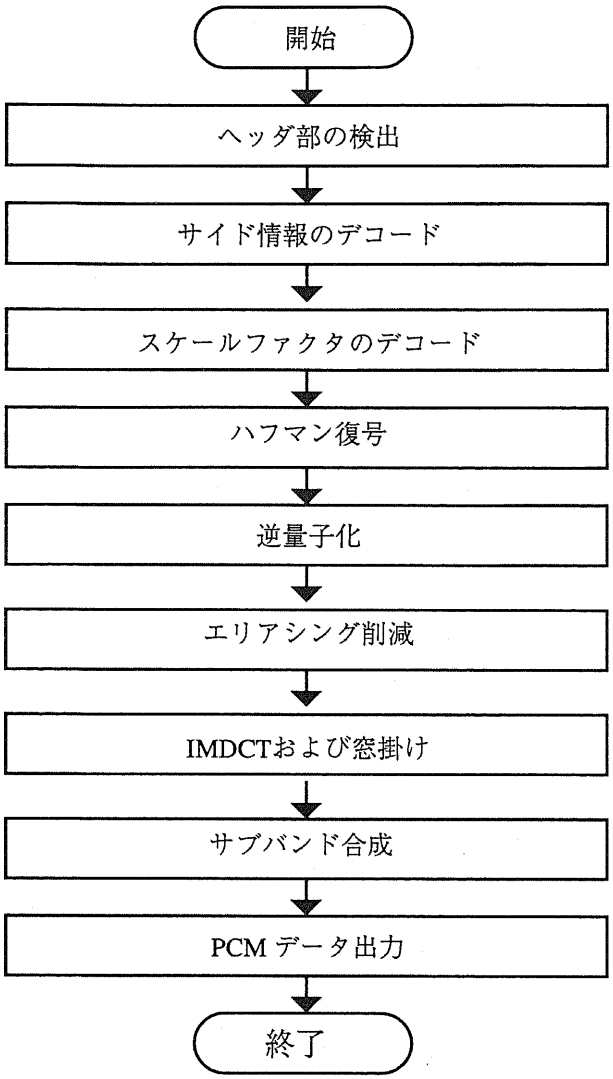


図 6.1　復号処理アルゴリズム

6.2.2　逆量子化

逆量子化はハフマン復号値から周波数領域サンプルを求めるための処理である。周波数領域サンプルは(6.1)式で計算される。

$$\mathrm{Xr}(i,j) = sign(is(i,j)) \times is(i,j) \times 2^{P},\ 0 \leqq i \leqq 31, 0 \leqq j \leqq 17 \tag{6.1}$$

(6.1)式で Xr(i, j) は逆量子化演算結果、$is\ (i, j)$ はハフマン復号結果、P はサイド情報とスケールファクタから求められる定数、$sign(a)$ は a の符号を表す。

6.2.3 IMDCT および窓掛け

エイリアシング削減を行った結果を X(k) とした時、IMDCT は(6.2)式で、窓掛けは(6.3)式で定義される。つまり、IMDCT は cos 係数との積和で定義され、窓掛けは IMDCT の出力値 Z(l) に窓係数を乗じることで定義される。

$$Z(l) = \sum_{k=0}^{N/2-1} X(k)C(l,k) \quad 0 \leqq l < N \tag{6.2}$$

ただし、

$$C(l,k) = cos\left\{\frac{\pi}{2N}\left(2l+1+\frac{N}{2}\right)(2k+1)\right\}$$

$$H(l) = Z(l)\,W(l)\,. \tag{6.3}$$

6.2.4 サブバンド合成

サブバンド合成の処理アルゴリズムを図 6.2 に示す。図 6.2 で行われる処理のうち、演算が行われるのは行列演算および 32 個の再生サンプルデータの計算である。

行列演算は32のサブバンドサンプルに直交変換行列 **N**を掛ける演算で(6.4)式で定義される。

$$V(m) = \sum_{i=0}^{31} N(m,i)\,S(i), \quad 0 \leqq m \leqq 63 \tag{6.4}$$

但し、

$$N(m,i) = cos\left\{\frac{\pi}{64}(16+m)(2i+1)\right\}$$

V(m)：周期加算信号
N(m, i)：係数
S(i)：32 サンプルのサブバンド入力

また、32 個の再生サンプルデータの計算は(6.5)式で計算される。

$$out(i) = \sum_{m=0}^{15} U(i+32m)\,D(i+32m), \quad 0 \leqq i \leqq 31 \tag{6,5}$$

但し、

out(i)：再生サンプル
U(k)：V(m) ($0 \leqq m \leqq 1023$) から生成されたデータ、$0 \leqq k \leqq 511$
D(k)：窓係数

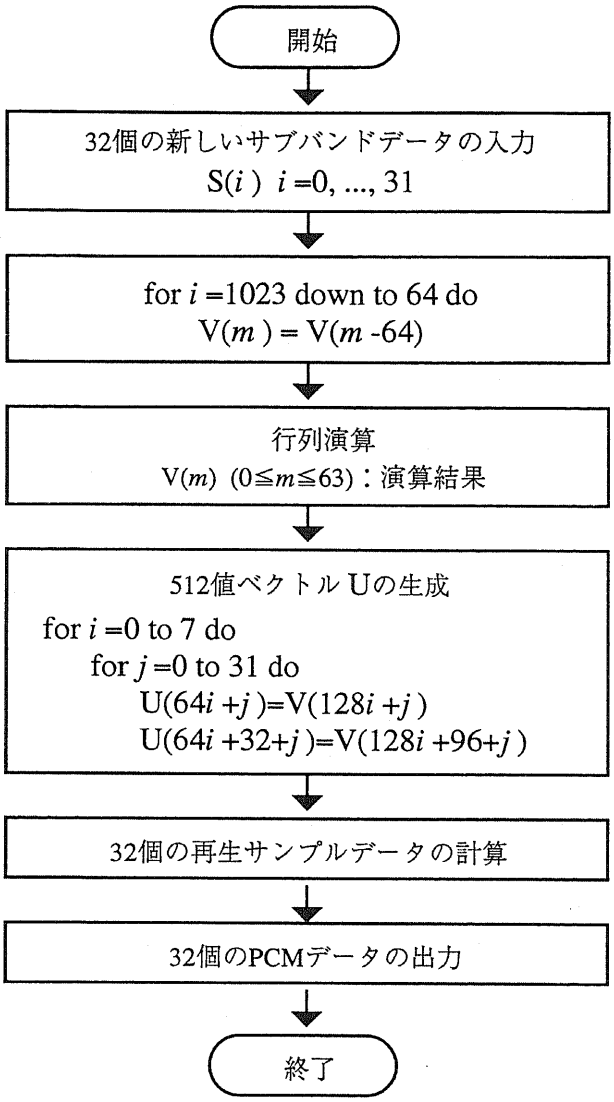


図 6.2　サブバンド合成フローチャート

6.3　マイクロコントローラと DRAM を用いた実現手法

6.2節で述べたアルゴリズムに工夫を加えて、今回使用したマイクロコントローラとDRAMを用いたシステムの特徴を活かしたMP3方式のオーディオ復号処理を実現し、更に評価システムを作成した。本節では、MP3復号処理の実現方法と評価システムについて述べる。

6.3.1 マイクロコントローラを用いた復号処理

MP3の復号処理では(6.1)式～(6.5)式に示すように多くの積和演算を必要とする。しかも、出力データの精度を20ビット程度に保つためには、16ビットを超えるデータを使った積和演算が必要である。しかし、組込み用途の32ビット・マイクロコントローラが、32×32ビットの乗算器を搭載していることは少ない。そのため、16ビットを超えるデータを使った積和演算は1命令で実現出来ず、(6.1)式～(6.5)式に従って復号処理を実現したのではリアルタイムでの復号は難しい。また、特別なアドレッシングモードを持たないマイクロコントローラとDRAMの組み合わせではデータロードのためのアドレス計算や、ランダムなメモリアクセスのオーバーヘッドが伴うという問題がある。そこで上記の問題を解決するために、

(1) 演算量を削減する

(2) データのメモリへの格納方法を工夫する

の2点に重点を置きMP3復号処理の高速化を試みた。

6.3.2 乗算サイクル数の削減

まず、復号処理の演算の多くは乗算または積和演算であることに着目して、乗算サイクル数の削減による復号処理全体の高速化を試みた。図6.3に今回採用した乗算の実現手法を示す。

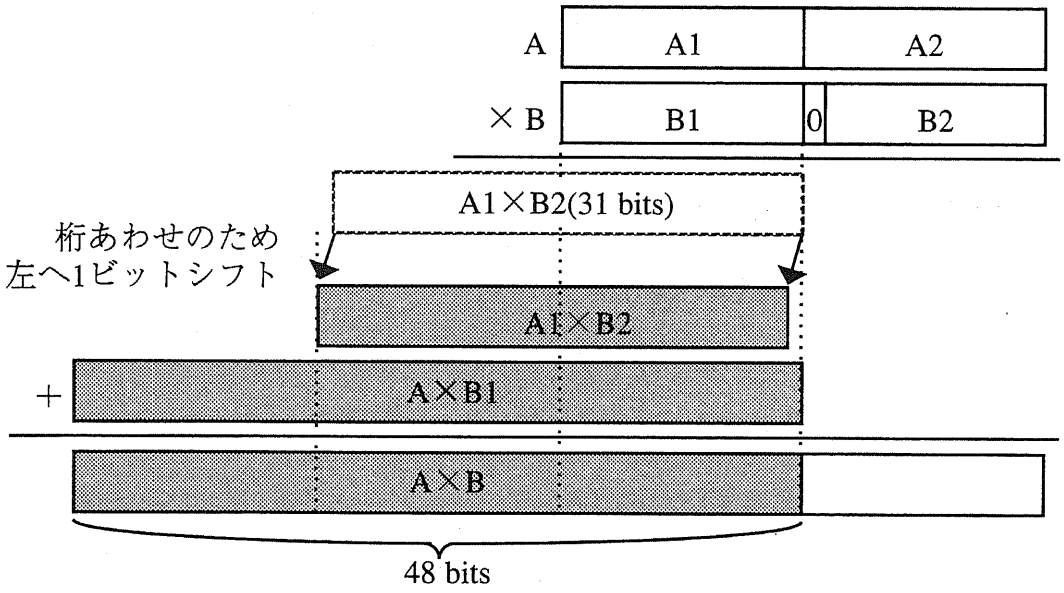


図6.3 32×31ビットの乗算で上位48ビットのみを計算する乗算方法

復号処理における出力データに必要な精度は20ビット程度であるため、計算途中の積和演算は32ビットを超える精度で実施する必要がある。今回使用するマイクロコントローラは32×16ビットの乗算器しか備えていないため、演算精度と処理時間の観

点から乗算の精度を工夫した。32×16ビットの乗算器を用いた積和演算の場合、32×32ビットの乗算は27ステップ、32×31ビットの乗算は8ステップ程度で実現できる。32×31ビットの乗算を使用してもデコード処理の演算精度には影響がないので32×31ビットの乗算を用いることにした。更に、32×31ビットで乗算を行う場合、計算途中でアキミュレータ内の値を退避させる必要がある。そこで32×16ビットの乗算で63ビットの精度が得られるところを、下位16ビットどうしの乗算を省略することで上位48ビットのみを計算し、乗算に必要なサイクル数を削減した。本手法を採用することにより、アキミュレータ内の値を退避させる必要もなくなる。また、退避させた値を格納しておくレジスタ等が不要になり、使用するレジスタ数も削減することができる。

6.3.3 IMDCT係数の対称性を利用した演算回数の削減

IMDCTの計算量を減らすために、(6.2)式で定義したIMDCTで用いる係数C(l, k)の性質を調べる。係数部分に着目すると、C(l, k)の定義より次式が導出できる。

$$
\begin{aligned}
C\left(\frac{N}{2}-l-1, k\right) &= cos\left(\frac{\pi}{2N}\left(2\left(\frac{N}{2}-l-1\right)+1+\frac{N}{2}\right)(2k+1)\right) \\
&= cos\left(\frac{\pi}{2N}\left(2N-(2l+1)-\frac{N}{2}\right)(2k+1)\right) \\
&= cos\left(\pi(2k+1)-\frac{\pi}{2N}\left((2l+1)+\frac{N}{2}\right)(2k+1)\right) \\
&= -cos\left(\frac{\pi}{2N}\left((2l+1)+\frac{N}{2}\right)(2k+1)\right) \\
&= -C(l, k), \qquad 0 \leqq l < \frac{N}{4} \\
C(N-l-1, k) &= cos\left(\frac{\pi}{2N}\left(2(N-l-1)+1+\frac{N}{2}\right)(2k+1)\right) \\
&= cos\left(\frac{\pi}{2N}\left(4N-\left(\left(2\left(\frac{N}{2}+l\right)+1\right)+\frac{N}{2}\right)\right)(2k+1)\right) \\
&= cos\left(2\pi-\left(\left(2\left(\frac{N}{2}+l\right)+1\right)+\frac{N}{2}\right)(2k+1)\right) \\
&= \cos\left(\left(\left(2\left(\frac{N}{2}+l\right)+1\right)+\frac{N}{2}\right)(2k+1)\right) \\
&= C\left(\frac{N}{2}+l, k\right), \quad 0 \leqq l < \frac{N}{4}
\end{aligned}
$$

つまり

$$
\begin{aligned}
&C\left(\frac{N}{2}-l-1, k\right) = -C(l, k) \\
&C(N-l-1, k) = C\left(\frac{N}{2}+1, k\right) \quad 0 \leqq l < \frac{N}{4}
\end{aligned}
\tag{6.6}
$$

が成立する。この(6.6)式を(6.2)式に適用することで、Z(*l*) は、

$$Z\left(\frac{N}{2}-l-1\right)=-Z(l)$$

$$Z(N-l-1)=Z\left(\frac{N}{2}+l\right) \qquad 0\leqq l<\frac{N}{4} \tag{6.7}$$

の関係が成り立ち、$0 \leqq l < N/4$、$N/2 \leqq l < 3N/4$ の範囲に関してのみ Z(*l*) を計算すれば残りの Z(*l*) は(6.7)式を用いることで簡単に求めるられる。つまりこの係数の対称性を利用し演算回数を半分にすることで MP3 復号処理の高速化が可能である。

例として、N=12、$k=0$ の場合の係数 C(*l*, 0) の値を図 6.4 に示す。この場合、(6.6)式は下のように記述できる。この係数の対称性を図に見ることができる。

$$C(5-l, 0) = -C(l, 0)$$
$$C(11-l, 0) = C(6+l, 0) \qquad 0\leqq l\leqq 2$$

従って、Z(0),Z(1),Z(2),Z(6),Z(7),Z(8) の6つの値を IMDCT により計算するだけですべての Z(*l*) が簡単に計算でき、演算回数がほぼ半分になることが分かる。

本節で述べた高速化手法を用いることによる1フレームあたりの乗算及び加算の回数の変化を表6.1 に示す。表6.1 より乗算および加算ともに回数が約半分になっていることが分かる。

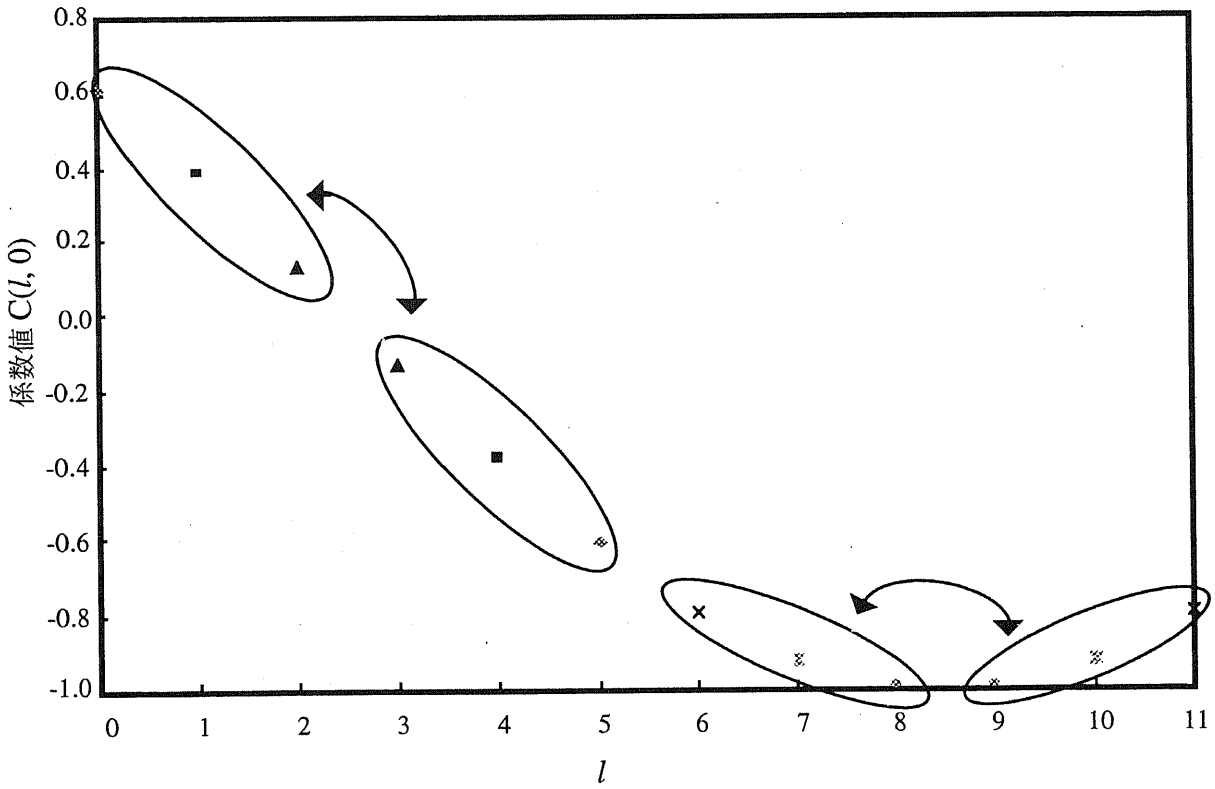


図 6.4　IMDCT 係数 C(*l*, *k*) (N=12, $k=0$ の場合)

表6.1 1フレームあたりの乗算及び加算回数の変化

	基本アルゴリズムに従った場合	提案した高速化手法を採用した場合
乗算回数	87552回	46080回
加算回数	82944回	41472回

6.3.4窓係数の配置の最適化

IMDCTに続いて行われる窓掛けに関しても、1回のIMDCTの計算結果から2つ分の窓掛け結果を計算することができる。DRAMを使用する場合、メモリアクセスをランダムではなく連続的にすることが処理速度の高速化につながる。そこで、上述のIMDCT演算の高速化にあわせ図6.5に示すように連続的にデータを読み出すことが出来るよう窓係数を格納した。同時に、符号反転も考慮し格納した。

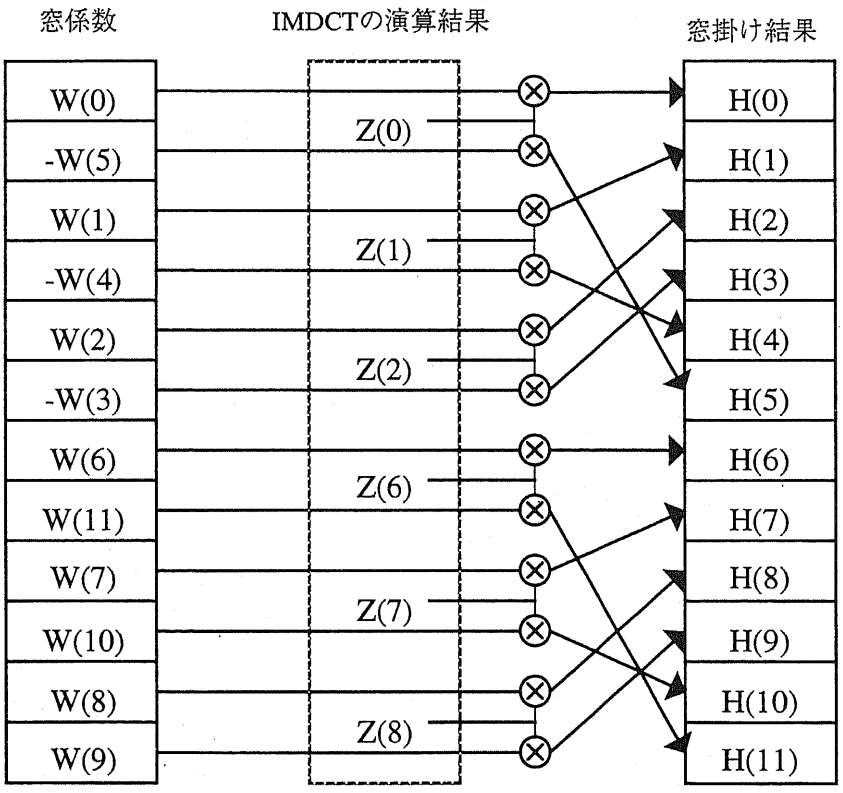


図6.5 窓係数の配置（N=12の場合）

6.3.5 サブバンド合成におけるデータ配置の最適化

6.1節でも述べたように、特別なアドレッシングモードを持たないマイクロコントローラとDRAMの組み合わせで(6.5)式の積和演算を行う場合、データロードのためのアドレス計算が必要となり、またランダムなメモリアクセスのオーバーヘッドが伴う。そのため、(6.5)式の積和演算を効率よく実行することができない。そこでこれらの問題を解決するために、データをメモリに格納する方法を工夫する。つまり(6.5)式が(6.8)式で計算できるように、V(m)の生成時にデータの格納順序を変える。

$$\mathrm{out}(i) = \sum_{m=0}^{15} \mathrm{V}(32i+m)\,\mathrm{D}(32i+m), \quad 0 \leqq i \leqq 31 \tag{6.8}$$

行列演算については、ソフトウェアで実現する場合、積和演算を32点DCTで実現し、高速アルゴリズムを使用する手法がある[5]。従って32点DCTの結果を格納するときにデータの格納順序を変える。

このメモリ格納方法により、連続的にデータを読み出しながら積和演算を効率的に実行することができる。特に、オートインクリメント機能付きロード命令をサポートするマイクロコントローラの場合は、この命令を使用することでアドレス計算を行うことなく連続的に読み出しができる。また、図6.2のベクトルUを生成せずにV(m)（$0 \leqq m \leqq 1023$）を使って(6.5)式を計算できるため、メモリとのロードおよびストア、それに伴なうアドレス計算を削減することができる。

6.3.6 ハフマン復号値の特性を利用した高速化手法

MP3復号処理で行われる演算の大半は積和演算である。そこで値が零のデータに対する積和は省略できることを利用して処理全体の演算量削減を図った。

2.1節よりハフマン復号値から周波数領域サンプルを計算する逆量子化処理は(6.1)式で表された。ここで、ハフマン復号値について考える。MP3の符号化では、人間が聞きやすい帯域の信号劣化を最小にするようなアルゴリズムが用いられている。言い換えれば、人間が聞くことのできる音のみを符号化し、特に高域成分のような聞こえない音の成分に関してはデータが削除されている。よってハフマン復号処理では、符号化されている部分のみをハフマンテーブルによって復号し、符号化されていない部分の復号値は零とおいている。このことから、高域成分に対応するハフマン復号値の多くは零であることが分かる。

そこで、このハフマン復号値の特性を利用した高速化手法を提案する。(6.1)式より、ハフマン復号値が零の場合は周波数領域サンプルが零となり、(6.1)式の計算が不要になることが分かる。つまり、ハフマン復号値の値に応じて(6.1)式の計算を選択的に実行できる。さらに、ハフマン復号値が零となっている符号化されていない高域成分の領域を検出することにより、逆量子化処理に続くエイリアシング削減のためのバタフライ演算やIMDCTを省略することができる。これにより、省略部分の処理に必要な演算量を削減でき、復号処理の高速化につながる。

6.4 評価結果

これらの提案した手法を実際に1.2.3節で述べたDRAM内蔵32ビット・マイクロコントローラを用いて実際にその効果を確認した。テストデータは文献6から入手したデータのうち、サンプリング周波数44.1kHzのジョイントステレオのもの3種類を用いた。

6.4.1 演算精度

3.2項で述べた乗算の方法で復号処理を行った結果に対する精度検証を行った。浮動小数点演算で復号処理を行った結果をもとに精度検証を行った結果、20ビットを超える精度を確認した。

6.4.2 処理速度

処理時間の測定結果を図6.5に示す。図6.5は、1フレームあたりの平均処理時間を示したもので、前節に述べたの高速化手法も併用した結果を表している。

従来手法を用いて実現したプログラムに対し、提案した4つの手法を適用することで約2倍の高速化が実現できた。その結果、1フレームあたりの復号処理時間が23ms程度となった。サンプリング周波数が44.1kHzの場合、約26ms/フレームで処理する必要があるため、汎用のマイクロコントローラでもリアルタイムの復号処理が可能となった。

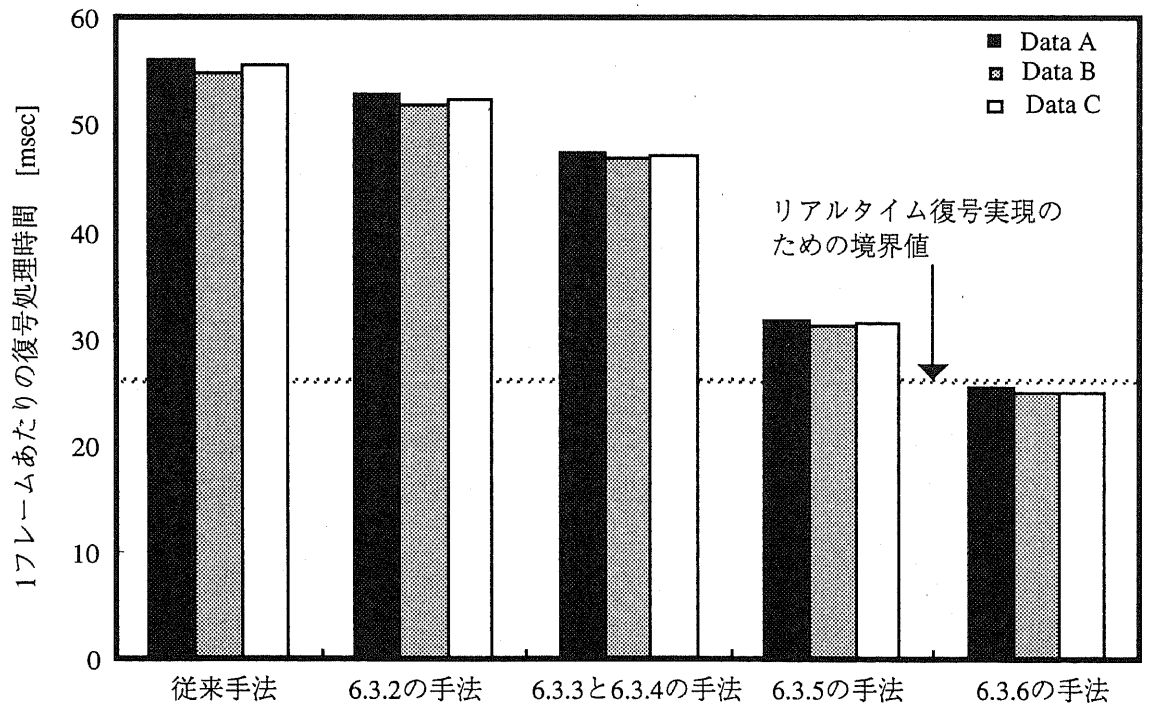


図6.5 処理時間の測定結果

6.5 まとめ

オーディオデータを従来よりも高い割合で圧縮し、しかも従来と同程度の再生時の音質を実現するためには、多くのフィルタ処理が必要となり、その実現には膨大な処理量が必要である。また、従来と同程度の音質を実現するためには、16ビット以上のデータを用い、最終的に16ビット以上の演算精度が必要である。

上述のようなオーディオデータ復号処理をデータサイズの小さな乗算器しか搭載しないマイクロコントローラを用いて実現するためには、乗算のサイズと演算精度を工夫した演算アルゴリズムの開発とフィルタ処理の際のデータや係数を格納するメモリの使用方法を工夫した高速化手法の検討という課題がある。そこで、組込み用32ビット・マイクロコントローラとDRAMを使ったMP3デコーダの実現を試みた。

本章では、先ず、MP3復号アルゴリズムについて述べ、復号アルゴリズムを実現する演算の計算式やデータの格納方式をマイクロコントローラに適したものすることを検討した。次に、検討したマイクロコントローラに適した演算の計算式やデータの格納方式を組込み型の32ビット・マイクロコントローラ上で効率的に実現する手法を検討した。そして、実際に組込み型の32ビット・マイクロコントローラを用いた実験システムを開発し、このシステムを用いて本方式の有効性を検証した。

その評価の結果、演算精度に関しては、浮動小数点演算で復号処理を行った結果をもとに精度検証を行い、20ビットを超える精度を実現できたことを確認した。また、処理速度に関しては、従来手法を用いて実現した場合に対し、提案した4つの手法を適用することで約2倍の高速化が実現できた。そして、1フレームあたりの復号処理時間が23ms程度となり、汎用のマイクロコントローラでもリアルタイムの復号処理が可能となった。

参考文献

1) ISO/IED 11172-3; "Information technology - Coding of moving pictures and associated audio for digital storage media at up to 1.5 Mbits/s. Part3-Audio Consumer Electronics", Vol.43, No.1, pp.40-47, February, (1997)

2) T. Tsai, T. Chen, and L. Chen; "An MPEG audio decoder Chip", IEEE Transactions on Consumer Electronics, Vol. 41, No.1, pp. 89-95, February, (1995)

3) L. Bergher, X. Figari, F. Frederiksen, M. Froidevaux, J. M. Gentit, and 0. Queinnec; "MPEG audio-decoder for consumer applications" CICC, pp. 413-416, (1995)

4) C. D. Murphy and K. Anandakrmer; "Real-time MPEG-1 Audio Coding and Decoding on a DSP Chip" IEEE Transactions on Consumer Electronics, Vol. 43, No, 1, pp. 40-47, F bruAry, (1997)

5) T. Sakamoto, M. Taruki and T. Hase; "A Fast MPEG-Audio Layer III algorithm for a 32-bit MCU", IEEE Transactions on Consumer Electronics, Vol. 45, No. 3, pp. 986-993, (1999)

6) MP3 Test Bitstreams at http://www.mpeg.org/index.html/mp3.html

第7章　結論

本論文の主要な目的は、マイクロコントローラの持つ制約から生じる画像・音響処理実現上の問題を、製品仕様から特定できる境界条件を利用しながら、解決するための画像・音響処理技術手法を示すことにある。本論文では、従来の画像・音響処理技術をDRAM内蔵マイクロコントローラに適用する際に特有の画像・音響処理技術の開発手法を示し、それを次に示す4つの分野に適用することで、従来の処理技術がDRAM内蔵マイクロコントローラを用いて実用になることを示した。

・デジタル・スチル・カメラ用撮像信号処理
・静止画像処理JPEG
・3次元グラフィックス処理
・オーディオ復号処理

本章では、第2章から第6章までに得られた主な研究成果を要約する。

(1)　マイクロコントローラ特有の画像・音響処理技術の開発手法の確立

第2章では、従来の画像・音響処理技術をマイクロコントローラを搭載した製品群に適用する上での問題点を、搭載されたハードウェア資源の制約と製品仕様から決まる処理の仕様に関する限定から、総合的に解析した。そして、ハードウェア資源の制約を受けながらも、画質や音質あるいは処理速度の面で実用的な範囲を実現するための手段や方法を検討した。特に、DRAM内蔵マイクロコントローラの持つハードウェア資源の活用、DRAMの特性を活かしたメモリ格納方法や人間の視聴覚モデルを用いた画像・音響処理の高速化等について検討した。その結果以下のような手段や方法を提案した。

(1) 定数のテーブル化
(2) 固定小数点化
(3) 乗算の実現方法と演算精度の検討
(4) 算術演算の精度検討
(5) 人間の視聴覚特性を考慮した演算精度の検討
(6) メモリ・アクセスを考慮した高速化手法

これらの手段や方法は、画像・音響処理全般に有効で、マイクロコントローラを対象とした画像・音響処理技術を開発するための一般的な開発手法として適用できる。これらの手法を実際の画像・音響処理に適用し、最大限の効果を得るためには、これらを基本的な手法として、適用の仕方を個別に工夫する必要がある。これに関しては、以降で具体的に述べた。

(2)　デジタル・スチル・カメラ用撮像信号処理への適用

第3章では、デジタル・スチル・カメラ専用のCCDイメージセンサの画素構造として多く用いられているベイヤー型原色方式に適用する補間方式について考察した。画素補間処理では、補間対象色と同一色の画素だけを用いて補間処理を行うのが一般的である。また近年の研究では、特に色信号の高解像度化に関する試みがなされている。例えば、色

の相関を算出するため、局所的な色の相関を利用し画素信号の補間処理を行う方式や、水平方向や2次元のLPFを用いる方式などが報告されている。そこで、先ず補間対象色と同一色の画素だけを用いて処理を行う画素補間方式について検討し、次に色の相関を用いて画素補間方式の改良を検討した。

先ず、同一色を用いた画素補間方式のうち代表的な8つの補間方式について検討を加えた。そして、画素補間により得られる再生画像の特徴と組込み用DRAM内蔵32ビット・マイクロコントローラを用いて、補間を実現するための演算量と、その結果得られる画質について考察した。

検討の結果、各補間方式の善し悪しは、解像度が高いかどうかよりも、高周波領域における偽色の発生が重要な要因になることがわかった。そこで、偽色が少なく、色の再現性がよいという観点では、相関を考慮した線形補間法が良いことがわかった。

次に画質と演算量を定量的に比較するために、実際にマイクロコントローラの演算量とS/N比の関係を調べた。演算量に関しては、VGAサイズの画像の補間処理を、動作周波数66MHzのDRAM内蔵マイクロコントローラで実行したところ何れの方式も0.2秒程度で実現できた。その結果、S/N比の観点においても線形補間方式が、演算量の割には良い画質が得られ、ディジタル・スチル・カメラのような民生用途では適していると考えられる。

次に、この結果を更に押し進め、色の相関を用いた画素補間方式をベイヤー型原色方式のCCDに適応したときの再生画像の特徴を、解像度、色の再現性の観点から調べた。

色の再現性に関してCZPを使って解析した結果、G信号については各色成分の周波数成分の比率が等しいと仮定した場合の再現性が良く、RおよびB信号については色信号間の関係として比の変化が少ないと仮定した場合の再現性が良いことがわかった。

また演算量とS/N比の関係について、CZPとSCID中の5種類の自然画を使って解析した結果、G信号について各色成分の変化量が等しいと仮定し、RおよびB信号については色信号間の関係として差の変化が少ないと仮定した場合の補間方式が演算量に対するS/N比が良いことがわかった。

(3) 静止画像処理JPEGへの適用

第4章では、第2章の議論を基に、静止画像処理JPEGをDRAM内蔵マイクロコントローラで実現する場合の高速化手法について述べた。

まず最初に、JPEGベースライン・システムの処理内容を分析し、全体の処理性能に大きな影響を与える処理を示した。そして、それらの処理内容からメモリ使用法の観点から高速化すべきものと、アルゴリズムの観点から高速化すべきものに分けた。

次に、メモリ使用法の観点から高速化すべき処理に対して、メモリ・アクセスを考慮した高速化手法について述べた。具体的には、MCUのデータに対するロードおよびストアの削減、ハフマン符号化時のジグザグ走査の削減、パックされた2個のデータのロード、そしてパックされた2個のデータの零判定に関する手法を述べた。そして、これらの手法を用いたJPEGプログラムを開発し、従来手法によるプログラムとの速度性能を評価した。その結果、提案した手法を採用することで2倍以上の処理性能が実現できることがわかった。

さらに、アルゴリズムの観点から高速化すべきDCT演算に対して、マイクロコントローラのアーキテクチャを活用した高速化手法について述べた。従来からのマイクロコントローラの場合、並列実行機能を持たない。この従来型の、乗算器を搭載し積和命令を持つマイクロコントローラの場合、積和演算を用いるChenのアルゴリズムを用いると効率的にDCT演算を実現できることがわかった。また一方で、最近のマイクロコントローラに取り入れられてきた並列実行機能を更に仮定すると、並列実行を効率的に活用できるAraiのアルゴリズムを用いると効率的にDCT演算を実現できることがわかった。これらの結果を用いたJPEGプログラムを開発し、速度性能を評価した。その結果、並列実行機能を採用することで、DCT演算は約40%、JPEG処理全体では約35%高速化できることが判明した。

このようにして、マイクロコントローラを用いたシステムに適した処理手法を用いることで、デジタル・スチル・カメラでよく用いられるVGAサイズの画像（サンプリング比 Y:Cb:Cr=4:2:2）の圧縮が、0.4秒（並列実行機能を持たない従来型のマイクロコントローラの場合、動作周波数66.6MHz）あるいは0.15秒（並列実行機能を持つのマイクロコントローラの場合、動作周波数100MHz）程度で実現できることが判明した。これにより、マイクロコントローラを用いてソフトウェアでJPEG方式による画像圧縮が実用的に実現できることを示すことができた。

(4) 3次元グラフィックス処理への適用

第5章では、32ビット×16ビットの乗算器を持った組込み用DRAM内蔵マイクロコントローラを用いて、たとえFPUや特別な3次元グラフィックス・エンジンを持たなくても、3次元グラフィックス機能をソフトウェアで実現する方法について提案し、実験機により評価した。

まず、3次元グラフィックスの全体の処理量を見積もった。その結果、高速化のためには、全体の処理時間のほとんどを占めるレンダリング処理を高速化する必要があることがわかった。さらに詳細に調べると、このレンダリング処理の約93%を占める浮動小数点演算を高速化することが、一番高速化のためには効果が大きいことがわかった。

そこで、浮動小数点演算処理を整数演算処理に置き換えたり、あるいは、RICSアーキテクチャを採用したマイクロコントローラにおいて多くの演算サイクルがかかる除算演算を削減することを検討した。この目的のために第5章では、小数点位置の工夫と、適応的な乗算演算を用いた固定小数点と、除算回数の削減の3つを提案した。

(1) 1/4 VGAの解像度で実用上問題のない座標値（X, Y, Z）の固定小数点表現を採用した。

(2) 処理内容を吟味し、それぞれの演算で必要な精度に応じて3種類の乗算ビット数を選択する。

(3) 実行サイクル数が多い除算演算の回数を減らすために、まず逆数を求め、求めた逆数との乗算処理に変更する。

次に、実際に実験機を作り、その上に、上述の提案をソフトウェアで実現して、その機能および性能の評価を行なった。

従来のソフトウェアパッケージを用いて浮動小数点演算部分を処理した場合は、およ

そ描画性能は100ポリゴン/秒であった。しかし、第5章で提案した固定小数点の手段を用いることにより、従来と比べて約13.8倍の描画性能を高速化することができた。この固定小数点の工夫に加えて、上述の提案した除算の削減を施すことにより、さらに約7.4倍高速化することができた。従って、これらの工夫により描画性能は10,000ポリゴン/秒が得られ、もとの浮動小数点演算部分においてソフトウェア・パッケージを用いて処理をした場合と比べて、およそ100倍高速化を実現することができた。

また、レンダリング処理を高速にするため視覚上問題のない範囲で、画質の劣化を許容することにした。これを前提として、各画素ごとにレンダリング処理を用いて陰面処理と色調計算を行うのではなく、レンダリング処理と画素補間処理を並用し、レンダリング処理の回数を減らす手法を提案した。つまり、レンダリング処理を行う画素を減らすことで全体の処理の高速化を図ろうというものである。レンダリング処理を行わない画素では、周囲の画素値からZバッファ法で求めたZ値を参照しながら補間処理で値を決定する。

この手法を評価した結果、提案した手法を用いてもS/N比から判断して画質の大きな劣化は見られないことがわかった。また、モデルを構成するポリゴンが大きい程、今回提案した手法の効果は大きく、約5倍の高速化が実現できたことがわかる。また、提案した手法はテクスチャーマップにも効果があり、大きなテクスチャーを付加する場合は、本手法は、テクスチャーマップの高速化手法として有効である。

(5) オーディオ復号処理への適用

第6章では、第2章の議論を基に、MPEGオーディオ・レイヤー3に基づくオーディオ復号処理をDRAM内蔵マイクロコントローラで実現する場合の高速化手法について述べた。

オーディオデータを従来よりも高い割合で圧縮し、しかも従来と同程度の再生時の音質を実現するためには、多くのフィルタ処理が必要となり、その実現には膨大な処理量が必要である。また、従来と同程度の音質を実現するためには、16ビット以上のデータを用い、最終的に16ビット以上の演算精度が必要である。

上述のようなオーディオデータ復号処理をデータサイズの小さな乗算器しか搭載しないマイクロコントローラを用いて実現するためには、乗算のサイズと演算精度を工夫した演算アルゴリズムの開発とフィルタ処理の際のデータや係数を格納するメモリの使用方法を工夫した高速化手法の検討という課題がある。そこで、組込み用DRAM内蔵32ビット・マイクロコントローラを使ったMP3デコーダの実現を試みた。

第6章では、先ず、MP3復号アルゴリズムについて述べ、復号アルゴリズムを実現する演算の計算式やデータの格納方式をマイクロコントローラに適したものすることを検討した。次に、検討したマイクロコントローラに適した演算の計算式やデータの格納方式を組込み型の32ビット・マイクロコントローラ上で効率的に実現する手法を検討した。そして、実際に組込み用DRAM内蔵32ビット・マイクロコントローラを用いた実験システムを開発し、このシステムを用いて本方式の有効性を検証した。

その評価の結果、演算精度に関しては、浮動小数点演算で復号処理を行った結果をもとに精度検証を行い、20ビットを超える精度を実現できたことを確認した。また、処理

速度に関しては、従来手法を用いて実現した場合に対し、提案した4つの手法を適用することで約2倍の高速化が実現できた。そして、1フレームあたりの復号処理時間が23ms程度となった。リアルタイムの復号処理には、1フレーム当たり26ms以下の時間で処理する必要があることから、汎用のマイクロコントローラでもリアルタイムの復号処理が可能となった。

以上に述べたように、本研究により、DRAM内蔵マイクロコントローラの持つ制約から生じる画像・音響処理実現上の問題を、製品仕様から特定できる境界条件を利用しながら、解決するための画像・音響処理技術の開発手法を確立した。そして、デジタル・スチル・カメラ用撮像信号処理、静止画像処理、3次元グラフィックス処理、オーディオ復号処理の4つの分野で、その開発手法に基づき従来の画像・音響処理技術を実用的にマイクロコントローラ適用できることを示した。

本論文に関する関連発表論文

1. 論文

(1) T. Sakamoto and T. Hase;
"JPEG Software Solution for a 32-bit MCU",
IEEE Transactions on Consumer Electronics, Vol. 43, No. 3, pp. 410-417, (1997)

(2) T. Okabe, T. Sakamoto and T. Hase;
"Full Audio Software Solution for a 16-bit DSP Corefor Digital Audio Decoder LSI",
IEEE Transactions on Consumer Electronics, Vol. 44, No. 1, pp40-47, (1998)

(3) K. Yoshida, T. Sakamoto and T. Hase;
"A 3D Graphics library for a 32-bit microprocessors for embedded systems",
IEEE Transactions on Consumer Electronics, Vol. 44, No. 3, pp. 1107-1114, (1998)

(4) Y. Ishikawa, Y. Kisuki, T. Sakamoto and T. Hase;
"Speech Synthesis Software for a 32-bit Microprocessor",
IEEE Transactions on Consumer Electronics, Vol. 44, No. 3, pp. 1173-1182, (1998)

(5) T. Sakamoto and T. Hase;
"Software JPEG for a 32-bit MCU with Dual Issue",
IEEE Transactions on Consumer Electronics, Vol. 44, No. 4, pp. 1334-1341, (1998)

(6) T. Sakamoto, C. Nakanishi and T. Hase;
"Software Pixel Interpolation for Digital Still Camera for a 32-bit MCU",
IEEE Transactions on Consumer Electronics, Vol. 43, No. 4, pp. 1342-1352, (1998)

(7) 中西 , 坂本 , 長谷 ;
"組込み形マイクロプロセッサに適した画素補間方式",
映像情報メディア学会誌 , Vol. 53, No. 4, pp.141-149, (1999)

(8) 浅井 , 坂本 , 長谷 ;
"Software Solution for GPS baseband Processing Using a 32-bit Embedded Microprocessor"
映像情報メディア学会誌 , Vol. 53, No. 5, pp.738 - 745, (1999)

(9) 坂本 , 森田 , 佐藤 , 長谷 ;
"組込み用マイクロコントローラを用いたPHSデータ通信機能の全ソフトウェア化"
映像情報メディア学会誌 , Vol. 53, No. 6, pp.901-904, (1999)

(10) T. Sakamoto, M. Taruki and T. Hase;
"A Fast MPEG-Audio Layer III algorithm for a 32-bit MCU",
IEEE Transactions on Consumer Electronics, Vol. 45, No. 3, pp. 986-993, (1999)

(11) Y. Ishikawa, Y. Kisuki, T. Sakamoto and T. Hase;
"Speech Synthesis Method based on Application-Specific Units and its Implementation on a 32-bit Microprocessor",
IEEE Transactions on Consumer Electronics, Vol. 45, No. 3, pp. 980-985, (1999)

(12) 郭, 浅井, 樽木, 坂本, 長谷;
"FPU を持たない 32bit MCU 用 GPS ナビゲーションプログラムの高速化",
映像情報メディア学会誌, , 1999 年 10 月号に掲載予定

(13) 浅井、影本、坂本、長谷, "32 ビット組込み型マイクロプロセッサを用いた GPS 衛星捕捉処理の高速化",
電気情報通信学会論文誌, 1999 年 11 月号に掲載予定

2. 国際会議発表

(1) T. Sakamoto and T. Hase;
"JPEG Software Implementation Techniques Based on a 32-bit RISC CPU",
IEEE International Conference on Consumer Electronics, Chicago, (1997)

(2) T. Asai, T. Sakamoto, and T. Hase;
"Software Solution of GPS Baseband Processing",
IEEE International Conference on Consumer Electronics, Los Angeles, (1998)

(3) K. Yoshida, T. Sakamoto, and T. Hase;
"A 3D Graphics Library for Embedded Systems",
IEEE International Conference on Consumer Electronics, Los Angeles, (1998)

(4) Y. Ishikawa, Y. Kisuki, T. Sakamoto and T. Hase;
"Speech Synthesis Software for a 32-bit Microprocessor",
IEEE International Conference on Consumer Electronics, Los Angeles, (1998)

(5) M. Taruki, T. Sakamoto and T. Hase;
"A Fast MPEG-Audio Layer III algorithm for a 32-bit MCU",
IEEE International Conference on Consumer Electronics, Los Angeles, (1999)

(6) Y. Ishikawa, Y. Kisuki, T. Sakamoto and T. Hase;
"Speech Synthesis Method based on Application Specific Synthetic Units and its implementation on a 32-bit Microprocessor",
IEEE International Conference on Consumer Electronics, Los Angeles, (1999)

謝辞

多くの方々のご指導ご鞭撻により本研究を纏めることができたことを感謝いたします。特に、静岡大学工学部 下平美文教授には本研究の纏めに際し卓越したご指導と本論文審査に当たり数々のご尽力を賜り心から謝意を表します。

また、本研究を纏めるに当たり、終始有益なご指導と、ご審査の労を賜りました静岡大学電子科学研究科長 畑中義式教授、静岡大学情報学部 中谷広正教授、静岡大学工学部 江上俊一郎教授、静岡大学電子工学研究所 川人祥二教授に深く感謝致します。

本研究は、三菱電機システムLSI開発研究所およびシステムLSI事業化推進センターにおいて、各方面の方々のご協力のもとに行われた筆者の研究を纏めたものです。本研究の機会を与えて戴くとともに本研究の遂行途上において激励くださいました同社 システムLSI開発研究所 部長 岩出秀平博士および元部長 斎藤和則博士に深く感謝致します。

三菱電機システムLSI開発研究所およびシステムLSI事業化推進センターで、本研究の討議および遂行にご協力戴きました、長谷智弘博士、吉田可奈子主事、木透康久主事、中西知嘉子主事、浅井敬氏、影本哲哉氏、池田雅彦氏、佐藤剛氏、樽木麻衣子氏に感謝致します。さらに、本研究の遂行にご協力戴きました、同社 情報技術総合研究所 石川泰専任、菱光コンピュータシステム株式会社 森田重主事および関係者各位に感謝致します。また、本研究を遂行するに当たり温かいご配慮を戴きました三菱電機システムLSI事業統括部 システムLSI第一部部長 清水徹博士およびシステムLSI第一部の 関係者各位に感謝致します。

本研究は、三菱電機システムLSI開発研究所、システムLSI事業化推進センターおよびシステムLSI事業統括部システムLSI第一部の関係者各位をはじめ多くの方々のご指導、ご理解によってなし得たものであり、重ね重ね心から感謝の意を表します。最後に、本研究を陰から応援してくれた妻 左斗子、父母および義父母に感謝致します。