修士論文

題目

動画像符号化における
重み付け動き補償の高効率化に関する
研究

指導教員

近藤 利夫 教授

2009年度

三重大学大学院 工学研究科 情報工学専攻
計算機アーキテクチャ研究室

堀田 勉 (408M521)

内容梗概

近年, ネットワークのブロードバンド化が著しい中, 通信と放送の両分野から高圧縮高精細の動画像符号化方式が求められ,2003 年 ITU-T により H.264/AVC が標準化された.

この H.264/AVC はこれまでの主流の符号化方式である MPEG-2 に比べ 2 倍以上の圧縮率が得られるものの圧縮率向上のために新たに導入, 拡張された符号化技術によって膨大な演算量が必要になっている. 重み付け動き補償はフレーム間で輝度が一様に変化するフェード画像について画質改善に成功しているものの, 照明によって輝度値が局所的に激しく変化する歌謡ショーのような画像では未だ十分な圧縮効率が得られていない.

本研究ではこのような画像に対する局所ごとの最適な重み付け動き補償による符号化効率の改善を最小限の演算量で実現することを目的とする.

本論文では重み付け動き補償の高度化に必要と考えられる輝度変化の影響を受けにくい動き探索法と輝度変化領域を考慮した領域分割法の 2 つを提案し, それらを H.264/AVC の参照ソフトウェアに実装し, 効果を評価したが, 優位な向上は見られなかった. 主原因は, 切り分け後の領域内に残る輝度変化による動き検出の乱れが押さえられていないことと考えられ, 今後はその解消法について検討していく必要がある.

Abstract

Recently, the remarkable development of broadband network leads to high-compression high-resolution digital video codec required from both telecommunication and broadcast fields. As a result, H.264/AVC is standardized by ITU-T in 2003.

H.264/AVC can provide more than twice of the compression efficiency obtained by the conventional coding, MPEG-2 , but it is also a drawback of large computational load, by introducing extended coding technology for compressibility improvement.

Weighted prediction makes the image quality of fade scene, which brightness is varying uniformly between the frames, fine, but compression efficiency is not yet fine in some sequences, one of which is varying suddenly and locally by the lighting of song and ballads show.

The purpose of this study is to achieve the improvement of coding efficiency by local optimal weighted prediction to such images with minimul amount of conputing.

In this paper, we propose 2 method, high-robustness to brightness change motion detecion method and to seperate the frame with regard for brightness change, we implemented them in H.264 reference software, and evaluated an effect, but significant improvement was not obtained.

目 次

図 目 次

表 目 次

1　序論

1.1　背景

近年の地上デジタル放送の開始や映像コンテンツ配信サービスの普及によって通信・放送に適した高圧縮高精細の動画像符号化規格が必要不可欠となり, 2003 年には ITU-T によって動画像符号化規格 H.264/AVC が標準化された.
H.264/AVC の特徴にはデブロッキングフィルタや重み付け動き補償, エントロピー符号化, 任意スライス順序などがある. これらの技術と従来技術の高度化あるいは相互の組み合わせによって, 伝送に必須となるより強いエラー耐性やこれまで主流であった MPEG-2 のおおよそ 2 倍以上の圧縮率を実現しており, すでにワンセグや携帯ゲーム機, ブルーレイディスクにおける動画像の圧縮といった近年の映像関連の分野で幅広い活躍をしている.
しかし, 従来の符号化技術の集大成であるこの規格でも, 未だ十分な圧縮効率が得られないままとなっている画像も存在する.

1.2　目標とその意義

先に述べたような圧縮効率が十分でない画像のうち本研究で注目するのは図 1.1 のような短時間で局所的に輝度変化が起こる画像である.
例えば比較的身近な歌謡ショーでは舞台照明が, そのような輝度変化を引き起こす.
図 1.1 に示す画像は中央のやや左側のあたりで照明により, 輝度値が局所ごとに変化している. このような場合は輝度変化領域において誤った動きベクトルを検出してしまい, 結果として動画像の圧縮効率が低下してしまう. そこで本研究では H.264/AVC から採用された重み付け動き補償方式を利用し, 複数箇所で発生する輝度変化による予測誤差を最小限に抑えることによって画像の圧縮効率を改善することを最終目標とする. そのために必要となるのが一定以上の領域における輝度値の変化の影響を受けにくい動きベクトルの検出方法と重み付け動き補償の効果を最大限に発揮させるための, 輝度変化の領域を適切に切り分ける領域の分割方法である. この論文では両者の手法を提案し, その実装と評価を行った.

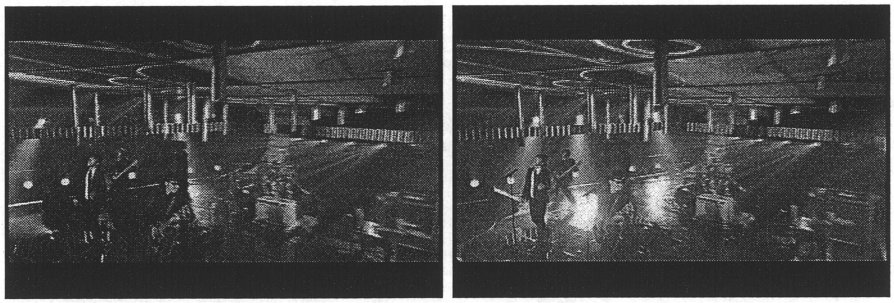

図 1.1: 複数の輝度変化領域を含む画像

2　関連技術と従来手法

2.1　動き補償

2.1.1　動き補償の重要性

動画像は複数のフレームが連続して表示されることによって成り立つ. そして連続するフレームはそれぞれが類似しているため, 動画像の圧縮の基本として連続する画像については差分の情報で表現する. 動き補償は連続するフレーム間における差分データのうち, 画像内のオブジェクト等の動きによって発生する情報をベクトルという形でデータ化し補うことで差分データを削減し, 符号発生量を低減する手法のことである. 近年の高圧縮率の動画像符号化方式の圧縮率の大幅な向上はこの動き補償の寄与が大きい. H.264/AVC では 1/4 画素精度までの動き補償を行うことでより詳細な動き情報を利用しており, 参照画像についても同一方向複数参照を許している. なお, 動き補償に必要となる動きベクトルを検出ための処理を動き探索と呼ぶ.

2.1.2　動き探索

動き探索によるベクトルの検出処理は演算量が非常に大きく膨大であり動画像符号化処理における最大のボトルネックとなっている. しかしこの処理は同時に動画像圧縮全体をを左右する最も重要な処理でもある. 従って, いかに演算量を少なくかつ, 正確にベクトルを求めるかが動画像符号化における最も重要な課題の１つといえる. この処理は画素の集合であるマクロブロック単位で行う.MPEG-2 ではこのマクロブロックサイ

ズが 16×16 で固定され,MPEG-4 においては 16×16, 8×8 の 2 種類のいずれかで行われる.
H.264/AVC ではこのブロックサイズが 16×16, 16×8, 8×16, 8×8, 8×4, 4×8, 4×4 の 7 種類から選択して符号化を行うことも可能となった. しかし, こういった符号化技術を用いることにより圧縮率は向上するものの元々膨大であった演算処理量が更に膨大になるという結果をもたらしている.

2.1.3 SAD(Sum of Absolute Difference)

SAD(Sum of Absolute Difference) とは差分絶対値和のことであり, 動き探索の際にブロックが移動した箇所かどうかを判別するために利用する. この処理はブロック同士の各画素値の差分をとり, それらの差分それぞれの絶対値を加算した総和の値 (式 1) がもっとも小さくなる箇所と元の場所との位置関係を動きベクトルとして検出する. 動き探索の中でもこの処理が大きなウェイトを占めており, 探索範囲やブロックのサイズによってその演算量は変動する.

$$SAD(B, v) = \sum_{r \in B} |I_{curr}(r) - I_{ref}(r + v)| \quad (1)$$

B : 符号化対象ブロック
v : 候補ベクトル
$I_{curr}(r)$: 符号化対象画像上で位置に存在する画素値
$I_{ref}(r)$: 参照画像上で位置に存在する画素値

2.2 重み付け動き補償

動き補償ではベクトルデータを考慮しない画素値の差分データに含まれるべき画像内の動き情報を動きベクトルという形で補うことでで符号発生量の削減を行うが, 重み付け動き補償では動きベクトルだけではなく重み係数とオフセット値からなる重み付けパラメータを下記のような数式に当てはめることでフレーム間の輝度値の変化の補償を行い, 更なる符号発生量削減を行う.

$WY' + D$ (Y' : 補償を行う信号, W : 重み係数, D : オフセット値) (2)

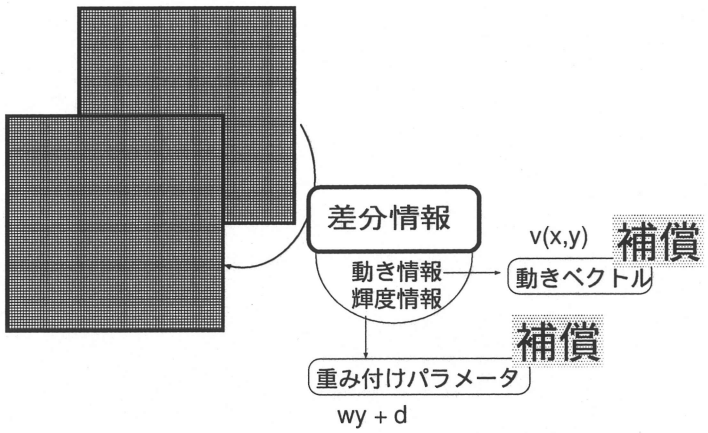

図 2.2: 重み付け動き補償の概略

適切な重み付けパラメータを与えることによりフェード画像のような輝度値がフレーム間で大きく変化する画像における輝度値の予測誤差を低減し, 符号発生量の削減をすることが出来る. 前述したように重み付け動き補償はフレーム間の輝度の変化を重み係数によって補償することから別名 "輝度補償" ともよばれる. だが一方で適当でない重み付けパラメータを与えると重み付けを行わない場合よりもかえって符号発生量を増加させてしまうことがあるため注意が必要である.

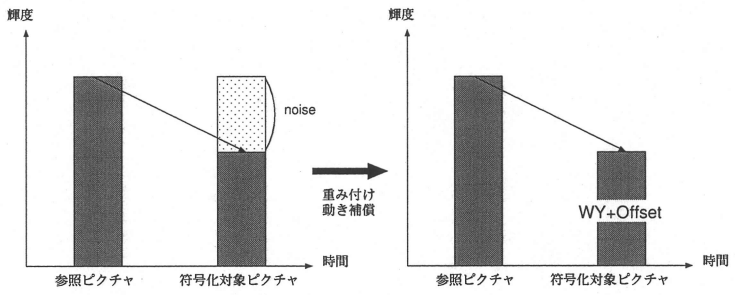

図 2.3: 輝度の補償概要

2.2.1 スライスへの分割

重み付け動き補償に必要な重み係数とオフセット値はピクチャから更に小さな単位であるマクロブロックの集合, スライスごとに与えられる. 従来の MPEG-2 などの符号化方式ではピクチャ単位で符号化処理を行っていたが H.264/AVC ではこのスライスが符号化処理を行う単位となる. また,H.264/AVC においてスライスの分割はスライスグループという方式で実現できる.
スライスグループについては表 2.1 を参照, 図 2.4 に例を示す. 重み付け動き補償に置いて, 重み係数とオフセット値はスライスごとに指定することを踏まえると輝度変化をもっとも適切な形で補償できる領域を判別しスライスへ分割し適切な重み係数とオフセット値を指定することが望ましい. そこで, 本論文では任意の形状を指定することのできる Explicit を指定する.

表 2.1: スライス・グループの種類

タイプ	名称	概要
0	Interleave	指定したマクロブロック数ごとに, スライス・グループを交互にならべる
1	Disperse	マクロブロックがピクチャ内で散らばるように 指定した数のスライスグループを交互に並べる.
2	ForGround/Reftover	複数の矩形のスライスグループを指定し, 残りを背景スライスグループとする
3	Box-out	マクロブロックを時計回り, または反時計回りに並べる
4	Raster	マクロブロックをラスタ順, または逆ラスタ順に並べる
5	Wipe	マクロブロックを縦に並べる
6	Explicit	各マクロブロックが属するスライスグループを示す テーブルを送ることで, 任意のスライスグループを作る

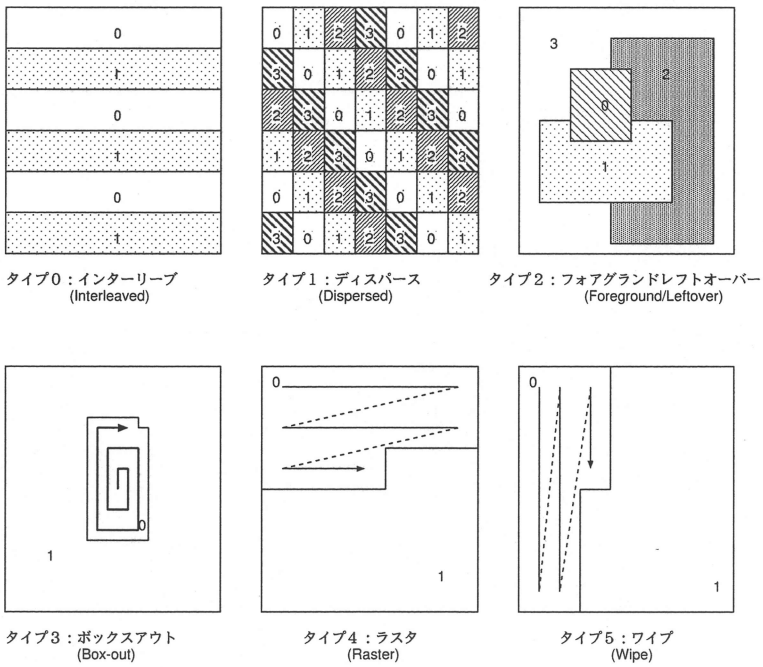

図 2.4: スライスグループの例

2.3 従来手法

2.3.1 平均値差分によるマッチング

輝度の変化による影響を避けるため動き探索のマッチング処理の際にブロックごとの画素平均値を差し引いたものを用いてマッチングを行う(文献 [2]). この処理の流れを図 2.5 に示す. 処理内容が明らかなように輝度値のダイナミックな変化による各画素値の予測誤差が低減できるため算出される SAD の値が改善される.
しかし, ブロックごとの平均画素値を求める演算量が通常の SAD を求める演算量と変わらないことと, 符号化対象ブロックと参照ブロックから平均が素値を減算する演算量を考えると通常の動き探索の 3 倍近くにまで演算量が増大してしまうため実用面で十分とはいえない.
また, 輝度変化領域の境界部分に関してはこの手法の性質上, 予測誤差による画質低下が改善されず動き探索の精度が低下してしまうのと, 平均画素値を差し引くことによる情報の損失のための探索精度低下も予想される.

$$SAD_{SA}(B,v) = \sum_{r\in B} |\,(I_{curr}(r) - Ave(B)) - (I_{ref}(r+v) - Ave(B_{pred}(v)))\,| \tag{3}$$

B : 符号化対象ブロック
$B_{pred}(v)$: 候補ブロック
v : 候補ベクトル
$I_{curr}(r)$: 符号化対象画像上で位置 r に存在する画素値
$I_{ref}(r)$: 参照画像上で位置 r に存在する画素値
$Ave(B)$: ブロック B の平均画素値

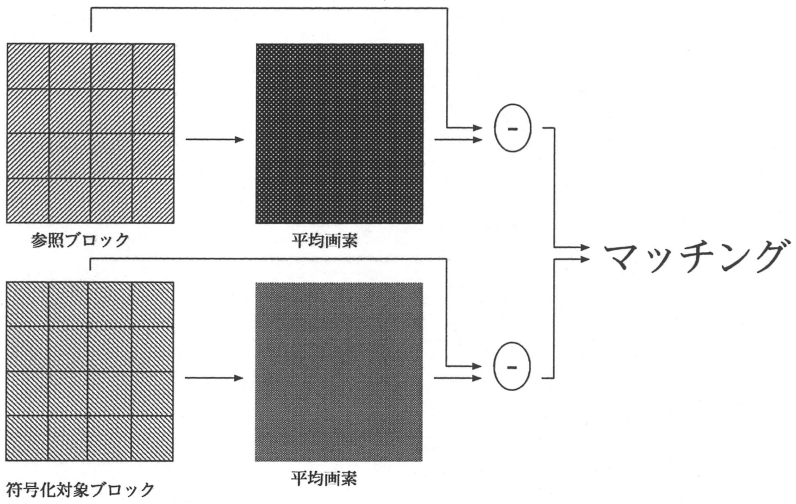

図 2.5: 提案手法の概略

2.3.2 k-平均法 (k-means) によるスライス切り分け

k-平均法による切り分けは画像中のマクロブロックを基準を中心としたk個の集合に分ける手法で, はじめにマクロブロックを適当なk個の集合へ分けておく. それらについて各要素, この場合はそれぞれのマクロブロックがどの集合へ属するかの判定を行いながら収束させていく. その手順は以下に示す.

1. 各マクロブロックの動きベクトルを求める (小々節 2.3.1 の手法を用いる)
2. 画像を適当にM個の集合に分類する ($U_m : m = 1, ..., M$)
3. U_m のそれぞれの重み付けパラメータ P_m を算出する
4. P_m を各マクロブロックに適用した場合の予測誤差がもっとも小さい U_m を選び, マクロブロックの属する集合を変更 (U_m を更新)
5. 符号化対象画像全体での予測誤差の絶対値総和が変化しなければ収束下と判定し終了, そうでなければ手順3へともどる

3　提案手法

3.1　輝度変化に対応した動き探索

3.1.1　エッジ抽出画像を利用したマッチング

今回提案する手法は動き探索の際にフィルタをかけて高周波成分を抽出した画像を用いる. この手法では画像での高周波にあたる輪郭部の抽出を行うことで輝度変化による影響を緩和する役目を果たす. 既存手法同様に輝度変化の境界部分での輪郭抽出で探索精度が低下することも考えられるが, ダイナミックな輝度変化によるSADで既存手法よりは予測誤差は低減できる. 今回高周波成分抽出には下記にしめすPrewittフィルタ (図3.6) でフィルタリングするものとし, 縦方向と横成分の高周波は下記の式(4) でフィルタ画像を生成する.

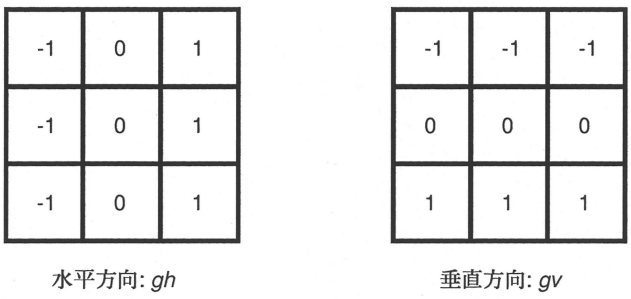

図 3.6: Prewittフィルタ

$$Y_{Prewitt} = \frac{|gh| + |gv|}{2} \tag{4}$$

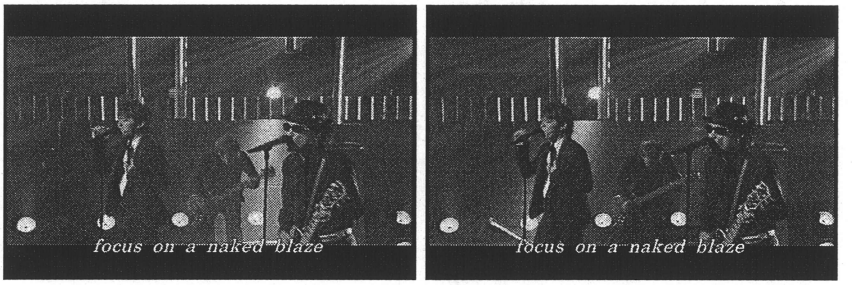

図 3.7: 照明の変化する連続画像

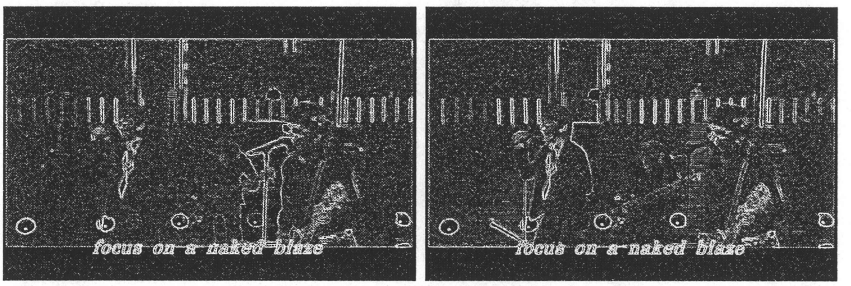

図 3.8: フィルタ画像

3.1.2　拡張テンプレート併用粗密階層型探索

エッジ抽出した画像は輝度変化に強いという特性を持つ分, オリジナルの画像の持つ情報量が削られており, その欠損の分が通常の動き探索の機能を阻害してしまうというリスクもある.
そこで今回, 高速動き探索アルゴリズムの 1 つである階層型探索を用いることでエッジ抽出画像を利用した際のリスクを抑えつつ, 輝度変化による動きベクトルの誤検出を可能な限り排した高速動き探索を実現する.
階層型探索は動き探索を行う際に符号化対象画像と参照画像のそれぞれの縮小した画像を対象とした予備探索を行う.(図 3.9 の上部参照) 予備探索の初めの 1 次探索として縮小したもっとも小さい画像に対して, 探索範囲内の探索を行う. その際に探索ブロックも画像の縮小規模に応じたサイズへ縮小し, それを用いて探索する. 続いて 1 次探索の結果を初期位置に 2 次探索を先に使用した縮小画像の次に大きな画像に対して行う. ここで,2 次探索以降の探索は直前の探索結果を中心とした初期位置から範囲を絞り込んで探索を行うものとする. こうすることで動き探索の総演算量を低減することが可能となる.
今回はエッジ抽出画像による探索を縮小画像対象の 1 次探索に限定することでエッジ抽出画像における情報欠損による動きベクトル精度低下のリスクと, 輝度変化による極端に誤った動きベクトルの検出を回避する.
加えて今回の探索ではブロックマッチングの際に用いるブロックのサイズ, テンプレートを本来のマクロブロックの 8 近傍分も加えたものを使用する. これにより細部の動きは捉えにくくなるものの, 大局的な動き検出精度の低下を防止する.

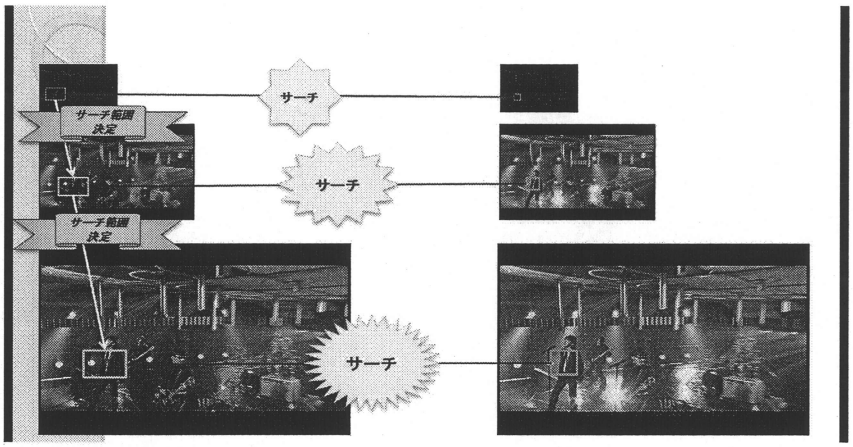

図 3.9: 階層探索の概略

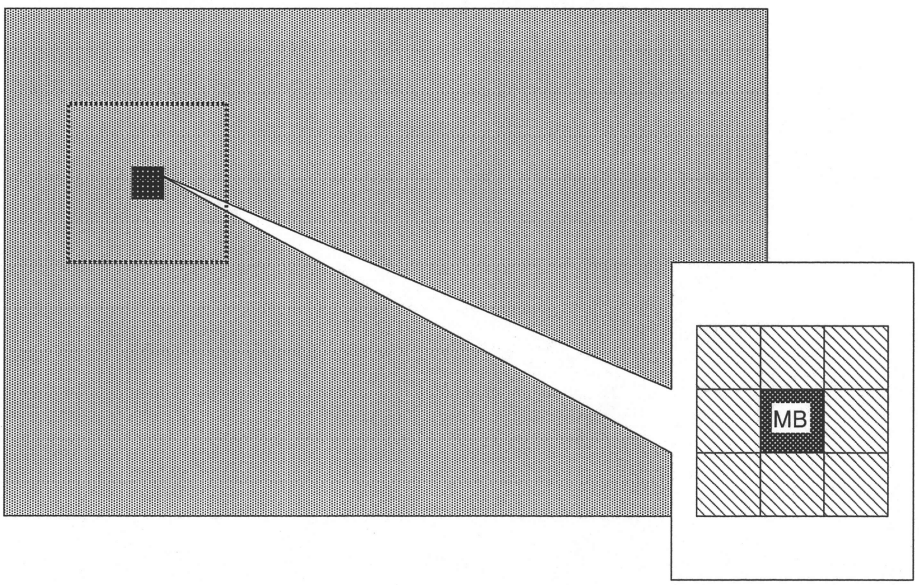

図 3.10: 拡張テンプレート

この手法による動きベクトルの検出結果を図 3.12 に示す. なお, その直前の画像が図 3.11 であり,2 つの画像間において照明の変化以外目立った変化は見られていない. 先に述べた提案手法の検出結果 (図 3.12) をエッジ抽出を使用しなかった場合の階層型探索による動きベクトルの検出結果 (図 3.13) と比較すると, 提案手法の動きベクトルが比較的安定していることが確認できる.
乱れたベクトルが完全になくなった訳ではないが, その多くは背景等の周囲との境界が認識しづらく, エッジがほとんど抽出できない領域に多い. そのような領域に関しては正確な動きが検出できなくても動き補償の効率低下は起こりにくいため, 許容範囲といえる.

図 3.11: 符号化画像の直前のワンシーン

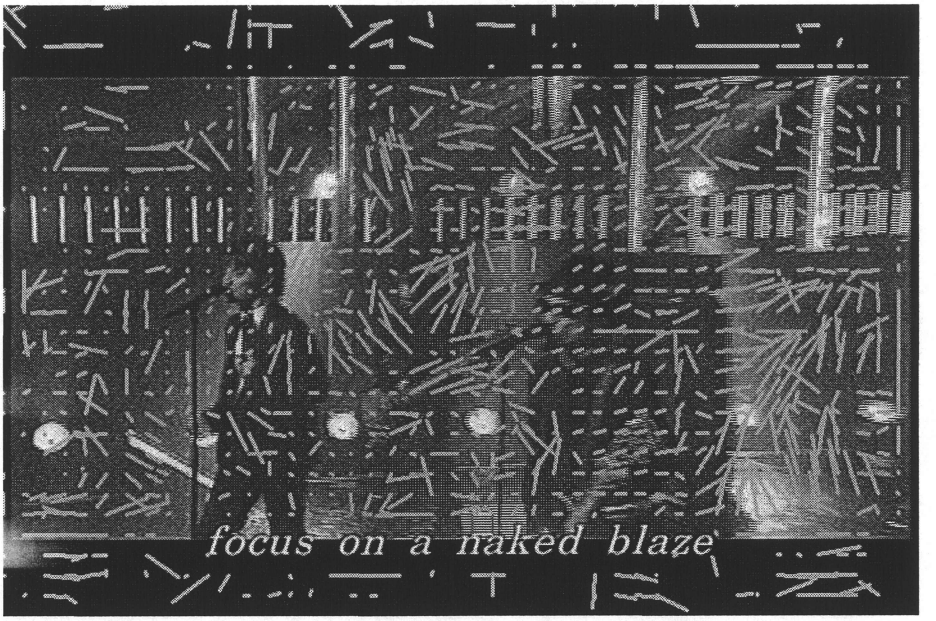

図 3.12: 動きベクトル検出結果 (エッジ画像使用)

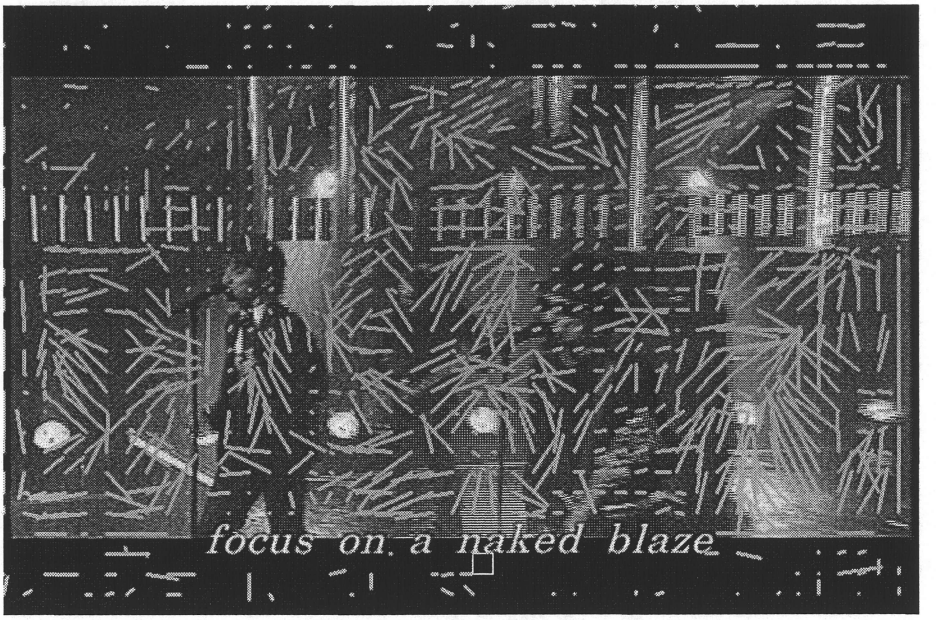

図 3.13: 動きベクトル検出結果 (エッジ画像未使用)

3.2 輝度変化ヒストグラムによるスライス切り分け法

同一の画面内で起こる輝度変化を分類すると, あまり変わらないところ, 明るくなったところ, 暗くなったところの 3 つに分けられる.
今回はそれらの 3 つに分類するために符号化対象画像と参照画像間の動き情報を適用した残差を利用する. 残差の実態については図 3.14(連続する画像間の一部から抜粋) を参照. 残差が 0 に近いほど白と黒の中間であるグレーで表示されている.
スライスへの切り分けを行うことを考慮し, スライスの構成要素であるマクロブロックごとの平均値を算出する.(図 3.15) さらに先に述べた残差の平均値からヒストグラムを作成する.(図 3.16) 作成したヒストグラムへ境界をもうけることでそこを閾値として領域の切り分けを行いスライスの形状を決定する. なお, 今回の手法ではヒストグラムの中心となる点から双方向へ固定長離れた点を境界とする. その際に中心となる点については画面全体が一様に輝度変化を起こす場合にヒストグラムが左右のどちらかに集中することも想定して, ヒストグラムの中央値 (メディアン) を利用する. また, 局所的な輝度変化があったとしてもその領域が一定以上の範囲で存在しない限り目につきづらいことや切り分けにより増加するスライスが含む情報分の符号量を鑑み, 画像全体の一定以下の割合しか大きさがない領域は隣接する領域へ統合する. これにより, 輝度変化がわずかな領域でしか発生しない画像やオブジェクトの変形等による残差情報の増大による過剰な数へのスライス分割を抑止することが可能となる.

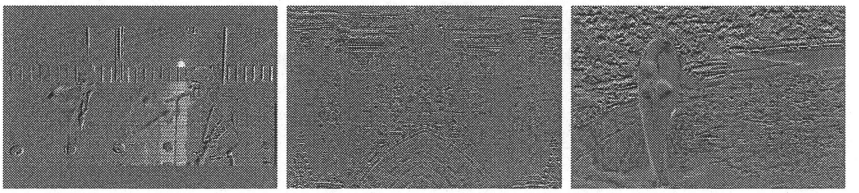

図 3.14: 残差情報

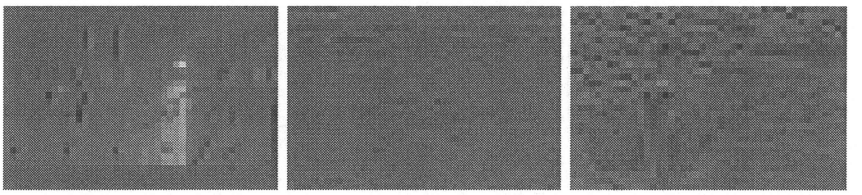

図 3.15: 残差ブロック平均

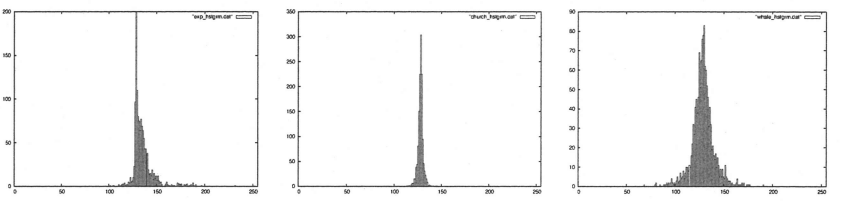

図 3.16: 残差平均値ヒストグラム

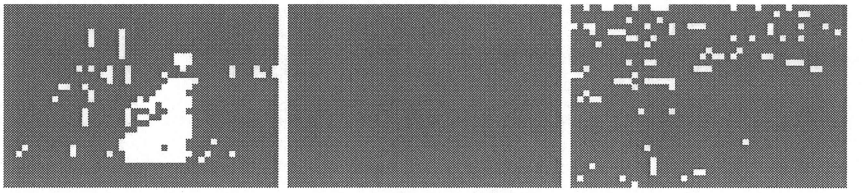

図 3.17: スライス形状 (d=11,r=0.0625)

提案手法の手順をまとめると

1. 各マクロブロックの動きベクトルを求める (小節 3.1 の手法を用いる)
2. 動き情報を適用した画像間の残差を計算する
3. 残差からマクロブロック単位の平均値を求める
4. 残差の平均値からヒストグラムを作成する
5. 作成したヒストグラムの中央値を求め, その値から正負に固定長の値 d だけ離れた点を閾値に各マクロブロックを 3 つの属性に分類する
6. 分類した同一属性のマクロブロックの集合が画像全体の一定割合 r 以下のとき, ヒストグラム上で隣り合うマクロブロックの集合と統合する

提案手法のフローチャートを図 3.18 に示す. 矢印の隣の数字は上述のの内容と一致する.
このうち, d と r は任意の値に設定が可能である. 今回は $d = 11$ と $r = 0.0625$ としている.

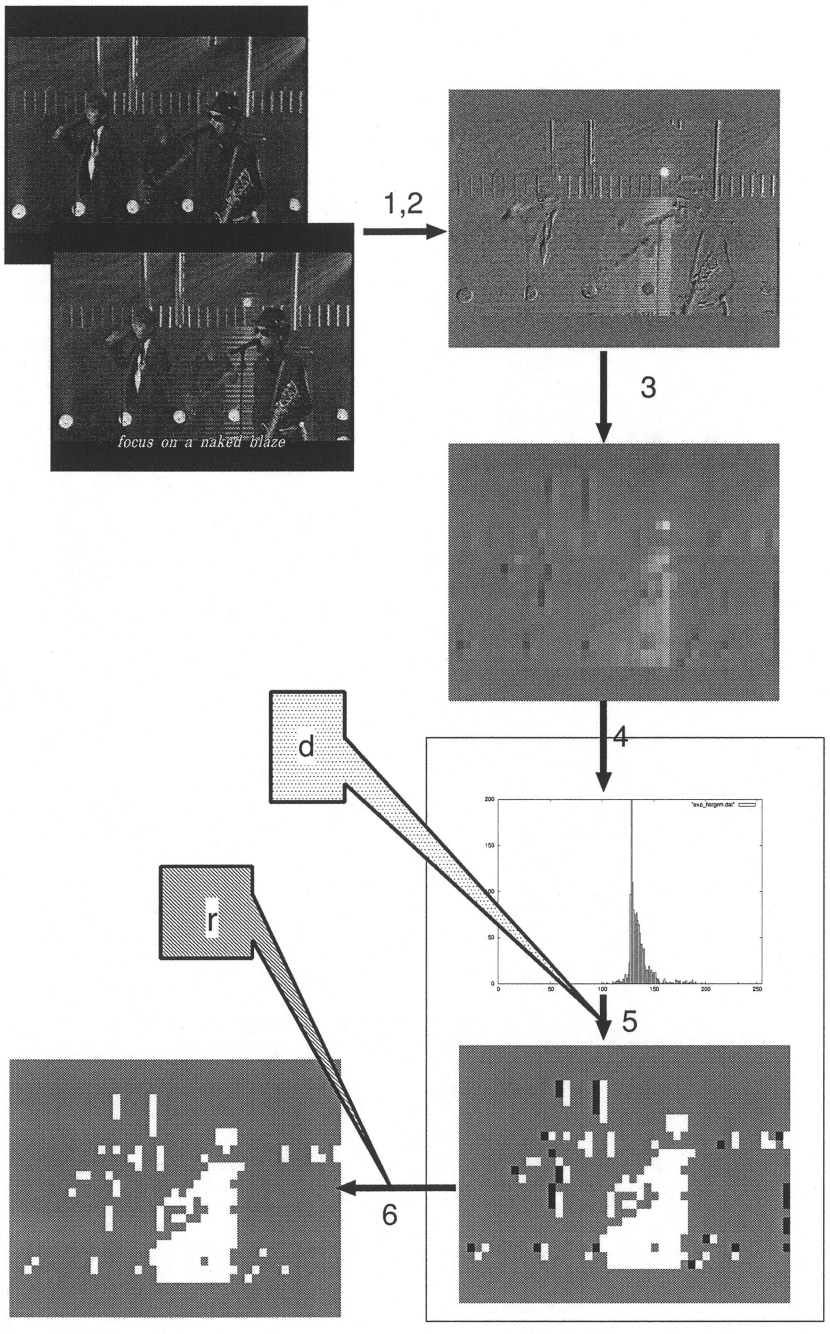

図 3.18: 提案手法のフローチャート

4　実験と評価

4.1　評価環境

WindowsXP 上で,H.264 の参照ソフトウェアエンコーダ (JM10.1) に今回の提案手法を実装し,Microsoft の開発環境 Visual C++ 2005 のコンパイラを使用. ビットレート固定は行わず量子化パラメータ Q_p を 28 に固定するとともに, 動き探索のマクロブロックサイズを 16×16 に固定した. 改善目的の対象である *Show*, 通常の画像として *Church* と *Whale_Show* の計3つを使用して重み付け動き補償による符号化効率の評価として,PSNR, ビットレートレートをにより画質と符号発生量を定量的に評価を行った. また, スライスの切り分けが有効かどうかと, 複数のスライスに対して妥当なパラメータが適用されているかをそれぞれについて目視により直接確認・評価を行う.

表 4.2: 評価環境

フレーム数	30
Q_p	28
マクロブロックサイズ	16×16
探索範囲	± 32
ピクチャ周期	IPPP...
スライス・グループ	Explicit

5　結果と考察

5.1　画像評価結果

今回の実験ではスライスの切り分けによる形状の確認と, それぞれのスライスに妥当と思われる重みづけパラメータが適用されているかの確認ができた. 図 5.19 の下部のスライス形状の白色の部分と上部の 2 つの画像間で照明がついたことによる輝度変化が発生していることが確認できる領域がおおよそ一致する. また, 同図 (5.19) の黒色の部分についても上部の 2 つの画像間で照明が消えたことによる輝度変化が確認でき, 同様に輝度変化の領域が一致している.

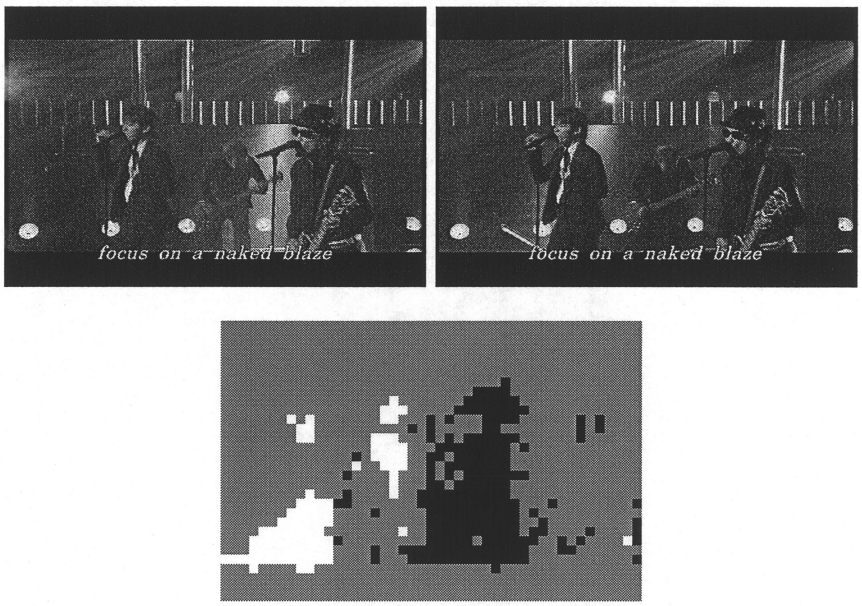

図 5.19: 動画中のシーンとそのスライス形状

加えて図 5.20 の左から順に灰色, 黒色, 白色のスライスの重み付けパラメータ適用ずみの参照画像を示している. 白色スライスでは輝度値が増加していることが先に述べたように確認できており, 黒色スライスでは輝度値が減少している. そのことをふまえて, 図 5.20 下部の画像は全体的に輝度が増しており, 中央の画像では輝度が減じていることから重み計数も適当な値が割り振られていることが確認できる.

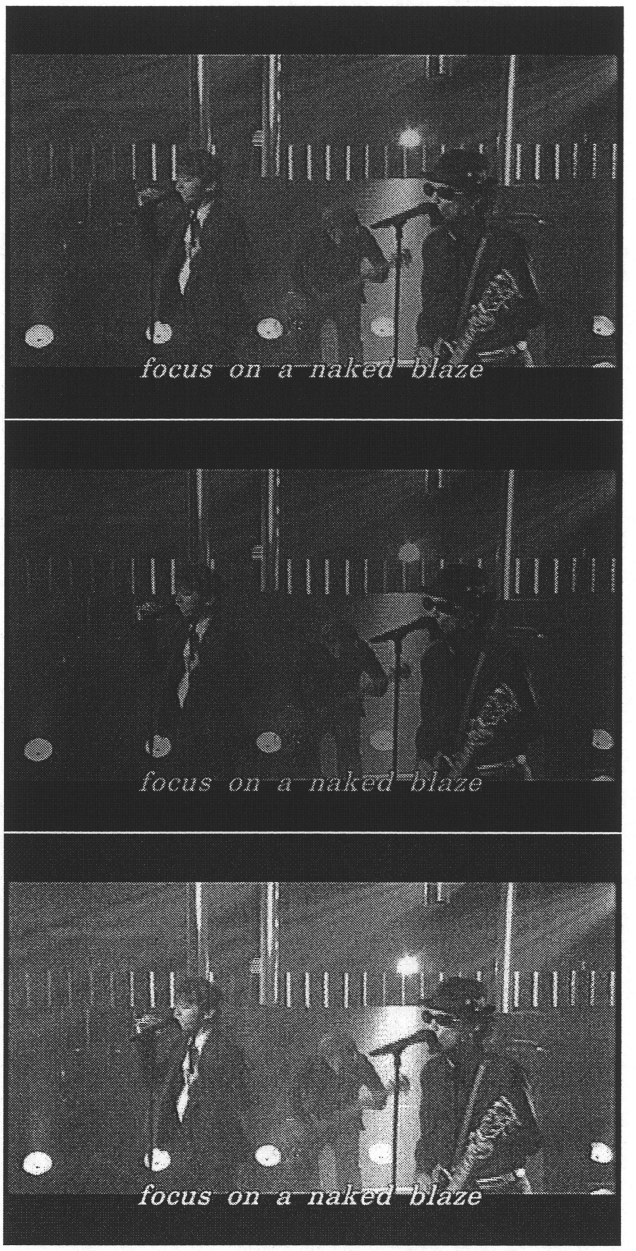

図 5.20: 重み付けパラメータを適用した参照画像

しかし,PSNR とビットレートを確認してみたところ同一の画像に対して, UMHexagonS アルゴリズムによる通常の探索結果と比べると改善がほぼ見られず一部の画像では符号量が増大する結果となってしまった.

5.2 考察

今回の手法における輝度変化検出について高速動き推定アルゴリズム UMHezagonS による同シーンの重み付け動き補償なしの場合の残差情報, 小節 5.1 と同じ画像間におけるものを次の図5.21 のに示す. さらに提案手法で途中使用した残差情報を図 5.22 に示し比較を行う.
この残差情報を見る限りにおいて, 対象の画像間における輝度変化の情報は正確に検出できておらず, 仮にこの残差情報をもとに今回の実験と同じ条件でスライス切り分けを行った結果は画像全体が1 枚のスライスとなったことからも提案手法による輝度変化の検出はほぼねらい通りに行えたといえる.

図 5.21: UMHezagonS による画像間の残差情報

今回の提案手法による重み付け動き補償を行った場合の残差情報を次の図 5.23 に示す.

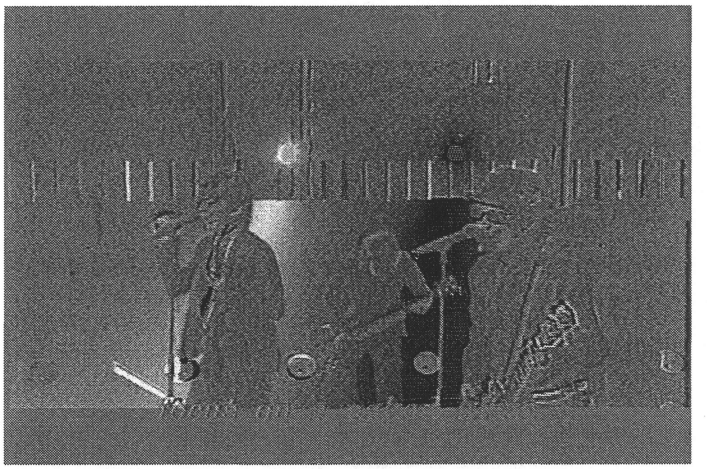

図 5.22: 提案手法のスライス切り分け途中で使用した残差情報

先の図 5.22 と比べると残差の大きい部分中央左部の白色部と中央右部の黒色部が緩和され, 全体で見ると残差が減少していることが確認できる.

しかしながら, 図 5.22 と図 5.23 それぞれの残差情報を用いて符号化を行った場合, さしたる改善が見られなかった. その主要因として, 輝度変化を今回は 3 つの種類に分類し, スライスの切り分けを 3 つまでとしたため, スライスを切り分けた後に領域に残る輝度変化に対して重み付け動き補償による予測誤差の改善が十分に発揮できなかったことが考えられる. 今後は分割可能数を増やし, 過剰な分割によるオーバーヘッドと局所への適切な重み付け動き補償による予測誤差低減の両者を調整し, 符号化効率改善につなげていかなくてはならない.
また, 今回使用した画像について写りが鮮明でないためか輪郭線がぼやけた部分も含まれているため, その部分の残差情報が重み付け動き補償による予測誤差の最小化を阻害している可能性も考えられるため, 鮮明なより多くの歌謡番組のシーンの採取も課題となる.

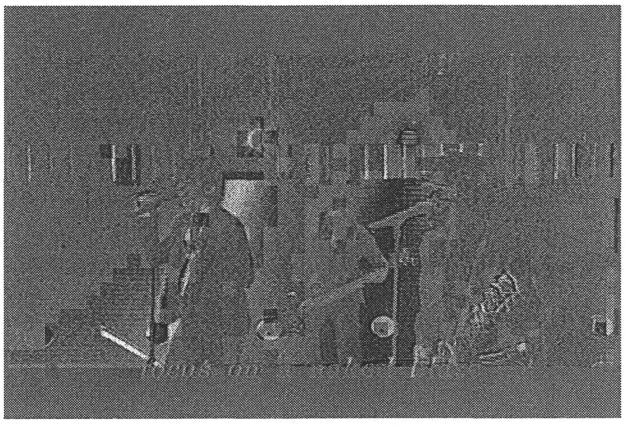

図 5.23: 提案手法による残差情報

6　まとめ

本研究では動画像符号化における重み付け動き補償の効率をより高めることを目的とし, 階層型探索にエッジ抽出画像を併用した動き探索と輝度変化情報から作成したヒストグラムを閾値で区切ることによりスライス切り分けを行う手法を提案し, 実装・評価を行った.
階層型探索の縮小画像への探索の際にエッジ抽出画像を用い, 縮小画像とエッジ抽出画像の使用による検出精度の低下軽減のため拡張テンプレート法を適用することで従来の動き探索 ([2]) に比べ約 100 分の 1 程度と実用的な演算量で輝度変化による動きベクトルの極端な誤検出をほぼ解消できることを確認した.
加えて, 画像間の輝度変化を基準としたヒストグラムの利用により, 従来の参考文献 [2] の k-平均法を利用する手法より演算量を排し, 直観的なスライス切り分けを行う手法では目視による確認により輝度変化の領域に一致したスライスへの切り分けが行われていることを確認した.
しかしながら, 参照ソフトウェア (JM10.1) への実装による画質評価の値 PSNR と符号量において有意な成果が確認できなかった.
今後の課題は輝度変化をより詳細に分類し, スライスの分割数を増やすことで重み付け動き補償による予測誤差の改善を向上させることとスライス分割によるオーバーヘッドを定量的に評価し, 符号化効率の改善を実現する手法と評価のための鮮明な歌謡ショーに代表される様々な輝度変化画像の採取が今後の課題である.

7　謝辞

本研究を進めるにあたり, 日々励みとなるご指導いただいた近藤利夫教授, 大野和彦講師, 佐々木敬泰助手に深く感謝申し上げます. また, 日頃ご支援お世話いただいた田中事務官に感謝申し上げます.
最後に計算機アーキテクチャ研究室の諸氏には本研究の様々な局面に置いて多大な御協力を頂きました. 個々に感謝の意を表します.

参考文献

[1] 大久保 榮 (監修)，角野 眞也, 菊池 義浩, 鈴木 輝彦, "H.264/AVC 教科書"

[2] 上倉 一人，木全 英明, 米原 紀子, 八島 由幸, "グローバル輝度変化補償 (Weighted Prediction) における複数輝度変化パラメータの推定法" 信学技報 ITS2003-104 IE2003-239 (2004-2))

[3] 加藤 晴久，中島 康之, "H.264/MPEG-4 AVC 重み付き動き補償における高速重み係数推定法の一検討", FIT2004(第 3 回情報科学技術フォーラム)

[4] 木全 秀明, 上倉 一人, 米原 紀子, 八島 由幸, "H.264/AVC 重み付けフレーム間予測符号化における重み付けパラメータ探索方式の検討", 信学技報 OIS2003-39, IE2003-64 (2003-9)

[5] 新井 英雄, 坂口 俊文, 阿部 良三, 綿谷 由純, "MPEG2 を用いたフェード画像符号化に関する検討", 1995 年電子情報通信学会情報・システムソサイエティ大会

付録

残差情報の表示や画像生成に関して

今回文中に登場した残差情報に関して, 参照ソフトウェアエンコーダ JM10.1 を使用して出力する方法に関しては画像としての出力はファイル構成の簡易さから PGM 形式で出力する.
残差情報に関しては参照ソフトウェアのソースファイル *macroblock.c* 中の関数 *LumaResidualCoding*8*x*8 中の残差算出部分があるためその内部にグローバル変数に追加した画像保存用配列に残差情報を保存するようにコードを変更する.
なお, その際に使用するマクロブロックの位置情報等は image parameters 構造体のメンバ変数 *opix_x*, *opix_y* で参照できる. そこからサブマクロブロック内の位置情報に関してはプログラム内部のループ部分を参考に利用すること.
以下にそのコードの一部を記載する為, 必要に応じて各自変更を行うこと.

```
//残差画像生成部
-------------------------------------------
for (j=0; j<4; j++)
{
 pix_y = pic_pix_y + j;
 for (i=0; i<4; i++)
 {
   保存用配列 [pix_y][pic_pix_x + i] =
   imgY_org[pix_y][pic_pix_x + i] - img->mpr[j+block_y][i+block_x];
 }
}
-------------------------------------------

//オリジナルのコード
//===== get displaced frame difference ======
if(!img->residue_transform_flag)
{
```

画像の生成はフレーム符号化関数 *encode_one_frame* 呼び出し直後に行っている.