Vol.2012-CG-149 No.17
Vol.2012-CVIM-184 No.17
2012/12/4

指定領域内の品質保持に対応した 可変型映像圧縮法および GPU による高速化

村上 敬亮[†1]　　安藤 英俊[†1]

リアルタイムの動画通信技術は放送技術の発展にともなって進歩し，当初はビデオ会議などの目的で使用されてきた．かつてはこうした技術は導入コストの高さなどから一部の機関での使用にとどまっていたが，現在ではインフラの整備や技術向上が進んだために娯楽や情報番組の配信にも利用され，日常生活においても珍しくないものとなっている．こうした動画配信技術は現在，医療などのクリティカルな分野においてもさらなる活躍が期待されているが，発展途上国や過疎地，被災地などの場所においてはインフラや設備の制約から，通信帯域や処理速度が課題となる．本研究では，こうした制約のある現場でのリアルタイム通信に適した動画の圧縮手法を提案し，また GPU によってエンコーダの高速化を行う．

The Method of Dynamic Video Compression Corresponding to The Quality Preservation and Accelerate Encoder Using GPU

KEISUKE MURAKAMI[†1]　HIDETOSHI ANDO[†1]

On time video communication technology has evolved along with the broadcast technology. Initially it has been used for the purpose, such as video conferencing. Although, at that time this technology have been tended to remain for use in some authority once from the high cost, now it used in entertainment or information program and is not unusual for general people because of developed infrastructure. The technology expected to apply in critical tasks like medical care but in developing countries, sparsely populated areas or disaster area there are some issues that communication band, processing speed and so on. In this study we suggest the new method to video compression in this case and accelerate the video encoder using GPU.

1. はじめに

リアルタイム動画配信システムの提供は，アナログ放送の技術を利用して早くから行われていた．当初は衛星の専用回線を利用するものが主で，導入や運用のコストも大きかったために，軍隊や NASA におけるビデオ会議など一部の機関での使用にとどまっていた．その後映像技術のデジタル化が行われ動画圧縮技術が確立すると，次第に整備されたインターネット回線を使用するシステムが現れ始める．さらに, PC のスペックが向上してくると次第に専用機材の必要性も薄くなっていき，現在では Skype®ビデオチャット，Ustream®，ニコニコ生放送®など，個人が無料で使用出来るリアルタイム動画通信サービスは，既に珍しいものではなくなっている．最近ではスマートフォンなどの携帯端末でも動画通信が実現可能なレベルに達しており，その技術はますます人々にとって当たり前のものになりつつある．

こうした状況の中, 日本政府は 2010 年に新たな情報通信技術戦略[1]を発表し，「光の道」構想を始めとする様々な IT 戦略についての方針が定められた．その 1 つとして映像コミュニケーションの普及があるが，これはリアルタイム動画配信技術を利用した遠隔会議，遠隔授業，遠隔医療などの普及拡大に力を入れていくというものである．リアルタイム動画通信技術にはより一層の活躍が期待されている．

インターネット回線を使用したリアルタイムの動画配信システムは，世界的に回線インフラが広く普及していることから，比較的その設置場所を選ばず，大掛かりな機材も必要ないため低コストに実現できるという利点がある．特に，過疎地や被災地，海外に目を向ければ発展途上国や戦地など，インフラや地域情勢，予算の関係上高価な専用設備を置くことの出来ない場所でも，動画通信は有用なものとなるはずである．

一方で，こうした利点を保ったままシステムを広く提供するにはまだまだ技術的な課題も多い．その代表的なものが，回線状況と用途を考慮した上での動画圧縮における圧縮率と圧縮時間である．本研究では，このような制約のある環境下でのリアルタイム動画配信において，ユーザーの求める情報を効率良く届ける圧縮手法を提案するとともに，近年実績を上げている[2]GPU による実装を行い、圧縮時間の短縮を図る．

2. 動画配信における圧縮の現状と提案手法

ネットワーク配信における動画圧縮で可能な限り品質の高い動画を提供するためには，変化する回線スループットを考慮して動的にビットレートを調整する必要がある．

従来のサービスでは，スループットの低下を検出すると動画全体の画質を落としているものが多い．これは，通信を途切れず継続するという意味では有用な方法であるが，ある程度の画質を保持しなければ意味が無い用途での使用

†1 山梨大学大学院
Yamanashi University

には向かない．例えば，柏木らの研究[3]にあるような過疎地を対象とした遠隔眼科診断のような状況では，高度な配信システムの設置は困難であるが，簡便なシステムで従来の品質調整が行われると診察に必要な画質が得られない．この場合ビットレートは落としつつ，部分的にでも必要最低限の画質を保持している必要がある．

そこで，本研究では回線スループットが低下した際，一部領域の画質のみを保ったまま，それ以外の領域の情報量を削減することでビットレートを調整する手法を提案する．概念図を図 1 に示す．

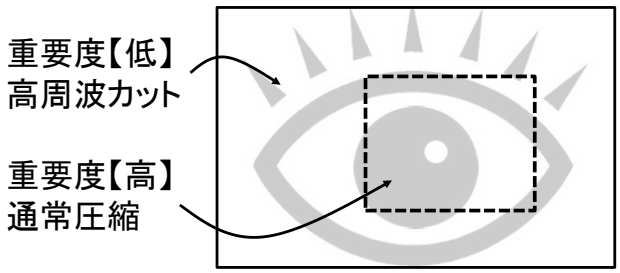

図 1　提案手法の概念図

画質を保持する領域はユーザーが手動で指定できるものとし，視線追跡装置など，将来的な各種インターフェイスとの連携による領域選択の自動化も視野に入れる．

3. 実装方法

3.1 基本方針

動画コーデックには様々なものがあるが，用途に応じて向き不向きを考慮する必要がある．例えば，日本の地上デジタル放送で使用されている MPEG2 は圧縮処理が速いために，HD やフル HD サイズの高解像度放送をリアルタイムで行うのには適している．しかし圧縮率は低いため，広帯域の獲得があまり期待できない民間のネットワーク回線しかない状況には不向きである．本研究では，圧縮率に優れ各種動画配信サービスでの使用実績もある H.264 を使用することとした．H.264 は圧縮に比較的時間がかかるのが難点であるが，処理速度に定評があるオープンソースのエンコーダ x264 を使用することでその欠点を緩和する．

提案手法の実現方法であるが，単に出力動画のビットレートを下げるだけであれば，予め周辺を画像処理的な手法で暈した動画をエンコーダに入力するという方法でも，圧縮アルゴリズムの特性上不可能ではない．しかし，それでは暈しの処理に無駄な計算リソースを使うことになりナンセンスである．本研究ではエンコーダの圧縮アルゴリズムに手を入れることとする．

H.264 には過去の動画コーデックにおいて実績を上げた多数の圧縮技術が複合的に用いられており，エンコーダ内のモジュール数も多い[4]．データの流れは図 2 に示すようになっており，入力された画像は基本的な圧縮処理を施した後，符号化・保存されるとともに次のフレームの圧縮を行う際の比較対象とするため再び展開される．本研究では，この中からすべてのフレームに対して行われ，かつ直接的なビットレート削減効果が得られそうなブロッキング，DCT，量子化の 3 つのモジュールの変更を行った．

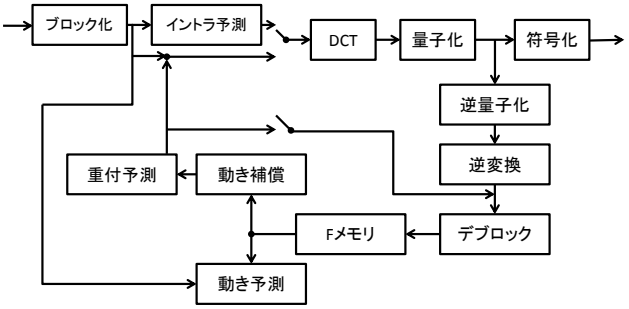

図 2　H.264 モジュールとデータの流れ

3.2 ブロッキング

エンコーダに入力された動画のフレームはまず，その画像を 8×8 ピクセル程度のブロックと呼ばれる小領域に分割され，種々の圧縮処理はこのブロックを基本単位として行われる．実際のブロックサイズはコーデックによって異なるが，H.264 の場合は 16×16 から 4×4 までの様々なサイズのブロックを画像の内容に応じて組み合わせ最適化が行われる．

提案手法では，選択領域のブロッキングは x264 標準の最適化アルゴリズムにより行うが，選択外の領域においては画像の内容を無視し，強制的に大きいブロックを指定することで動画全体のブロック数を削減している．これは，視覚的にはモザイクが掛かったような効果を生み出す．

3.3 離散コサイン変換（DCT）

離散コサイン変換（DCT）は，多くの音声・画像圧縮コーデックで利用されている技術である．フーリエ変換から派生したもので，離散信号を周波数別のコサイン波に分離する．DCT を施した信号は低周波成分にデータが集中しやすく，高周波成分は削除しても視覚上大きな劣化を感じないことから，不可逆圧縮に広く用いられている．

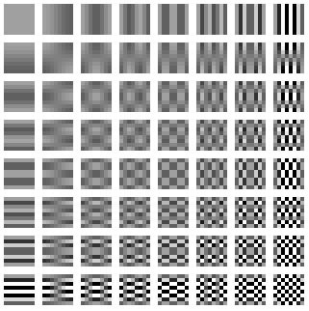

図 3　DCT の基底行列

H.264 では，各ブロックの画素信号は DCT によって図 3 に示す基底行列に対応する 8×8 の係数行列に変換される．

本研究の提案手法では，映像中の指定外の領域に関して，係数行列右下寄りの高周波成分を削除する．これは，見た目上は指定外領域の映像がぼやけることを意味する．

3.4 量子化

DCT の係数行列に変換された各ブロックは，さらに量子化によってそのビット数を削減してから符号化および保存される．H.264 では，各ブロックに量子化パラメータという調整値が存在し，ブロックによって量子化の粗さを変化させることが出来る．

提案手法では，この量子化パラメータも領域外のブロックについては変更を行い，大幅にビット数を削減することとしている．

3.5 領域サイズと内部パラメータ

本研究では映像に対して 3 種類の領域を設けた．中心がユーザーの指定する選択領域で，ここは通常のエンコードが行われる．その外側は提案手法が適用され，2 段階で映像が粗くなる．具体的な設定は内側から順に，8×8 ブロック強制領域，16×16 マクロブロック強制領域で，DCT 高周波カットと量子化パラメータ調整に関しては 2 領域に対して同様の設定を施した．ユーザーが選択領域を移動させると，8×8 ブロック強制領域は追従する．領域区分の概略を図 4 に示す．図中の矢印に付与した数字は，動画の横幅を 1 とした時の各領域のサイズを表している．

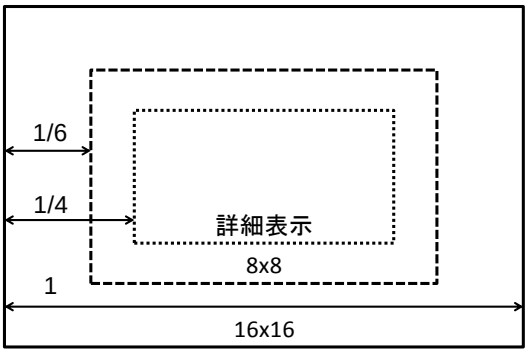

図 4　領域区分の概略

4. 実験

4.1 手順

実験では，同じ動画を通常の x264 と，提案手法を実装したものとでそれぞれエンコードする．この際，エンコーダの設定は目標画質基準で統一する．得られた 2 つの圧縮動画を元動画と比較し，映像評価指標の 1 つである PSNR を算出する．その際，選択領域内のみを対象とした PSNR を求め評価に用いる．選択領域内の画質を固定しているため，期待通りであれば提案手法にてエンコードした動画の PSNR は通常のものと遜色なく，かつ出力された動画のビットレートは提案手法の方が小さくなるはずである．

PSNR（ピーク信号対雑音比）は，エンコード前後の画像を比較し，信号が取りうる最大値とノイズの比率を算出するものである．結果は dB を用いて表すことが多い．値が大きいほどノイズが小さく，品質が良いとされる．また，PSNR に 0.2dB 以上の差があると，映像を目視した際，主観的に違いを感じると言われる．ただし，PSNR が悪くても視覚的に綺麗だと感じる場合もあるため，あくまでも参考指標の 1 つとして捉える必要がある．

エンコード時の x264 は High プロファイル, チューン film を設定．リアルタイム通信を想定することから，前方・後方の両フレームを参照する B フレームは挿入しない．動画のサイズは SD (720×406) とした．画質基準モードによる圧縮になるため，動画のビットレートを正確に指定できるわけではないが，おおよそ提案手法による圧縮動画が 1Mbps 前後になるように調節を行った．これは，1Mbps であればかなり劣悪な回線でもリアルタイム通信が可能であり，多くのケースにおいて場所を選ばず利用することが出来るとの想定である．

4.2 ビットレート評価

実験の結果，最終的に出力された動画の平均ビットレートは，図 5 のようになった．40%程度の削減効果があったと言える．この結果は，選択領域を固定しても，あるいは人間が操作する程度の速度で移動させても変化はなく，実用上問題ないようである．

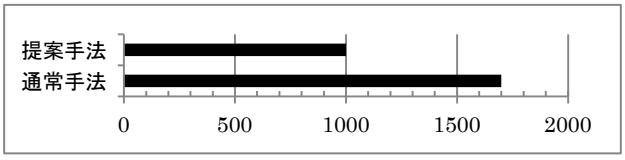

図 5　ビットレート比較（単位 kbps）

参考までに，ビットレート条件による拘束を排除し，高画質設定で提案手法によるエンコードを行なってみたところ，ビットレートは通常の場合と比較して 60%程度削減されることが分かった．これは，高画質設定の方が全体的に映像の高周波成分を残すため，提案手法の効きが強かったのだと解釈できる．

4.3 PSNR 評価

提案手法を用いて出力された人間の目の画像を図 6 に示す．出力動画を目視で評価しても，外縁部が大きく暈けていることはあまり気にならない．これは，もともと人間の視覚が中心以外を精密に捉えていないことに起因すると考えられる．視線追跡装置との連携は，最も提案手法の効果

Vol.2012-CG-149 No.17
Vol.2012-CVIM-184 No.17
2012/12/4

を発揮するであろう．

選択領域内だけを取り出して計測した PSNR は，通常のものも提案手法のものも，凡そ 30dB 前後と揃った．両者間の差も 0.2dB 以内で主観的に差がないレベルである．細かく見ていくと，わずかながら提案手法の方がノイズは小さくなっていた．

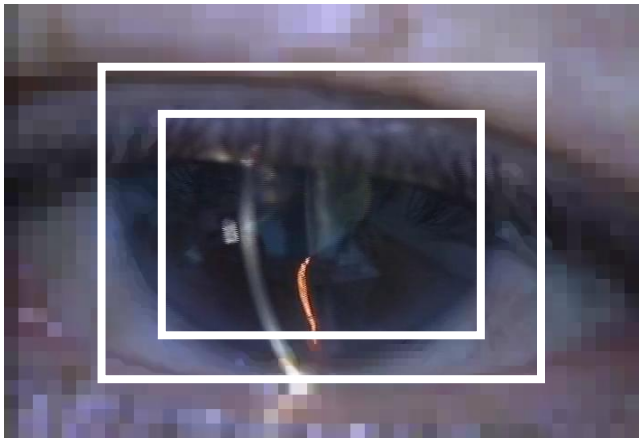

図 6　提案手法の出力画像

4.4 実行速度

今回の実験では，Core2 Duo 2.67GHz の CPU を用いて Windows マシンでエンコードを行った．

提案手法を実装した x264 では，コードの最適化が崩れてしまうためか，元のものと比較して若干実行速度の低下が見られる．今回の実験においては，エンコード時の処理速度は約 24fps であった．これは，アニメーションなどの映像の再生フレームレートに相当する．実験に使用したマシンが 2012 年現在, 決してハイエンドのものではないことを考えると，SD サイズ 24fps の単純なエンコード処理のみであれば提案手法においてリアルタイムで行えると判断して良いであろう．しかしながら，実用上は前述の視線追跡装置に代表されるような，様々な処理を追加する必要があり，また今後より高い解像度やフレームレートが要求される可能性はある．従って，次章に記す GPU による圧縮処理の高速化を試みる．

5. GPU による高速化

5.1 GPU の特性

GPU (Graphic Processing Unit) は，コンピュータにおけるグラフィック関連の処理を行う専用デバイスとして誕生したものである．近年ではグラフィック処理にとどまらず，汎用計算にも積極的に用いられるようになった．内部には 100～1000 単位のプロセッサが組み込まれており，それらが並列処理を行うことで CG の頂点や画素など，大量のデータを独立して扱うタスクを高速化している．プロセッサ単体の実行速度は CPU のものよりも劣るため，GPU は逐次処理には弱いが並列処理には強いという特徴を持つ．GPU プログラミングによってハードウェアの性能を引き出すには，この特性をよく理解し，適切なマルチスレッドプログラミングを行う必要がある．

5.2 基本方針

x264 の中で最も処理時間のかかるモジュールは動き予測で，実にエンコード時間全体の 81%となっている．動き予測の処理は膨大なデータを扱う一方で各々の処理は独立なため，並列化が容易で GPU には適している．従って，本研究では動き予測モジュールを GPU 用コードによる実装に置き換える．実装には NVIDIA 社製 GPU 用の開発キットである CUDA®を使用する．

x264 の動き予測を CUDA に移植した先行研究が Lawrence ら[5]によって既に行われているが，このソースコードは GPU のアーキテクチャを考慮した最適化が甘いことから, さらなる処理速度の改善が見込める. 本研究では，このコードに多少の最適化を加えて用いることとする．

5.3 動き予測

動き予測は，多くの動画コーデックにて採用されている圧縮用モジュールの 1 つである．その目的は，動画内の前後のフレームを比較して生成されるフレーム間差分信号を小さく抑えることにある．

一般的に，動画内の連続したフレームの内容は似通っていることが多い. そこで動画圧縮では，前述の DCT に代表されるような 1 フレーム内での冗長性を省く処理の他に，複数のフレーム間での冗長性を省く処理も合わせて行う．最も単純な方法は，各フレームの信号をそのまま保存する代わりに，先に処理したフレームとの画素差分値をとっておくというものである．これがフレーム間差分である．

そして，差分をとる際にさらなる工夫として加えられているのが動き予測である．動き予測では，処理対象のフレームに存在する各ブロックと似通ったものを比較対象のフレームから探索し，対応するブロック間での差分をとる．すなわち，保存される情報はブロックの相対位置（動きベクトル，図 7）と，ブロック間差分値である．画面内の動きに着目した手法であり，多くの場合，単に同一座標上のフレーム間差分をとるよりも差分値が小さくなる．

図 7　比較画像（左）と処理画像の動きベクトル（右）

H.264 では，キーフレーム（I フレーム）については完全な画像に対して DCT をかけるが，差分フレーム（P，B フレーム）に関しては動き予測を使ったフレーム間差分に対して DCT を施して保存している．

5.4 実装の最適化

動き予測の処理はブロックごとに独立しており，GPU 上で並列化することによって高速な計算が可能であるが，コードを最適化するにはいくつか注意しておかねばならない点がある．ここからは，先行研究のソースコードに対して行った最適化の説明を行う．

(1) コアレスアクセス化

GPU 上の処理は並列計算が基本であり，かつ同時に実行されるスレッド数が CPU に比べて膨大であるため，メモリアクセスに注意を払わなければ競合が起こって大幅にその性能を低下させてしまうことになる．近年の GPU はキャッシュメモリのサイズも大きくなってきたため，以前ほどコーディングによる差は出にくいとされるが，それでも未だ手動での最適化による効果は小さくない．

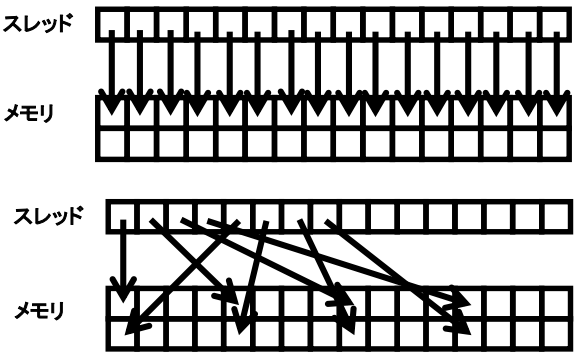

図 8　コアレスアクセス（上）とランダムアクセス（下）

CUDA にはコアレスアクセスと呼ばれる機能が搭載されている．これは，複数のスレッドがメモリから同時にデータを読み込む際，アクセスされるメモリ領域が連続であれば，1 回分のアクセス時間ですべてのスレッドにおける読み込み処理を完了させることができるというものである．メモリアクセスの様子を図 8 に示す．

先行研究においてはメモリアクセスのタイミングが考慮されておらず，ほぼランダムアクセスとなっていたため，コアレスアクセスに改めた．

(2) シェアードメモリの使用

NVIDIA 社の GPU には複数種類のメモリが搭載されている．特に宣言せず一般的に使用するグローバルメモリや，自動的に管理されているキャッシュメモリなど，CPU にて馴染みのあるメモリの他，テクスチャ画像などの扱いに長けるテクスチャメモリ，読み込み処理に特化したコンスタントメモリなどがある．

基本的なメモリとして使用されるグローバルメモリは，1～2GB と容量は大きいものの，1 回のアクセスには約 300 クロックほどかかるため，何も考えずにグローバルメモリのみを使用していてはプロセッサを待たせることとなり，GPU で処理を行う意味が無い．

本研究では，シェアードメモリを使用してメモリアクセスを高速化する．シェアードメモリはグローバルメモリと比較して圧倒的にサイズが小さく，またデータを共有できるスレッドの範囲も狭まるものの，アクセスは 3 クロックほどで済む．データの共有範囲とそのサイズが限定的な処理の場合に使用すると，処理速度を高速化できる可能性がある．

今回，シェアードメモリには動き予測の処理を行なっているブロックの画素値を置くことにした．

(3) リダクションの改善

配列の各要素に格納されたデータを集約して 1 つの情報を得るような処理をリダクション（縮約）と言う．逐次的なプログラミングでは通常，リダクションはループを回して配列要素を 1 つずつ参照することで実現することが多い．しかしこれを GPU 上でそのまま行うと大量のプロセッサが遊んでしまい，大きなボトルネックとなる．リダクションも可能な限り並列化するべきである．

本研究では，図 9 に示すようなトーナメント式のリダクションアルゴリズムを導入して改善を図ることとした．

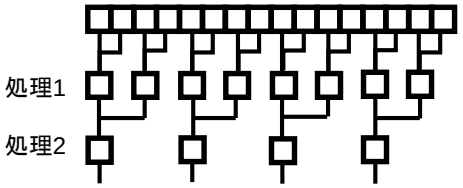

図 9　トーナメント式リダクション

(4) その他

先行研究のソースコードでは，GPU 上のメモリに動画データを移行する際，フレーム毎に領域の確保と破棄を繰り返していた．このような処理は CPU 用のアプリケーションでは普通に行われることであるが，やはり GPU の処理速度が早いために相対的にボトルネックとなる．本研究では，作業用の領域確保は 1 回に止め，エンコードが終了するまでその領域を使いまわす仕様へと変更した．

また，細かい関数については統合を行い，関数呼び出しのオーバーヘッドを削減した．

5.5 速度評価

速度評価においては，無加工の x264 エンコーダと，先行

研究における CUDA 実装版，そして本研究で最適化を行ったものの 3 つについて実験を行った．使用した GPU は NVIDIA GeForce GTX580 である．エンコードする動画の解像度についても，SD，HD，Full-HD の 3 種類を用意し，エンコードにおけるフレームレートを測定した．表 1 はその結果を示したものである．

通常，動画の（再生における）フレームレートはアニメーションなどで 24fps，実写映像では 30fps である．表 1 の結果によると，先行研究では SD サイズの動画は余裕を持ってリアルタイムエンコードできる．CPU 版では SD サイズのエンコードでも 20fps 程度しか出せなかったことから，改善はしているものの，HD 以上については処理速度に難があった．

本研究で最適化を行ったものについては，これに加えて HD サイズ圧縮時フレームレートが 30 を上回っており，余裕を持ってリアルタイムエンコードが可能となった. なお，Full-HD のリアルタイムエンコードにはまだ速度が足りないようである．

表 1　GPU 実装版のエンコード速度 (fps)

	Full-HD	HD	SD
先行研究	10.42	21.09	48.8
本研究	19.18	36.81	90.57

前述した実装の最適化のうち，大きな効果があったのはシェアードメモリの使用とコアレスアクセス化であった．それぞれ単独で行った場合，動き予測モジュールの処理時間は従来比で 1.7 倍と 1.5 倍である．

最適化を行った CUDA 版の x264 においては，動き予測モジュールの処理時間は全体の 11%程度まで落ち込んだ．従って，さらなる高速化を目指す場合には他のモジュールを GPU 版に差し替える必要があると考えられる．

6. まとめ

本研究では，設備投資などに制約のある環境下でのリアルタイム動画通信を想定した動画圧縮手法の提案と，GPU によるエンコーダの高速化を行った．

提案手法では，選択領域の画質を保持したまま外縁部の情報量を削減することによって動画のビットレートを削減し，部分的にではあるものの精細度を保った動画圧縮が行えるようにした．その結果，1Mbps 前後の動画においては 4 割程度のビットレート削減効果があった．

また，GPU による高速化では，処理時間の占める割合が大きかった動き予測処理の高速化を試みた．結果，動き予測の処理時間を大幅に短縮することに成功し，HD サイズ以下の動画については余裕を持ってリアルタイムエンコードが可能となった．

今後の展望について述べる．まずは提案手法を実装したエンコーダを用いてネットワーク通信網を形成し，通信を含めた性能を評価する必要がある．これについては，現在オープンソースのメディアプレイヤーである VLC メディアプレイヤーとの連携作業を行なっている．

また，実際の現場では単純な 1 対 1 通信だけでなく，1 対多通信や，手元のストレージへの保存を並行して行う要求も存在し得る．それぞれの回線や保存媒体に合わせ異なる品質調整を行った圧縮動画を同時配信できるよう実装を施し，その上で性能調査を行う必要がある．

そして GPU による高速化についても, 他のモジュールにおける可能性を検討する．圧縮処理の占める時間が小さくなれば, その分追加処理を行う余裕が生まれ, 実用的な様々な機能を搭載することも可能になるはずである．

参考文献

1) 高度情報通信ネットワーク社会推進戦略本部: 新たな情報通信技術戦略，http://www.kantei.go.jp/jp/singi/it2/100511honbun.pdf（2012 年 11 月 5 日閲覧）
2) 安藤英俊, 望月雄太: GPU によるウェーブレット変換を用いた高精細映像の低遅延ネットワーク配信，情報処理学会研究報告, Vol.2009-CG-136 No.10 (2009)
3) 柏木 賢治，郷 健太郎: 眼科遠隔診療ロボットの診断能力の検討，日本遠隔医療学会雑誌, Vol.5, No.2, pp.256-257 (2009).
4) 大久保 榮，角野 眞也，菊池 義浩，鈴木 輝彦: H.264/AVC 教科書 改定三版，p86，impress R&D (2009)
5) Lawrence Chan, Jae W. Lee, Alex Rothberg, and Paul Weaver: Parallelizing H.264 Motion Estimation Algorithm using CUDA, IAP 2009 CUDA@MIT/ 6.963 (2009)