J_009

画素ビット長拡張による動画像符号化効率改善方式

Improving Video Coding Efficiency by Pixel Bit-depth Increase

野田 玲子† 中條 健†
Reiko Noda Takeshi Chujoh

1. まえがき

現在，最も高い圧縮性能を誇る映像符号化方式の一つとして，H.264/MPEG-4 AVC[1]がある．H.264 では，半画素精度の動き補償フィルタとして，6 タップのフィルタを用いて処理を行うため，高次の周波数まで再現が可能ある．しかし，この処理は，複数画素の画素値にフィルタ係数を掛けて足し合わせた後，入力画像と同じ画素ビット精度に丸める処理がなされるため，丸め誤差が生じ，予測の精度を高める上で十分とは言えない．本稿では，入力画像の各画素の値を左にビットシフトして 2 の N 乗倍の値に変換し，入力画像の画素ビット精度を拡張することで，動き補償フィルタの処理で生じる丸め誤差を小さく抑えて符号化効率を向上させる方式を提案する．また，入力画素ビット長を拡張することにより，符号化内部での演算精度が増加するため，データパスの一部を元の入力画像のビット精度に制限することで，メモリや演算精度を制限しつつ，符号化効率を改善する方式についても検討を行った．

2. H.264 における動き補償フィルタと丸め誤差

H.264[1]では，半画素位置を補間するフィルタとして{1/32, -5/32, 20/32, 20/32, -5/32, 1/32}という 6 タップの補間フィルタを用いて，隣接する複数画素の画素値にフィルタ係数の分子を掛けて足し合わせた後 32 で除算し，小数点以下を四捨五入で丸めることにより，入力画像と同じビット精度で半画素位置の画素値を求めている．ここで生じる丸め誤差によって本来の半画素位置の値よりもずれた画素値に補間されることになる．

図 1に，8bitの入力画像の画素値に対し，H.264 の動き補償フィルタを用いて，8bitで半画素位置を補間した場合と，入力画素の各画素の値を左に 4bitシフトし，12bitに拡張して半画素位置を補間したときの画素値を求めた例を示す．横軸は画素位置，縦軸左及び縦軸右はそれぞれ 12bitに拡張した画素値，8bitでの画素値を示している．入力画素を 12bitに拡張したことにより，補間フィルタにおいて 32 で除算した際の四捨五入による丸め誤差が小さくなり，8bit での補間に比べ，より入力画像に忠実に補間されることが判る．

以上から，入力画素ビット長を拡張することで，動き補償フィルタの演算精度が向上し，結果として予測残差を小さく抑えることができると考えられる．H.264 では，入力画素ビット長に対し 5bit の拡張を行うことで，半画素精度の動き補償フィルタの演算精度を十分向上させることが可能である．また，動き補償フィルタだけでなく，画面内予測やループフィルタの精度を向上させる効果もあると考えられる．

†株式会社 東芝 研究開発センター

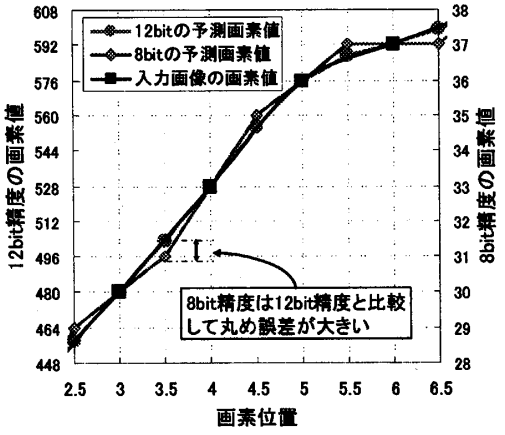

図 1 半画素精度の予測画素値

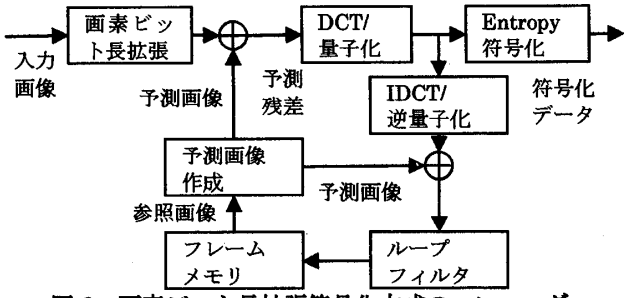

図 2 画素ビット長拡張符号化方式のエンコーダ

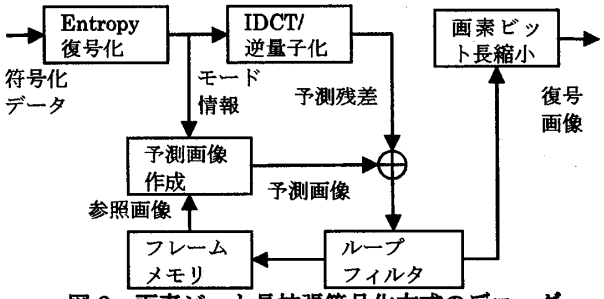

図 3 画素ビット長拡張符号化方式のデコーダ

3. 画素ビット長拡張符号化方式

3.1 画素ビット長拡張符号化方式の基本構成

図 2，図 3に画素ビット長拡張符号化方式を含むエンコーダ及びデコーダの構成を示す．符号化側では，まず 8bit の入力画像の各画素をMビット左シフトしてからエンコーダ本体に入力する．画素ビット長が拡張された入力画像は，H.264 とほぼ同じ処理で符号化される．

一方，復号側では，入力された符号化データを H.264 とほぼ同じ処理で復号化した後，復号画像の各画素を M ビット右シフトによって 8bit に変換してから出力する．この際，丸めの処理は四捨五入で処理を行う．

符号化及び復号化の内部では，画像信号が M ビット拡張されているため，フレームメモリを(8+M)bit 以上の画素ビット長で確保する必要がある．また，DCT/IDCT，量子化/逆量子化，ループフィルタなどは，拡張されたビット数に応じた演算精度が必要となる．これらの処理は 8bit 以上の入力に対応した High 10 以上の FREXt の Profile[1]と同等の処理を適用した．

3.2 フレームメモリ／演算コストの削減

画素ビット長符号化方式は，前述のとおり，フレームメモリと符号化内部での演算精度が増加するため，前節で述べた方式のようにすべてのデータパスにおいて拡張したビット長で処理を行うのではなく，一部のパスの演算精度を制限する方式について，以下の 3 方式を検討した．

(1) フレームメモリを 8bit に制限
(2) 直交変換と量子化の演算精度を 8bit 入力相当に制限
(3) (1),(2)の双方を適用

(1)については，ローカルデコード画像を 8bit に変換してからフレームメモリに格納し，予測画像を作成する際に(8+M)bit に拡張すると，フレームメモリの容量を 8bit に抑えたまま予測画像生成の際の動き補償フィルタの精度を高めることができる．3.1 の方式において，画素ビット長を(8+M)bit(0<M≦8)に拡張した場合，バイトアライメント処理を行うとすると，フレームメモリとして 1 画素につき 2 バイト必要であるが，この方式は 1 画素につき 1 バイトで格納することができるため、3.1 の方式の 1/2 のメモリサイズで実現できる．

(2)については，(8+M)bit で作成した予測画像信号を 8bit に戻して 8bit の入力画像との残差信号を生成することで，直交変換と量子化の演算精度を 8bit 入力相当に制限したまま，予測画像生成の際の動き補償フィルタの精度を高めることができる．3.1 の方式で画素ビット長を(8+M)bit に拡張した場合，デコーダでの直交変換と量子化の演算精度は(16+M)bit 必要であるが，この方式は 16bit の演算精度で実現でき，回路規模の削減やソフトウェア処理における並列化が容易となる．

(3)については，(1)(2)の制限を組み合わせることにより，フレームメモリ及び直交変換と量子化演算精度の双方を抑えつつ，予測画像生成の際の動き補償フィルタの精度を高めることができる．

4. 性能評価

画素ビット拡張符号化方式を適用したときの効果を確認するため，8bitの評価素材(1280x720@30p, 289frame) 4 種類に対し，H.264 High Profileと 3.1 の方式（提案方式(a)）および 3.2 の 3 方式(1)(2)(3)（それぞれ提案方式(b)～(d)とする）で画素ビット長を 12bitに拡張して符号化を行い，原画像(8bit)と復号画像(8bit)とのPSNRの比較を行った．符号化条件は表 1のとおりである．

表 2にH.264 High Profileに対する提案方式の符号量削減率を示す．また，図 4にShuttle StartのR-D曲線を示す．提案方式はH.264 High Profileに比べて，左上に位置していることが判る．また，レート歪曲線から，提案方式は高レートで効果が高いことが確認できた．

また，すべてのデータパスを 12bit に拡張して符号化した場合(a)と比較して，フレームメモリや予測残差を 8bit に制限する方式(b)(c)(d)の R-D 曲線は右下に位置するものの，ビット拡張を行わないで符号化した場合に比べると左上にある．これらの結果から，フレームメモリや予測残差など，符号化のデータパスの一部を 8bit に制限すると，若干効果が少なくなるものの，依然ビット拡張の効果があることが確認できた．

また，Shuttle Startについて，表 1の符号化条件で方式(a)における拡張画素ビット長を 10bit, 14bitとした場合の符号量削減率は、それぞれ 7.98%, 9.32%となった．このことから，4bit程度の拡張を行えばビット拡張の効果が十分得られることが分かる．これは動き補償フィルタの丸め誤差の符号化効率への影響が 4bit程度の拡張で十分小さくなるためと考えられる．

表 1 符号化条件

エントロピー符号化	CABAC
符号化予測構造	M=3 (IBBPBBPBB...)
量子化パラメータ	(16, 20, 24, 28)
参照フレーム枚数	2 枚

表 2 符号量削減率(12bit に拡張した場合)

方式	(a)	(b)	(c)	(d)
Big Ships	3.31%	2.22%	1.94%	2.02%
City	5.46%	4.25%	2.93%	3.41%
Crew	3.35%	1.52%	1.94%	0.88%
Shuttle Start	9.56%	6.07%	5.57%	5.50%
Average	**5.42%**	**3.51%**	**3.10%**	**2.95%**

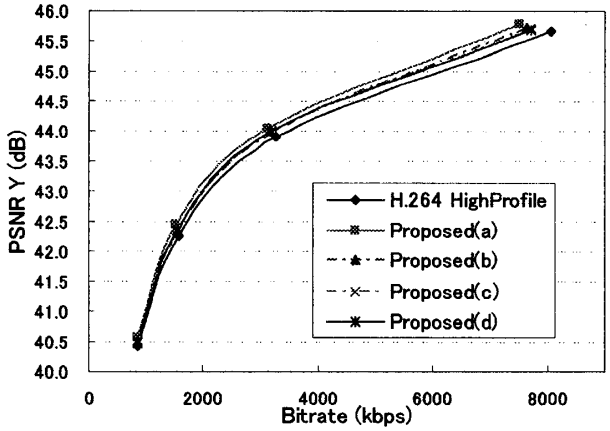

図 4 Shuttle Start の R-D 曲線

5. まとめ

H.264 の動き補償フィルタの丸め誤差に着目し，画像の画素ビット長を拡張することで，この丸め誤差を小さく抑え，符号化効率を向上させる方式について検討を行った．8bit の入力画像に対し，提案方式により，画素ビット長を 12bit に拡張して符号化することで，H.264 High Profile と比較して最大 9.56%, 平均 5.42%の符号量削減効果がある．また，フレームメモリや予測残差を 8bit に制限しても，平均 3%前後の符号量削減効果が得られ，効果は小さくなるが，フレームメモリや一部の演算精度を制限しても効果があることも確認できた．

参考文献

[1] Recommendation H.264 (03/05) , ISO/IEC 14496-10 : 2005.