UDC 621.397.3＋681.3.06＋006

超高速JPEGコーデックの開発

浅井　有人*，Ta thi Quynh Lien**，野中　俊一郎*，羽田　典久*

Very High Speed JPEG Codec Library

Arito ASAI*, Ta thi Quynh Lien**, Shunichiro NONAKA*, and Norihisa HANEDA*

Abstract

This paper proposes a high-speed method of directly decoding and scaling down high-resolution JPEG images to low-resolution and small ones, which achieves a dramatic improvement in speed. The algorithm has an insignificant problem which causes noise in the rightmost and undermost pixels of output images in some minor cases. We have worked out some methods to lessen this problem, while hardly slowing down the processing speed.

Experiment results show that our approach, including the above related work, has successfully achieved a 5 to 10 times increase in speed, in comparison with methods using the de-facto standard JPEG library libjpeg.

1. はじめに

近年，デジタルカメラや携帯電話に搭載されたカメラの高解像度化により，大容量高解像度の画像が数多く生成され，利用されている。しかしながら，これら高解像度画像データは，その展開および表示に多くの計算機時間を要するため，取り扱いは容易ではない。特に，現実の表示デバイスが対応する解像度は，それら高解像度画像よりも低いため，高解像度画像の展開，縮小，表示という手順を毎回要している。

この展開，縮小，表示にかかる時間は，ありふれた画像の一覧，検索画面などのように大量のサムネール画像を同時に多数表示するユーザインターフェースにおいて，著しいユーザビリティーの低下を招いている原因にもなっている。

本論は，JPEGデータのDCT（Discrete Cosine Transform）係数上での間引きを行なうことによって，高解像度JPEGデータから直接に低解像度の縮小された画像を展開処理することで，飛躍的な高速化を実現した。また，本方式にまつわる諸々の問題点の解決方法について報告する。

2. JPEGデータの展開コストと縮小コスト

本論ではIJG（Independent JPEG Group）のlibjpegをもとに，高速化の道筋をたどる。Fig. 1は，libjpegを用いてJPEGを単純にRGB展開するのに要する時間コストを示したものである。これからわかるように，JPEGデータの展開に要する時間は，展開後画像の画素数に大きく依存する。

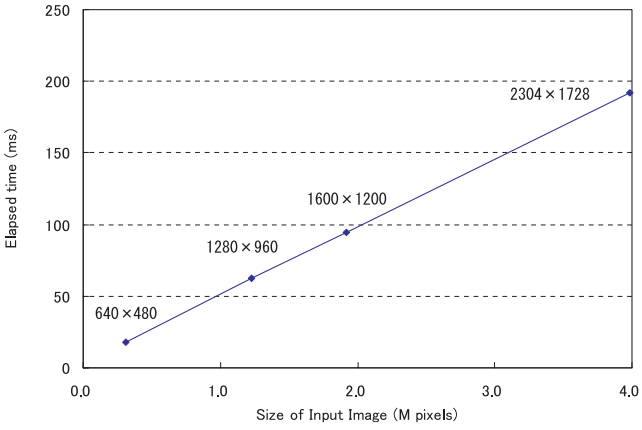

Fig. 1 Orthodox JPEG decompression to RGB bitmaps.

本誌投稿論文（受理2007年12月20日）

*富士フイルム（株）新規事業開発本部
ネット応用ビジネス推進部
〒351-8585　埼玉県朝霞市泉水3-13-45

*Internet Business Development Division
New Business Development Division
FUJIFILM Corporation
Senzui, Asaka, Saitama 351-8585, Japan

**富士フイルムソフトウエア（株）
〒215-0004　神奈川県川崎市麻生区万福寺1-2-2

**FUJIFILM Software Co., Ltd.
Manpukuji, Asou-ku, Kawasaki, Kanagawa 215-0004, Japan

さらに，高解像度で展開された画像を所定の解像度に縮小するための時間コストをFig. 2，Fig. 3に示す。用いた縮小アルゴリズムはBiCubic法に基づく補間アルゴリズムである。このアルゴリズムは畳み込みによる積和計算を主とするので，SSE2に代表されるようなSIMD命令が非常に有効に働き，およそ3～5倍の性能向上を得ることができた。また，ここからわかる通り，縮小コストは元画像サイズのピクセル数と，縮小後のピクセル数の両方に大きく依存する。

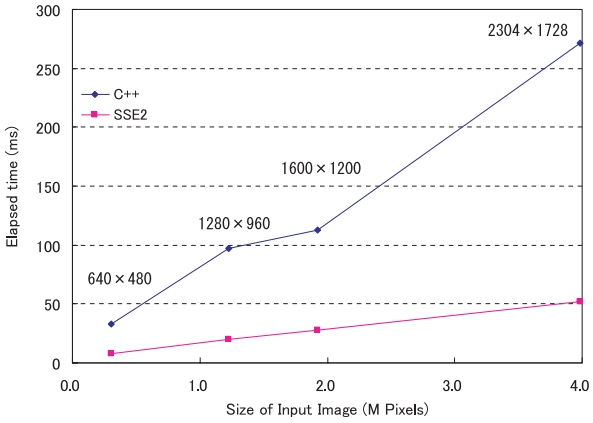

Fig. 2 RGB bitmap scaling (fixed output size QVGA).

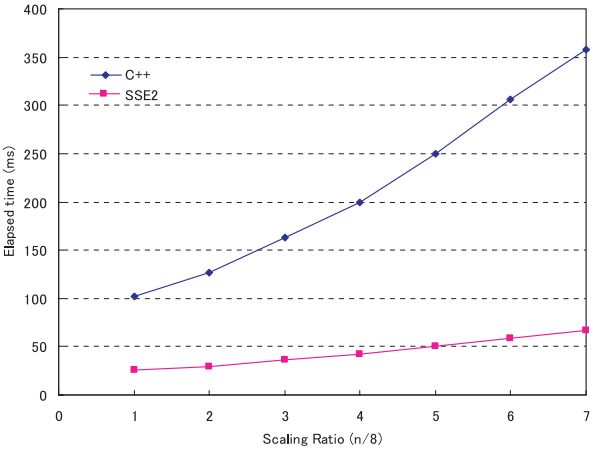

Fig. 3 n/8 RGB bitmap scaling (fixed input size 1600×1200).

3. DCT係数の間引きによる縮小展開

JPEGデータは8×8ピクセルのブロック単位でDCT処理を行ない，周波数成分上で不可逆圧縮をしている。展開においては，これらのDCT係数の逆DCT変換を行なうことになるわけであるが，最終的に縮小画像を取り出したいという要求に対して，縮小画像に含まれ得ない高周波成分のDCT係数を展開することは無駄な処理であり，必要な周波数までを含むDCT係数まで間引いてJPEGを展開することによって，この無駄な計算時間と記録容量を節約することができる。DCTの性質により，DCT係数の低周波部分の係数のみを残し，残された係数に対して残された次数の逆DCTを行なうことで，縮小展開されたデータを得ることができる。$1/2^n$の縮小については，FFT（Fast Fourier Transform）におけるバタフライ演算と同様に高速な逆DCTが知られており，高速に処理を行なうことができる。これにより，8×8のブロックを直接に1×1，2×2，4×4のブロックに縮小展開できる。縮小率はそれぞれ1/8，2/8，4/8となっている。これを$1/2^n$縮小展開と呼ぶ。

一歩進めて，3/8，5/8，6/8，7/8の場合を考えてみよう。これらの場合については高速な逆DCTが知られていなかったため，縮小展開を行なっても逆DCTがネックになりメリットがなかった。われわれは高速なn次（n=3, 5, 6, 7）の逆DCT演算法を発明し，n/8に縮小展開しながら高速にJPEGデコードを行なうことを可能にした。これと前述の$1/2^n$縮小展開を併せて，n/8縮小展開が実現された。この方式と等倍で展開して，BiCubic法（SSEを用いる）で縮小した場合の時間コストの比較はFig. 4の通りで，2～6倍の高速化を達成した。

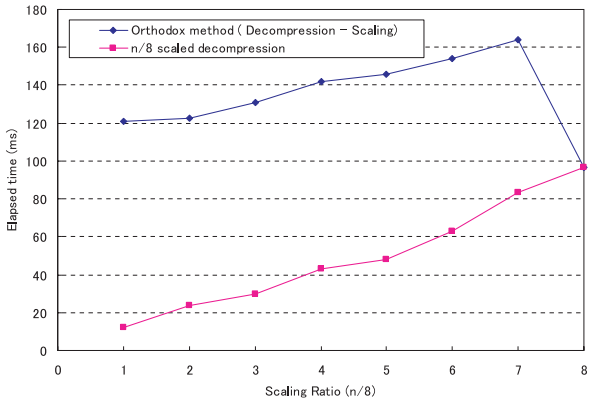

Fig. 4 n/8 Scaled JPEG decompression (fixed input size1600×1200).

4. 画像端の諸問題

4.1 パディング領域

JPEGデータは前述の通り8×8ピクセルのブロック単位で処理が行なわれるため，幅もしくは高さのピクセル数が8の倍数とならない場合，8の倍数まで切り上げられ，切り上げられた領域にはパディング要素が挿入される。このパディング領域にはどのようなデータが挿入されるべきかは規格上明確な定義がない。

通常のJPEG展開においては，パディング領域も含めて全画素を展開した後，本来のピクセル数でパディング領域が切り落とされる。しかしながら，われわれのn/8縮小展開においては，パディング領域を正確に1ピクセルの精度で切り落とすことができないため，画像端（右端と下端）にパディング要素による影響が現れることがある（Fig. 5）。

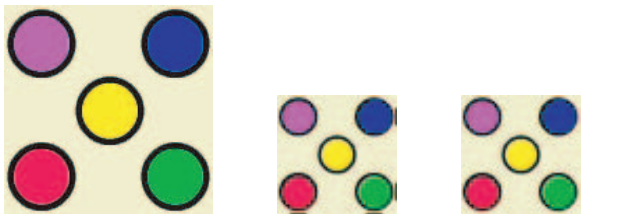

Fig. 5 Sample image with padding error.

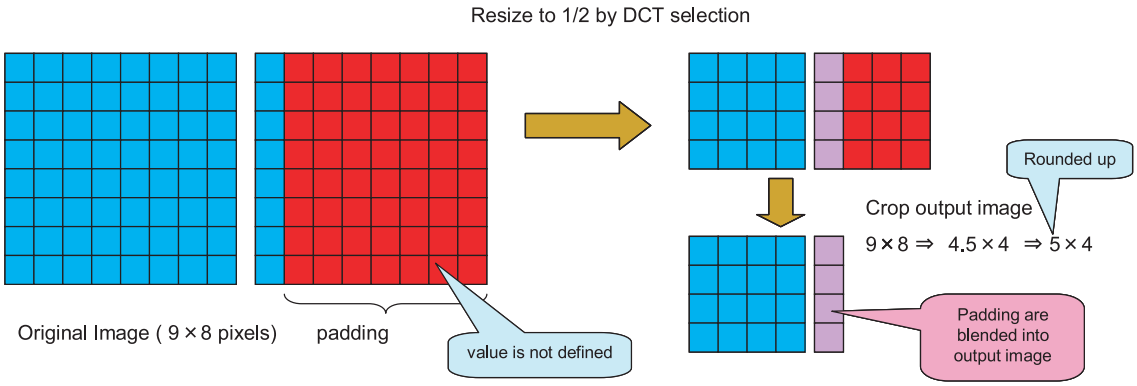

Fig. 6 Principle of padding-blended error.

この現象の発生原理をFig. 6に示す。図では画像の幅が8の倍数+1の場合であり，右端ブロックは幅1ピクセルの正しい画像データと幅7ピクセルのパディング領域からなる。この画像を，例えば，1/2に縮小展開する場合，DCTの間引きにより得られた4×4ピクセルの縮小ブロックにおいて，縮小後解像度の小数以下を切り上げた部分を結果として得，残りをパディング領域として切り落とす。図の場合は，幅1ピクセルを結果として，残りの幅3ピクセルを切り落とす。しかしながら，結果として得た幅1ピクセルの画素の値には，パディングとして付与されたパディングデータを50％含んでしまうことになり，これが問題となりうる。

なお，小数以下を切り上げる理由は，実用上のJPEGデータには1ピクセル幅の縁取りがなされている場合が少なくなく，切捨てた場合には縁線が失われるので好ましくないからである。

本報告ではこの問題を次のように解決した。パディング領域を含むブロックにおけるn/8縮小展開を断念し，これら端ブロックに関しては次のように処理を行なうこととした。①通常通り8×8ピクセルに等倍展開，②パディング要素を端ピクセル画素で上書，③領域平均法によりn/8倍に縮小を，④所望の画素で切り落とす。ここで，等倍展開した後にパディング領域の切り落としを行わないことには理由がある。等倍展開後にパディングを切り落としてから，所定の解像度に縮小を行なうと，このブロックの縮小率が丸め誤差により厳密にn/8に一致しなくなるからである。

また，縮小アルゴリズムに関しては，領域平均法としたが，BiCubic，BiLinear法と比較した場合，DCT係数の間引きによる縮小と周波数特性が最も近く，端ブロックにおいても画像のつながりが最もよいという評価を得た結論になっている。

さらに，本処理に伴う計算時間コストについて考察を行なう。本方式では，幅もしくは高さのピクセル数が8の倍数でない場合，右端，および左端ブロックに関してn/8縮小展開の高速化のメリットを享受することができない。しかしながら，画像の総ピクセル数Sに対して，端ブロックの数は$\sqrt{S}$のオーダーである。つまり，元画像が高解像度であればあるほど端ブロックの割合が急速に減少する。すなわち，元画像が十分に大きければ本方式による時間コストの増加は微々たるものとなる。例えば，2メガピクセルの画像において端ブロックは全体の1.2％に満たない。

4.2 端ピクセル解像度問題

n/8縮小展開アルゴリズムは，8×8のピクセルブロックをn×nのピクセルブロックに展開するという性質上，縮小率は厳密にn/8と一致する。しかしながら，幅および高さのピクセル数が8の倍数とならない元画像の場合，丸め誤差によって正確にn/8倍の幅，高さを得ることができない。それでは，この丸め誤差はどこに行くのであろうか。

前章で，パディングデータの再作成を行なってn/8倍縮小後，切り落としを行なった。つまり，画像の右端，下端のパディング要素と隣接する1ピクセルは複製されたパディング要素と共に縮小され出力されることになっている。このことは，n/8倍の縮小の丸め誤差がすべて右端，下端の1ピクセルに凝縮されることを意味する。別の言い方をすれば，これら端ピクセルの縮小率は，他の領域の縮小率と一致しない。

この結果，縮小率に非常に近い周波数成分において，端ピクセルとその他のピクセルにおいて異なるという現象を引き起こす。実際の例をFig. 7に示す。問題画像の右端ピクセルラインの模様が異なっていることがわか

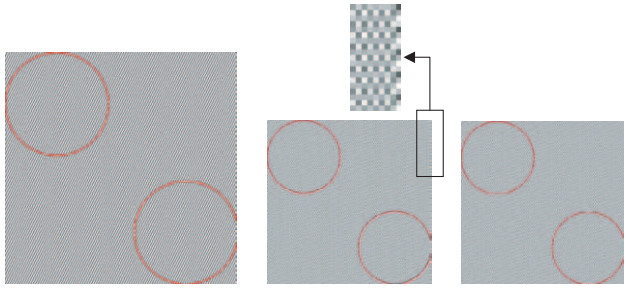

Fig. 7 Sample image with rounding error.

る。その発生原理はFig. 8の通りである。

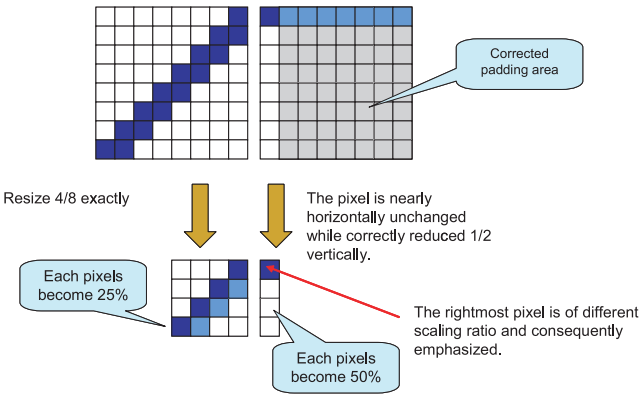

Fig. 8　Influence of rounding error margin in the right-hand edge of image.

本問題を本質的に解決することはできないが，軽減させるためにわれわれは次のようなアルゴリズムを挿入した。

① 前章までの方法でn/8縮小展開（小数点以下切り上げ）の画像Sを求める。
② 四捨五入によるn/8倍サイズの画像バッファKを用意し，SをKにコピーする。
③ Sの端ピクセルの本来のサイズ（小数部分）を求め，隣接ピクセルと按分計算を行ない，修正された画素値を求める。
④ 上記で求めた画素値をKの端ピクセルに上書きする。

本アルゴリズムは，端ピクセルの縮小率が異なることによる元画像からの過剰な寄与分を補正し，隣接ピクセルと分け合うことによって，端ピクセル画素のみが強調される現象を抑制する。

本処理に伴う計算コストの増加分は前節の考え方と同様に相対的に小さい。

4.3　処理時間への影響

本章で述べた端ブロック，および端ピクセルの補正処理によって増加する時間コストはTable 1の通りであり，高解像度画像の高速処理を目的とする限り影響は非常に小さいことがわかる。

Table 1　Elapsed Time for1/8 Scaled Decompression.

Input Image	before (ms)	after (ms)	rate
633×473	2.46	2.61	106%
1273×953	8.53	9.46	111%
1593×1193	12.31	12.69	103%
2297×1721	21.96	22.74	104%

5.　JPEGエンコーダ

前章までは，高解像度のJPEGデータをRGB展開する処理について述べたが，高解像度のJPEGデータを低解像度のJPEGデータに変換するというニーズは大きい。

本章では，このようなニーズにおいてさらに高速化する方法としてYCC処理を導入する。JPEGデータはRGBをYCC空間に変換して，YCC空間上でDCT変換されている。さらに，JPEG圧縮のために，Cr，Cb成分についてはさらにサブサンプルされて保存されている場合が通例である。YUV420のサブサンプル比のJPEGデータでは，Cr，Cb成分は元の画像解像度の1/2となっており，面積にして1/4となっている。すなわち，RGB展開する場合に比べて，YCC展開をした場合，出力データサイズはRGBが3に対して，YCCは1+1/4+1/4=3/2となり半分にまで節約することができる。また，デコード時のYCC→RGB変換と，再エンコード時のRGB→YCC変換の処理を同時に省略することができるため，さらなる高速化が可能である。目的解像度がn/8でない場合は，YCC表現上でそれぞれのプレーンをBiCubic法で縮小することで目的の解像度のYCCデータを得ることができ，これをJPEGエンコーダに直接入力できる。

本方式により，高解像度JPEG画像から低解像度JPEG画像を得るための時間コストはFig. 9のように改善される。

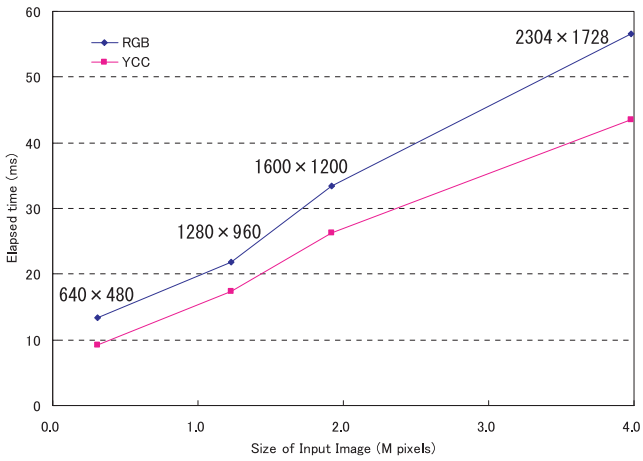

Fig. 9　JPEG decompression－QVGA scaling－JPEG compression sequence.

6.　ビットレートコントロール

携帯電話に代表されるようなデバイスにJPEG画像を提供するためには，JPEGデータのデータサイズに制限をつける必要が発生する。JPEGデータのデータサイズは，画素要素，および量子化係数によって変化する。外部から与えることができるのは量子化係数であり，この係数を増減させることでJPEGデータのデータサイズ，そして，画質もコントロールすることができる。しかしながら，量子化係数とビットレートの関係は複雑であり，画素要素に大きく影響されるため，これをあらかじめ予見することはできない。

従って，JPEGデータのビットレートコントロールには，近似式メトリクス上での二分検索を行ない，所望の範囲に入る量子化係数を反復実行して求めることにな

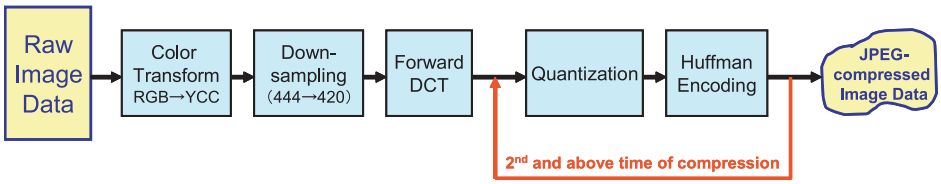

Fig. 10 Compression sequence and retrying pattern.

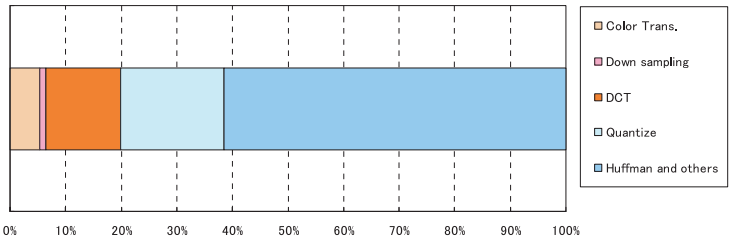

Fig. 11 Percentage of elapsed time for each JPEG compression sequence.

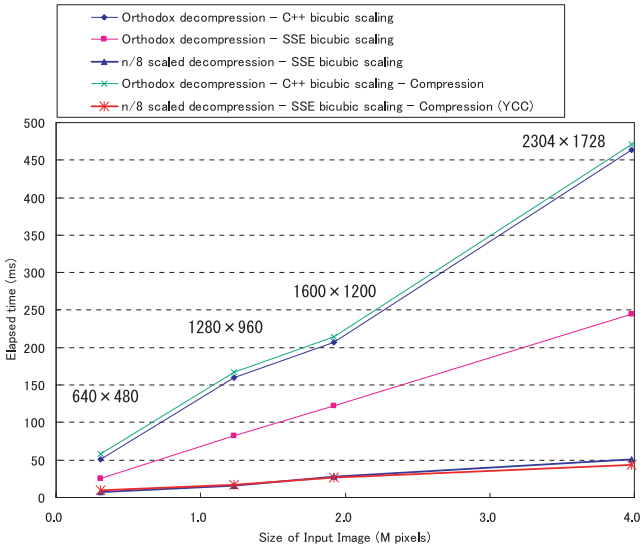

Fig. 12 Time required to convert JPEG into QVGA.

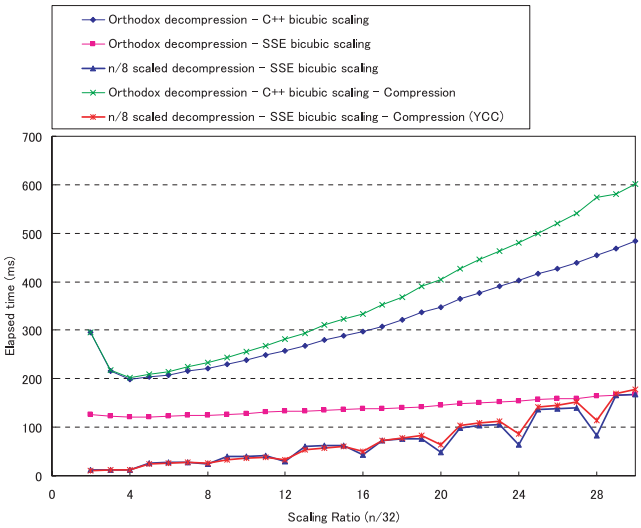

Fig. 13 Time required to compress 1600×1200-sized JPEG into each size.

る。JPEGエンコーダの一連の処理ロジックで量子化係数を変えながらの反復実行を行なうに当たって，RGB→YCC変換，およびDCT変換は初回のみの実行とし，2回目以降の反復試行においては省略することができる(Fig. 10)。JPEGエンコードにおけるDCT係数の計算処理までのコストは，Fig. 11に示す通り20％程度である。

7. まとめ

これまでに述べたように，数々の手法を取り入れることで，高解像度画像を高速かつ高品質に低解像度展開し，表示もしくは再JPEGエンコードすることが可能となった。さらに，その際の端点ピクセルに見られるような画質劣化を回避している。まとめとしてのベンチマークをFig. 12，Fig. 13に示す。縮小率がn/8にならない部分に関しては，SSE実装によるBiCubic法を用いて2段階縮小を含んでいる。そのため，Fig. 13におけるn/8のポイントと，それ以外のポイントでは差が見られる。

このように，高解像度画像の縮小展開，およびサムネイルJPEG作成にかかる計算時間コストは5倍～10倍と大幅に高速化することができた。本技術は，当社の携帯電話向け画像変換サービスKeitai Pictureに搭載されているほか，携帯電話に始まる表示デバイスサイズへの高速縮小，サムネイル一覧画像の高速作成など，さまざまなアプリケーション分野に応用，利用されている。今後，さらに多くの画像アプリケーションに展開されると考えられる。

参考文献

1) ISO/IEC 10918-1:1994, Information technology -- Digital compression and coding of continuous-tone still images: Requirements and guidelines.
2) Thomas G. Lane, USING THE IJG JPEG LIBRARY, libjpeg.doc included in jpegsrc.v6b.tar.gz.